Systems Engineering Challenges in Engineering-In Software Assurance to the System Acquisition Lifecycle

21st Annual Systems and Mission Engineering

Dr. Kenneth E. Nidiffer

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM18-1225

Challenges: Engineering-In Software Assurance to the System Acquisition Lifecycle

1. Increasing software-enable systems are a significant strategic resource
2. Satisfying interconnected operational mission and business needs
3. Addressing the expanding code base
4. Finding software assurance measures that scale
5. Mitigating the challenges to our technical base
6. Working in the infancy of the software enabled-systems
7. Addressing a moving target
8. Designing-in software assurance over the lifecycle
9. Understanding attack patterns, vulnerabilities, and weaknesses
10. Improving our acquisition processes for faster delivery of capabilities

* SDLC: System Development Lifecycle

1. Increasing software-enable systems are a significant 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

Streamlined Acquisition, Commercial Practices, and Iterative Development are Getting High Emphasis From Executive Management…

USD (A&S)

Hon. Ellen M. Lord
Under Secretary of Defense for
Acquisition, Technology, and Logistics

Software is the "thread that runs through all our programs. It's the functional area that I have focused on."

"both the department and industry are behind the curve in terms of modernization of software practices."

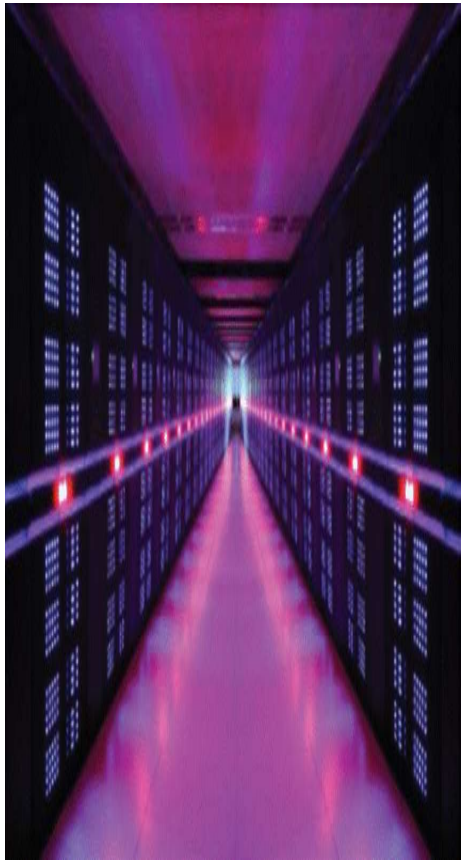
"I believe we are at an inflection point in terms of doing things differently. We are pivoting from the traditional waterfall software development methodology to agile and DevOps. So we are coding every day, testing every night."

Jeff Boleng
Software Engineering Institute
Special Assistant, Software Acquisition

2. Satisfying Interconnected Operational Mission and Business Needs

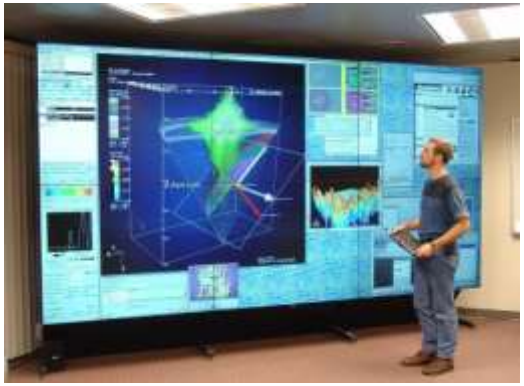
Manufacturing

Finance



Research

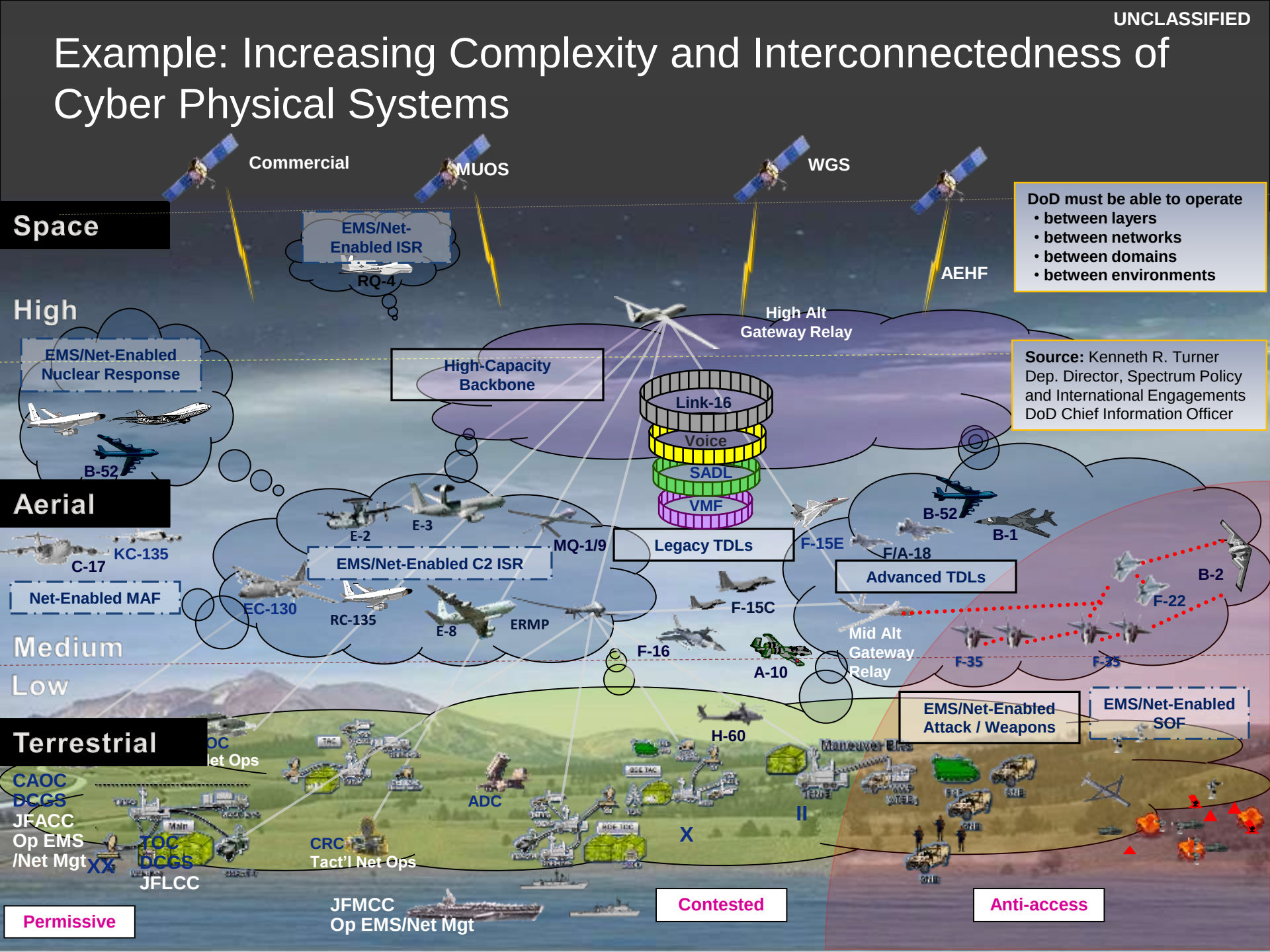
Space and Aviation



Engineering

Source: SEI

Example: Increasing Complexity and Interconnectedness of Cyber Physical Systems



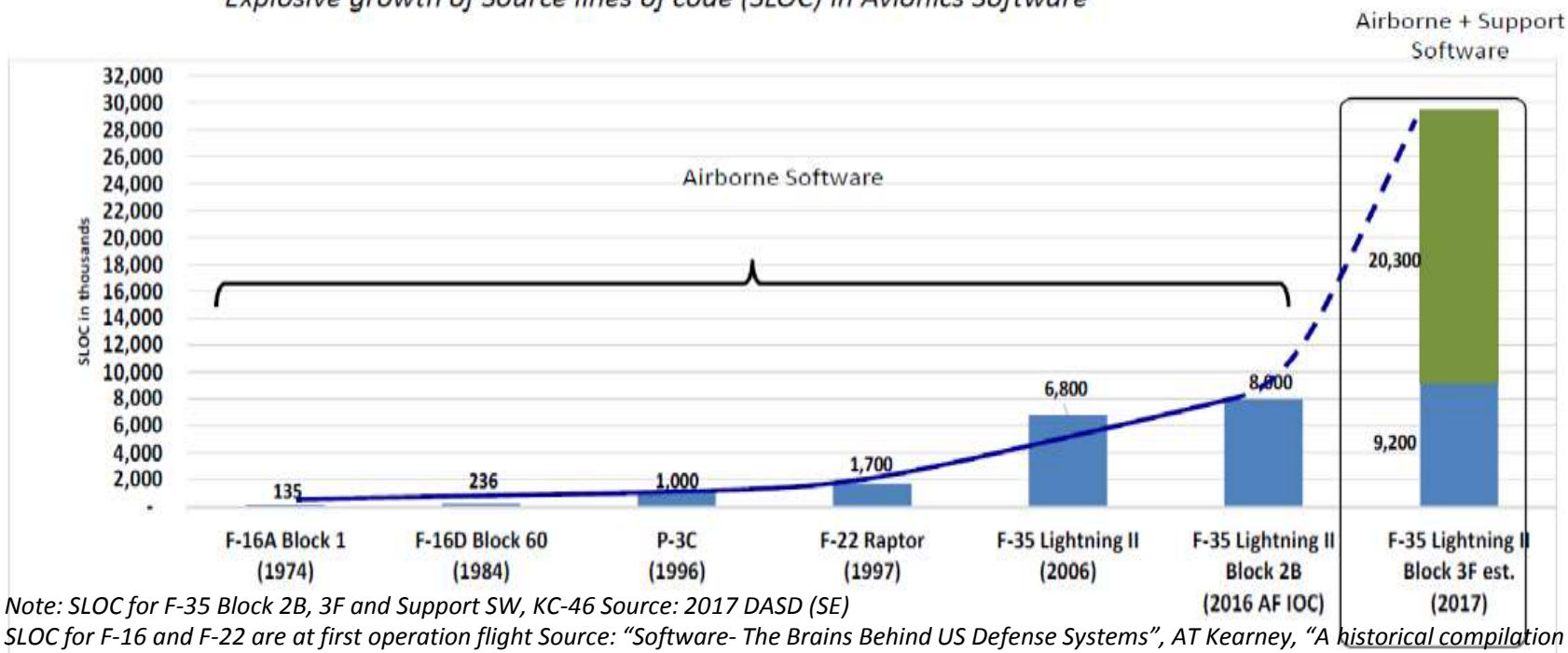
3. Addressing the Expanding Code Base

Driving Revolutionary Change in DoD Software Design and Acquisition

DoD Software Growth

- DoD Software complexity and size rapidly growing

Explosive growth of Source lines of code (SLOC) in Avionics Software



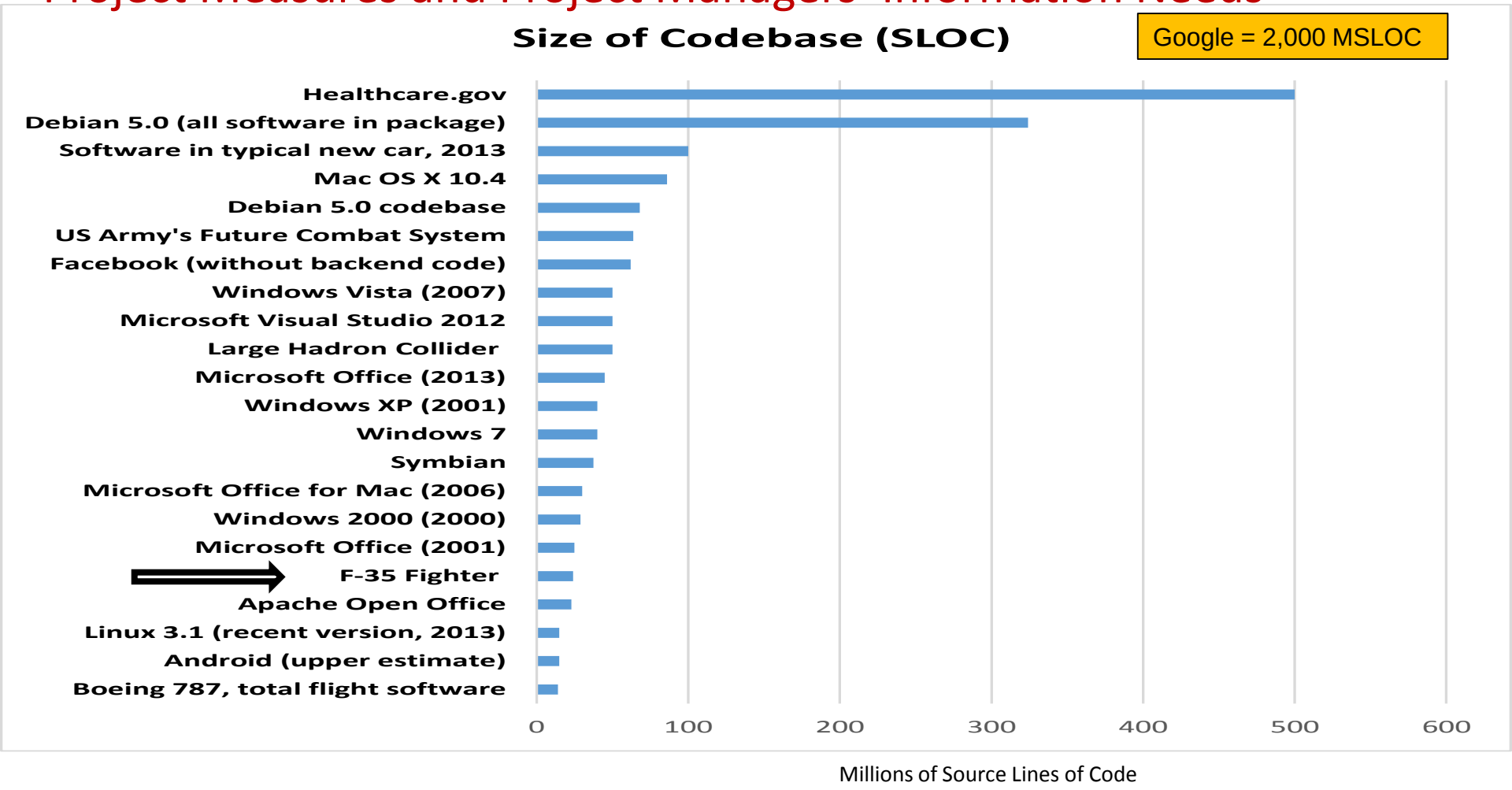
Note: SLOC for F-35 Block 2B, 3F and Support SW, KC-46 Source: 2017 DASD (SE)

SLOC for F-16 and F-22 are at first operation flight Source: "Software- The Brains Behind US Defense Systems", AT Kearney, "A historical compilation of software metrics with applicability to NASA's Orion spacecraft flight software sizing", Judas, Paul A, and Prokop, Lorraine E., Innovations in Systems and Software Engineering: A NASA Journal, DOI 10.1007/s11334-011-0142-7, 2011 NASA

Source: DoD Defense Science Board Study Design and Acquisition of Software for Defense Systems, February 2018

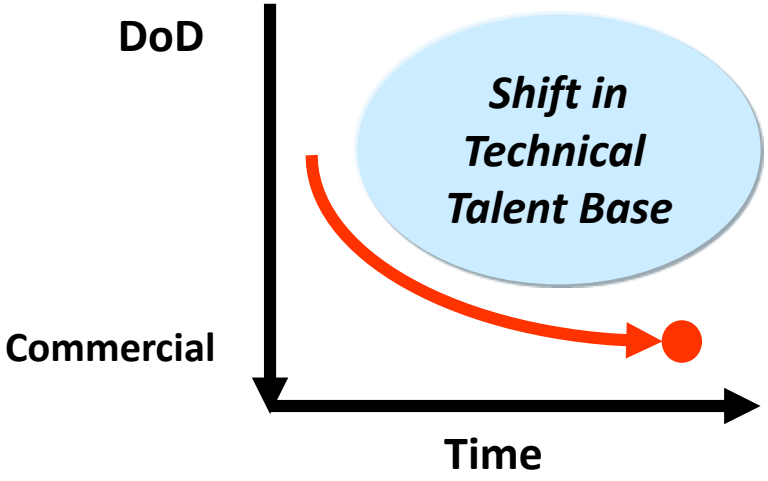
4. Finding Assurance Measures that Scale is a Key Assurance Issue

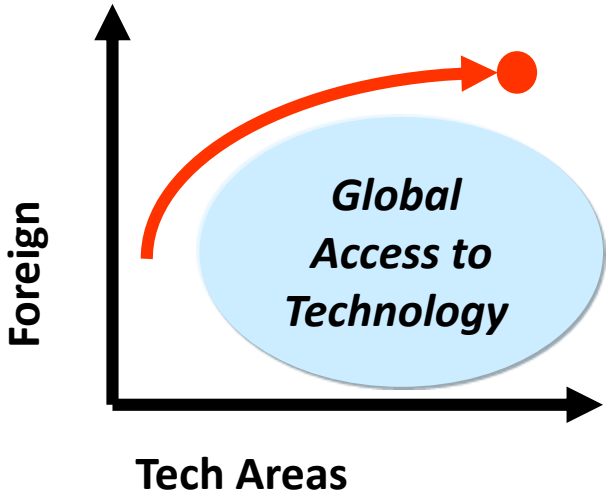
Growing Gap Between Information Obtained Using Traditional Project Measures and Project Managers' Information Needs

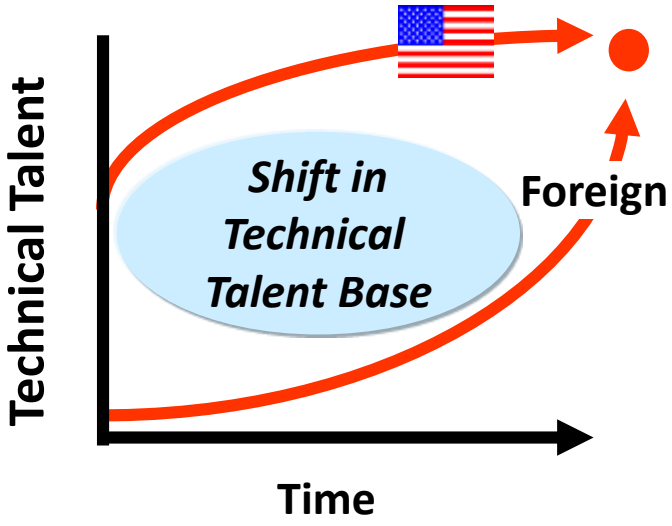


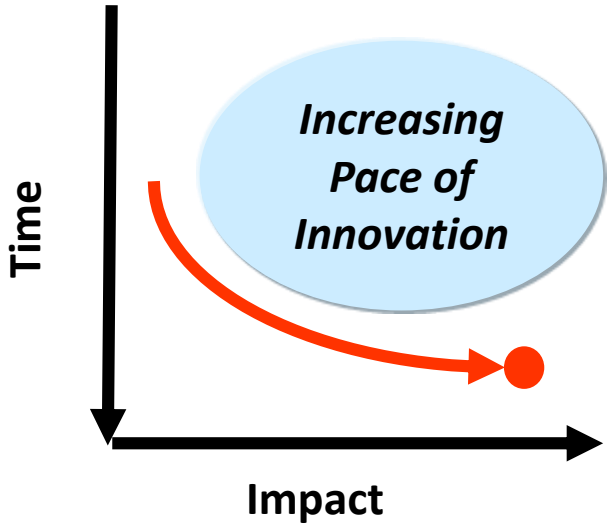
Source: David McCandless, "Information Is Beautiful," 29 August 2018, web retrieval

5. Mitigating the Challenges to Our Technical Base









Source: The Honorable Zachary J. Lemnios, Director, Defense Research and Engineering, October 2010

6. Working in the Infancy of Software-Enable Systems

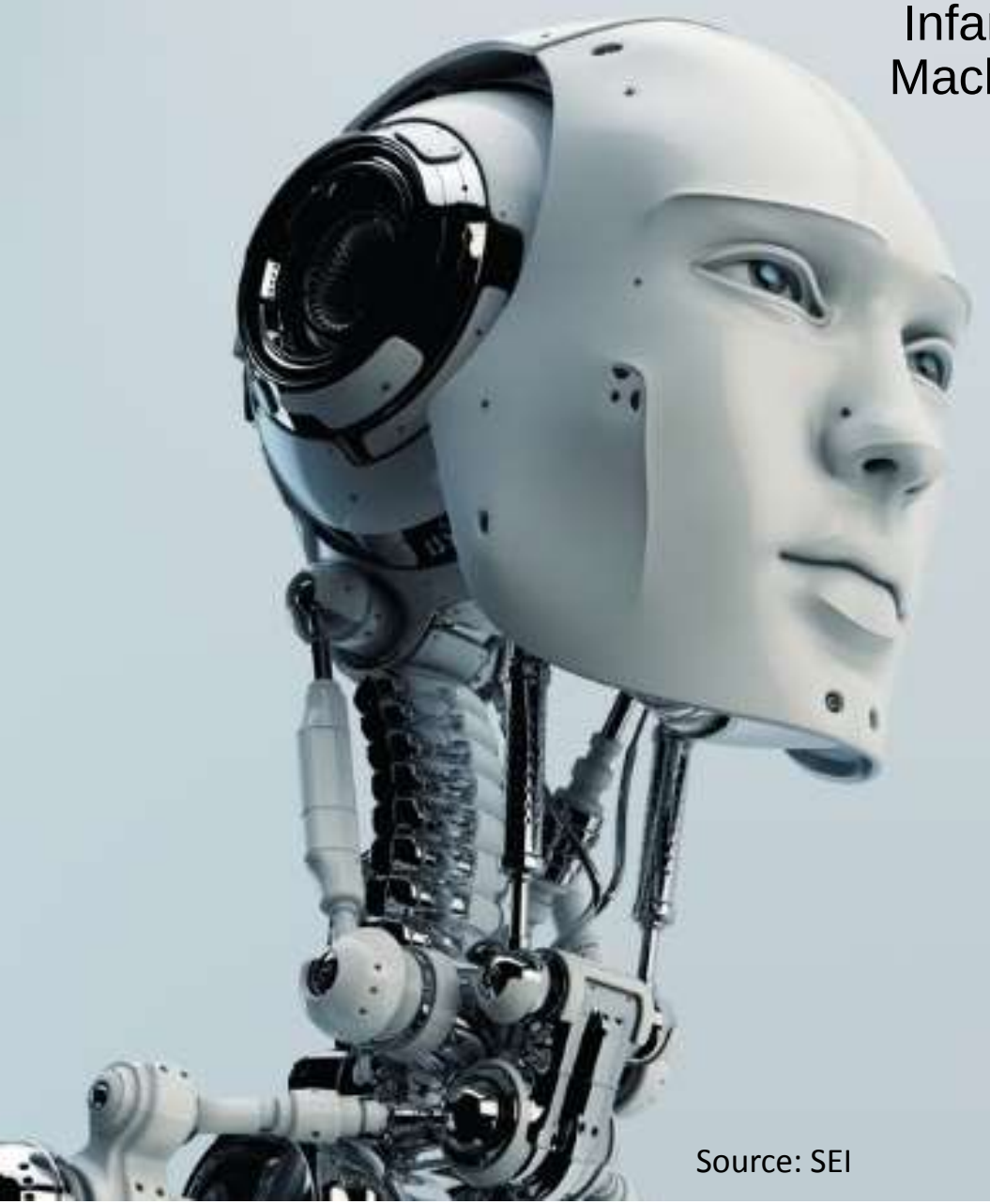
Assessing Future Workforce Needs in Terms of Software Competencies and a Software Career Field

	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 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

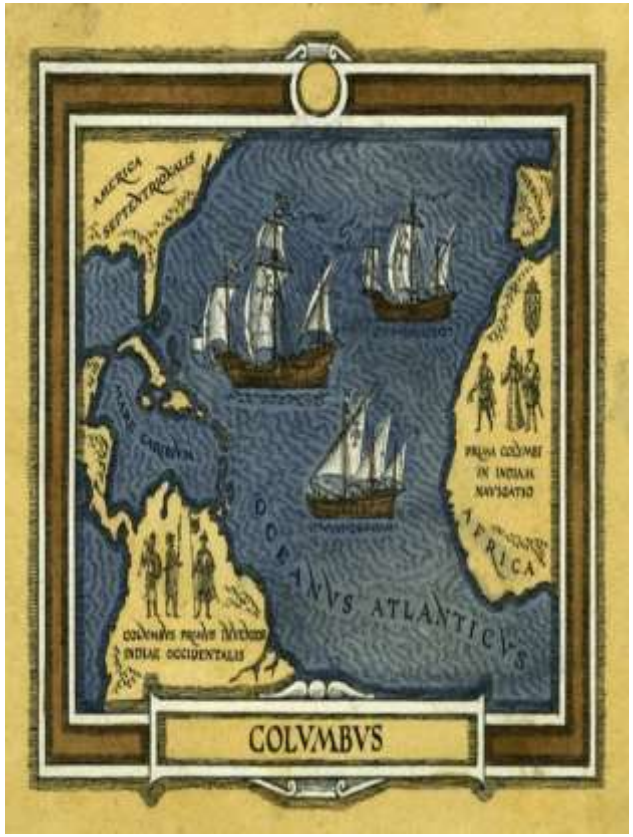
Related to need for explainability and predictability

Source: SEI

7. Increasingly, Software Assurance is a Moving Target

- **SwA Definition*:** The level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software throughout the lifecycle

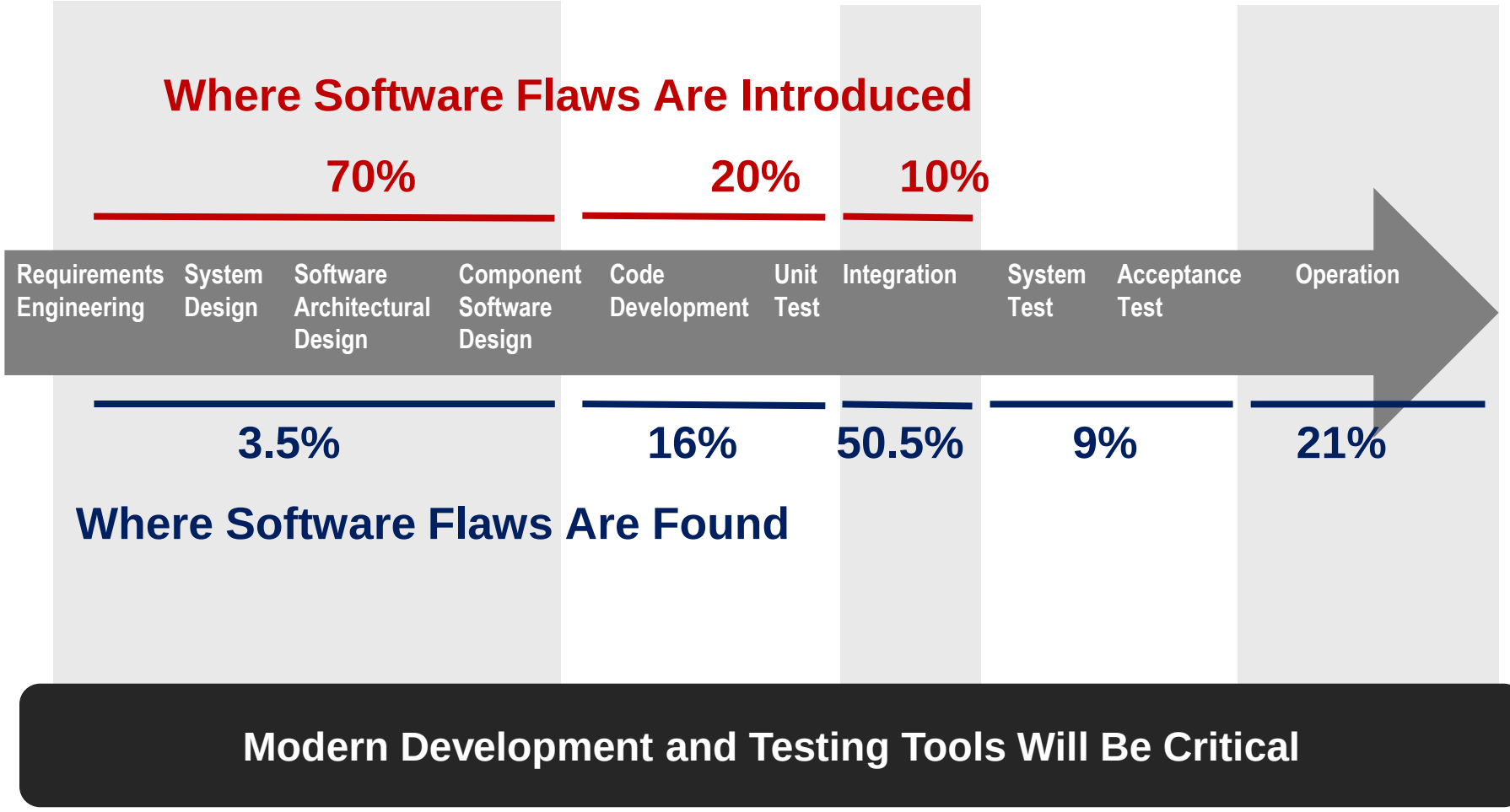
- **Moving Target:** The changing and expanding role that software plays in our society means that the development of software-enabled systems must continue to evolve while we pursue software quality



*Source: DoDI 5200.44, Protection of Mission Critical Functions to Achieve Trusted Systems and Networks (TSN) and 2013 NDAA S933

8. Designing-in Software Assurance Over the Lifecycle

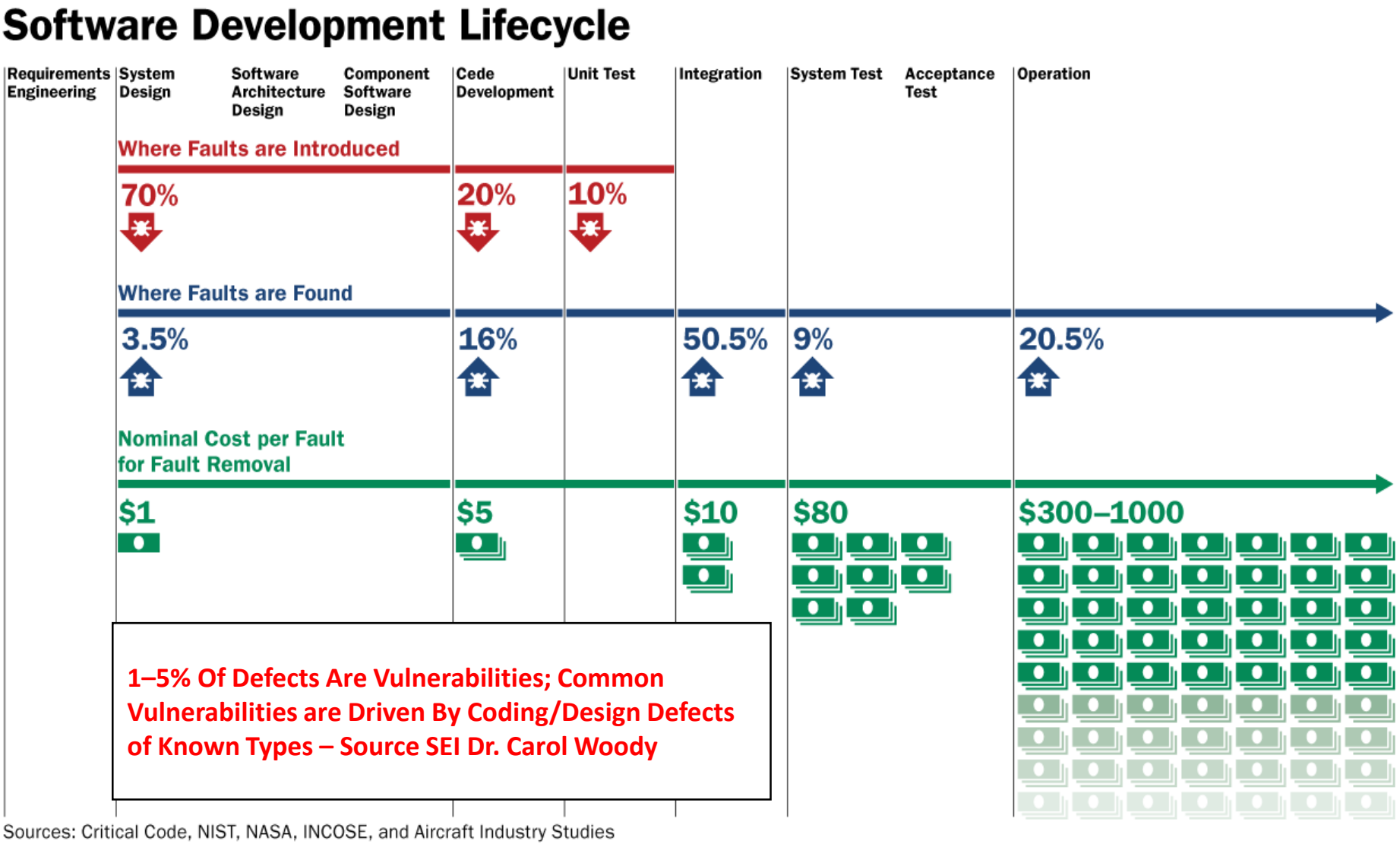
Humans Make Mistakes in Software Engineering Due to Lack of Training, Tools, Experience and Because They Are Fallible



Modern Development and Testing Tools Will Be Critical

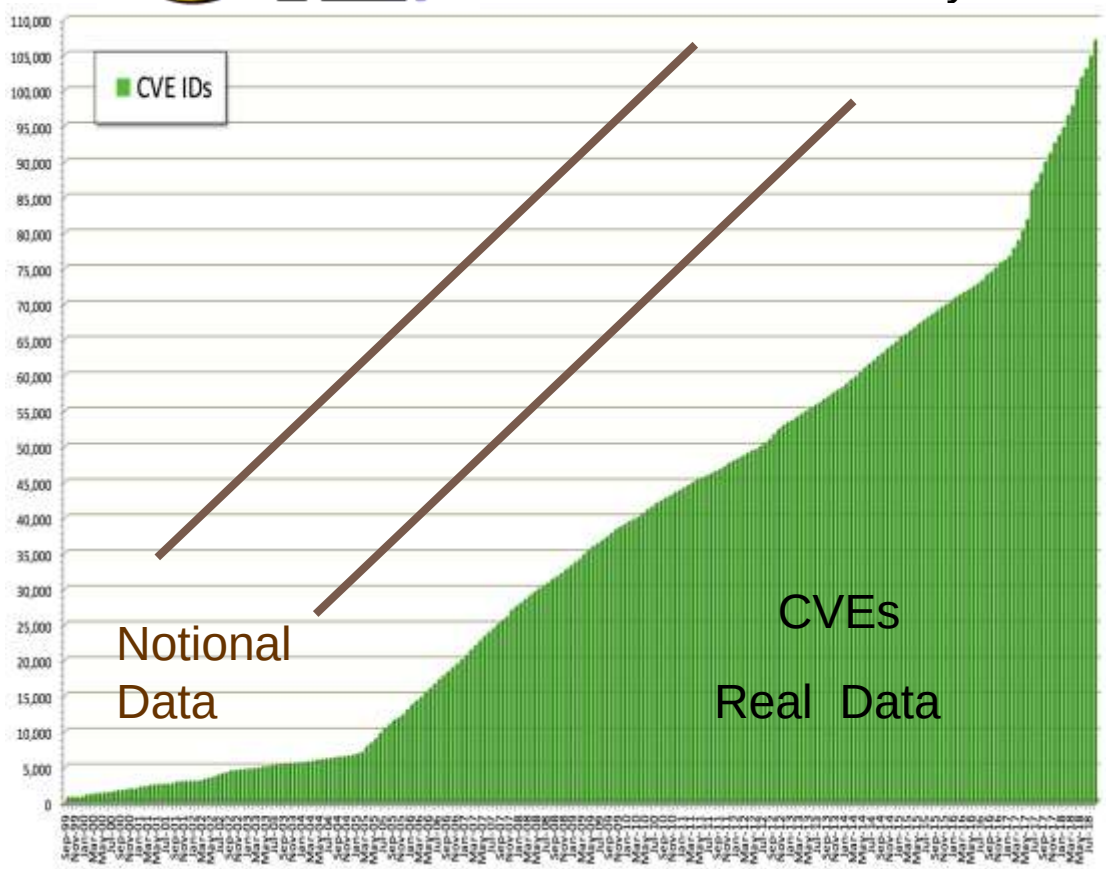
Sources: *Critical Code,* NIST, NASA, INCOSE, and aircraft industry studies

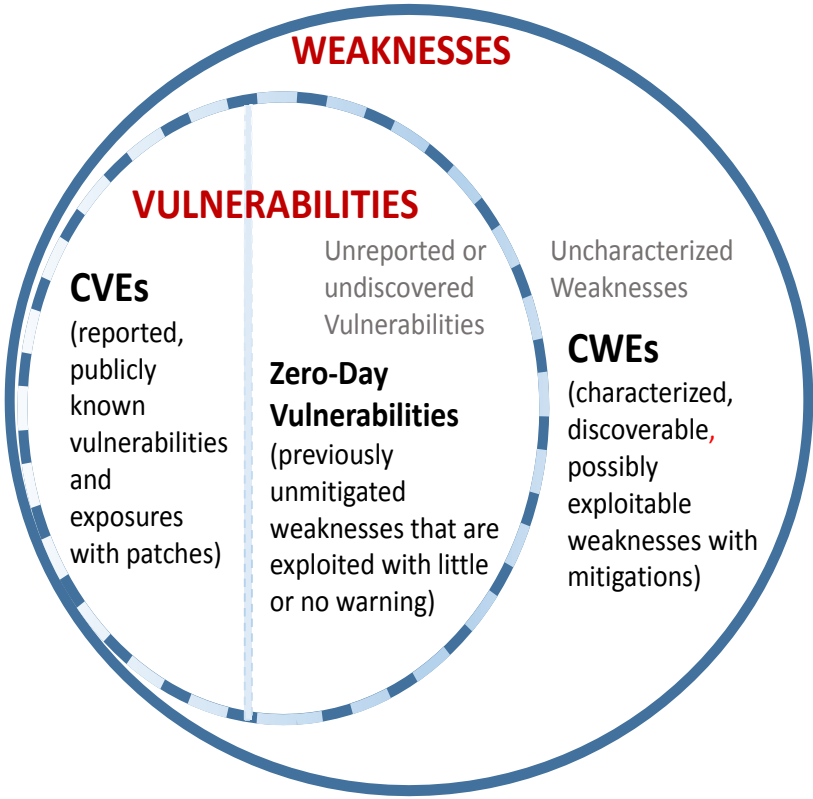
Reducing Technical Debt by Improving Software Development



9. Understanding Attack Patterns, Vulnerabilities, and Weaknesses

CVE 1999 to 2018: Reported Common Vulnerabilities and Exposures (CVE)

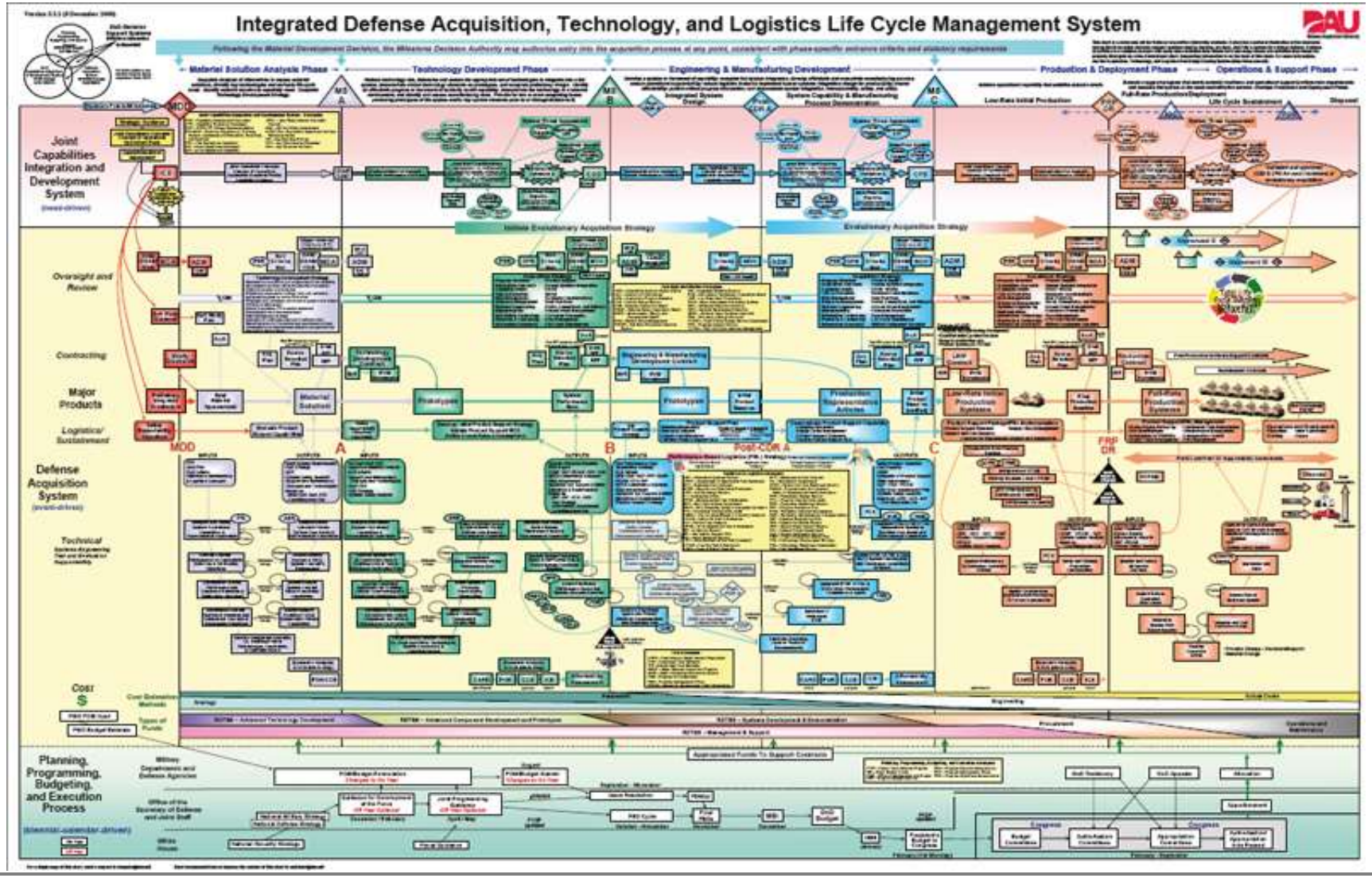




Source: Dr. Robert A. Martin, MITRE Corporation, August 2018

10. Improving Acquisition Processes for Faster Delivery Of Capabilities

An Effective Process For Major Defense Systems – But Not Very Agile



Several On-Going Studies to Streamlined Acquisition

DoD and Congress are Mandating Rapid Iterative Software Development for Defense Acquisition

Workforce Initiatives

- Defining SW Competencies
- Characterizing the Workforce and Common Practices

DIB/DSB Studies and Pilots

- (872) Software Acquisition & Practice SWAP
- (882) Implement Recommendations DSB Software Acquisition of Software
- (873) Pilots for Transitioning to Modern SW Practice

DoD Experiments

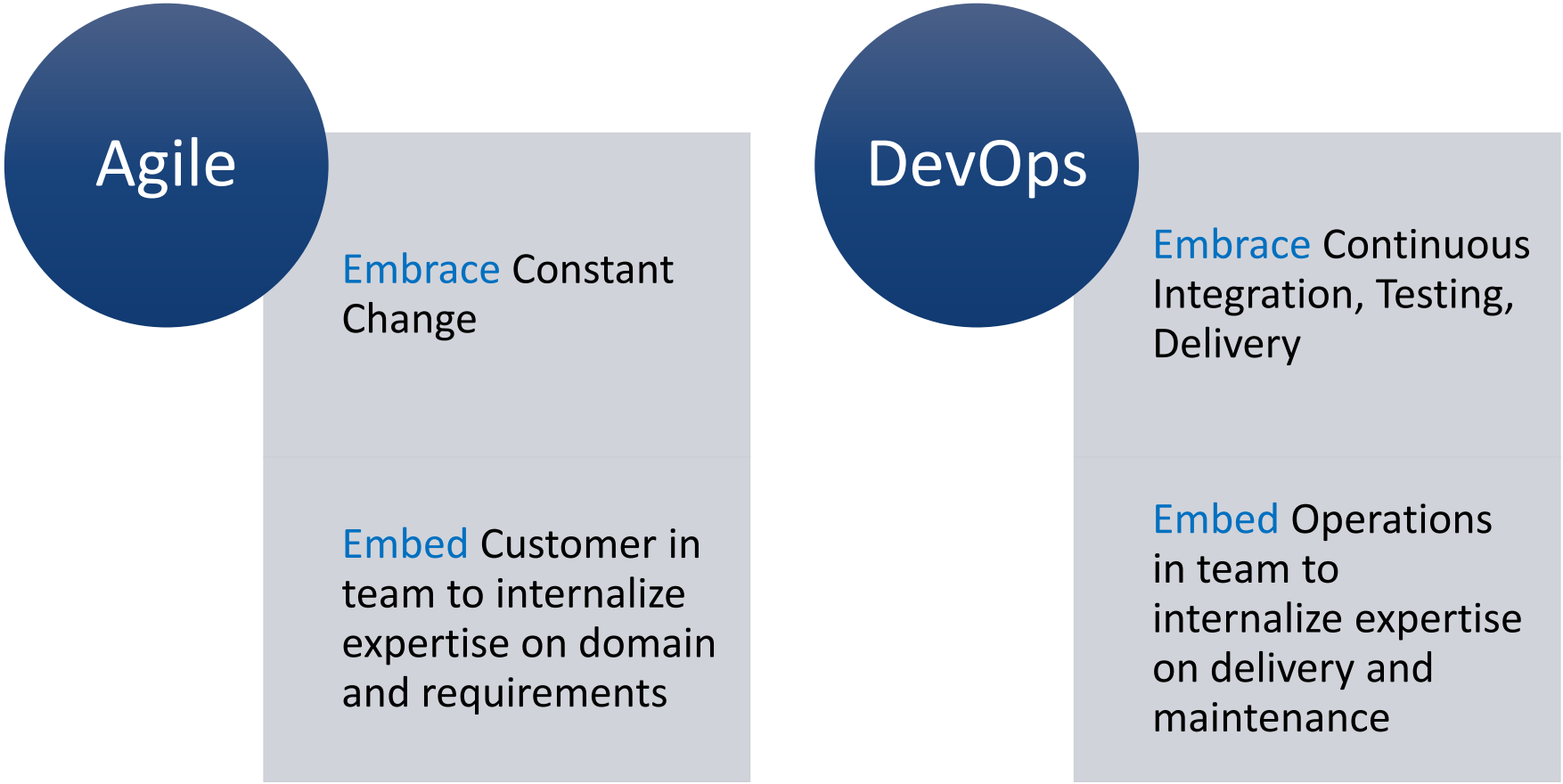
- AOC Pathfinder
- Agile ALIS
- Joint Improvised-threat Defeat Org. (JIDO)

Modernized SW Training

Modernized Acquisition Regulations and Approaches

Modernized DoD Systems

Engineering-In the Benefits of DevOps and Rapid Delivery of Capabilities over the Systems Engineering Life Cycle



Source: SEI Briefing - Integrating the Risk Management Framework (RMF) with DevOps, Tim Chick, March 2018

Source: SEI Agile Collaboration Group, Contact Eileen Wrubel (eow@sei.cmu.edu)

Establishing the Joint Federated Assurance Center (JFAC)

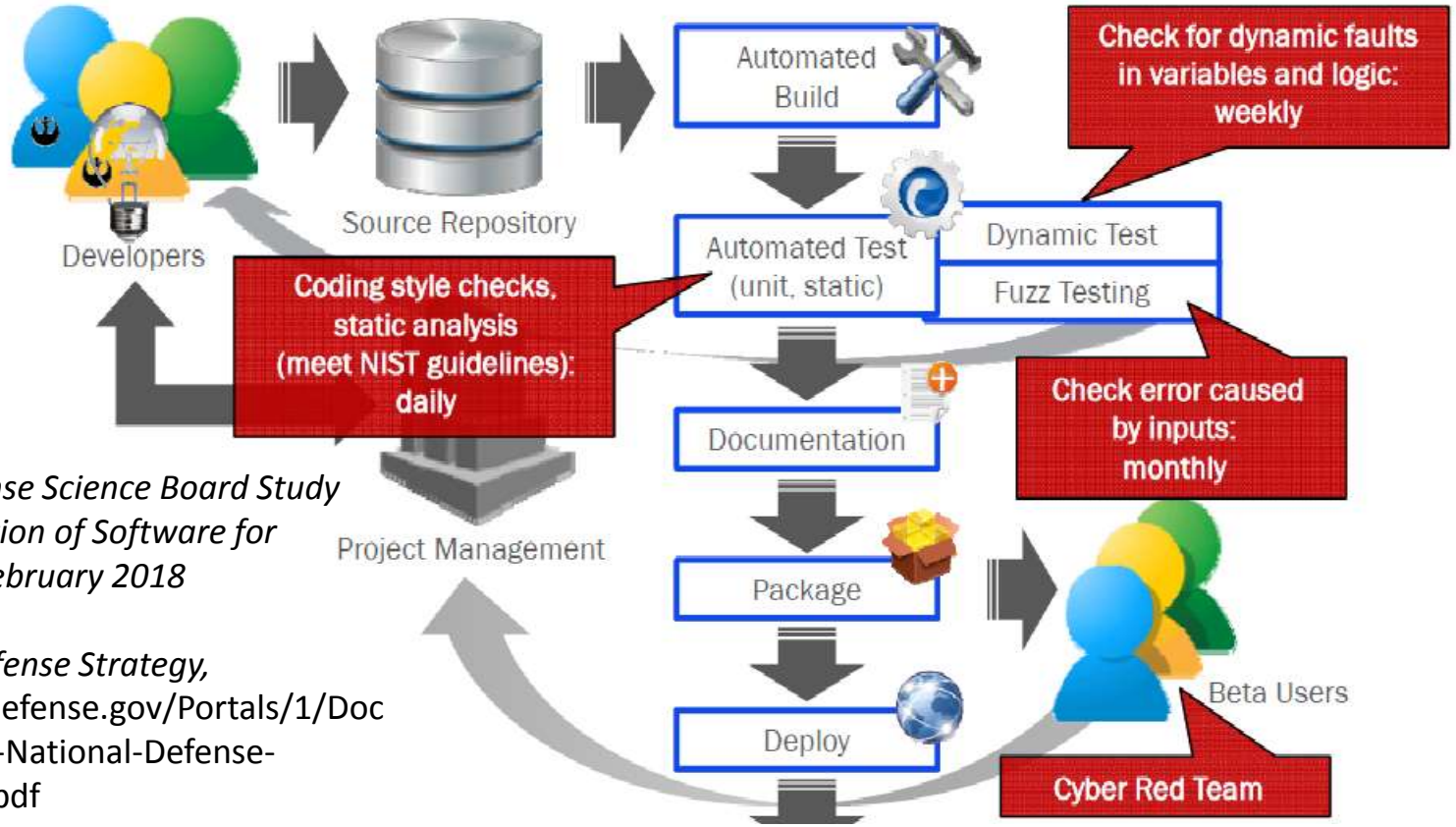
Improving Modern Software Development Assurance Expertise in Program Offices and the Broader Functional Acquisition Workforce.

- Identifies PM software assurance responsibilities critical in defending software-intensive systems
- Presents actions a PM must take to ensure that software assurance is effectively addressed throughout the acquisition lifecycle.

- Helps software developers understand expectations for software assurance.
- Summarizes standards and requirements that affect software assurance decisions and provides pointers to key resources that developers should consult.

Software Factory *

"Deliver Performance at the Speed of Relevance…" **



**Source: DoD Defense Science Board Study Design and Acquisition of Software for Defense Systems, February 2018*

*** The National Defense Strategy, 2018*https://www.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf

Machine Learning*

"Improved Software Engineering and a Focus on Artificial Intelligence will Accelerate DoD's Speed…" **

Architectures And Applications For Advanced, Pervasive And Invisible Analytics

- Predictive Analytics And Prescriptive Analytics — More About Predicting The Future
  - Ensemble Learning And Deep Learning

Architectures, Assurance And Applications For Smart Machines — Integration Of Machine Learning And Autonomy

- Deep Analytics Applied To An Understanding Of Context
- Advanced Algorithms For Understanding The Environment, Learning, And Acting Autonomously

Assurance is a Learned Behavior – Behavior Drives Attitude Change

*Source: SEI - A Quick Look at Emerging Technologies for Software-Reliant Systems, Dr. Grace Lewis, Nov 2015
** Source: Statement of Dr. Eric Schmidt, House Armed Services Committee, April 17, 2018
https://docs.house.gov/meetings/AS/AS00/20180417/108132/HHRG-115-AS00-Wstate-SchmidtE-20180417.pdf

So Where Does This Lead Us?

- A more robust software assurance approach is needed
  - Nature of emerging threats leaves us with a very short (or no) time to prepare
- Decision makers need insight and understanding on how to achieve software assurance
- As software-dominated systems become larger in scope/complexity, making better decisions will become more important
  - Critical to shift from only asking "*what happened*?"
  - To seeking insight via effective measures on asking "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 career field that includes software assurance core competencies
- There are no shortcuts on this path – JFAC* is a definite enabler

* Joint Federated Assurance Center (JFAC)

Final Thought: Advanced Software with Operational Participation

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

Source: SEI

…or the Borg

Source: SEI

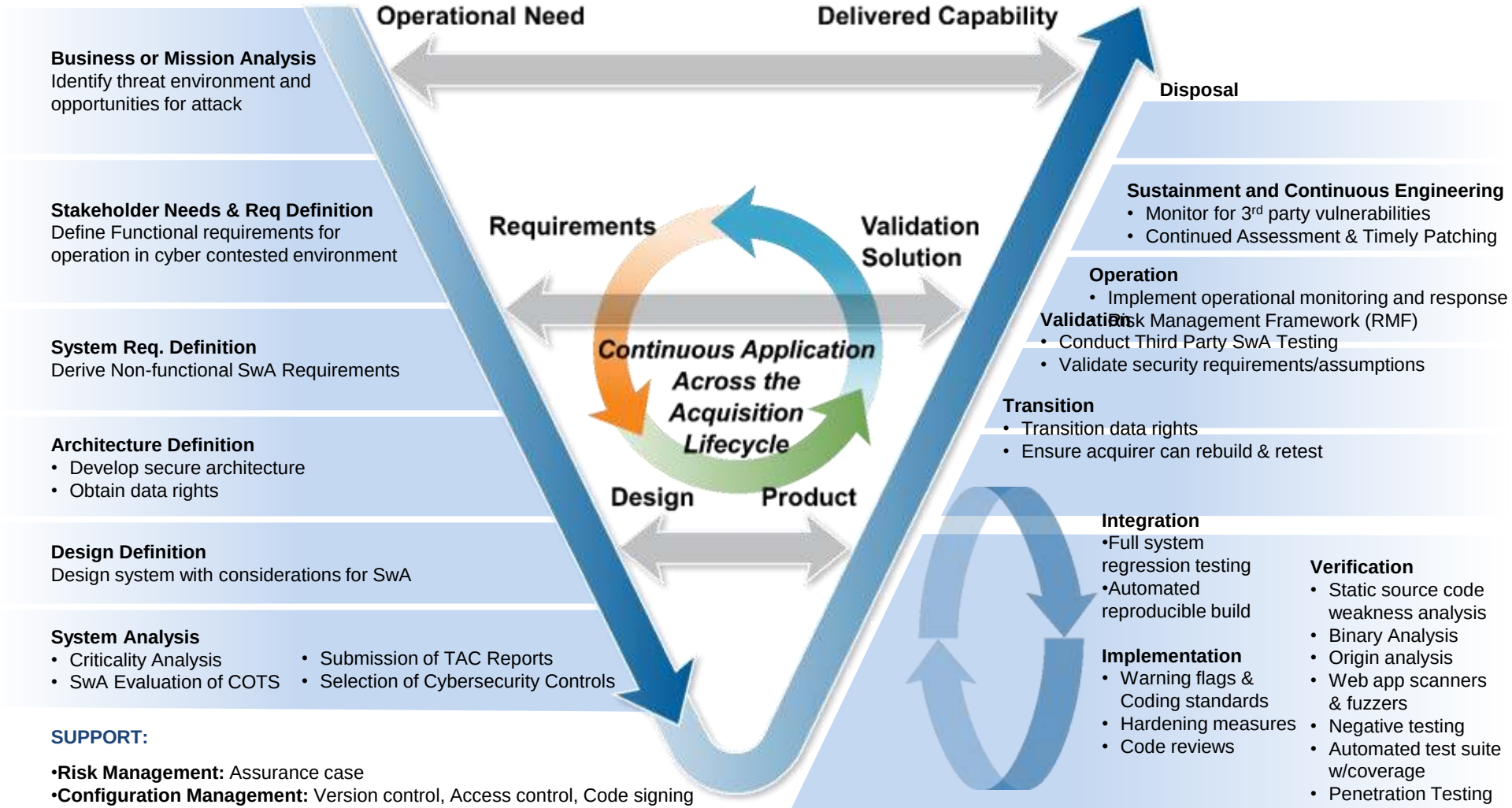
Contact Information

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

Systems and Software Engineering Across the Lifecycle

Amount of Effort & Software Development Depends on SDLC Phase's Purpose



Business or Mission Analysis
Identify threat environment and opportunities for attack

Stakeholder Needs & Req Definition
Define Functional requirements for operation in cyber contested environment

System Req. Definition
Derive Non-functional SwA Requirements

Architecture Definition
- Develop secure architecture
- Obtain data rights

Design Definition
Design system with considerations for SwA

System Analysis
- Criticality Analysis
- SwA Evaluation of COTS
- Submission of TAC Reports
- Selection of Cybersecurity Controls

SUPPORT:

- **Risk Management:** Assurance case
- **Configuration Management:** Version control, Access control, Code signing
- **Measures & Metrics**

Disposal

Sustainment and Continuous Engineering
- Monitor for 3rd party vulnerabilities
- Continued Assessment & Timely Patching

Operation
- Implement operational monitoring and response Risk Management Framework (RMF)

Validation
- Conduct Third Party SwA Testing
- Validate security requirements/assumptions

Transition
- Transition data rights
- Ensure acquirer can rebuild & retest

Integration
- Full system regression testing
- Automated reproducible build

Implementation
- Warning flags & Coding standards
- Hardening measures
- Code reviews

Verification
- Static source code weakness analysis
- Binary Analysis
- Origin analysis
- Web app scanners & fuzzers
- Negative testing
- Automated test suite w/coverage
- Penetration Testing

NOTE: *Implementation, Integration & Verification are often performed continuously & simultaneously with the aid of IDEs & other tools.*

NOTE: *Lifecycle processes typically occur simultaneously, **not** in sequence; see ISO/IEC 15288 & 12207*

Source: Bradley Lanford, Egility Corporation and Dr. David Wheeler, IDA

Right Conditions for Agile

CONDITIONS	FAVORABLE	UNFAVORABLE
Market Environment	Customer preferences and solution options change frequently.	Market conditions are stable and predictable.
Customer Involvement	Close collaboration and rapid feedback are feasible. Customers know better what they want as the process progresses.	Requirements are clear at the outset and will remain stable. Customers are unavailable for constant collaboration.
Innovation Type	Problems are complex, solutions are unknown, and the scope isn't clearly defined. Product specifications may change. Creative breakthroughs and time to market are important. Cross-functional collaboration is vital.	Similar work has been done before, and innovators believe the solutions are clear. Detailed specifications and work plans can be forecast with confidence and should be adhered to. Problems can be solved sequentially in functional silos.
Modularity of Work	Incremental developments have value, and customers can use them. Work can be broken into parts and conducted in rapid, iterative cycles. Late changes are manageable.	Customers cannot start testing parts of the product until everything is complete. Late changes are expensive or impossible.
Impact of Interim Mistakes	They provide valuable learning.	They may be catastrophic.

Source: DoD Defense Science Board Study Design and Acquisition of Software for Defense Systems, February 2018, Darrell K. Rigby, Jeff Sutherland, and Hirotaka Takeuchi, "Embracing Agile," *Harvard Business Review* (May 2016), https://hbr.org/2016/05/embracing-agile.