Struggles at the Frontiers: Achieving Software Assurance for Software-Reliant Systems

Dr. Kenneth E. Nidiffer

The 27th Annual IEEE Software Technology Conference

Long Beach, California, USA
12 October - 15 October 2015

Meeting Real World Opportunities and Challenges through Software and Systems Technology

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Software is the foundation of the cyber environment, enabling explorations into new frontiers

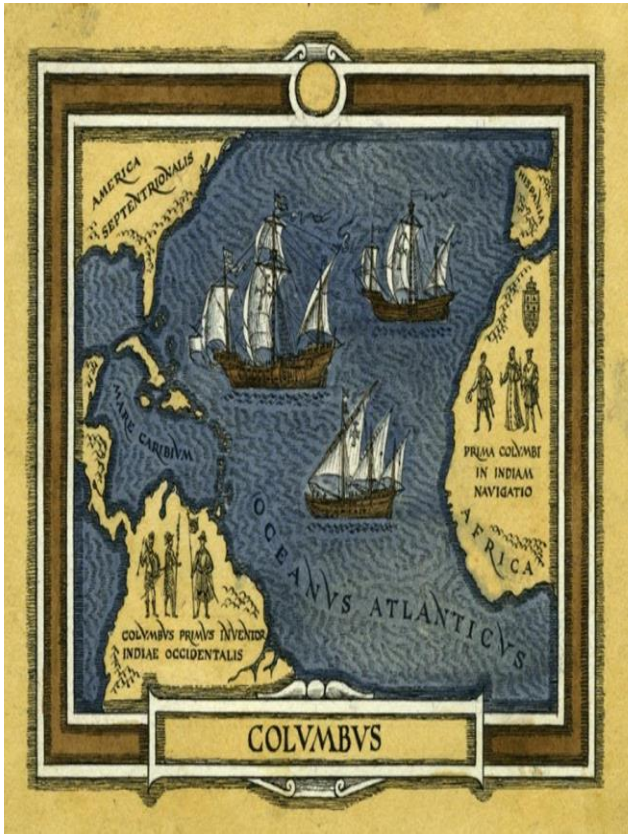
… Software Quality is a property/attribute of a system – must be engineered/designed-in!

Tianhe-2 Supercomputer

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

Content

- Context: Software Quality Is a Constant Purpose and Software Is a Moving Target
- Perspectives: Struggles in the Persistent Pursuit of Software Quality Assurance
- Future: Software Is the Underpinning of the Cyber Environment, Enabling Explorations into New Frontiers


AMERICA SEPTENTRIONALIS
HISPANIA
MARE CARIBIVM
PRIMA COLVMBI IN INDIAM NAVIGATIO
AFRICA
OCEANVS ATLANTICVS
COLVMBVS PRIMVS INVENTOR INDIAE OCCIDENTALIS
COLVMBVS

Source: SEI

Context: Software Quality Is a Constant Purpose and Software Is a Moving Target

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

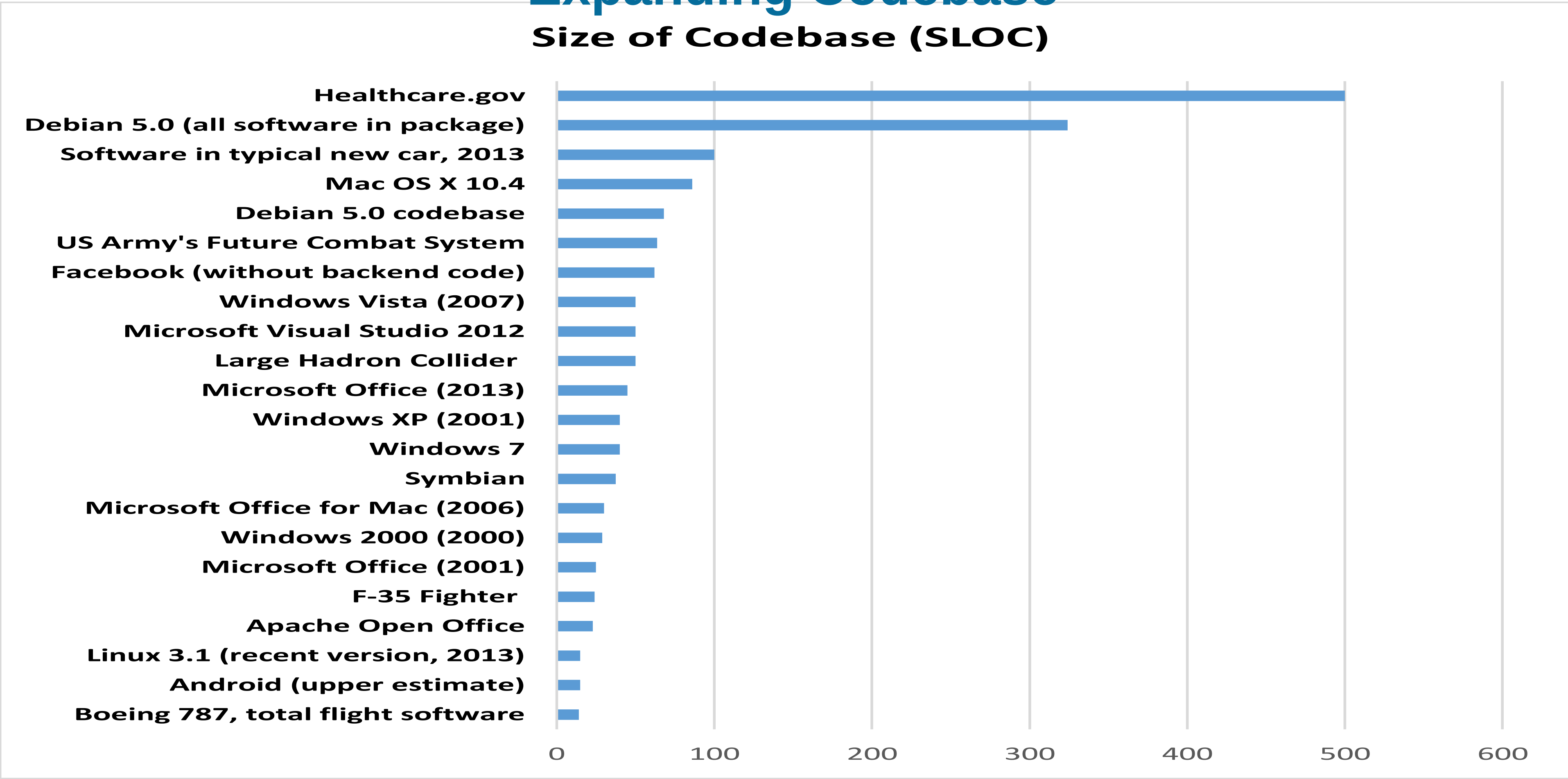
Context: Software Quality Is a Constant Purpose and Software Is a Moving Target

- Constant Purpose
  - Software Assurance: To provide the level of confidence that software functions as intended (and no more) 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 roll of software plays in cyberspace means that software engineering must continue to evolve in the ongoing pursuit of software quality.

*** NDAA 2013 Section 933**

Context: Software Is a Moving Target

Expanding Codebase

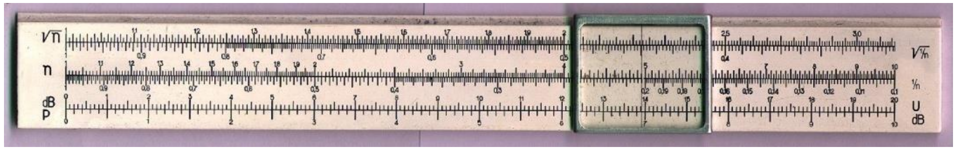


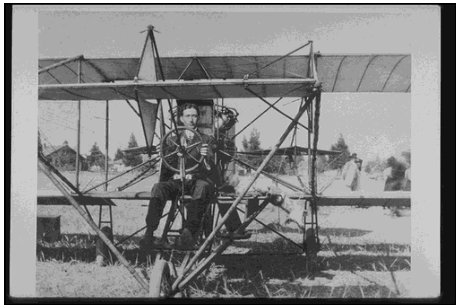
Millions of Source Lines of Code

Source: David McCandless – Software is Beautiful, 12 August 2015 Web Retrieval

Context: Software Is a Moving Target – Aircraft Growth of Software Over Time

In The Beginning





1960s

F-4A
1,000 LOC

1970s

F-15A
50,000 LOC

1980s

F-16C
300K LOC

1990s

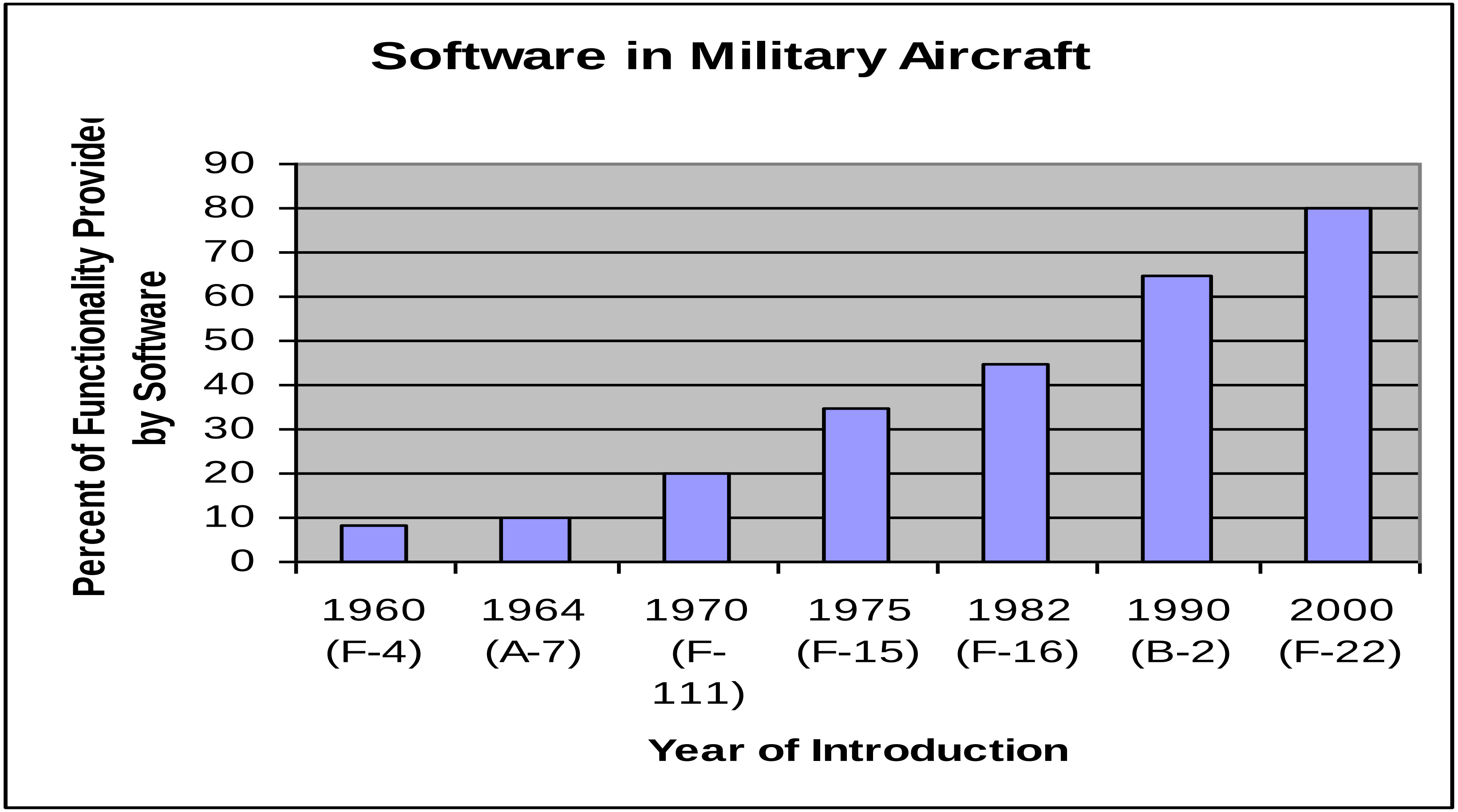
F-22
1.7M LOC

2000+

F-35
>6M LOC

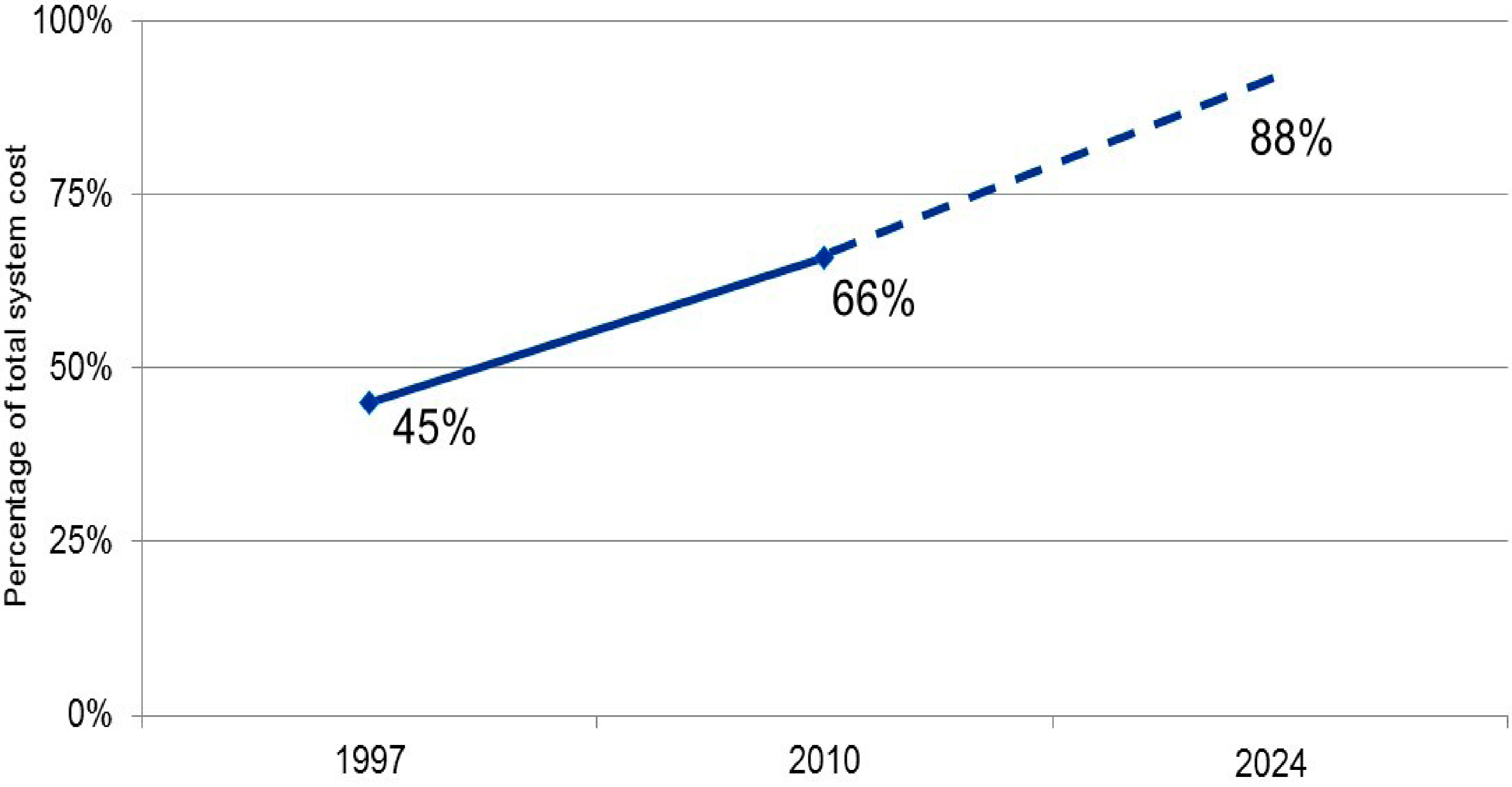
Permission provided for use by author by Lockheed Martin Corporation

Context: Software Is a Moving Target - Percent of Functionality Provided by Software



Source: NASA Planetary Spacecraft Fault Management Workshop, April 14-16, 2008, New Orleans

Context: Software Is a Moving Target - Aircraft Software Development and Rework Cost



Reference: 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.

Context: Software Is a Moving Target – Importance of Software Engineering

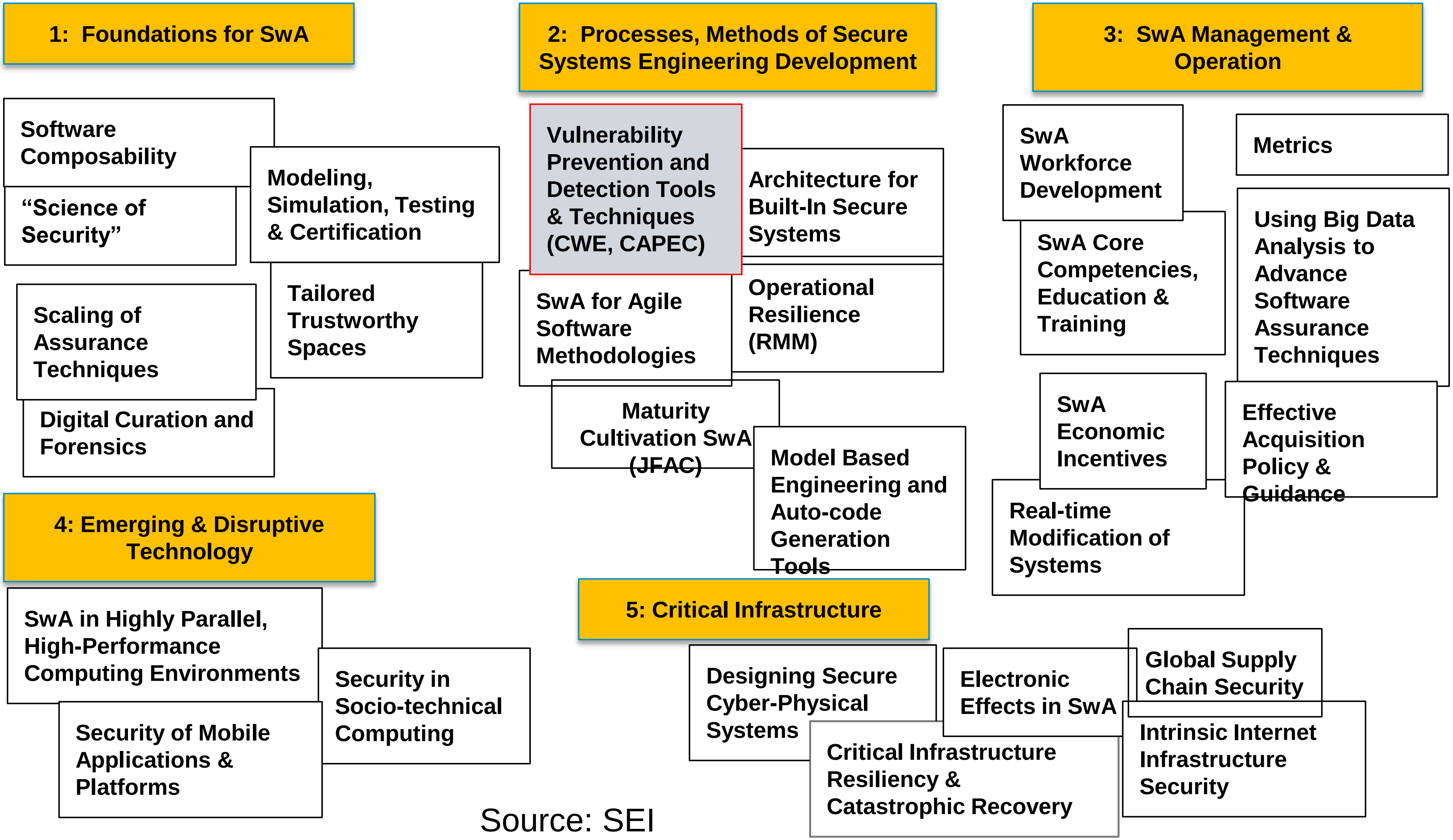
Argument: Need to advance the state of the practice of **software engineering** to improve the **quality** of systems that depend on software

- **Quality is a property/attribute of a system – must be designed-in!**

Software engineering requires analysis and synthesis

- **Analysis: decompose a large problem into smaller, understandable pieces**
  - **abstraction is the key**
- **Synthesis: build (compose) a software from smaller building blocks**
  - **composition is challenging**

Context: Software Is a Moving Target – Importance of Software Engineering



1: Foundations for SwA

- Software Composability
- “Science of Security”
- Modeling, Simulation, Testing & Certification
- Tailored Trustworthy Spaces
- Scaling of Assurance Techniques
- Digital Curation and Forensics

2: Processes, Methods of Secure Systems Engineering Development

- Vulnerability Prevention and Detection Tools & Techniques (CWE, CAPEC)
- Architecture for Built-In Secure Systems
- SwA for Agile Software Methodologies
- Operational Resilience (RMM)
- Maturity Cultivation SwA (JFAC)
- Model Based Engineering and Auto-code Generation Tools

3: SwA Management & Operation

- SwA Workforce Development
- Metrics
- SwA Core Competencies, Education & Training
- Using Big Data Analysis to Advance Software Assurance Techniques
- SwA Economic Incentives
- Effective Acquisition Policy & Guidance
- Real-time Modification of Systems

4: Emerging & Disruptive Technology

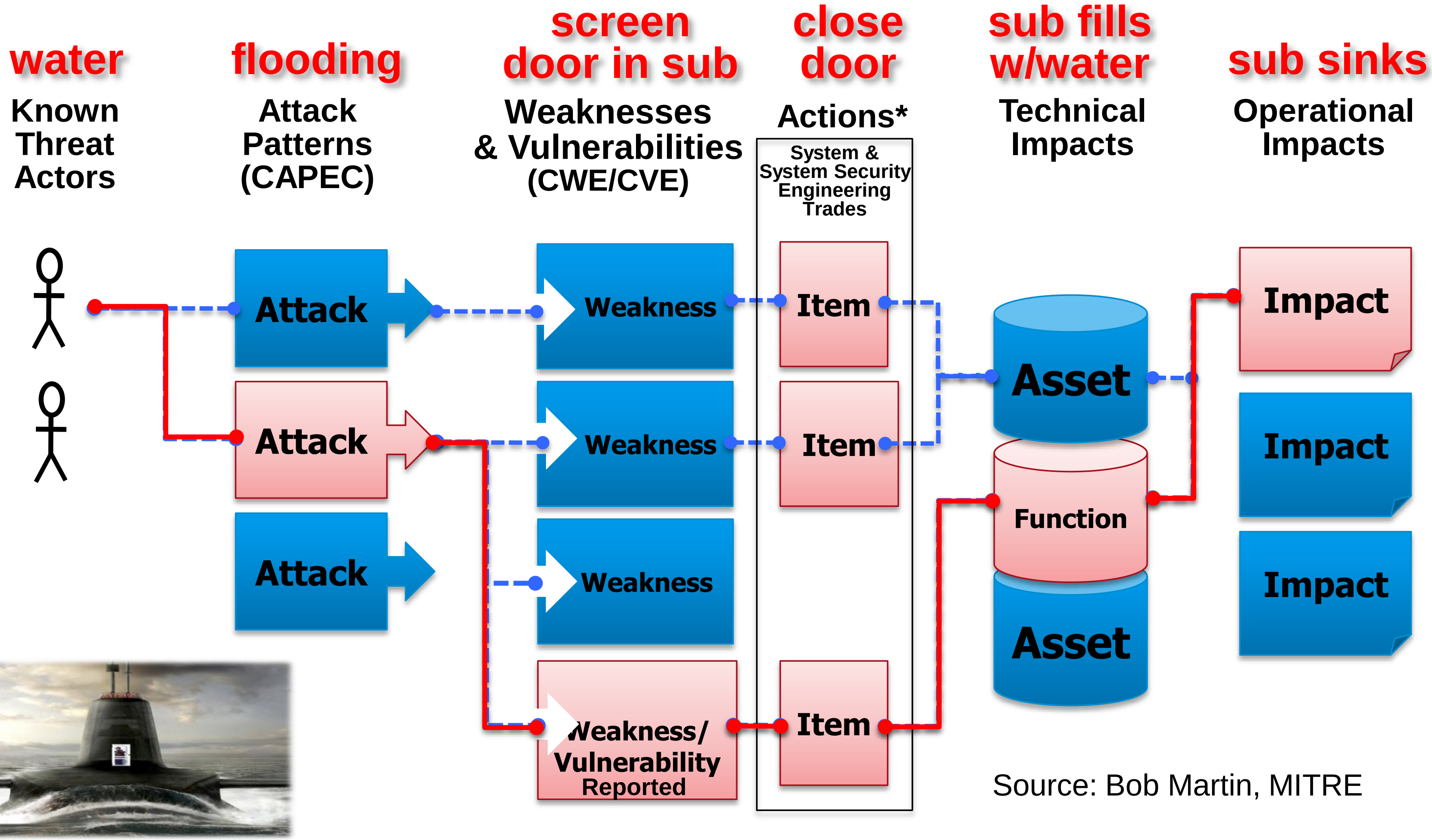
- SwA in Highly Parallel, High-Performance Computing Environments
- Security in Socio-technical Computing
- Security of Mobile Applications & Platforms

5: Critical Infrastructure

- Designing Secure Cyber-Physical Systems
- Electronic Effects in SwA
- Global Supply Chain Security
- Critical Infrastructure Resiliency & Catastrophic Recovery
- Intrinsic Internet Infrastructure Security

Source: SEI

Context: Software Is a Moving Target – Importance of Software Engineering

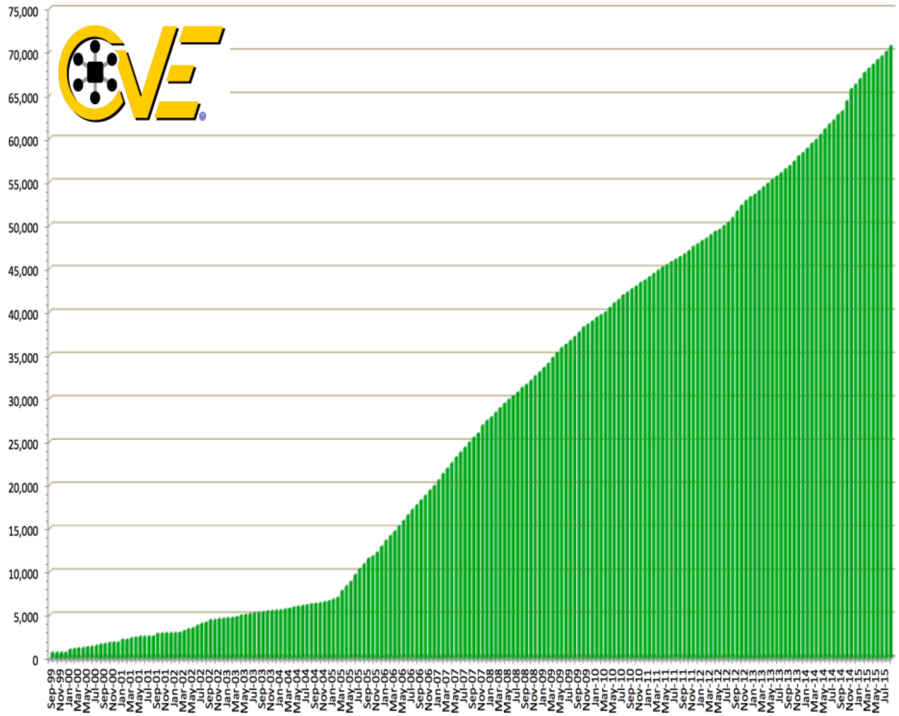


Source: Bob Martin, MITRE

* "Actions" include: architecture choices; design choices; added security functions, activities & processes; physical decomposition choices; static & dynamic code assessments; design reviews; dynamic testing; and pen testing

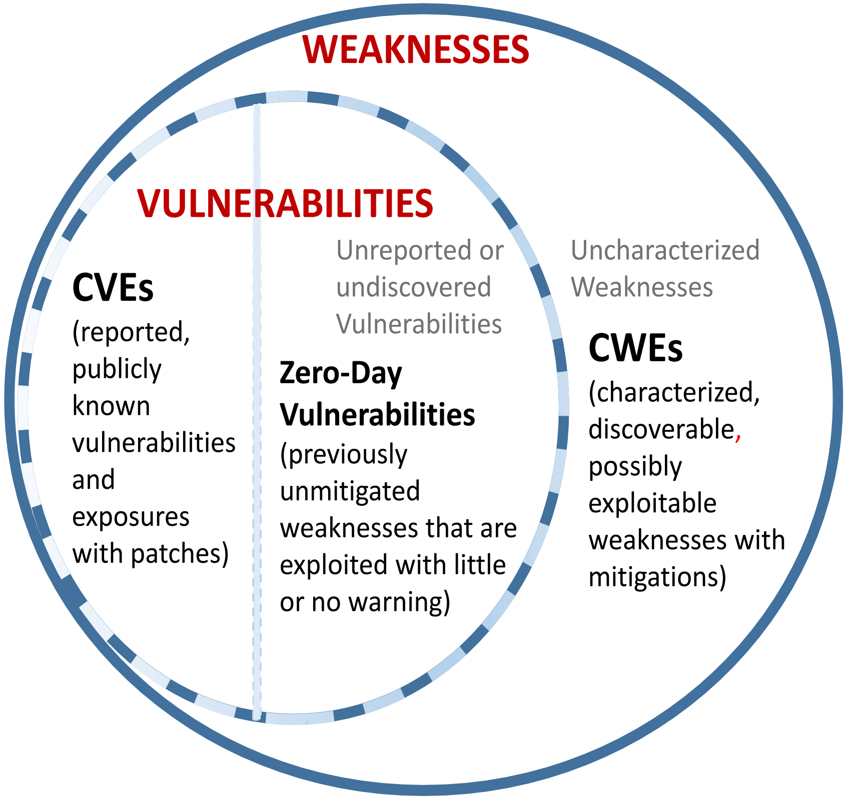
Context: Software Is a Moving Target – Reported Common Vulnerabilities and Exposures (CVE)

CVE 1999 to 2015



Source: Dr. Robert A. Martin, MITRE Corporation, August 2015

*



WEAKNESSES

VULNERABILITIES

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

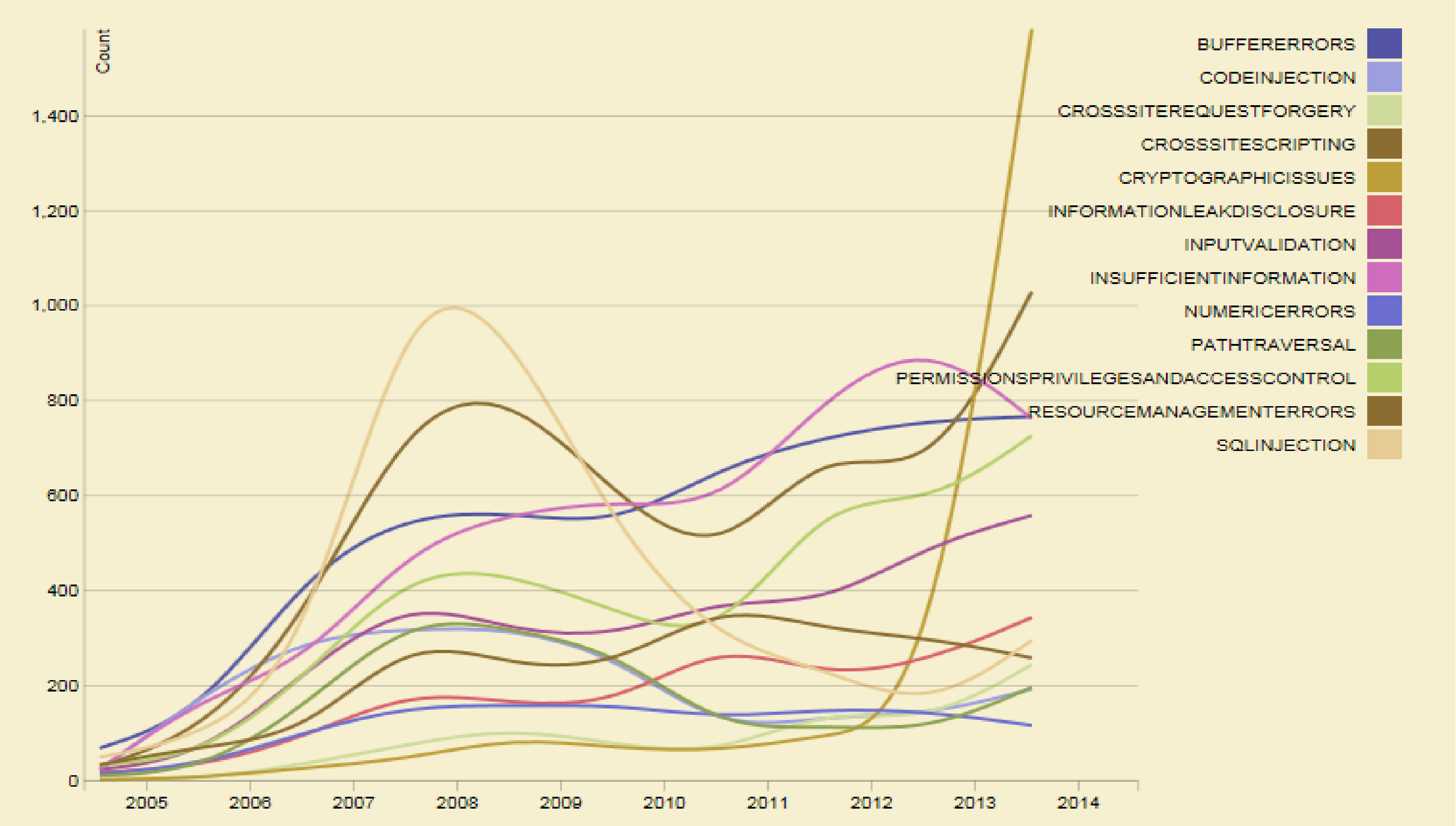
Unreported or undiscovered Vulnerabilities

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

Uncharacterized Weaknesses

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

Context: Software Is a Moving Target - Common Weakness Enumeration (CWE*)



Source: NIST, National Vulnerability Database, 12 August 2015 web retrieval

* CWE provides a unified, measurable set of software weaknesses

Perspectives: Struggles in Software Engineering and the Persistent Pursuit of Software Quality

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

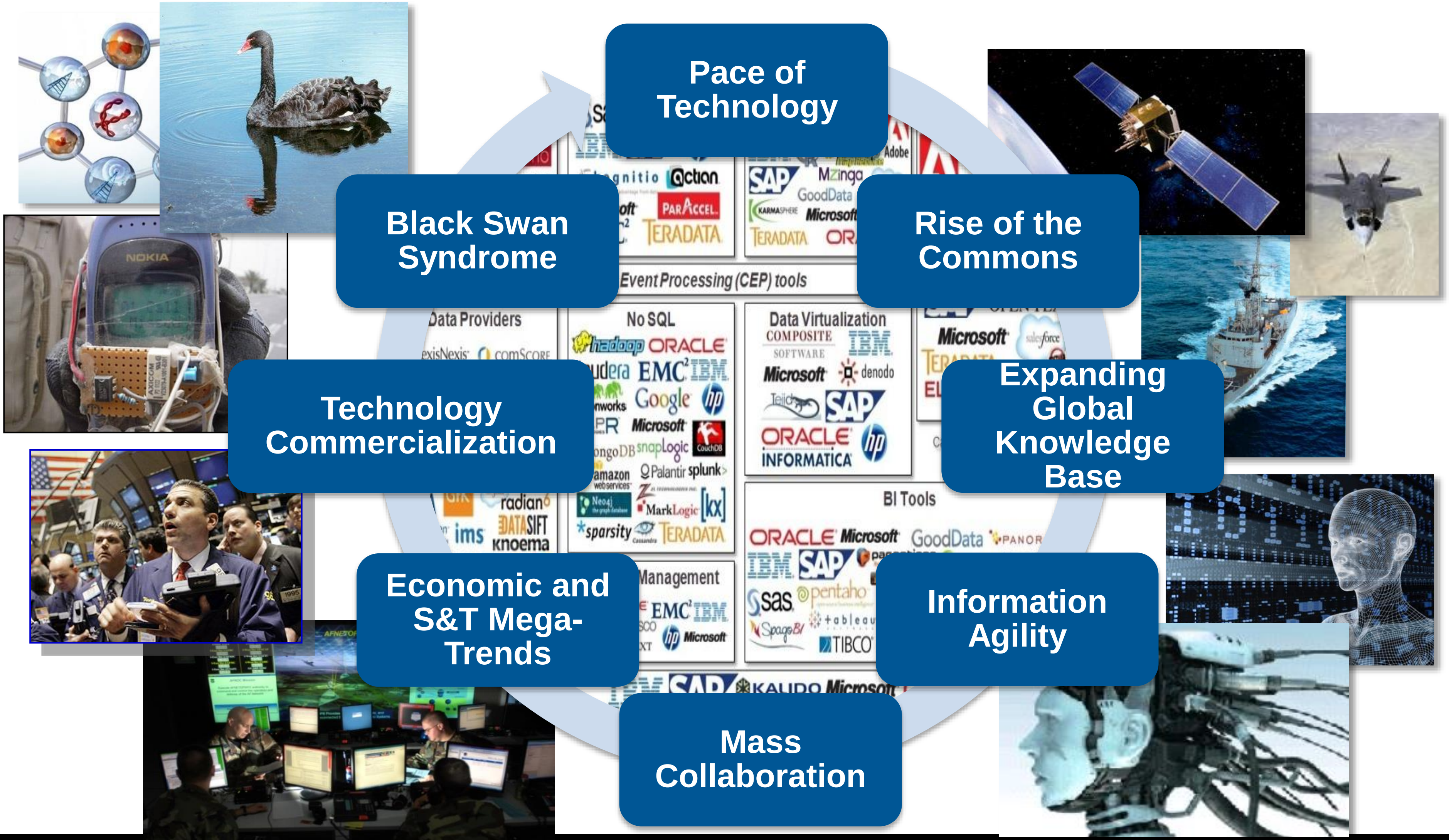
Software Engineering Institute | Carnegie Mellon University

Struggles in Software Engineering and the Persistent Pursuit of Software Quality - Some Things We Know About Software

- Ubiquitous
- Codebase is increasing
- Vulnerabilities (Defects, Flaws) increasing
- Represents increasingly more system functionality and cost
- Research needed to address significant challenges
- Software-reliant systems are becoming more complex and intertwined
- Nationally and globally important
- Need to manage software systems better
- Software quality must be engineered/designed in

Pursuit of software quality is increasingly more important!

Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Globalization



Pace of Technology

Rise of the Commons

Expanding Global Knowledge Base

Information Agility

Mass Collaboration

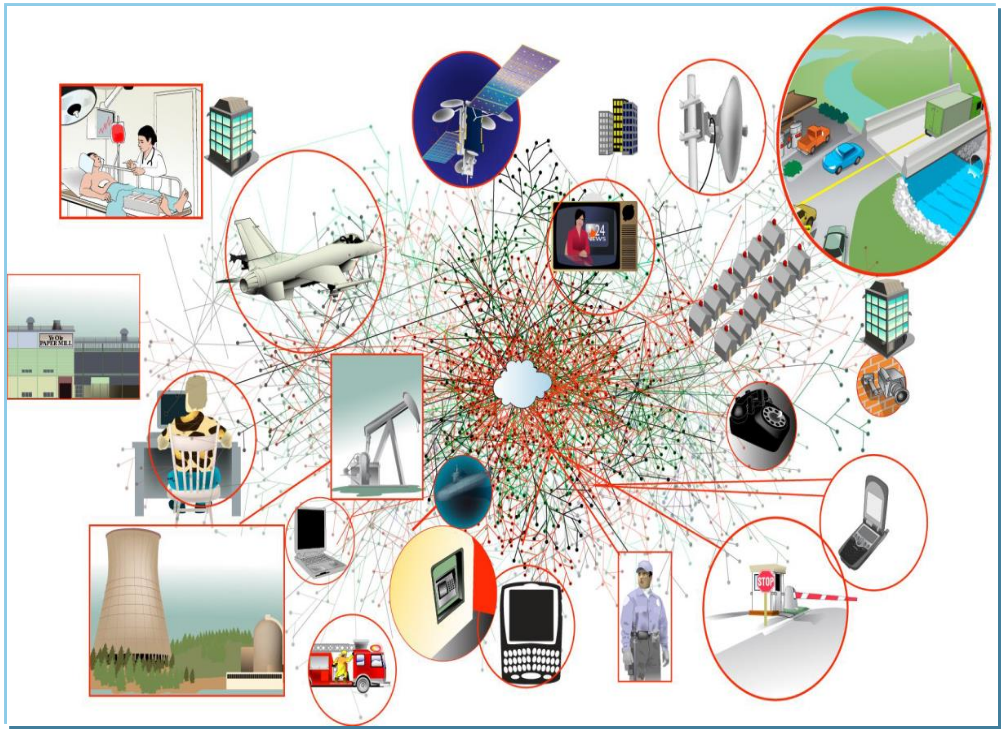
Economic and S&T Mega-Trends

Technology Commercialization

Black Swan Syndrome

Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Critical Infrastructure

- More Efficient and Agile Development of Software-Reliant Capabilities
- Improved Globalization/Supply-Chain Management
- Reduced Risk Due to Software Vulnerabilities
- More Resilient Cyber Systems and Networks
- Reduced Sustainment Cost
- Improved Workforce Competencies



Transportation Architectures

+

Healthcare Architectures

+

Banking & Financial Architectures

+

Energy & Utilities Architectures

+

Communications Architectures

...

Source: SEI

Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Essential Difficulties

According to Fred Brooks,* software projects are difficult because of accidental and essential difficulties

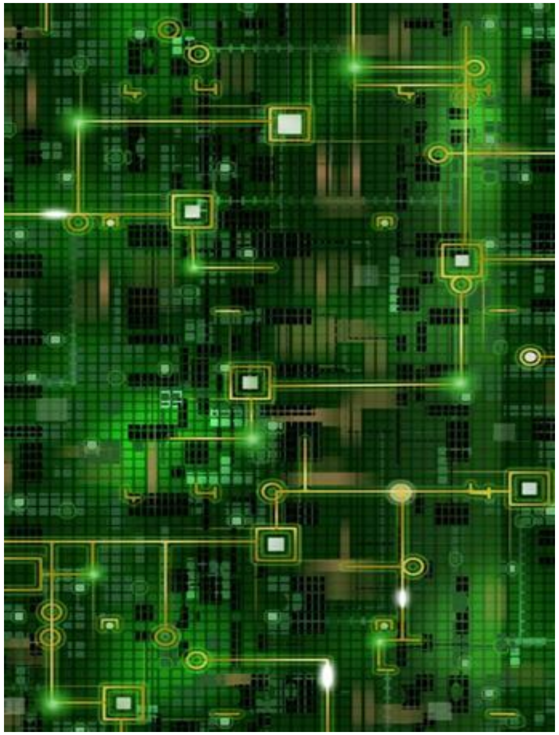
- Accidental difficulties are caused by the current state of our understanding
  - of methods, tools, and techniques
  - of the underlying technology base
- Essential difficulties are caused by the inherent nature of software
  - invisibility – lack of physical properties
  - complexity – for its size
  - conformity
  - changeability

“the massive dissemination of error-loaded software is frightening”
– Edsger Dijkstra, 1968

* ***The Mythical Man-Month*** **by Fred Brooks, Addison Wesley, 1995**

Struggles in Software Engineering and the Persistent Pursuit of Software Quality - Complexity

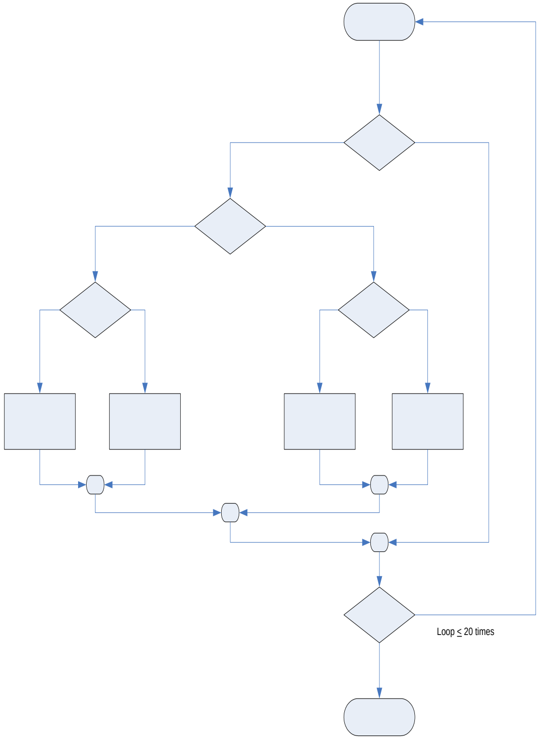
- Due to interaction of components, number of possible states grows much faster than lines of code
- For its size, software is very complex compared to other engineering artifacts
- Hardware is complex, but the laws of physical science usually tell us what to expect for a known input



Source: SEI

Struggles in Software Engineering and the Persistent Pursuit of Software Quality - Changeability

- The flowchart might correspond to a 100 LOC module with a single loop that may be executed no more than 20 times.
- There are approximately 10^{14} possible paths that may be executed!
- For any but the smallest programs, complete path coverage for defect detection is impractical.
- Limited natural governance



Lehman Laws:

1. The Law of Continuing Change – programs must change to be useful

2. The Law of Increasing Complexity – programs that change become more complex

Adapted from Pressman, R.S., *Software Engineering: A Practitioner's Approach, Third Edition*, McGraw Hill, 1992

Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Infancy of Software Engineering

Source: SEI

Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Infancy of 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/ treatment	• Several areas of study: computer science, software/ systems engineering, IT, HCI, social dynamics, AI • All nodes attached to/relying on netted system
R&D and Launch Cycle	10-20 years	10-20 years	Significantly **compressed**; solution time to market needs to happen very quickly

Source: SEI

HCI: Human Computer Interaction; AI: Artificial intelligence

Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Demographics of Workforce Provide Different Views of the Frontiers

Source: SEI

Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Demographics of Workforce Provide Different Views of the Frontiers

- Demographics of workforce are changing, and different views may emerge with multiple generations to consider
- Generation Y professionals are technically savvy and can better leverage IT capabilities for improved efficiencies and productivity; however, they may lack the systems engineering knowledge, skills, and abilities

Silent Generation 1928-1945	Baby Boomers 1946-1964	Generation X 1965-1980	Generation Y/Millenials 1981-2000
Hard worker	Workaholic	Technically advanced	Technically savvy
Respects authority	Questions authority	Prefers informality	Embraces diversity
Work is obligation	Works efficiently	Needs structure and direction	Requires supervision
Formal communicator	Competitive	Direct/immediate communicator	Indirect/virtual communicator
Work/family separation	Little work/life balance	Seeks work/life balance	Demands work/life balance

Sources: SEI, Recommendations for Improving Acquisition Training, May 2010

Achieving Effective Acquisition of Information Technology in the Department of Defense, National Academy of Sciences , 2010

Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Software Is Everywhere with Limited Natural Governance

Source: SEI

Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Software Is Everywhere with Limited Natural Governance

Laws of physics

Laws of software

Challenge of algorithms

Difficulty of distribution and concurrency

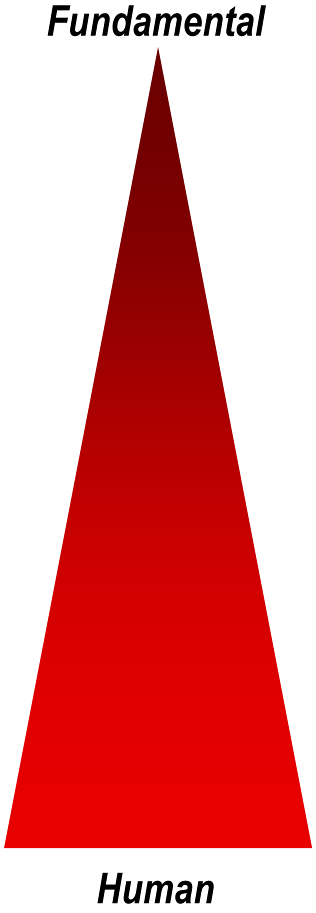
Problems of design

Importance of organization

Impact of economics

Influence of politics

Limits of human imagination



Source: IBM

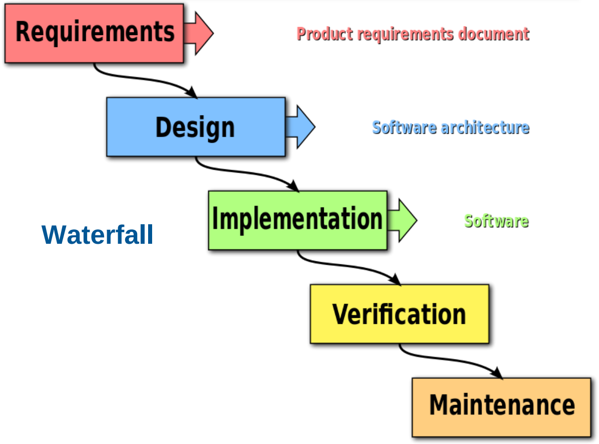
Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Increasing Use of Innovative Processes, Methods and Tools (Accidental Difficulties)

Predictive Models

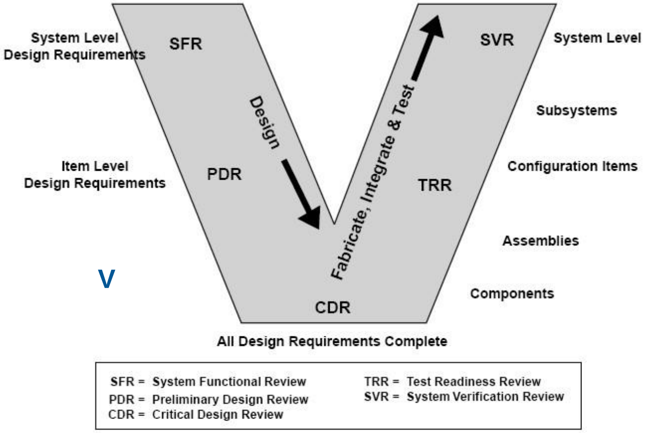
Iterative Models

Adaptive Models

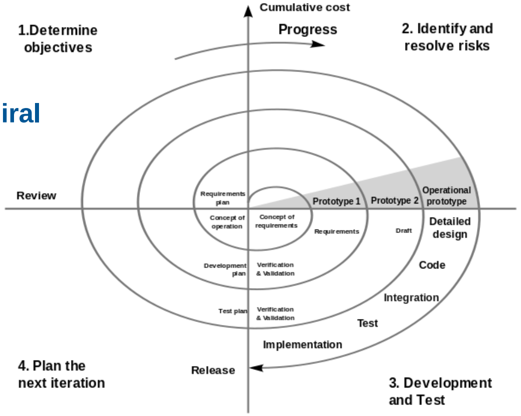
Waterfall



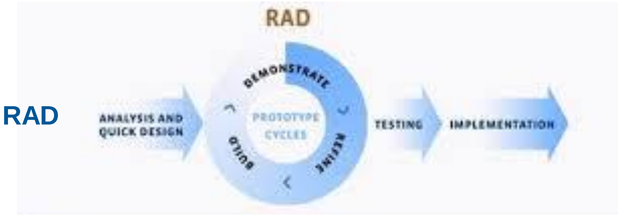
V


System Level
Design Requirements
SFR
SVR
System Level
Design
Fabricate, Integrate & Test
Subsystems
Item Level
Design Requirements
PDR
TRR
Configuration Items
Assemblies
Components
CDR
All Design Requirements Complete
SFR = System Functional Review
PDR = Preliminary Design Review
CDR = Critical Design Review
TRR = Test Readiness Review
SVR = System Verification Review

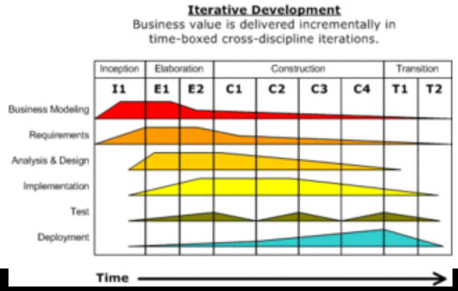
Spiral



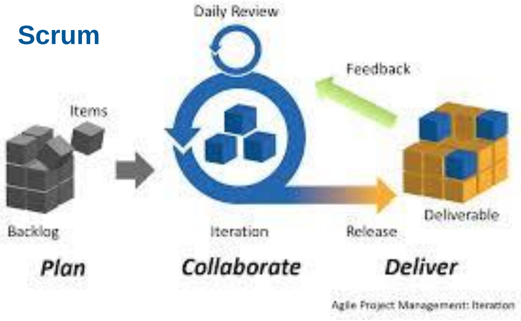
RAD



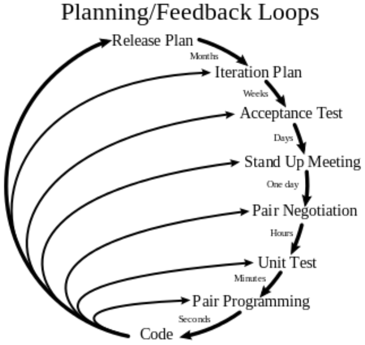
RUP



Scrum

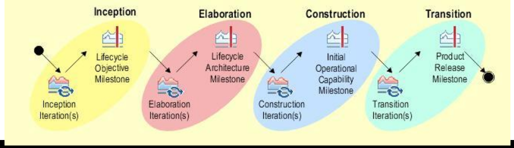


XP



OpenUP

Source: Noblis


Inception
Elaboration
Construction
Transition
Lifecycle Objective Milestone
Lifecycle Architecture Milestone
Initial Operational Capability Milestone
Product Release Milestone
Inception Iteration(s)
Elaboration Iteration(s)
Construction Iteration(s)
Transition Iteration(s)

Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Software Connects Us in Near Real Time, Creating Different Decision Mechanisms

Source: SEI

Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Software Is Becoming a More Personal and Valued Utility

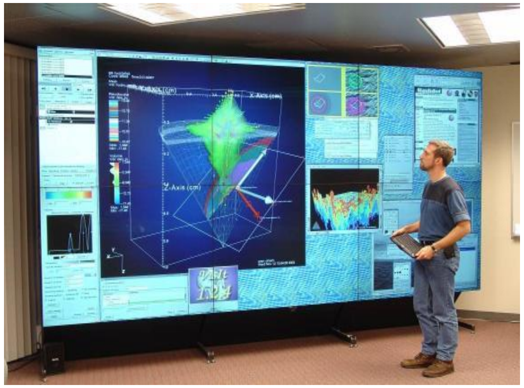
Source: SEI

Struggles in Software Engineering and the Persistent Pursuit of Software Quality – Software Is Globally Important

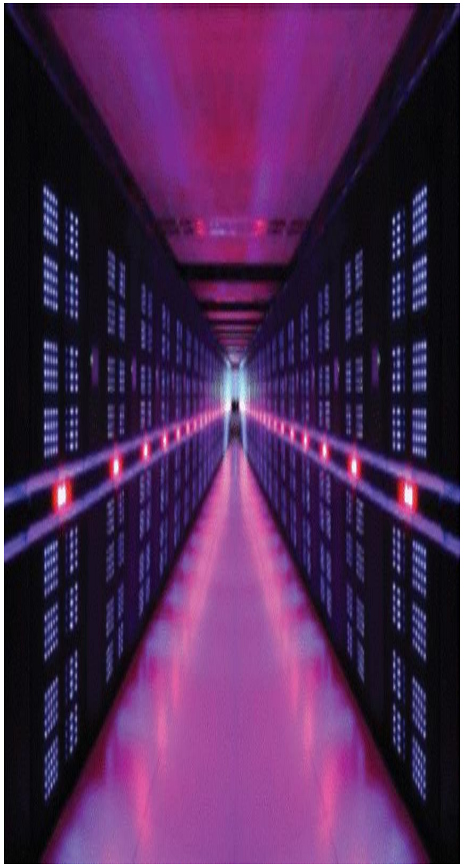
Manufacturing

Finance

Space and Aviation



Engineering



Research

Source: SEI

Future: Software Is the Underpinning of the Cyber Environment, Enabling Explorations into New Frontiers

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Software Is the Underpinning of the Cyber Environment Enabling Explorations into New Frontiers – Software Is Today's Strategic Resource

Increasing Globalization

Source: SEI

Software Is the Underpinning of the Cyber Environment Enabling Explorations into New Frontiers – by Providing Great Capabilities to Bifurcated Communities

Source: SEI

Software Is the Underpinning of the Cyber Environment Enabling Explorations into New Frontiers – Software Engineering and Cybersecurity Are Now Inseparable

- Cybersecurity is now not only one of a software system’s essential qualities, but also a factor that expands the meaning of software quality
- The pursuit of software quality now also must consider the risks from potential actions of an adversarial/malicious user throughout the software lifecycle
- Cybersecurity needs to be included in activities from the onset of the acquisition, designed, and built into the software systems
- Cybersecurity needs to be considered a prime concern as the system is fielded and sustained

Software Is the Underpinning of the Cyber Environment Enabling Explorations into New Frontiers – Software Engineering and Cybersecurity Focus on Providing Effective Business Solutions

"You can spend all sorts of money finding problems…and if you don't fix what you find, you have not solved the problem.

…Key things you should be doing…

1. Code Reviews (with good tools)
2. Architecture Risk Analysis
3. Penetration Testing"

Dr Gary McGraw, fmr member, IEEE CS Brd of Governors, Keynote to HP Protect 2013.

"We really need to be able to analyze what programs are up to, whether they were authored as malware, or whether they were authored as non-malware but have vulnerabilities…I'm implying the ability to inspect a code artifact and determine if

(1) it has vulnerabilities and

(2) if it resembles other things we already know, and

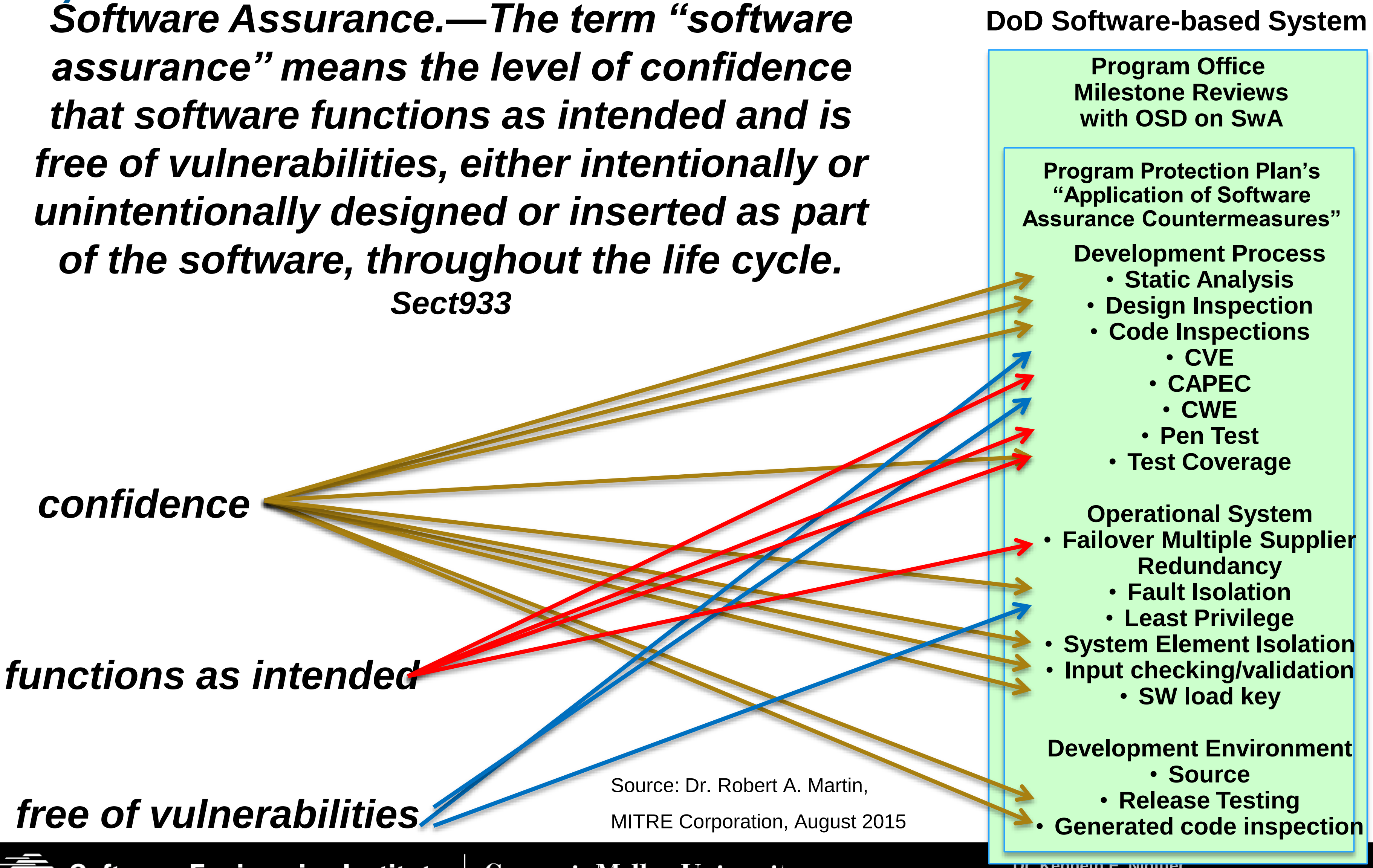
(3) indicators of what it might do."

Dr Kevin Fall, CTO, SEI, Oct 2013.

Software Is the Underpinning of the Cyber Environment Enabling Explorations into New Frontiers –Public Law 113-239 "Section 933 - Software Assurance" & OSD Guidance DoDI 5000.2 (Program Protection Plan)

Software Assurance.—The term "software assurance" means 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 life cycle.

Sect933



DoD Software-based System

Program Office
Milestone Reviews
with OSD on SwA

Program Protection Plan's
"Application of Software Assurance Countermeasures"

Development Process
- Static Analysis
- Design Inspection
- Code Inspections
- CVE
- CAPEC
- CWE
- Pen Test
- Test Coverage

Operational System
- Failover Multiple Supplier Redundancy
- Fault Isolation
- Least Privilege
- System Element Isolation
- Input checking/validation
- SW load key

Development Environment
- Source
- Release Testing
- Generated code inspection

Source: Dr. Robert A. Martin,
MITRE Corporation, August 2015

DoD Program Protection Plan (PPP) Software Assurance Methods

Countermeasure Selection

Source: Dr. Robert A. Martin,

MITRE Corporation, August 2015

Development Process

Apply assurance activities to the procedures and structure imposed on software development

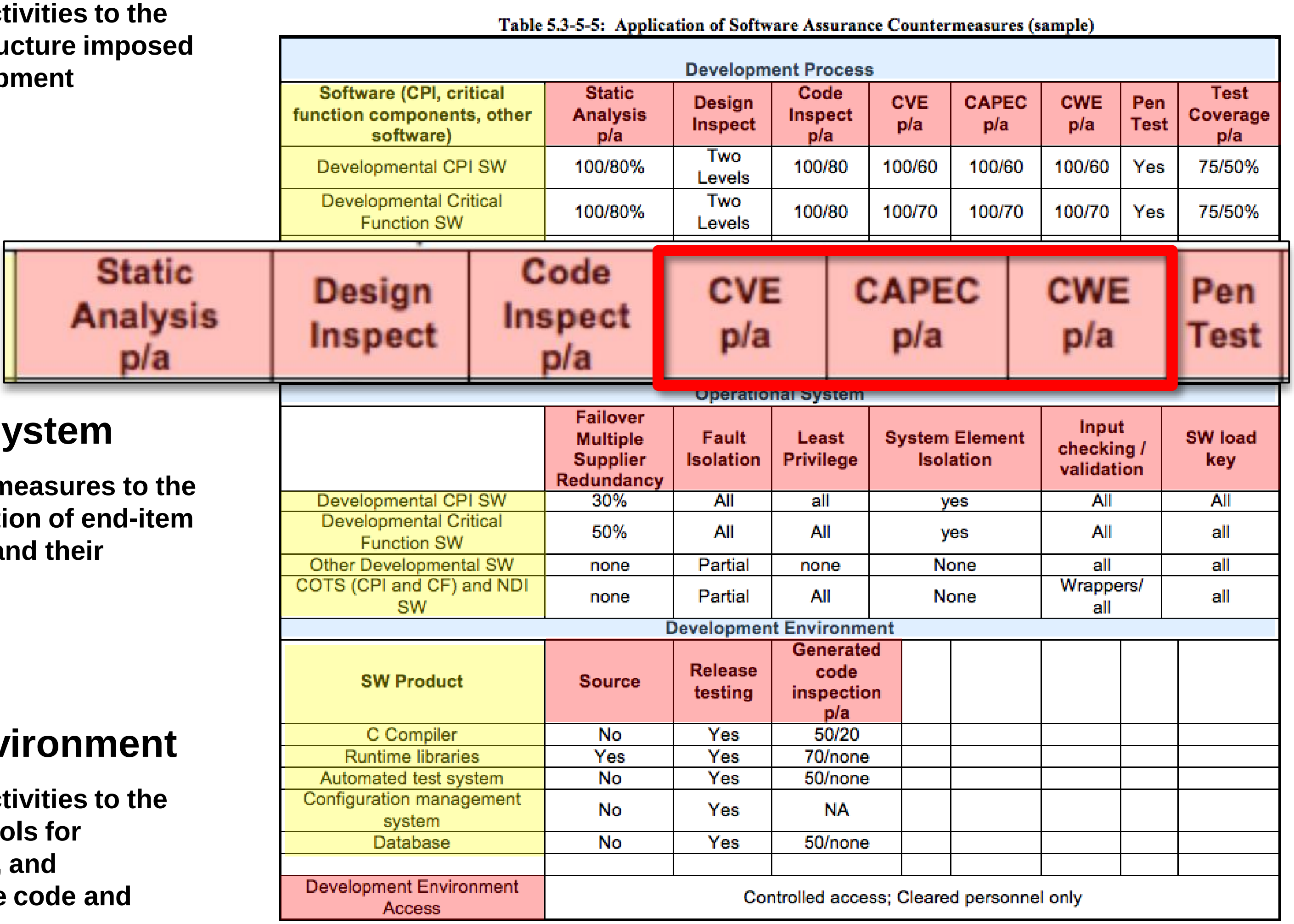
Operational System

Implement countermeasures to the design and acquisition of end-item software products and their interfaces

Development Environment

Apply assurance activities to the environment and tools for developing, testing, and integrating software code and interfaces

Table 5.3-5-5: Application of Software Assurance Countermeasures (sample)



Development Process

Software (CPI, critical function components, other software)	Static Analysis p/a	Design Inspect	Code Inspect p/a	CVE p/a	CAPEC p/a	CWE p/a	Pen Test	Test Coverage p/a
Developmental CPI SW	100/80%	Two Levels	100/80	100/60	100/60	100/60	Yes	75/50%
Developmental Critical Function SW	100/80%	Two Levels	100/80	100/70	100/70	100/70	Yes	75/50%

Static Analysis p/a	Design Inspect	Code Inspect p/a	CVE p/a	CAPEC p/a	CWE p/a	Pen Test

Operational System

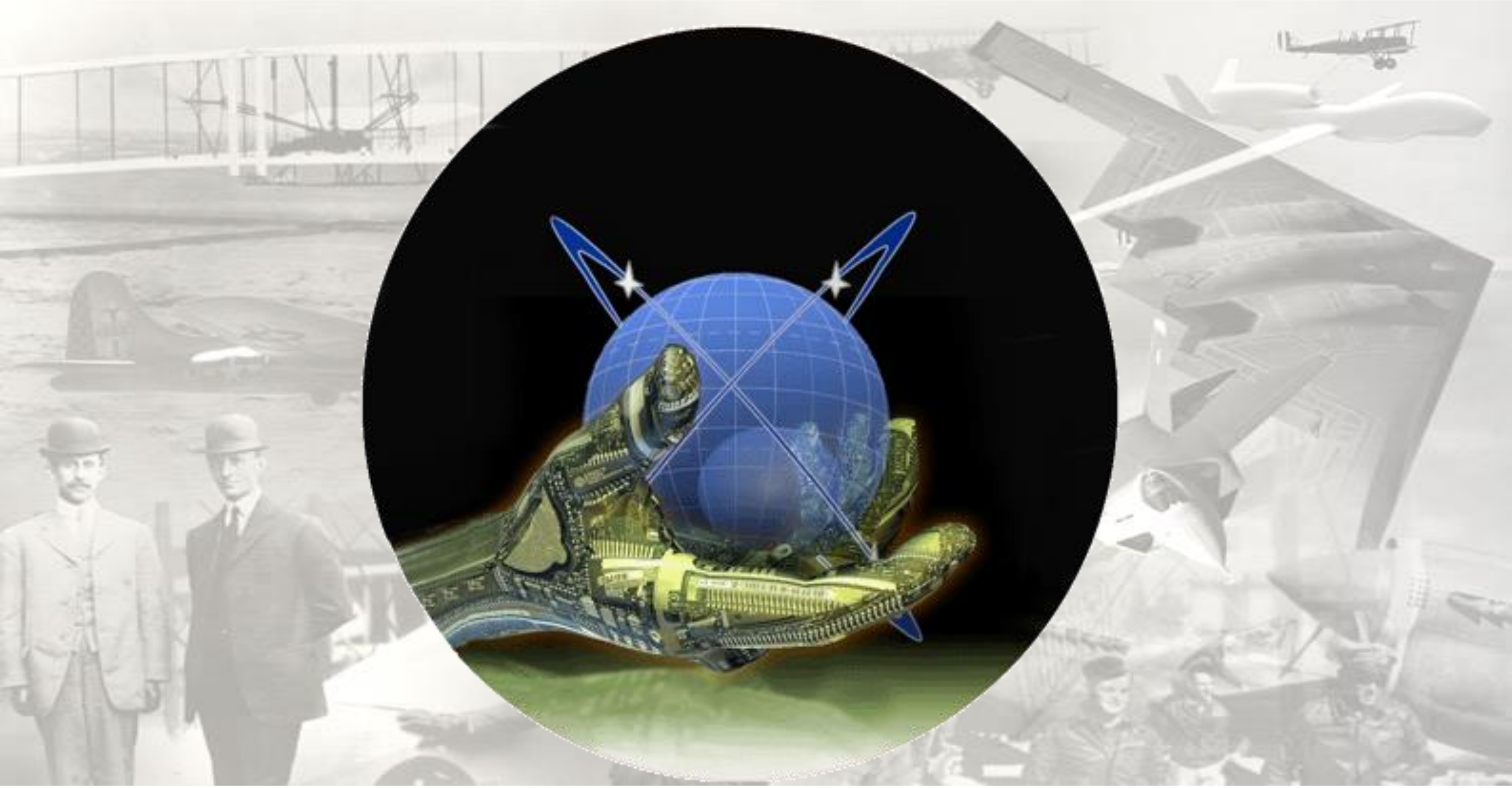
	Failover Multiple Supplier Redundancy	Fault Isolation	Least Privilege	System Element Isolation	Input checking / validation	SW load key
Developmental CPI SW	30%	All	all	yes	All	All
Developmental Critical Function SW	50%	All	All	yes	All	all
Other Developmental SW	none	Partial	none	None	all	all
COTS (CPI and CF) and NDI SW	none	Partial	All	None	Wrappers/ all	all

Development Environment

SW Product	Source	Release testing	Generated code inspection p/a
C Compiler	No	Yes	50/20
Runtime libraries	Yes	Yes	70/none
Automated test system	No	Yes	50/none
Configuration management system	No	Yes	NA
Database	No	Yes	50/none
Development Environment Access	Controlled access; Cleared personnel only		

Additional Guidance in PPP Outline and Guidance

Questions?



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

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.

DM-0002663