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!
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

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

Size of Codebase (SLOC)

<table>
<thead>
<tr>
<th>Software</th>
<th>Millions of Source Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthcare.gov</td>
<td></td>
</tr>
<tr>
<td>Debian 5.0 (all software in package)</td>
<td></td>
</tr>
<tr>
<td>Software in typical new car, 2013</td>
<td></td>
</tr>
<tr>
<td>Mac OS X 10.4</td>
<td></td>
</tr>
<tr>
<td>Debian 5.0 codebase</td>
<td></td>
</tr>
<tr>
<td>US Army's Future Combat System</td>
<td></td>
</tr>
<tr>
<td>Facebook (without backend code)</td>
<td></td>
</tr>
<tr>
<td>Windows Vista (2007)</td>
<td></td>
</tr>
<tr>
<td>Microsoft Visual Studio 2012</td>
<td></td>
</tr>
<tr>
<td>Large Hadron Collider</td>
<td></td>
</tr>
<tr>
<td>Microsoft Office (2013)</td>
<td></td>
</tr>
<tr>
<td>Windows XP (2001)</td>
<td></td>
</tr>
<tr>
<td>Windows 7</td>
<td></td>
</tr>
<tr>
<td>Symbian</td>
<td></td>
</tr>
<tr>
<td>Microsoft Office for Mac (2006)</td>
<td></td>
</tr>
<tr>
<td>Microsoft Office (2001)</td>
<td></td>
</tr>
<tr>
<td>F-35 Fighter</td>
<td></td>
</tr>
<tr>
<td>Apache Open Office</td>
<td></td>
</tr>
<tr>
<td>Linux 3.1 (recent version, 2013)</td>
<td></td>
</tr>
<tr>
<td>Android (upper estimate)</td>
<td></td>
</tr>
<tr>
<td>Boeing 787, total flight software</td>
<td></td>
</tr>
</tbody>
</table>

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

Software in Military Aircraft

<table>
<thead>
<tr>
<th>Year of Introduction</th>
<th>Percent of Functionality Provided by Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>1960 (F-4)</td>
<td>0</td>
</tr>
<tr>
<td>1964 (A-7)</td>
<td>10</td>
</tr>
<tr>
<td>1970 (F-111)</td>
<td>20</td>
</tr>
<tr>
<td>1975 (F-15)</td>
<td>30</td>
</tr>
<tr>
<td>1982 (F-16)</td>
<td>40</td>
</tr>
<tr>
<td>1990 (B-2)</td>
<td>50</td>
</tr>
<tr>
<td>2000 (F-22)</td>
<td>80</td>
</tr>
</tbody>
</table>

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

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"
- 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
- Operational Resilience (RMM)
- SwA for Agile Software Methodologies

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

4: Emerging & Disruptive Technology
- SwA in Highly Parallel, High-Performance Computing Environments
- Security of Mobile Applications & Platforms
- Security in Socio-technical Computing
- Source: SEI

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

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

Known Threat Actors

Attack Patterns (CAPEC)

Weaknesses & Vulnerabilities (CWE/CVE)

Actions*

System & System Security Engineering Trades

Technical Impacts

Operational Impacts

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

© 2013 MITRE
CVE 1999 to 2015

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

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

Unreported or undiscovered Vulnerabilities

Uncharacterized Weaknesses

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
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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

- Pandemic
- 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

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

<table>
<thead>
<tr>
<th></th>
<th>PHYSICAL SCIENCE</th>
<th>BIOSCIENCE</th>
<th>COMPUTER/SOFTWARE/CYBER SCIENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origins/History</td>
<td>Begun in antiquity</td>
<td>Begun in antiquity</td>
<td>Mid-20th Century</td>
</tr>
<tr>
<td>Enduring Laws</td>
<td>Laws are foundational to</td>
<td>Laws are foundational to</td>
<td>Only mathematical laws have proven foundational to computation</td>
</tr>
<tr>
<td></td>
<td>furthering exploration in</td>
<td>furthering exploration in the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the science</td>
<td>science</td>
<td></td>
</tr>
<tr>
<td>Framework of</td>
<td>Four main areas:</td>
<td>Science of dealing with</td>
<td>• Several areas of study: computer science, software/ systems</td>
</tr>
<tr>
<td>Scientific Study</td>
<td>astronomy, physics,</td>
<td>health maintenance and</td>
<td>engineering, IT, HCl, social</td>
</tr>
<tr>
<td></td>
<td>chemistry, and earth</td>
<td>disease prevention/ treatment</td>
<td>dynamics, AI</td>
</tr>
<tr>
<td></td>
<td>sciences</td>
<td></td>
<td>• All nodes attached to/relying on netted system</td>
</tr>
<tr>
<td>R&amp;D and Launch Cycle</td>
<td>10-20 years</td>
<td>10-20 years</td>
<td>Significantly <strong>compressed</strong>; solution time to market needs to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>happen very quickly</td>
</tr>
</tbody>
</table>

Source: SEI

HCl: 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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Silent Generation</td>
<td>Hard worker</td>
<td>Workaholic</td>
<td>Technically advanced</td>
<td>Technically savvy</td>
</tr>
<tr>
<td>Baby Boomers</td>
<td>Respects authority</td>
<td>Questions authority</td>
<td>Prefers informality</td>
<td>Embraces diversity</td>
</tr>
<tr>
<td>Generation X</td>
<td>Work is obligation</td>
<td>Works efficiently</td>
<td>Needs structure and direction</td>
<td>Requires supervision</td>
</tr>
<tr>
<td>Generation Y/Millenials</td>
<td>Formal communicator</td>
<td>Competitive</td>
<td>Direct/immediate communicator</td>
<td>Indirect/virtual communicator</td>
</tr>
<tr>
<td></td>
<td>Work/family separation</td>
<td>Little work/life balance</td>
<td>Seeks work/life balance</td>
<td>Demands work/life balance</td>
</tr>
</tbody>
</table>

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**
- Requirements
- Design
- Implementation
- Verification
- Maintenance

**Iterative Models**
- 1. Determine objectives
- 2. Identify and resolve risks
- 3. Development and Test
- 4. Plan the next iteration

**Adaptive Models**
- Scrum
- XP
- OpenUP

**Source:** Noblis

---

Dr. Kenneth E. Nidiffer
The 27th Annual IEEE Software Technology Conference
Struggles in Software Engineering and the Persistent Pursuit of Software Quality – **Software Connects Us in Near Real Time, Creating Different Decision Mechanisms**
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**
Software Is the Underpinning of the Cyber Environment Enabling Explorations into New Frontiers – by Providing Great Capabilities to Bifurcated Communities

Source: SEI
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 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

Source: Dr. Robert A. Martin, MITRE Corporation, August 2015
DoD Program Protection Plan (PPP) Software Assurance Methods

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)

<table>
<thead>
<tr>
<th>Development Process</th>
<th>Static Analysis p/a</th>
<th>Design Inspect</th>
<th>Code Inspect p/a</th>
<th>CVE p/a</th>
<th>CAPEC p/a</th>
<th>CWE p/a</th>
<th>Pen Test</th>
<th>Test Coverage p/a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developmental CPI SW</td>
<td>100/80%</td>
<td>Two Levels</td>
<td>100/60</td>
<td>100/60</td>
<td>100/60</td>
<td>Yes</td>
<td>75/50%</td>
<td></td>
</tr>
<tr>
<td>Developmental Critical Function SW</td>
<td>100/80%</td>
<td>Two Levels</td>
<td>100/70</td>
<td>100/70</td>
<td>100/70</td>
<td>Yes</td>
<td>75/50%</td>
<td></td>
</tr>
</tbody>
</table>

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.

Additional Guidance in PPP Outline and Guidance

Countermeasure Selection

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