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

Dr. Kenneth E. Nidiffer

20th Annual Systems Engineering Conference
Waterford Conference Center
Springfield, VA
October 23–26 2017

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Dynamics of Integrating Software Assurance Engineering Activities into the System Acquisition Life Cycle

Copyright 2017 Carnegie Mellon University. All Rights Reserved.
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.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.
This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM17-0799
Software-Enabled Systems Are Today’s Strategic Resource

“Software is the building material for modern society”

Dr. Bill Scherlis*

Manual Labor

Water

Steam

Oil

Software

Increasing Globalization, Productivity, and Complexity

Source: SEI

*Dr. Bill Scherlis
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

* 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 as a strategic initiative
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: DoD Stakeholders and Different Perspectives on Software Assurance

Software Assurance
Cyber Resiliency
Cybersecurity
Mission Assurance

J2 - Intel
It’s Hacking!
J3 - Ops
It’s Network Defense!
PM
Cybersecurity is Operational!
SE
J6 – CIO
It’s Program Protection!
IT
It’s Electronic Warfare!
T&E
LOG

Source: DAU

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

Source: SEI
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
Deputy Director, Spectrum Policy and International Engagements
DoD Chief Information Officer

UNCLASSIFIED

DoD must be able to operate:
- between layers
- between networks
- between domains
- between environments

Source: Kenneth R. Turner
Deputy Director, Spectrum Policy and International Engagements
DoD Chief Information Officer

UNCLASSIFIED
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

Addressing System Sustainment
Strategic Trends Impacting Software Sustainment

• The future is software-centric…dependent…

• Advanced DoD capabilities manifested in algorithms/software
  – Autonomy, machine learning, collaborative systems, competitive networking, system resiliency, IoT, hybrid cloud computing, human-computer interaction, digital security, analytics for decision making…
  – Anticipating/adapting to new dynamic threats and requirements

• Warfighter needs coupled with technology innovation dynamics drive need for strategic enterprise focus on evolving a DoD enterprise life-cycle software strategy to be aligned with this changing environment
  – Policies
  – Current and future infrastructure capabilities
  – Business models
  – Investment strategies
Handling the Expanding Code Base
Software is dramatically expanding with limited natural governance

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

- **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.

Source: Bob Martin, MITRE
Increasing Vulnerabilities: CVE 1999 to 2017, Reported Common Vulnerabilities and Exposures (CVE)

Source: Dr. Robert A. Martin, MITRE Corporation, May 2017
Designing-in Software Assurance Throughout the System Life Cycle

<table>
<thead>
<tr>
<th>Phase</th>
<th>Where Software Flaws Are Introduced</th>
<th>Where Software Flaws Are Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Engineering</td>
<td>70%</td>
<td>3.5%</td>
</tr>
<tr>
<td>System Design</td>
<td>20%</td>
<td>16%</td>
</tr>
<tr>
<td>Software Architectural Design</td>
<td>10%</td>
<td>50.5%</td>
</tr>
<tr>
<td>Component Software Design</td>
<td></td>
<td>9%</td>
</tr>
<tr>
<td>Code Development</td>
<td></td>
<td>21%</td>
</tr>
<tr>
<td>Unit Test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acceptance Test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Special emphasis needed up-front in the system life cycle

Sources: *Critical Code*, NIST, NASA, INCOSE, and Aircraft Industry Studies
Reducing Technical Debt Over the System Life Cycle

Reducing Technical Debt

May Not Be Right Direction if Pushing Costs to Sustainment

Right Direction if More Assured Programs

Reducing Technical Debt

Software Development Lifecycle

<table>
<thead>
<tr>
<th>Where Faults are Introduced</th>
<th>Where Faults are Found</th>
<th>Nominal Cost per Fault for Fault Removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>70%</td>
<td>3.5%</td>
<td>$1</td>
</tr>
<tr>
<td>20%</td>
<td>16%</td>
<td>$5</td>
</tr>
<tr>
<td>10%</td>
<td>50.5%</td>
<td>$10</td>
</tr>
<tr>
<td></td>
<td>9%</td>
<td>$80</td>
</tr>
<tr>
<td></td>
<td>20.5%</td>
<td>$300–1000</td>
</tr>
</tbody>
</table>

1–5% of defects are vulnerabilities; common vulnerabilities are driven by coding/design defects of known types.

Sources: Critical Code, NIST, NASA, INCOSE, and Aircraft Industry Studies
Reducing Technical Debt: Engineering-in Software Assurance Activities Across the Life Cycle

DoD PM’s Guidebook for Integrating Software Assurance Engineering Activities into the Systems Acquisition Lifecycle (Being Developed)
## 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**

<table>
<thead>
<tr>
<th></th>
<th>Physical Science</th>
<th>Bioscience</th>
<th>Computer/Software/Cyber Science</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Origins/History</strong></td>
<td>Begun in antiquity</td>
<td>Begun in antiquity</td>
<td>Mid-20th century</td>
</tr>
<tr>
<td><strong>Enduring Laws</strong></td>
<td>Laws are foundational to furthering exploration in the science</td>
<td>Laws are foundational to furthering exploration in the science</td>
<td>Only mathematical laws have proven foundational to computation</td>
</tr>
</tbody>
</table>
| **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 |

HCl: 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