Software Assurance, Trustworthiness, and Rigor

Don O'Neill, Independent Consultant

Abstract. Trustworthiness requires a commitment to rigor in both software production and its verification. No soft skill, rigor has a hard edge. In an environment of neglect and unmet need, the introduction of rigor would be a game changer, one that requires both cultural change and engineering know how with pass/fail training certification.

Introduction

Software Assurance only has meaning in the context of trustworthiness, that is, worthy of being trusted to fulfill the critical requirements needed for a particular software component, system, or system of systems [1]. Software Assurance demands two capabilities associated with trustworthiness, the capability to produce trustworthy software products and the capability to verify that software products are trustworthy. Each depends on engineering and technology rigorously applied.

The kernel of the layered defense approach to Software Assurance is Build Security In and Structured Programming with its rigorous and provably correct use of zero and one predicate prime programs along with proper programs composed of multiple prime programs limited to single entry and single exit [2].

The layered defense approach described here includes Defense in Depth, Build Security In, Risk Management Calculation, and Resilience processes, engineering, and tool automation dependent on an elevated state of software engineering rigor and precision [3]. There is much room for improvement in the Software Assurance methods and practices that assure such rigor.

Software Assurance Definition

From DOD ASD (R&E) the term "software assurance", or "SwA", means the level of confidence that software functions as intended, and only as intended, and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software, throughout the lifecycle. According to the Department of Homeland Security, software assurance spans:

1. **Trustworthiness -** No exploitable vulnerabilities exist, either maliciously or unintentionally inserted.
2. **Predictable Execution -** Justifiable confidence that software, when executed, functions as intended.
3. **Conformance -** Planned and systematic set of multi-disciplinary activities that ensure software processes and products conform to requirements, standards, and procedures.

Accordingly then, the objectives of this article are to be better able to pinpoint the factors and sources of risk involved in achieving Software Assurance, to identify concrete Software Assurance objectives involved and consequential outcomes sought in meeting each of the goals associated with software production and its verification, and to visualize the operations of Software Assurance through the use of assurance assertions and indicators. Concrete Software Assurance objectives with consequential outcomes include components that are provably correct; demonstrably free of weaknesses and Cyber Security vulnerabilities; and demonstrably free of improper proprietary information, copyrighted material, and trade secrets.

Background

On July 31, 2007, the Software Assurance State of the Art Report [4] described the emerging activities, approaches, and technologies thought to be the foundation of the discipline of software security. While the report mentioned Inspections, Correctness by Construction, Static Analysis, Function Extraction for Malicious Code (FX/MC), Common Weakness Enumeration (CWE), and Standard of Excellence Product Checklists, it did not include Structured Programming, Clean Room Software Engineering, or Statistical Testing. Build Security In depends on the adoption of all of these methods. Also not mentioned in SOAR were Technical Debt and Build Security In except for reference to a portal name. Importantly, SOAR identified the need for security extensions to the CMMI®, so far only discussed and not acted upon. Resilience was mentioned in the context of resistance to and tolerance of attack, fault tolerance, and system failure recovery but not in the context of anticipating, avoiding, withstanding, mitigating, and recovering from the effects of adversity whether natural or manmade under all circumstances of use including both systems and systems of systems.

Situation is Dire

The current state of Software Assurance is dire and stems from a combination of neglect and unmet need [1,5]. The future state of Software Assurance needs to feature smart and trusted methods and metrics including Software Assurance risk calculation that is precise, accurate, timely, and complete.

Some unmet need is due to lack of adoption and some, lack of technology. Lack of adoption falls on both industry practitioners and government acquisition specialists. Neglect is on glaring display in the intentional practice of Technical Debt spurred on by a misguided culture of acceptance [6, 7]. Furthermore, the shortfall in STEM workforce yields a persistent level of unintentional neglect based on ignorance. Beyond Software Assurance adoption, the focus areas most in need of attention are Build Security In, Resiliency, Advanced Technology, and renovation of the CMMI to address Software Assurance.

The improper use of proprietary software involving proprietary information, copyrighted material, and trade secrets increasingly goes undetected. Uncovering such use and detecting specific instances is a technical challenge, one that usually requires full and ready access to the Dirty System source code for best results as well as the wherewithal and means to express the proprietary information, copyrighted material, and trade secrets in a precise, rigorous, and trusted abstract manner suitable for computer searching and comparison [8].

Focus Areas	Known Practice	Underutilized Practice	Underutilized Technology	Advanced Technology Needed
Technical Debt	Discontinue this practice since it intentionally fosters vulnerabilities and neglect			
CMMI	Excellent platform for achieving widespread adoption yet incomplete with respect to Software Assurance, Cyber Security		Software Assurance, Cyber Security	
Defense In Depth	Encryption, identity management, access control, authorization management, accountability management, incident management, configuration management, and security assurance operations	Encryption, authorization management, security assurance operations	Three factor authentication: what you are, what you have, what you know	Trusted encryption advances needed
Build Security In	Structured Programming with emphasis on proof of correctness and correctness by construction	Emphasis on proof of correctness and correctness by construction		
	Cleanroom Software Engineering and Statistical Testing	Cleanroom Software Engineering	Statistical Testing	
	Software Inspections Process with well defined Standard of Excellence for SDLC software artifacts	Well defined Standard of Excellence: completeness, correctness, style, rules of construction, multiple views		
	Static Analysis		Approximate Matching to disambiguate multi-vendor tool results	Static Analysis commonality and interoperable tool vendor results post processing
	Function Extraction	Function Extraction	Function Extraction applied to large programs and systems	All languages and instruction set architectures
	Common Weakness and Known Vulnerability Evaluation	Common Weakness and Known Vulnerability Evaluation		Integration of Common Weaknesses and Known Vulnerabilities with Static Analysis tool automation
	Proprietary information, copyrighted material, and trade secret detection	Rigorous and systematic abstraction, filtration, comparison	Approximate Matching using text strings to detect fragments Function Extraction for abstracting intended function	Hypernion using Behavior Specification Units (BSU's) for detecting intended function
Risk Management Calculation	Goal identification and actually calculating risks based on systematically distinguishing factors, sources of risk, and problems using levels of confidence	Actually calculating risks based on systematically distinguishing factors, sources of risk, and problems using levels of confidence	Counterfeit and tainted component detection confidence level Trusted chain of custody confidence level	Cognitive Systems, that is, IBM Watson, with emphasis on levels of confidence used in calculating risk
Resilience	Defense in Depth, Business Continuity, Survivability and RMA Engineering, and Resiliency featuring: coordinated recovery time objectives, digital situation awareness, distributed supervisory control, selective availability, operation sensing and monitoring, information and data recovery, and interoperability of data and information exchange	Coordinated recovery time objectives, digital situation awareness, selective availability, data and information exchange		Cognitive Systems, that is, IBM Watson, with emphasis on anticipation, avoidance, and digital situation awareness

Table 1. Software Assurance Practice and Technology Assessment

Software Assurance Layered Defensive Approach

The kernel of the layered defense approach to Software Assurance is Structured Programming and the rigorous and provably correct use of zero and one predicate prime programs along with proper programs composed of multiple prime programs limited to single entry and single exit [2]. The layered defense approach recommended here includes Defense in Depth, Build Security In, Risk Management Calculation, and Resilience.

1. Defense in Depth is composed of encryption, identity management, access control, authorization management, accountability management, incident management, configuration management, and security assurance operations.
2. Build Security In is composed of Structured Programming with emphasis on correctness [2, 9], Cleanroom Software Engineering and Statistical Testing [10, 11], Software Inspections [12], Static Analysis tool use [13], Function Extraction tool use [14], common weakness [15] and known vulnerability evaluation [16].
3. Risk Management calculation includes goal identification and actually calculating risks based on systematically distinguishing factors, sources of risk, and problems using levels of confidence.
4. Resiliency prerequisites are Defense in Depth, Business Continuity, and Survivability, and features of Resiliency include coordinated recovery time objectives, digital situation awareness, distributed supervisory control, selective availability, operation sensing and monitoring, information and data recovery, and interoperability of data and information exchange [17, 18, 19].

These layered defense tactics are assessed in Table 1 in terms of Software Assurance practice and technology distinguished by known practice, underutilized practice, underutilized technology, and advanced technology needed. A framework of cyber tactics including anticipation, detection, attribution, and counter measures is needed to reason about tools, tiers, and threat categories [20]. Cause and effect chain elements spanning goals, weaknesses, attributes of building security in, attack outcomes, bad actors, and consequences must be identified and traced for selected Cyber Tactics. Consequences must be traced for selected goals and attack outcomes. In the spirit of continuous improvement and consistent with the erratic and rapid pace of both technology and threat emergence, all methods and means including tools need to be assessed and evaluated continuously. Refreshment and update is not so much periodic; instead refreshment is on demand in accordance with the "check everything every time" principle [21].

Obstacles and Underutilized Technology

The practice of Technical Debt fosters permissible and persistent neglect and needs to be discontinued and eliminated [7]. Technical Debt is the organizational, project, or engineering neglect of known good practice that can result in persistent public, user, customer, staff, reputation, or financial cost [6].

While the CMMI provides an excellent platform for achieving widespread adoption of software process maturity, it is incomplete with respect to Software Assurance and Cyber Security process. The importance of underutilized technology lies in the potential of the CMMI and its widespread infrastructure of usage to serve as the platform for Software Assurance adoption and dissemination [4].

With respect to Defense in Depth, encryption, authorization management, and security assurance operations are underutilized practices. In addition three factor authentication based on what you are, what you have, and what you know is an underutilized technology. Finally, trusted encryption advances are needed.

Build Security In

Quite specifically, Structured Programming with the rigorous and provably correct use of zero and one predicate prime programs along with proper programs composed of multiple prime programs limited to single entry and single exit is the kernel of trustworthy Software Assurance [2].

With respect to Build Security In, there are numerous underutilized practices including Structured Programming with insufficient emphasis on correctness and proof of correctness and Inspections with well defined standard of excellence spanning completeness, correctness, style, rules of construction, and multiple views [22, 23]; Cleanroom Software Engineering with its underutilized practice and Statistical Testing and its underutilized technology; Function Extraction and its underutilized practice and technology and its need for advanced technology; the underutilized practice of common weaknesses and known vulnerabilities and the need for additional advanced technology; and the advanced use case of detecting proprietary information, copyrighted material, and trade secrets using Carnegie Mellon University's (CMU) Function Extraction [14], NIST's Approximate Matching [24], and Oak Ridge National Laboratory's (ORNL) Hydernion [25].

Promising Areas for Research

Defense in Depth awaits trusted encryption advances. Static Analysis needs commonality and interoperable tool vendor results post processing advances to disambiguate multi-vendor results and decimate the number of false positives. Function Extraction needs to be able to process large programs and systems such as, those found in legacy software systems, COTS, reuse libraries, and commercial software releases. In addition, Function Extraction needs to be able to process source code from a wide range of languages and object code from various instruction set architectures. Common weakness and known vulnerabilities need to be integrated with Static Analysis tool automation.

Proprietary information, copyrighted material, and trade secret detection can potentially be determined using NIST's Approximate Matching using text strings to detect fragments. More advanced, CMU's Function Extraction for abstracting intended function and ORNL's Hypernion using Behavior Specification Units (BSU's) for detecting intended function offer promise. Proprietary software is licensed under exclusive legal right of the copyright holder with the intent that the licensee is given the right to use the software only under certain conditions, and restricted from other uses. In the legal community, the Abstraction-Filtration-Comparison (AFC) test[1] is a three-step process for determining substantial similarity of the non-literal elements of a computer program. Abstraction distinguishes which aspects of the program constitute its expression and which are the ideas. Filtration removes from consideration aspects of the program that are not legally protectable by copyright, such as, elements associated with efficiency, external factors, and the public domain. Comparison considers whether just those aspects of the program that constitute its expression and not those aspects not legally protected by copyright are present in the Clean System.

Risk Management Calculation needs to apply Cognitive Systems, that is, IBM Watson, with its emphasis on levels of confidence useful in calculating risk. More specifically, with respect to Risk Management Calculation, while goal identification is commonplace, actually calculating risks based on systematically distinguishing factors, sources of risk, and problems using levels of confidence is an underutilized practice. Beyond that, determining the level of confidence in assigning a failed state to a factor is assisted by evidence-based assurance assessment questions.

Resilience needs to apply Cognitive Systems, that is, IBM Watson, with its potential for anticipation, avoidance, and digital situation awareness. More specifically, with respect to Resilience, coordinated recovery time objectives, digital situation awareness, selective availability, and data and information exchange represent underutilized practices and features.

Line of Sight for Adoption

The line of sight for a target organization adopting Software Assurance is guided by a five level maturity value map including Defense in Depth, Build Security In, Risk Management Calculation, and Resilience.

* **Level 2:** Defense in Depth including standards for existing methods and practice for assuring security focuses on protection from vulnerabilities and threats through security in depth. Protective measures include encryption, identity management, access control, authorization management, accountability management, incident management, configuration management, and security assurance operations.
* **Level 3:** Build Security In including standards for existing methods and practice for trustworthiness focuses on building security in by constructing software components, systems, and systems of system to an engineered standard of excellence based on completeness, correctness, and rules of construction. Beginning with discontinuing the practice of Technical Debt, Build Security In practices include structured programming with an emphasis on correctness, Clean Room software engineering and statistical testing, software inspections process with well defined standard of excellence, the application of static analysis tools, the use of common weaknesses and known vulnerabilities, and the application of Function Extraction (FX).
* **Level 4:** Risk Management Calculation including standards for existing methods and practice for calculating software assurance uncertainty focuses on factors, sources of risk and problems evaluated for each goal, a count of factors that might

serve as sources of problems in goal achievement, and a count of factors that represent objectives that are in a failed state. Determining the level of confidence in assigning a failed state is assisted by evidence-based assurance assessment questions. For each goal, the calculated risk is the number of problems divided by factors evaluated expressed as a percent.

* **Level 5:** Resilience of Systems of Systems [18,19] and Supply Chains [26, 27] including standards for existing methods and practice for assuring resiliency under stress focuses on anticipating, avoiding, withstanding, mitigating, and recovering from the effects of adversity whether natural or manmade under all circumstances of use. Features of resiliency engineering include coordinated recovery time objectives, digital situation awareness, distributed supervisory control, selective availability, operation sensing and monitoring, information and data recovery, and interoperability of data and information exchange.

Software Assurance maturity seeks some of the same consequential outcomes as traditional organizational maturity, such as, more efficiency, more predictability, less risk, and less chaos. Accordingly efforts would be directed at incorporating Software Assurance and Cyber Security into the CMMI in order to utilize its existing effective dissemination and adoption infrastructure [28, 4]. While Software Assurance certification programs and university degree programs may successfully impact and meet the developmental needs of individuals, the organizational capability and its systematic appraisal and assessment would be tied to a Software Assurance-enabled CMMI.

Summary and Conclusion

The layered defense approach recommended here includes the Defense in Depth, Build Security In, Risk Management Calculation, and Resilience engineering and processes dependent on an elevated state of software engineering rigor. There is much room for improvement in the rigor of Software Assurance methods and practices capable of delivering consequential outcomes including components that are provably correct; demonstrably free of weaknesses and Cyber Security vulnerabilities; and demonstrably free of improper proprietary information, copyrighted material, and trade secrets.

Fueled by STEM shortfall, Software Assurance is impeded by the acceptance of Technical Debt, a practice that breeds neglect and should be discontinued. Advancing Software Assurance adoption would be stimulated by renovating the CMMI to include Software Assurance and Cyber Security process capabilities.

Strengthening Defense in Depth encryption, authorization management, and security assurance operations practices is needed. Build Security In emphasis on correctness proofs, correctness by construction, and the defined standard of excellence for completeness, correctness, style, rules of construction, and multiple views is the required foundation for the increased rigor being sought. The full utilization of statistical testing, Function Extraction, and static analysis automation would verify and calibrate the improvement in rigor in a cost effective and schedule efficient manner enabling the Next Generation Software Engineering goal of doing more with less… fast [29]. Much needs to be done on Resilience including the widespread implementation of Resiliency functions and features.

ABOUT THE AUTHOR

Don O'Neill served as the President of the Center for National Software Studies (CNSS) from 2005 to 2008. Following twenty-seven years with IBM's Federal Systems Division (FSD), he completed a three-year residency at Carnegie Mellon University's Software Engineering Institute (SEI) under IBM's Technical Academic Career Program and has served as an SEI Visiting Scientist. A seasoned software engineering manager, technologist, independent consultant, and expert witness, he has a Bachelor of Science degree in mathematics from Dickinson College in Carlisle, Pennsylvania. His current research is directed at public policy strategies for deploying resiliency in the nation's critical infrastructure; disruptive game changing fixed price contracting tactics to achieve DOD austerity; smart and trusted tactics and practices in Supply Chain Risk Management Assurance; and a defined Software Clean Room Method for transforming a proprietary system into a Clean System devoid of proprietary information, copyrighted material, and trade secrets and confirming, verifying, and validating the results.

Phone: 301-990-0377
E-mail: oneilldon@aol.com

NOTES

1. Wikipedia, "Abstraction-Filtration-Comparison test", <http://en.wikipedia.org/wiki/Abstraction-Filtration-Comparison_test>

REFERENCES

1. "Software 2015: A National Software Strategy to Ensure U.S. Security and Competitiveness," Center for National Software Studies, May 2005 <http://www.cnsoftware.org/nss2report/NSS2FinalReport04-29-05PDF.pdf>
2. Linger, R.C., H.D. Mills, B.I. Witt, "Structured Programming: Theory and Practice", Addison-Wesley Publishing Company, Inc., 1979
3. O'Neill, Don, "Software Assurance Trustworthiness and Rigor", 2013, <http://youtu.be/N2IT84hEhkU>
4. Goertzel, Karen Mercedes, Theodore Winograd, Holly Lynne McKinley, Lyndon Oh, Michael Colon, Thomas McGibbon, Elaine Fedchak, Robert Vienneau, "State of the Art Report: Software Security Assurance", IATAC and DACS , July 31, 2007 <http://iac.dtic.mil/csiac/download/security.pdf>
5. O'Neill, Don "Software 2015: Situation Dire YouTube", 2013, <http://youtu.be/qKBKbhH_J64>
6. O'Neill, Don, "Technical Debt in the Code: Cost to Software Planning", Defense AT&L Magazine, March-April 2013 <http://www.dau.mil/pubscats/ATL%20Docs/Mar_Apr_2013/O%27Neill.pdf>
7. O'Neill, Don, "Technical Debt YouTube Presentations", 2013, <http://youtu.be/1z6LPnRL4wU>
8. O'Neill, Don, "A Defined Software Clean Room Method for Transforming a Dirty System into a Clean System", <http://youtu.be/UTfHD1ORjOs, 2014>
9. Kourie, D.G., B.W. Watson, "The Correctness-by-Construction Approach to Programming", Springer, ISBN: 978-3-642-27918-8, 264 pages, 2012
10. Linger, Richard C. and Carmen J. Trammell, "Cleanroom Software Engineering Reference Model", Version 1.0, Technical Report CMU/SEI-96-TR-022, Carnegie Mellon University Software Engineering Institute, November 1996 <http://resources.sei.cmu.edu/asset_files/TechnicalReport/1996_005_001_16502.pdf>
11. Trammell, Carmen J., Richard C. Linger and Jesse H. Poore Stacy J. Prowell "Cleanroom Software Engineering: Technology and Process", SEI Series in Software Engineering, Addison-Wesley, March 19, 1999, 390 pages
12. O'Neill, Don, Software Inspections Situations Video, <http://youtu.be/BvIvh5RD4Bk>, 15:22 minutes
13. Okun Vadim, Aurelien Delaitre, Paul E. Black, "Report on the Static Analysis Tool Exposition SATE IV, NIST Special Publication 500-297, January 2013 <http://samate.nist.gov/docs/NIST_Special_Publication_500-297.pdf >
14. Hevner, Alan R., Richard C. Linger, Rosann W. Collins, Mark G. Pleszkoch, Stacy J. Prowell, Gwendolyn H. Walton, "The Impact of Function Extraction Technology on Next Generation Software Engineering", July 2005, CMU/SEI-2005-TR-015 <http://www.sei.cmu.edu/reports/05tr015.pdf>
15. "Common Weakness Enumeration" MITRE, 2013, <http://cwe.mitre.org>
16. "Common Vulnerabilities & Exposures Database". 2013, <http://cve.mitre.org>
17. Resilient Military Systems and the Advanced Cyber Threat, Department of Defense Defense Science Board Task Force Report, January 2013
18. O'Neill, Don, "Maturity Framework for Assuring Resiliency Under Stress", Build Security In web site, Operated by Software Engineering Institute and Sponsored by Department of Homeland Security, July 2008
19. O'Neill, Don, "Meeting the Challenge of Assuring Resiliency Under Stress", CrossTalk, The Journal of Defense Software Engineering, September/October 2009 <http://www.crosstalkonline.org/storage/issue-archives/2009/200909/200909-ONeill.pdf>
20. O'Neill, Don, "Cyber Strategy, Analytics, and Tradeoffs: A Cyber Tactics Study", CrossTalk, The Journal of Defense Software Engineering, September/October 2011 <http://www.crosstalkonline.org/storage/issue-archives/2011/201109/201109ONeill.pdf>
21. Mead, Nancy R. and Carol A. Sledge, "Foundations of Software Lecture", Software Assurance for Executives, MSWA Curriculum, CERT, Software Engineering Institute, Carnegie Mellon University, 2013 <http://www.cert.org/mswa/SAE-course-material-list.html>
22. O'Neill, Don, "Software Inspections Tutorial", Software Engineering Institute, Technical Review, 1988
23. O'Neill, Don, "Peer Reviews", Encyclopedia of Software Engineering- Volume 2, Second Edition, Edited by John Marciniak, John Wiley & Sons, Inc., January 2002, pp. 929-945
24. "Approximate Matching: Definition and Terminology", Draft NIST Special Publication 800-168, January 2014, <http://csrc.nist.gov/publications/drafts/800-168/sp800_168_draft.pdf>
25. "The Hyperion System: Computing the Behavior of Software", Cyber and Information Security Research Group, Oak Ridge National Laboratory, 2014
26. Croll, Paul R., "Supply Chain Risk Management: Understanding Vulnerabilities in Code you Buy, Build, or Integrate", CrossTalk, The Journal of Defense Software Engineering, March/April 2012 <http://www.crosstalkonline.org/storage/issue-archives/2012/201203/201203-Croll.pdf>
27. O'Neill, Don, "Software and Supply Chain Risk Management (SSCRM) Assurance Framework", 2013, <http://youtu.be/1uUnG6HY1_c>
28. O'Neill, Don, "Cyber Security and CMMI", 2013, <http://youtu.be/evkdv0j4ZRo>
29. O'Neill, Don, "Preparing the Ground for Next Generation Software Engineering", IEEE Reliability Society, Annual Technology Report 2008, pp. 148-151, June 2009
30. O'Neill, Don, "Calculating Security Return on Investment", Build Security In web site, Operated by Software Engineering Institute and Sponsored by Department of Homeland Security, February 2007
31. Software Analysis and Forensic Engineering (SAFE), <http://www.safe-corp.biz/products_codesuite.htm>
32. O'Neill, Don "IBM Watson Experimental Prototype Challenge: Supply Chain Risk Management Assurance", 2013, <http://youtu.be/QycPEI0RY20>