NASA and DOD thrust on Software Assurance to assure to assure high quality, safety, securety and reliability of mission-critical systems

Computers are increasingly being introduced into safety-critical systems, and, as a consequence, have been involved in accidents. Between June 1 985 and January 1987, six known accidents involved massive overdoses by the Therac-25 – with resultant deaths and serious injuries. The cause was attributed to the errors in control program, excessive trust in software when designing system, lack of hardware safety measures (interlocks) and lack of appropriate software engineering practices.

The world has become reliant on software-enabled systems and components. In addition, software is now embedded in the cyberspace domain that enables defense military, intelligence, and business operations. DOD systems are constantly under threat from Nation-state, terrorist, criminal, or rogue developer who can exploit vulnerabilities remotely or gains control of systems through supply chain opportunities.

As a result, the DoD is keenly aware of the increasing importance of software and the critical need to achieve software quality," says Paul D. Nielsen, Software Engineering Institute.

The purpose of software assurance is to assure that software products are of sufficiently high quality and operate safely, securely and reliably. The software assurance process is the

planned and systematic set of activities that ensure conformance of software life cycle processes and products to requirements, standards, and procedures. Software assurance assures that the software and its related products meet their specified requirements, conform to standards and regulations, are consistent, complete, correct, safe, secure and as reliable as warranted for the system and operating environment, and satisfying customer needs.

DOD defines Software Assurance as 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. Failures in software assurance can be of particularly high consequence for defense systems due to their growing roles in protecting human lives, in war fighting, and in safeguarding national assets.

Software has become critical for Military Capability

Software is important for the DoD because it promotes lower cost and improved agility in deploying and reconfiguring systems. One result is reflected in the DoD's ability to now program systems that were once fixed-function to meet changing mission needs. Sensor networks, field programmable gate arrays, software-defined networking, software-defined radios, and embedded controllers represent a few of these now-programmable areas.

"Another result is that software enables the interconnectivity that is central to accomplishing system-of-systems configurations. Systems of systems support network-centricity, aiding DoD mission goals for information superiority. "

A third result is that software enables a shift from stovepipe ("platform-centric") systems to modular ("framework and apps") approaches. To exploit the flexibility of modular development,

the DoD continues to explore the use of an open systems architecture approach that will shift development focus more to payloads and less to platforms.

The overwhelmingly large role of software in safety-critical air systems (defense and commercial) provides an appropriate illustration.

The Air Force vision document Global Horizons traces the percentage of capability in air systems reliant on software through generations of aircraft. By the mid-1970s, when theF-16 went into production, software accounted for about 40percent of capability. A generation later, the F-22 relied on software for 80 percent of capability. Software may contribute 90 percent of capability for today's premier fighter, the F-35.In addition, millions of lines of software are required to supportF-35 Lighting II ground functions. Software's critical role in delivering capability is driving commercial aircraft makers toseek a new development paradigm.

Software Quality includes Cybersecurity

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 system, and considered a prime concern as the system is fielded and sustained.

Cybersecurity concerns for software quality must also account for a software supply chain that is diverse and complex—even global. Consider the variety in these supply chains: physical

components, integrated components such as network routers, software, the prime contractor organization, subcontractor organizations, and other supply chains for the commercial products used . Each component might be deemed to have sufficient quality, but the integration of components with different levels of software quality ratchets up cybersecurity—and mission—risk for the system.

The introduction of malware by a supply chain partner also suggests insider threat concerns. Recent high profile incidents such as Edward Snowden's actions and the Target Corporation breach heighten awareness of the threat that insiders (malicious or unintentional) pose from fraud, sabotage, or theft of intellectual property. While Snowden, working as an NSA contractor, appears to have acted intentionally, the theft of credit card information from Target is reported to have resulted from a mistake by an employee at a supplier that had access for electronic billing to the firm's network

"While observed in the useful (easy, safe, reliable) operation of a software-reliant system, software quality is determined by practices, tools, technologies, and methods that result from software engineering research and development," says Paul D. Nielsen.

Eliminating common vulnerabilities during software development can result not only in more secure software but also in a large cost reduction, because less effort will be expended to repair code. Government, industry, and academic cybersecurity researchers are forming and promoting the adoption of international secure coding standards for some common software programming languages, including C, C++,

It is important to prevent errors through adherence to secure coding standards; however, rigorous testing is also advisable. For instance, vulnerabilities may emerge as software components are integrated, in commercial off-the-shelf (COTS)

and custom-developed software, or in patches sent out to eliminate already discovered vulnerabilities. An advanced level of software testing would include full penetration testing by organic or external experts.

Some costly Software errors

A 2.6 billion rouble ($A58 million) Russian weather satellite and nearly 20 micro-satellites from other nations were lost following a failed launch of the Meteor-M from Russia's new cosmodrome in the far east on November 28. And in another blow to the Russian space industry, communications with a Russian-built communications satellite for Angola, the African nation's first space vehicle, were lost following its launch in Dec 2017.

Deputy Prime Minister Dmitry Rogozin, who oversees Russia's military industrial complex and space industries, said in a television interview on Wednesday that the November 28 launch from the new Vostochny launch pad in Russia's far east failed because the rocket had been programmed to blast off from the Russia-leased Baikonur launch pad in Kazakhstan instead of Vostochny. He accused the Russian space agency Roscosmos of "systemic management mistakes", adding the failure had been caused by "human error".

The $500 million Ariane 5 space rocket self-destructed after a fault occurred 40 seconds from launch in 1996. The main cause was inappropriate software reuse; the code was taken from the Ariane 4, without proper analysis.

The Mars space probes Mars Climate Orbiter disintegrated on entry to Mars atmosphere, 1998 and in Mars Polar Lander landing gear prematurely activated on entry to atmosphere also in 1998. The causes in Mars Climate Orbiter were attributed to mismatch between use of anglo-american and metric units, multiple process errors and lack of formal interfaces. The

causes of Polar landed were due to lack of integration testing, shock was interpreted as landing the engines were stopped and lander fell

NASA’s Approach to Software Assurance

NASA performs high-risk functions in the process of achieving its goals and objectives. The Program/Project Manager plans the best risk mitigation strategy for the entire project, of which software is a part. Software assurance is an umbrella risk mitigation strategy for safety and mission assurance of all of NASA’s software.

Software assurance assists in risk mitigation by helping expose potential defects in products and processes, thus preventing problems from evolving. However, it also, through its metrics, tracking and analyses activities, enables improvement of future products and services. Software assurance often serves as the corporate memory from project to project, sharing potential problem areas and lessons learned.

Software assurance reviews and analyzes all processes used to acquire, develop, assure, operate and maintain the software independently; evaluating if those processes are appropriate, sufficient, planned, reviewed, and implemented according to an adequate plan, meeting any required standards, regulations, and quality requirements.

It provides a consistent, uniform basis for defining the requirements for software assurance programs to be applied and maintained throughout the life of that software, that is, from project conception, through acquisition, development, operations and maintenance, and then evaluates if the software is properly retired.

US DOD’s Software assurance

In response to a mandate from Congress, Deputy Secretary of Defense Robert O. Work chartered the Joint Federated Assurance Center (JFAC) as a federation of U.S. Military Department and agency software assurance (SwA) and hardware assurance (HwA) organizations and capabilities.

According to this charter, the JFAC is charged with supporting program offices throughout the life cycle with SwA and HwA expertise, capabilities, policies, guidance, and best practices. The JFAC is responsible for coordinating with DoD organizations and activities that are developing, maintaining, and offering software and hardware vulnerability detection, analysis, and remediation support.

Other roles and responsibilities of the JFAC include:

- Conducting SwA and HwA analyses and assessments in support of defense acquisition, operations and sustainment activities;
- Advocating for the advancement of DoD interests in SwA and HwA research, development, and test and evaluation activities; and
- Building relationships with other communities of interest and practice in SwA and HwA such as other government organizations, academic environments, and private industry.

System Security Engineering (SSE) Software Assurance

DOD’s objective is to establish software assurance as an accepted SE discipline within the Department. The requirement is to use SwA tools and methodology across DoD system life cycle.

- Is a cross-cutting, multi-disciplinary area of interest
- Impacts not only security, but SW development, test, deployment, and operation techniques and practices
- Has tools and techniques that support cyber security, software design, software development techniques and practices, software test, and supply chain risk management
- Is a growing area of importance in industry
- Requires cooperative research, participation, innovation, and engagement

Challenges are:

– Translating systems engineering requirements into SwA contract language

– Identifying effective contract language and verifying results

– Specifying metrics for security risks, vulnerability detection, and validated mitigation

– Training and educating the workforce

– Building efficacy/scalability of tools and techniques

– Integrating SwA capability into engineering disciplines

DESIGN AND DEVELOPMENT PROCESS FOR ASSURED SOFTWARE

In general, the primary reason for software project failure is often not due to the lack of technical expertise by the software development engineers; but rather it is due to poor project estimation, planning and control.

The key to success in estimating software efforts is to establish and maintain detailed historical data on cost, schedule and technical performance.

Data Driven Management and Technical Execution Best Practices

Mature data-driven best software project management and technical engineering practices are required to consistently achieve the goal of delivering high quality, safe, secure, and reliable systems on schedule and within budget. The software project management processes and technical development processes must be documented, institutionalized and enforced. The
software development plan must specify the steps, activities, roles and responsibilities, and required reviews and metrics that are used for both the initial system development (pre-IOC) and sustainment (post-IOC) efforts.

According to Joe Heil, Naval Surface Warfare Center Dahlgren Division (NSWCDD), “At a minimum, each software development organization must collect, maintain, share and report on a frequent, regular and structured basis the quantitative and qualitative information to address all of the critical execution questions listed below:

1. Are the expected system requirements stable and understood?

2. Is the scope and size of the effort understood?

3. Is the activity adequately staffed?

4. Is the activity making the required progress?

5. Is the activity being executed within budget?

6. Is the activity meeting technical performance, assurance, and quality goals?

7. Is the activity formally successfully identifying and

mitigating risks?

8. Is the activity continually improving efficiency and effectiveness?

Continuous improvement requires the software teams to maintain awareness of and apply emergent best practices which include tools, techniques, methods, technologies, etc. For example, a few proven best sw engineering technical practices include:

- User Centered and Model-based system and software engineering.
- Documented traceability between requirements, design, code and test artifacts.
- Multi-Discipline-expert peer reviews of artifacts (specifications, code, tests, etc.).
- Build-a-Little Test-a-Little (Rapid prototyping, Agile development, etc.).
- Automated testing (at CSCI level) and simulators for go/fault/stress testing.
- Tracking defect detection and removal in each development phase.
- Regular causal analysis of defects to improve earlier detection and removal.

Software assurance (quality AND resiliency against cyber vulnerabilities) must be engineered-in throughout all development activities. This entails much more than applying the latest COTS security patches prior to delivery. SW assurance requirements must be defined, the software design must not only defend against cyber intrusions, but also be resilient enough to detect and complete mission critical functions after intrusion; coders must be trained on and apply secure coding techniques; multiple tools must be integrated

into all activities to identify and remove vulnerabilities as early as possible; and all testing phases should include penetration testing.

Formal Risk management

A formal risk and opportunities board and process must be established and executed with discipline. The process must facilitate risks being identified and communicated on a frequent, regular interval and at the appropriate levels of leadership. All risks must always be addressed from the three perspectives of cost, schedule and technical performance impact.

Risks must be formally documented via the standard 5×5 risk cubes; and each risk must have a documented mitigation plan with assigned individual(s) responsible for driving the risk to closure. All status and risk reviews must have an assigned leader and well defined agenda and required participants. The discussions must be supported by objective data (planned vs actual cost and schedules, technical performance, and quality indicators, open versus closed risks over time, etc.) rather than subjective “red, yellow, green stoplight” type indicators.

The common keys to success include utilizing a software system acquisition approach that relies on government software engineers to not just monitor/review industry software efforts, but also perform hands-on architecting, designing, coding, integrating, and testing of a subset of the complex software components for mission critical systems. This teaming approach combined with data-driven project-management and technical execution best practices has been successfully utilized for decades for several mission critical warfare programs and has consistently resulted in the delivery of high

quality, safe, reliable, multi-mission-platform capable and operationally successfully software systems that were developed within cost and schedule constraints, writes Joe Heil.

References and Resources also include:

http://static1.1.sqspcdn.com/static/f/702523/26194068/1430694655517/201505-Nielsen.pdf

http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=274940

http://www.news.com.au/technology/science/space/failed-space-launches-haunt-russia/news-story/cba6b6296ed952cc4b14593a96884ff4

https://www.csiac.org/wp-content/uploads/2017/07/CSIAC_Journal_V5N2_web_opt.pdf