Dr. Paul D. Nielsen

Director and Chief Executive Officer
of the Software Engineering Institute (SEI), Carnegie Mellon

Before the
United States House of Representatives
Defense Acquisition Reform Panel of the Committee on Armed Services

July 9, 2009
8:00 a.m.

Chairman Andrews and Ranking Member Conaway, thank you for the opportunity to participate in this hearing before the Defense Acquisition Reform Panel of the Committee on Armed Services about the challenges facing effective acquisition and management of information technology systems. After serving in the Air Force for 32 years, including my final assignment as the commander of the Air Force Research Laboratory at Wright-Patterson Air Force Base in Ohio and after the past five years as the director of the Software Engineering Institute, I have experienced firsthand the challenges and opportunities that we face in defense acquisition.

Growth in Software Reliance

Today our military men and women train, operate and fight in a cyber environment that is based upon global IT architectures, applications and services. Gone are the days of IT being a support infrastructure to a few basic missions. Today and in the future, IT architectures and services have created and enabled the cyber environment within which a majority of key DOD missions and functions are conducted, for example: Command and Control, Operations, Logistics, Medical, Personnel Management, and Intelligence. Almost everything a soldier, sailor, marine or airmen does has some interaction with IT and software—from recruitment to retirement, from getting paid to getting medical help, from planning operations to executing them. So when the IT acquisition process and programs fail to deliver quality integrated architectures and systems that are reliable, assured, and secure – every major military mission is potentially impacted. And now, to further complicate both acquisition and operations, we all work in a global cyber environment with all the benefits and vulnerabilities of the connectivity enabled by our IT systems. So thank you for making IT acquisition the focus of this hearing.

The Carnegie Mellon Software Engineering Institute (SEI) is a Department of Defense Federally Funded Research and Development Center (FFRDC). Our work is centered on the software component of our systems, the core of their functions and capabilities – often totaling one third or more of the overall cost of an acquisition program. In fact, one of the largest challenges we see facing the Department of Defense is the effective management of the software foundation upon which all military platforms, capabilities and systems of systems are built and run. For operational success these capabilities must all function in real-time or near-real-time and are often networked. Our military can only operate at its peak capability with reliable, high quality, and secure information technology. Software is the heart of our command and control capabilities and services. It's the glue that connects our systems and integrates our systems. It's critical to the warfighter that our software intensive systems perform as flawlessly as possible.

Due to the levels of complexity that now exist in software development, engineering, and security, software is often the least understood and most neglected component of our DOD IT architectures today; and although major DOD systems encompass more than just software, it is the software that fundamentally allows them to function. Especially worrisome is that at present, and for the foreseeable future, both our global combat and peacekeeping engagements rely on software and systems that may contain components produced outside the United States.

The percentage of weapon system functionality that is dependent upon software has sky rocketed and will only continue to grow. Figure 1 shows this growing reliance on software for functionality in military aircraft:

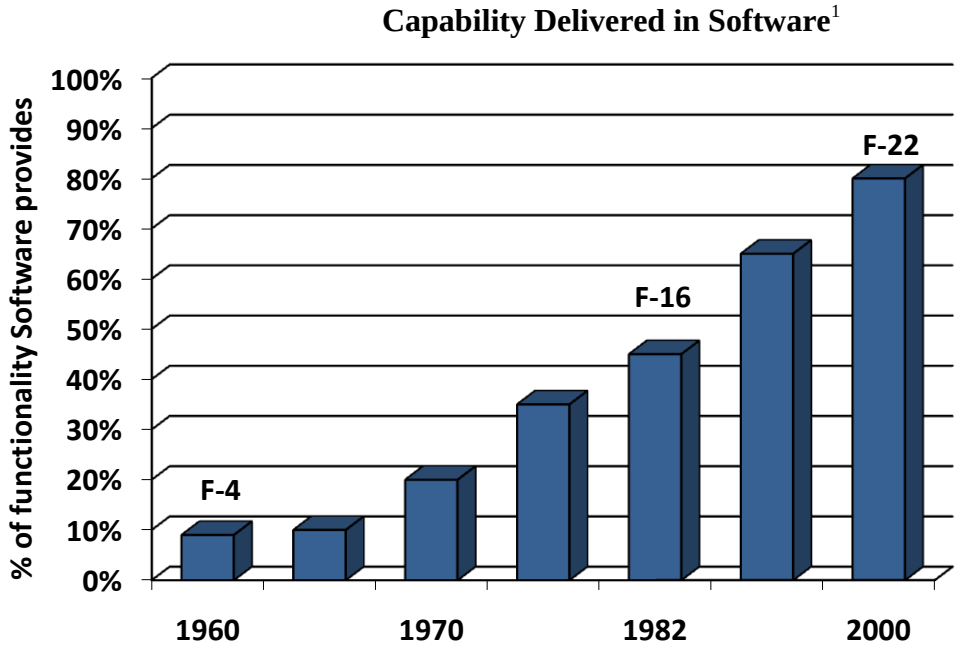

Figure 1

Why is Software Acquisition so Difficult?

The "unlimited" complexity of software is neither well understood nor well appreciated by many. Software is not rooted in the physical world like other engineering disciplines--civil, aeronautical, electrical, or mechanical engineering. Without physical constraints, the design space is so vast for large programs that you need strong architectural principles, disciplined processes, and talented people to be successful. The larger the program, the more important this is—and, as you are aware, many defense programs are very large.

Additionally, software is invisible; people don't buy code - they acquire systems that satisfy requirements. And software is intangible—you can't touch it or kick its tires. Nevertheless, in most defense systems, software is critical to the very success of the program. The systems just don't work without software.

The challenges of acquiring software-reliant systems continue to grow along with their expanding functionality and complexity, but software code is rarely designed and produced entirely within an acquisition program any longer. We use a great deal of commercial off-the-shelf (COTS) software especially in our IT systems: commercial operating systems, data base software, enterprise business solutions, e-mail and office productivity software. The government has gained a lot of from the use of commercially available software. But now, in an era where we worry more about malicious code, we don't always know where this commercial software was written, what support software and hardware was used to create it, and ultimately how trustworthy the code really is.

Without solid software engineering, software issues often don't become evident until late in an acquisition, such as during integration and test, which results in significant program slips. With software and hardware technologies continuing to evolve rapidly, very few program managers — or key decision

[1] Watts S. Humphrey, Winning with Software: An Executive Strategy (Boston: Addison-Wesley 2002), 4

makers — have a deep understanding of software technology, even as demands increase for further integration, interoperability, system-of-systems capabilities, service oriented architectures, and cloud computing. Additionally, mounting expectations for software systems to be connected, configurable, and interoperable add to the challenges of ensuring their security.

In fact, the current challenges we face in software engineering and integration are daunting. In addition to core issues with standard and comprehensive acquisition policy and approaches, we face the development of extremely large systems with several million software lines of code (SLOC) that often utilize insecure and unassured software engineering. Amplifying development complexity is the creation of "hybrid" systems which integrate legacy re-use (a task often misconceived as being "easy"), COTS software, and new unverified technologies. Complications are further augmented when multi-contractor teams are using different software processes, dispersed engineering, and separated development and operational locations (to include ever growing software development in China and India) to complete developments.

Therefore a solid foundation for IT software acquisition includes not only the required technical expertise, management oversight and quality assurance processes but also the adequate budget, schedule, and staff needed to carry them out. Of course this is always a challenge due to pressured timelines and cost schedules. At the SEI we have presented a preliminary framework of activities focused on building security into the government's major software reliant acquisition systems and architectures, spanning the acquisition life cycle from identification of a mission or business need to system delivery. We continue to work to refine this work to ensure our IT architectures and systems are based upon secure software of high quality.

Ten Key Reasons Software Acquisitions Fail

The SEI has almost 25 years of experience working with DOD and other government acquisition programs managing developments that include persistently growing amounts of increasingly complex software code. This experience provides the basis upon which we compiled this list of Ten Key Reasons Software Acquisitions, and systems which are software dependent, fail to meet their goals on time and/or on budget.

1. Technology key to program success is new to the organization
2. Software issues are considered too late in the system-development process
3. Inadequate planning and estimating
4. Size matters – large projects get into trouble more frequently than smaller ones, all projects grow larger over time
5. Software objectives are not fully understood or specified; they change frequently (and grow) during the project
6. Inadequate experiences and trained project management
7. Inadequate process emphasis and erosion of process discipline
8. Inadequate contract incentives to encourage use of proven software engineering practices
9. Acquirers and developers lack experience working as a team or the resources to do so
10. Insufficient senior staff and inexperienced software engineering cadre

How to Better Manage Large Software Efforts

Increasing focus on the following six steps would, in my opinion, significantly assist efforts to successfully manage the development of large software systems and software intensive systems.

1. Incorporate Software Early and Continuously in the Systems Engineering Process

Recognize that software represents a major risk and is a critical technology that needs to be watched like other technologies. It is vital to the success of a program that software engineering efforts start early as a key part of the systems engineering process. This means including software in early CONOPS, architecture, and requirements activities, (e.g., concept/trade studies). Competing demands on the budget between early adoption of good software engineering practices and "producing" something tangible can be alleviated by obtaining independent software cost and schedule estimates and maintaining an executable software cost, schedule, and requirements baseline. Software security and resiliency engineering must also be included early in development. Too often, software security receives insufficient attention in critical, early life cycle activities, leaving systems open to attack and putting the warfighter directly in harm's way when systems are deployed.

One of the most troubling aspects of software development is that typically more than half of the development cycle is devoted to fixing errors, often not discovered until the system test. The cost to correct software errors – in both time and money – incrementally increases as the development stage advances. The rising costs of fixing defects late in the development cycle – or worse, after deployment to our warfighters – indicates the need for more effective development methods that avoid injecting errors or find them much earlier. The solution is to build quality in with effective software engineering—and do it from the beginning of the program.

2. Recruit and Develop a Trained Software Staff within the Program Office

To ensure personnel requirements, both in quality and quantity, can be met with acceptable staff, use professionals from the military, civil service, and Federally Funded Research and Development Centers (FFRDC) to help train, monitor and work with developmental contractors on architecture, design, and other issues. If possible, aim for a "critical mass" so software concerns have sufficient voice; ensuring personnel requirements are clearly identified, including stipulations that all software players should have the software architecture embedded in their thinking. To reduce software integration risk, verify that the other domain experts, especially systems engineering and management, have at least a "reading level" understanding of software engineering and establish a common understanding of crucial software concepts across the program.

Designate leadership – programs should have a chief software engineer, a chief software architect (one person cannot really do both) and a software security engineer. All three positions must be provided with the appropriate responsibility and authority. Their roles are as follows:

- Chief Software Engineer – day-to-day technical management: a project engineer devoted to software. (Analog : general contractor for a building)
- Chief Architect – responsible for formulating vision for form, function, and usage of software in system. (Analog: architect for an office building)
- Designated security engineering function- led by an experienced systems/software security engineer, within the program office. Someone to continuously identify threats and vulnerabilities in the emerging operational environment and solution space

All science and engineering disciplines need men and women who are active and up to date in their field. It's very important that government software professionals attend professional education classes and conferences. These events, most three to five days long, facilitate new ideas and approaches while helping these professionals improve and sustain their skills; but too often there are no funds for these activities.

3. Ensure Adequate Emphasis on Software Architecture

In recent studies by OSD, the National Research Council, NASA, and the NDIA, architectural issues were identified as a systemic cause of software problems in DOD systems.

Poorly designed software architectures result in greatly inflated integration and test costs and an inability to sustain systems in a timely and affordable manner. Furthermore, without the right architecture, systems will not only lack robustness but will likely exhibit undesired, disparate behaviors at the system and system-of-systems levels. No one would attempt to build a home or a major office building without an architect and without several architectural views, but time and again not enough time and effort are spent on the software and system architectures of our defense programs.

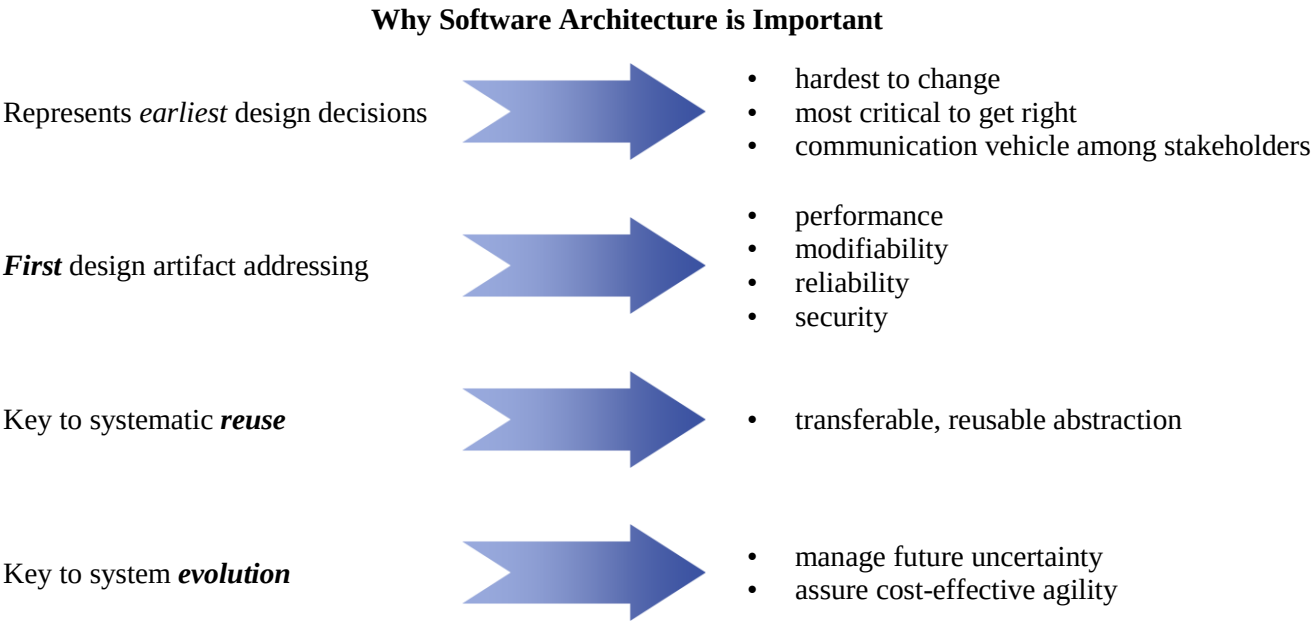

Figure 2

Software architecture drives software development throughout the life cycle and therefore must be central to software development activities. To design a software architecture that satisfies constraints, meets functional requirements, and fulfills key quality attributes, it is imperative that the appropriate stakeholders are involved throughout to ensure both mission and business goals are used to explicitly identify and characterize fundamental features. Assuring adequate consideration is given to all of the quality attributes including safety and security, will result in a more robust and resilient system.

4. Software Security Must be a Main Concern

Many systems are designed and developed with little initial concern for the software's security properties, directly increasing the danger and exposure for users. Security is seen as a patch not a design feature. This situation is made even worse by the trend that more software development is being outsourced off-shore. Commercially available software is often delivered with unintentional security vulnerabilities due to defects and, occasionally, with hidden malware—purposely designed vulnerabilities.

This is due, in part, to the lack of emphasis on security as an integral part of software in the acquisition process, including the lack of accountability for acquisition officials to verify software assurance. The rampant worldwide increase in exploitation of software vulnerabilities demands that development practices minimize software errors and other vulnerabilities that give our adversaries an open door to exfiltrate data and to deny or degrade services.

Today's software engineering practices permit dangerous error, whether they are introduced by mistake, poor practices, or with the intent to cause harm; enabling hundreds of attack programs to compromise millions of computers every year. Software is an underpinning of all critical infrastructures, including public systems such as health IT, energy, and banking and finance. Exacerbating the problem are the advanced and persistent cyber threats that are increasingly targeting our defense industrial base.

As our adversaries—both foreign and domestic—become increasingly sophisticated in their ability to insert malicious code into critical software we must work to minimize their ability to introduce and exploit vulnerabilities. [2]

5. Establish an Effective Process

"*History has shown a direct relationship between the effectiveness of the processes used to manage a project and how well that project meets its cost, schedule, and performance objectives—projects with strong processes have a far greater probability of meeting their objectives than projects that have weak processes. These disciplined processes are based on the best practices identified by the Software Engineering Institute (SEI) … and other experts, which have been proven to reduce the risk in implementing systems."* — General Accounting Office[3]

A well established process can help programs support the goals of the military by enabling repeatability, insight and oversight, control and tracking, measurement, improvement, training and transformation (via consistency, integration, coordination). Using the right processes to manage and develop software can dramatically reduce the risk of acquiring software-intensive systems. For it is just as true for software, as it is in many other high-tech industries, that measured and managed processes, designed and tailored to support the development of a system, provide the best possible performance for a given workforce and technology. Our research has shown that the right process improves the predictability of development cost and schedule, as well as the quality and reliability of delivered systems. It reduces time to field, lowers development and ownership costs, and increases the probability of mission success by providing early warning of issues while there is still time to act. Both the acquirer's and the supplier's ability to achieve a successful outcome are improved, and the greatest positive impact is realized when best practices are applied across all the engineering domains, including acquisition, in every stage of development.

For success, action plans should be established early to increase efficiency and effectiveness of program processes as well as produce measured progress. Interactions between multiple government and contractor stakeholders should be considered as high risk areas. The key is not so much about having the "best process," it's about having the *right* process to deliver the right capability to the warfighter when they need it.

---

[2] The Department of Defense (DOD) and Department of Homeland Security (DHS) Software Assurance (SwA) Acquisition Working Group, *Software Assurance in Acquisition: Mitigating Risks to the Enterprise, A Reference Guide for Security-Enhanced Software Acquisition and Outsourcing*, October 22, 2008

[3] GAO-07-1157R DHS Posthearing Questions

Benefits of Process

We continuously study and evaluate the improvements organizations have seen when they focus on disciplined development so that we may better understand and advance software engineering best practice. We have provided the committee some of the resultant reports showing the benefits, but summarizing from these reports (Figure 3), we know that dramatic improvements in cost, schedule, predictability, and the quality of delivered software can and have been achieved, and that the return on investment (Figure 4) for these results is rapid and significant.[4]

Measured Benefits Utilizing Process

Benefit Measured[5]	Median Improvement
Cost	38%
Schedule	50%
Quality	50%
Customer Satisfaction	14%
Return on Investment	3:1

Figure 3

Process Improvement Pay-Off [6]:

Data Source	Process Improvement	ROI/Benefit Conclusions
Raytheon study of multi-site internal improvement effort	Implementing SW CMM and migrating to CMMI	• 36% drop in Cost-Performance Index variability • 70% drop in Schedule Performance Index variability • Continuous productivity gains: 1997-1999 - 30%; 2000 - 9%; 2001 - 11%; 2002 - 6%
US Air Force F-16 Logistics Operations Division - AFMC	Processes for defect analysis and removal to reduce Operations and Sustainment (O&S) costs	• $43 M in documented sustainment cost savings (2002) • $900 M projected savings for the life of F-16 program
DACS (on behalf of DOD)	Review of commercial and government process improvement results and ROI for measured defect identification and removal, reduced rework, increased productivity, decreased cycle time, etc	• 54% Fewer validation cycles (Schlumberger) • 100% Productivity Increase (Schlumberger) • 51% Increase in on-time delivery (Schlumberger) • 25% Post-release defects ((Schlumberger) • Most defects eliminated before testing (Boeing STS) • 31% Reduced rework ─ 7.75:1 ROI (Boeing STS) • 55% Reduced software development costs (NASA SEL) • 40% Decreased cycle time (NASA SEL)

Figure 4

[4] JWO suggestions for this list include CMU/SEI-2003-TR-009, CMU/SEI-2003-TR-014, CMU/SEI-2006-TR-004, CMU/SEI-2007-TR-013
[5] Source for this table is CMU/SEI-2006-TR-004
[6] Source - Benefits of Improvement Efforts, CMU/SEI-2004-SR-010, Oct 2004

6. Continued Software Engineering, Research and Development

Software engineering is a relatively new discipline, subject to ongoing and immediate advances in capability, constantly changing to support the development of larger and more complicated systems, guaranteeing that what is "effective" or "right" today will not be appropriate in the future. Therefore, it is essential for the DOD to continue to work to establish and incentivize the adoption of the most effective practices and processes available.

In an endlessly evolving world it is crucial to US security and future operations that we fund the necessary research to meet the increasing demands of the warfighter and at the same time stay ahead of our adversaries, whose capabilities and tradecraft transform daily. The commercial world will continue to invest in technologies and innovations for which they can see a business case, but they will not always have a complete overlap on military requirements. The military often works in very hostile and remote environments, lives depend on its systems, and the nation's success in meeting its goals and objectives are enabled by its systems. It is very important that the DOD invest in research in software engineering with an eye on its unique requirements.

In conclusion, none of these techniques are "silver bullets." Acquisition is a "team sport" and the development of complex systems of systems demands an acquisition team with a capable acquisition manager teamed with capable development managers. The team includes the senior acquisition officials in the Pentagon—and the appropriate congressional committees. We all have a responsibility to the soldiers, sailors, marines and airmen to support them with affordable, secure, operable systems that meet the nation's needs.

I would like again to thank the Committee for providing me the opportunity to testify on this topic of critical importance to our Nation and the Department of Defense.