SYM-AM-15-089

PROCEEDINGS
OF THE
TWELFTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM

THURSDAY SESSIONS
VOLUME II

DoD Software-Intenstive Systems Development: A Hit and Miss Process

Brad Naegle, NPS

Published April 30, 2015

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
30 APR 2015		**00-00-2015 to 00-00-2015**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
DoD Software-Intenstive Systems Development: A Hit and Miss Process	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Graduate School of Business & Public Policy,555 Dyer Rd,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**22**	

The research presented in this report was supported by the Acquisition Research Program of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

DoD Software-Intensive Systems Development: A Hit and Miss Process

Brad Naegle, LTC, U.S. Army (Ret.)—is a senior lecturer and academic associate for Program Management Curricula, Graduate School of Business and Public Policy, Naval Postgraduate School. While on active duty, LTC (Ret.) Naegle was assigned as the product manager for the 2 1/2-ton Extended Service Program (ESP) from 1994-1996 and served as the deputy project manager for Light Tactical Vehicles from 1996-1997. He was the 7th Infantry Division (Light) division materiel officer from 1990-1993 and the 34th support group director of security, plans, and operations from 1986-1987. Prior to that, LTC (Ret.) Naegle held positions in test and evaluations and in logistics fields. He earned a master's degree in systems acquisition management (with distinction) from the Naval Postgraduate School and an undergraduate degree from Weber State University in economics. He is a graduate of the Command and General Staff College, Combined Arms and Services Staff School, and Ordnance Corps Advanced and Basic Courses. [bnaegle@nps.edu]

Introduction

> From remotely piloted aircraft and smart bombs to autonomous vehicles and advanced fighter jets, software is crucial to the success of today's weapon systems. Focusing solely on developing and maintaining military hardware is no longer an option. With shrinking defense budgets and increasingly complex systems, the defense industry and services must fight to deliver on this ambitious objective, the military must drastically transform its approach to software. New organizational structures, operating models, and tools will be essential to modernizing and sustaining the U.S. weapon systems. (Hagen, Hurt, & Sorenson, 2013, p. 31)

Although the Department of Defense (DoD) has developed some very successful software-intensive systems, such as the Aegis, Tomahawk Missile, and F/A-18 Hornet, we continue to struggle with successfully developing like systems. The software development in the F-35 Joint Strike Fighter (JSF) continues to be problematic. The GAO (2012) stated that

> JSF software development is one of the largest and most complex projects in DoD history, providing essential capability, but software has grown in size and complexity, and is taking longer to complete than expected. Developing, testing, and integrating software, mission systems, and logistics systems are critical for demonstrating the operational effectiveness and suitability of a fully integrated, capable aircraft and pose significant technical risks moving forward. (p. 7)

The report went on to state, "This program [JSF] has modified the software development and integration schedule several times, in each instance lengthening the time needed to complete work" (GAO, 2012, p. 11). The results of the software development problems have contributed to a two-year delay and increased costs of about one billion dollars.

When software-intensive systems encounter developmental problems, it is easy to see the symptoms: schedule overruns, acquisition cost overruns, systems delivered with less capability than desired, and unaffordable software sustainment costs. The actual causes of the visible symptoms are often much more difficult to determine.

Cost and schedule overruns in software development are often the result of poor initial software size estimates and unforeseen software redesign. In the case of the JSF,

> The lines of code necessary for the JSF's capabilities have now grown to over 24 million—9.5 million on board the aircraft. By comparison, JSF has about 3 times more on-board software lines of code than the F-22A Raptor and 6 times more than the F/A-18 E/F Super Hornet. This has added work and increased the overall complexity of the effort. The software on-board the aircraft and needed for operations has grown 37 percent since the critical design review in 2005 … almost half of the on-board software has yet to complete integration and test—typically the most challenging phase of software development. (GAO, 2012, p. 11)

The report goes on to state that typical software size growth in DoD systems development ranges from 30% to 100%.

JSF design changes were originally supposed to taper off and be completed by January 2014. Actual design changes through September 2011 failed to taper off and continue at a significantly high rate. The projections in the GAO (2012) report indicated that the revised design change projections will continue, and actually grow in number, until January 2019 (p. 16). Given this level of redesign, the software and system complexity growth are likely to continue.

The DoD Software-Intensive System Development Problem and Research Technique

Problem

From a systems management perspective, the overarching problem is that the DoD Acquisition Management System produces both successful and unsuccessful software-intensive systems. The management oversight, structure, and discipline offered do not produce repeatable success in complex, software-intensive systems development.

Primary Research Question

The problem previously identified drives this primary research question: Why does the DoD Acquisition Management System produce both successful and unsuccessful software-intensive systems?

Secondary Research Questions

I analyze the DoD software-intensive system development challenge by addressing these secondary research questions:

- Does the DoD acquisition environment provide opportunity for variable results in software-intensive system development?
- How does the software engineering environment impact DoD software intensive system development?
- Is the DoD requirements development and communication process sufficient for potential software developers?
- How is the software-intensive system architecture developed to ensure warfighter capabilities are designed and prioritized?

DoD Acquisition Environment

At the top level, there are the three primary decision support systems used within the DoD, and the interaction within these systems significantly decides the acquisition of products or services (DoD, 2013b). The three systems are the Joint Capabilities Integration and Development System (JCIDS), which provides the acquisition requirements documents;

the Defense Acquisition System (DAS), which provides the processes to develop and acquire the needed products to fulfill the requirement; and the Planning, Programming, Budgeting, and Execution (PPBE) process, which is the funding resource management.

Software-intensive systems are most impacted by the JCIDS and the DAS Decision Support Systems, and the PPBE process has no particularly unique impact on software intensive systems development. This research, therefore, focuses on elements of the JCIDS and DAS systems.

Requirements Generation

The Joint Capabilities Integration and Development System (JCIDS) was designed to assess capability requirements and associated capability gaps and risks (CJCS, 2012, p. A-1). Capability gaps may be identified in one or more of the following areas: Doctrine, Organization, Training, Materiel, Leadership Policy and Education, Personnel, Facilities, and Policy (DOTMLPF-P). Materiel-related capability gaps become the basis for the requirements process that drives the acquisition community to develop and acquire platforms designed to bridge all or part of the identified gap. JCIDS is designed to be an iterative process, beginning with a validated Initial capabilities Document (ICD), triggering the acquisition community to begin an Analysis of Alternatives (AoA) on candidate systems that potentially address the capability need. The Capabilities Design Document (CDD) refines and adds necessary detail to support the technical design of the system sought. The final document in the series is the Capabilities Production Document (CPD), which further refines the user requirements and adds detail supporting the production planning for the system. Although JCIDS is designed to refine well-defined requirements, there is clearly an opportunity for requirements creep with this iterative user requirements process.

After the user community completes each JCIDS iteration, the program/project/product manager (PM) or materiel developer is prompted into action. As stated, the ICD prompts an AoA identifying the possible systems that could be procured or developed to meet the capability need. The CDD is a key document in the requirements generation cycle and is the user community's primary input for the PM's development of the performance specification for the Request for Proposal (RFP). The CPD is the user's key document for driving production decisions, and the PM's production strategy is significantly influenced by the CPD.

One of the PM's most critical functions is developing the performance specification for inclusion in the RFP. This requires the PM team to translate the user-stated needs from *capabilities*-based language to performance-based language that is used to drive the design efforts of potential system developers, usually contractors. This is critical because the RFP is the basis for the potential contractors' proposals containing the estimated cost, schedule, and technical performance they plan to achieve. The submitted proposals are evaluated and compared during the labor-intensive source selection process, resulting in a contract award based on proposal merit. If the performance specification is incomplete, vaguely stated, or misunderstood, then the source selection process and contract award is based on incorrect proposals and the effort is significantly wasted.

The selected contractor accepts the terms of the contract based on the assumptions and estimates contained in the proposal. To develop the proposal, the contractor translates the PM's performance specification into a basic detailed specification so that the scope of work can be estimated for the proposed cost and schedule. Correcting these performance specification deficiencies later puts the government at a significant disadvantage, as the contract has been awarded and necessary changes to the contract are negotiated without competition. Changes, additions, or even clarifications to the performance specification after

contract award are likely to impact the terms of the contract, resulting in a negative impact to the cost, schedule, or performance of the desired system.

The Defense Acquisition System

The DoD Acquisition, Technology, and Logistics Life Cycle Management System is the framework for control and management of DoD systems development, based on the SEP. The model features development phases that define activities, and milestones that serve as control and decision points. These phases and milestones are established very early in the development cycle using the information available during early Materiel Solution Analysis (MSA), which is obviously very limited. Overwhelmingly, the PM responsible for establishing this strategy is not the individual responsible for executing it. Funding requirements, including amount, type, and *period of execution*, are established in the Program Objective Memorandum (POM), submission and a congressionally-approved funding profile is established for the entire acquisition strategy within the PPBE process. At this point, the schedule becomes very rigid as Congress must approve significant changes to the funding profile, including when the funding is to be executed. Although there are obviously known and unknown risks associated with an acquisition strategy formulated this early, there is no provision for a management reserve of funding to address these risks.

The Interim DoD Instruction 5000.02, dated November 26, 2013, shows alternate versions of the DAS phases and milestones (see Figure 3) that attempt to address the impact that software imparts on the development process. The interim instruction depicts the following variants of the model: Defense Unique Software Intensive Program; Incrementally Fielded Software Intensive Program; Hybrid Program A (Hardware dominant); and Hybrid Program B (Software Dominant) models (DoD, 2013b, pp. 10–14). The new models indicate an understanding that software impacts the system development process differently than typical hardware systems do. As these are all newly developed, their impact on future development is unknown.

Performance Specifications and the Work Breakdown Structure

Since the implementation of acquisition reform in the nineties, detailed specifications have been replaced with performance specifications in order to leverage the considerable experience and expertise available in the defense contractor base. In most hardware-centric engineering disciplines, the expertise that the DoD seeks to leverage includes a mature engineering environment in which materials, standards, tools, techniques, and processes are widely accepted and implemented by industry leaders. This engineering maturity helps to account for derived and implied requirements not explicitly stated in the performance specification. Three levels of the work breakdown structure (WBS) may provide sufficient detail for vendors to develop a desired system in a mature engineering environment, such as the automotive field. For example, an automotive design that provides for easy replacement of wear-out items such as tires, filters, belts, and batteries obviously provides sustainability performance that is absolutely required. Most performance specifications do not explicitly address this capability as they would be automatically considered by any competent provider within the mature automotive engineering environment.

The *Department of Defense Handbook: Work Breakdown Structures for Defense Materiel Items* (MIL-HDBK-881A), recommends a minimum of three levels be developed before handoff to a contractor (DoD, 2005). If a program is expected to be high cost or high risk, it is critical to define the system at a lower level of the WBS (DoD, 2005, p. 3). Complex weapon systems are nearly always high cost, and the complex software development that these systems require almost always means that the development effort is high risk as well. The WBS and performance specification must, consequently, be significantly more

developed to provide the software engineer enough information and insight to accurately estimate the level of effort needed—cost and schedule—and to actually produce the capabilities needed by the warfighter. Contracts resulting from proposals that are based on underdeveloped, vague, or missing requirements typically result in catastrophic cost and schedule growth as the true demands of the software development effort are discovered only after contract award.

Technology Readiness Assessment and Risk Management

Another important management aspect is addressing the readiness of the key technologies for successful development and deployment. A Technology Readiness Assessment (TRA) is required for most Major DoD Acquisition Programs (MDAPs; DoD, 2011, p. 1-1) The purpose for conducting a TRA is to address the risk of attempting to develop a system with a key technology that is too immature to successfully deploy the system when needed by the warfighter. To benchmark the assessment, Technology Readiness Levels (TRLs) have been developed in a nine-level model, with a goal of ensuring that a system's key technologies achieve at least a TRL level 6 to reduce risk to an acceptable level.

There are software TRLs established, and level 6 is defined as "Module and/or subsystem validation in a relevant end-to-end environment." The level 6 description specifies the "level at which the engineering feasibility of a software technology is demonstrated. This level extends the laboratory prototype implementations on full-scale realistic problems in which the software technology is partially integrated with existing hardware/software systems" (Blanchette, Albert, & Garcia-Miller, 2010, p. 35).

The software TRL level 6 description presents several problems in performing the TRA on a software-intensive system. Weapon system software is typically engineered from scratch with few reused elements, which means that there is very little-to-nothing on which to perform the assessment. There will likely be software developed for similar systems that would meet the level 6 description, but assessing like-software built for another system will not significantly reduce the software technology risk of the proposed system. For example, the F-35 is built by the same manufacturer as the F-22, and they are both high-performance military aircraft with different but overlapping missions. Yet the F-35 is experiencing more software development problems than its predecessor and already has three times more software than the F-22 (Hagen et al., 2013, p. 26).

Software TRLs do not appear to be providing the same type readiness indicator as hardware-related TRLs, leaving software technology risks substantially unknown. In a 2010 U.S. Army workshop report from the Software Engineering Institute (SEI), the participants noted that "though marginally useful, these efforts have only confirmed for the participants the futility of continuing to base [technology] readiness decisions for software aspects of systems on the DoD software TRLs" (Blanchette et al., 2010, p. 2). The software TRLs clearly do not seem to be effective at reducing risk for the TRA.

To help with early risk management in lieu of effective software TRLs, a software developer maturity assessment is mandated for most software-intensive systems, through attaining level 3 in the SEI's Capability Maturity Model Integrated (CMMI), or equivalent, assessment methodology (DoD, 2013a, p. 92) The concept recognizes that the software build is a product of the process, and more mature organizations—those with successful past performance, demonstrated engineering discipline, stable development staffs, and effective management structures—reduce system development risk.

SEI also has the Software Acquisition Capability Maturity Model® (SA-CMM), which is designed to evaluate the maturity of software acquiring organizations such as the DoD's

software-intensive system PM offices (Cooper & Fisher, 2002). The SA-CMM is also a five-level model, similar to the CMMI. The DoD currently has no requirement for PM offices to undergo an evaluation or achieve any SA-CMM level, but the maturity of the team responsible for communicating the system requirements and managing the development has an impact on risk.

Findings Summary

In summary, the DoD acquisition environment features a requirements flow-down process that involves user-stated capabilities-based requirements translated to performance-based requirements, then translated to the detailed design specifications. This requirements translation process is the basis for the resource-intensive source selection and binding contracting processes, which are critical for accurate cost and schedule estimates. Although DoD acquisition is based on the event-driven SEP, the schedule becomes rigid very early in the process when time-specific funding is attached. The subsequent system PMs are charged with managing the cost, schedule, and performance set by the initial PM with no funding provided for managing the associated risk. To reduce risk, PMs are directed to perform TRAs early in the process, with a goal of achieving at least TRL 6 on key technologies. Software TRLs do not appear to be effective, and software developer maturity assessments are conducted to help reduce system development risk. The latest Interim DoD Instruction 5000.02 (DoD, 2013a) depicts newer phases and milestone models that attempt to address the differences that software development causes in the management of the DAS.

DoD Acquisition Environment Analysis

Does the DoD acquisition environment provide opportunity for variable results in software-intensive system development?

The DoD acquisition environment appears to remain vulnerable to significant variability when developing software-intensive systems, similar to the problems currently plaguing the F-35 JSF program. Although the new phases and milestones models address the software component development, other critical management functions remain unchanged. Requirements generation, performance specification development, RFP, source selection, and contracting processes have yet to adapt to the unique challenges presented when managing software-intensive system development. Early program risk management assesses key technology readiness, but the software TRLs are ineffective for predicting software development risk. Evaluating the software developer's maturity helps reduce some risk but fails to include the critical DoD entities in any maturity assessment.

Early risk management through the TRA and achieving a desired TRL is ineffective for the software component. Assessing the contractor (software developer) maturity through CMMI or equivalent evaluation appears to be effective in reducing the developer risk but does not address the DoD acquisition community maturity. As the software developer is significantly dependent on the government's ability to effectively generate and clearly communicate a comprehensive set of requirements, quality attributes, and critical design elements, assessing just the developer's maturity addresses only part of the risk.

Software Engineering Environment

Software Engineering

The software engineering environment is not mature, especially when compared to hardware-centric engineering environments. Dr. Philippe Kruchten (2005) of the University of British Columbia remarked, "We haven't found the fundamental laws of software that would play the role that the fundamental laws of physics play for other engineering

disciplines" (p. 17). Software engineering is significantly unbounded because there are no physical laws that help define environments. There is significant evidence for software engineering immaturity, and it is nearly impossible to find widely accepted, industry-wide development standards, protocols, architectures, or formats. There is no dominant programming language, design and development process, standard architectures, or software engineering tools, which means that reusable modules and components rapidly become obsolete. All of these combine to make it nearly impossible to institute a widely accepted software reuse repository. Without significant software architecture and code reuse in developing software-intensive weapon systems, each development process essentially starts from scratch. This fact is one of the main reasons that the TRA and the software TRLs are ineffective in predicting software development risk (Naegle & Petross, 2007).

The software engineering state-of-the-practice currently is wholly dependent on the requirements that are passed to the software development team. From the requirements, a software architecture is designed, and the requirements "flow down" through that architecture to the individual modules and computer software units that are to be constructed. The software build focuses on the requirements that flowed down to that level and the integration required for functionality. The standards, protocols, formats, languages, and tools used for the build will likely be unique to the contractor developing the software, and will most certainly not be universally accepted or recognized across the software industry.

The software architectural design is the basis for all of the current and future system performance that the system will achieve, and the current state-of-the-practice in software engineering has each project design a unique architecture. Like hardware, the software design will significantly impact system attributes that are important to the warfighter, including maintainability, upgradability, interoperability, reliability, safety, and security. Most hardware-oriented engineering environments address these critical areas through widely accepted industry standards.

Findings Summary

With software, virtually all of the performance and quality attributes developed come directly from the requirements received, and the immature software engineering environment will likely not compensate for any desired performance, such as system sustainability, that is not clearly specified in a requirement. Unlike hardware-oriented engineering environments, where the widely accepted industry standards will be employed whether or not they are specified, with software, you get what you specify and very little else.

The software architectural designs suffer from the immature engineering environment as well. Each software design is unique and driven by the requirements received with no industry-standard architectures available. All current and future system attributes impacted by the architecture must be communicated to the software design staff to ensure they are considered in the design process.

Software Engineering Environment Analysis

How does the software engineering environment impact DoD software-intensive system development?

As illustrated in the previous section, the lack of software engineering maturity impacts both requirements development and design of the architecture. To compensate for the relative immaturity of the software engineering environment, the DoD must conduct

significantly more in-depth requirements analysis and provide potential software developers detailed performance specifications in all areas of software performance and sustainability. This is a significantly different mind-set than the hardware-dominated systems acquisition of the past.

In addition to the performance requirements, software architectures must be similarly shaped to include system attributes expected by the warfighter. Many DoD user representatives and acquisition professionals have grown accustomed to the engineering maturity levels offered by the hardware-oriented systems that dominated past acquisitions. Providing the system requirements in the same fashion may not drive the architecture for needed attributes. As demonstrated by the F-35 JSF redesign problems, changing software architectures during the development cycle will likely be costly in terms of schedule and funding.

DoD Acquisition Environment: Impact on Software Development and Quality Attributes

DoD Requirements Generation Process

The DoD requirements generation process was described earlier as part of the DoD acquisition environment and consists of three major processes: user-generated requirements in the form of *capability needs* using the JCIDS; PM-generated requirements in the form of *performance specifications;* and finally, contractor-generated *detailed specifications*, developed generally in that order. Two major requirements language interpretations are required to get from the warfighters' needs to the system built to meet those needs, leaving significant opportunity for misinterpretation, omission, and misunderstanding of weakly articulated and vaguely stated language. To do this effectively, the PM must accurately interpret user capability language (for example, warfighter requires the capability to … in all mission environments) and translate that into performance language (for example, system shall achieve xxx performance … in these specific conditions). The contractor then translates the performance language into the system build-details that meet or exceed the performance specified.

The importance of system software requirements development to the potential success of software-intensive systems development cannot be overstated. Underdeveloped, vaguely articulated, ill-defined software requirements elicitation has been linked to poor cost and schedule estimations, resulting in disastrous cost and schedule overruns such as what the F-35 JSF is currently experiencing. In addition, the resulting products have been lacking important functionality, are unreliable, and have been costly and difficult to effectively sustain (Naegle, 2006).

Systems Engineering Process

Using the SEP approach, the explicit user capabilities requirements specified in the Joint Capabilities Integration and Development System (JCIDS) provides the input for system requirements analyses. These analyses are intended to illuminate all system-stated, -derived, and -implied requirements and quality attributes necessary to achieve the capabilities needed by the warfighter. The WBS is a methodology for defining ever-increasing levels of performance specificity using the SEP to guide the development of each successive layer (DoD, 2005, pp. 1–5).

> Just as it supports hardware development, the Systems Engineering Process (SEP) is essential in the development of software design. In software development, good quality and predictable results are paramount goals in creating the specified warfighter capabilities within cost and schedule

> constraints. To accomplish those goals, we examine the methods, tools and processes the software developer uses in building the software with the intent of attaining a product that provides all of the necessary functionality and is supportable, efficient, reliable and easy to upgrade. (Naegle & Petross, 2007, pp. 14, 15)

Software Engineering Maturity Impact on Requirements Generation

The immature software engineering environment, discussed earlier, can be compensated for only by a requirements generation system that does not leave any gaps in performance or quality attributes needed. Having all of the requirements clearly communicated is critical, but the software engineer must also understand the requirements in context. Both essential and enhancing features are communicated to the system and software developers as requirements and, as such, appear to have equal weight. The critical difference between "essential" and "enhancing" may not be clear to the software development team, which may result in a poorly performing and possibly dangerous design. The distinction needs to be made clear, but there is no definitive method for identifying requirements as system "essential" or "enhancing."

System Operational Context

To gain some insight into the operational environments that the system is expected to operate within, the DoD provides an Operational Mode Summary/Mission Profile (OMS/MP). The OMS/MP provides some basic insight into the operational profile, threat profile, environmental profile, and the terrain/sea state/undersea/air environment profile, which adds some context to the requirements, but is not usually scenario based. It typically lacks sustainability activities, interoperability profiles, system life-cycle profiles, planned or anticipated upgrades, or operation in stressful, degraded, or emergency situations. There is no prioritization of the operational modes or configurations, nor identification of critical and non-critical systems.

The software development team would likely continue to be missing important information that it needs to adequately design the software and to predict the funding and schedule resources necessary to build the software the warfighter expects.

The OMS/MP documents do not typically provide any information regarding system life-cycle changes such as pre-planned product improvement (P3I) programs, planned upgrades and technology refreshments, future interoperability requirements, or plans for future integration into tactical and logistical networks. These life-cycle events, while known or anticipated, are not effectively communicated to potential developers for inclusion in the proposal process and are often omitted from the software system design.

Impact on Software and Quality Attributes Analysis

Is the DoD requirements development and communication process sufficient for potential software developers?

The DoD requirements generation process that was purposefully designed to garner the maximum contractor innovation and flexibility appears to provide too little information for the software developer to adequately predict the resources necessary to develop the system software. It is clear that the current state of the software engineering environment is mostly incapable of compensating for missing, vaguely stated, or weakly articulated requirements. At the same time, the current DoD requirements generation system provides ample opportunity to inadvertently omit requirements and to provide vaguely stated or weakly articulated requirements through the capabilities-oriented JCIDS documents and the performance-based specifications derived from them.

Without fully understanding the requirements in a detailed operational context, the software design and development effort and resources remain significantly unknown. The typical OMS/MP provides some operational context to the requirements, but is not sufficiently detailed to provide the design drivers needed by the software engineers. Developing a proposal with this limited information will likely result in a significantly underestimated software development effort. After contract award, more operational details are typically provided through program and design reviews, and the cost and schedule for the software effort are likely to inflate significantly to accommodate the new understanding of the requirement in a non-competitive environment.

The lack of operational context typically provided by the government during the RFP process appears to have significant negative impacts on the software design for reliability and maintainability. The OMS/MP documents' lack of information regarding significant planned and anticipated life-cycle changes, system sustainment activities and burden, and operations under unusual conditions will likely mean that the system software design will not easily accommodate known changes. There is no prioritization of the operational modes or configurations that would impact system design considerations. This information would also help differentiate critical systems from enhancing (non-critical) systems, providing a priority in the software design effort.

Software-Intensive System Architecture Development Analysis

> How is the software-intensive system architecture developed to ensure warfighter capabilities are designed and prioritized?

The DoD system architectural process, with all of its tools, techniques, and discipline, appears to be ineffective in driving repeatable, successful software designs. Within the SEP, there are three DoD processes that drive the system architecture: the requirements generations system, the WBS, and the OMS/MP.

There appears to be significant opportunity to omit requirements, or to provide vague or weakly articulated requirements through the translation process from the user capability-based requirements, to the PM's performance specification, and finally to the contractor's detailed specification. This problem is exacerbated by the immature software engineering environment described earlier, which is solely focused on requirements as provided.

The process of developing the WBS appears to be similarly flawed in effectively communicating the functional architecture to a sufficient level for the software developers. The overarching philosophy for both requirements generation and the WBS, in order to garner the maximum flexibility and innovation, is purposely not to be specific. Due to the immature engineering environment, the software components need significantly more specificity than the hardware counterparts to produce realism in the cost and schedule provided in the contractor's proposal.

The operational context information that the government provides appears to be insufficient for the potential software developers to have an understanding of the requirements within the context of the operational environment, constraints, and life-cycle events of the proposed system. The OMS/MP typically provides only a vague understanding of the operational environment and significantly more information is required to design and build the system actually needed by the warfighter. This additional information is likely to be added in program and design reviews conducted after the contract is awarded, so resulting changes impacting the software development can cause significant increases in the cost and schedule, all negotiated without the advantages of a competitive environment.

Conclusions

The DoD acquisition process provides the environment for both successful and unsuccessful software-intensive systems development. Specific elements of the DoD acquisition process that contribute to the variable environment include the following:

- ***The DoD Requirements Generation Process*.** The translation process from JCIDS capabilities-based language to the RFP/contract performance-based language, and finally to the specification-based detailed language creates ample opportunity for misinterpreted requirements to be communicated. This process was designed to garner innovation from mature engineering fields that leverage widely accepted materials, processes, and standards—attributes that the software engineering field does not yet have.
- ***Communicating Operational Context*.** The Operational Mode Summary/ Mission Profile (OMS/MP) provides some insight into a system's intended operational context but provides far too little information for the complex software design process. This lack of detail, again, cannot be compensated by the immature software engineering environment and so impacts software-intensive systems more than hardware-centric ones.
- ***Failure to Compensate for the Immature Software Engineering Environment*.** As demonstrated by the first two bullets, one of the major differences between successful and unsuccessful software-intensive systems development is recognizing and compensating for the immature software environment. The DoD Acquisition System policies, guidelines, and controls do not provide a framework to ensure that essential software attributes are sufficiently revealed and effectively communicated to the contractors that will design and build the software systems.
- ***The DoD Acquisition System***
    - The DAS is designed to leverage industry innovation by providing performance specifications that are designed to allow mature industrial engineering environments to develop the best-value technologies that meet the performance specifications. This is effective when the engineering environments are mature and can offer viable, mature technology alternatives that are considered industry standard. There are insufficient DAS processes for recognizing and compensating for immature engineering environments, such as exists in the software field.
    - The schedule and funding profile are initially set by the first system PM, and the program depends significantly on how well the requirements generation process accurately identified the bulk of the requirements. Once funding is linked to milestones, the program cost and schedule become very rigid, which exacerbates problems with software-intensive system developments that have late requirements creep due to insufficient understanding of the effort in the proposal preparation.
    - Software Technology Readiness Levels (TRLs) are ineffective in reducing risks associated with the system software development. Because there are few reusable software components, limited industry-wide standards for architecture and supportability, and rapidly

emerging languages, protocols and tools, the software TRLs, based on past efforts, are not reliable predictors of software readiness.

- Software development significantly adds to the system development risk. The DAS is designed to reduce development risk, but cannot eliminate all associated risks. Some risk is accepted with the expectation that the PM team will effectively manage those risks, yet there is no funding management reserve provided to do so. Any risk management mitigation effort that involves funding has the opportunity to create a cascade of management actions resulting from funding reductions in other planned and necessary activities.

Recommendations

General

As part of this research, I searched for tools, techniques, and procedures that would address the software-intensive system development problems and integrate well with the Defense Acquisition System (DAS) while supporting the Systems Engineering Process (SEP). The tools, techniques, and procedures recommended in this section are not particularly new and many programs may have used some, most, or all of these in the development of their systems. The major recommendation is that DoD formalize and institute the use of these tools, techniques, and procedures (or similar ones) for the development of software-intensive systems. There would almost certainly be a benefit when applied to hardware-centric system development, too, and certainly there would be no detriment in using them for all complex system development.

One of the findings of this research was the lack of a PM management reserve fund to address accepted development risks, but a significant policy and political change would be required to provide a management reserve in program funding. I believe this course of action to be unlikely, but the implementation of the recommendations would significantly reduce software-intensive system developmental volatility and risk, and reduce the need for the management reserve.

Each of the tools, techniques, and procedures are valuable in assisting the systems development process, but when used together, provide a synergistic effect to the vital front-end analyses that directly impact the shortcomings revealed in this research. Implementing these tools does not require any major adjustments to the DAS or the SEP, and in fact become major enablers for both.

Tools, Techniques, and Processes

The following tools, techniques, and processes are briefly described in this section:

- The Software Engineering Institute's (SEI's) Quality Attribute Workshop (QAW)
- The Maintainability, Upgradability, Interoperability, Reliability, & Safety and Security (MUIRS) analytic technique
- The Software Engineering Institute's Architectural Tradeoff Analysis Methodology (ATAMsm)
- The Failure Modes and Effects Criticality Analysis (FMECA)
- Software Management Readiness Levels (MgtRL)

Quality Attribute Workshop

The QAW is primarily a method for more fully developing system software requirements and is intended to provide stakeholders' input about their needs and expectations from the software (Barbacci et al., 2003, p. 1). As the system requirements are developed, software quality attributes are identified and become the basis for designing the software architecture.

The SEI's QAW is implemented before the software architecture has been created and is intended to provide stakeholder input about the needs and expectations from the software (Naegle, 2007). The QAW process provides a vehicle for keeping the combat developer and user community involved in the DoD acquisition process, which is a key goal of that process. In addition, the QAW includes scenario-building processes that are essential for the software developer to design the software system architecture (Barbacci et al., 2003, pp. 9–11). These scenarios will continue to be developed and prioritized after contract award to provide context to the quality attribute identified for the system.

Primary Software Acquisition Problem Area Addressed

The QAW process is primarily designed to more fully develop system software requirements so that the government RFP is clearer to potential contractors. In turn, the resulting proposals should be more accurate and realistic, reducing requirements and project scope creep.

Maintainability, Upgradability, Interoperability/Interfaces, Reliability, and Safety/Security Analytic Technique

The MUIRS analytic technique is designed to provide a framework for better understanding of essential supportability and safety/security aspects that the warfighter needs and expects but often doesn't communicate clearly with the capabilities-based JCIDS documents. This analytic technique helps compensate for the immature software engineering environment as the MUIRS analysis illuminates the derived and implied requirements that the immature environment cannot.

Much of the software supportability and safety/security performance that typically lacks consideration and is not routinely addressed in the software engineering environment can be captured through development and analysis of the MUIRS elements. Analyzing the warfighter requirements in a QAW framework for performance in each MUIRS area will help stakeholders identify software quality attributes that need to be communicated to potential software contractors (Naegle, 2006, pp. 17–24).

The MUIRS analysis assists the QAW process by focusing on those elements that are too often typically overlooked during the requirements generation process. The QAW and MUIRS analysis are critical to the software design process, discussed in the next section.

Primary Software Acquisition Problem Area Addressed

MUIRS primarily addresses the immature software engineering environment as it provides an analytic approach for critical sustainment and safety/security attributes often missing, weakly articulated, or vaguely stated in the requirements produced. With its capabilities and performance-based requirements processes, the DoD significantly depends on mature engineering environments to fill the gaps left from the requirements generation and communication processes, but the software engineering environment is unable to do so. The MUIRS analysis is also an enabler for the QAW and ATAMsm architectural processes discussed next.

Architectural Tradeoff Analysis Methodologysm

The SEI's ATAMSM is an architectural analysis tool designed to evaluate design decisions based on the quality attribute requirements of the system being developed. The methodology is a process for determining whether the quality attributes are achievable by the architecture as it has been conceived before enormous resources have been committed to that design. One of the main goals is to gain insight into how the quality attributes trade off against each other (Kazman, Kleim, & Clements, 2000, p. 1).

Within the SEP, the ATAM provides the critical Requirements Loop process, tracing each requirement or quality attribute to corresponding functions reflected in the software architectural design. Whether ATAM or another analysis technique is used, this critical SEP process must be performed to ensure that functional- or object-oriented designs meet all stated, derived, and implied warfighter requirements. In complex systems development such as weapon systems, half or more than half of the total software development effort is expended in the architectural design process. Therefore, the DoD PMs must ensure that the design is addressing requirements in context and that the resulting architecture has a high probability of producing the specified warfighters' capabilities described in the JCIDS documents.

The ATAM focuses on quality attribute requirements, so it is critical to have precise characterizations for each. To characterize a quality attribute, the following questions must be answered:

- What are the stimuli to which the architecture must respond?
- What is the measurable or observable manifestation of the quality attribute by which its achievement is judged?
- What are the key architectural decisions that impact achieving the attribute requirement? (Kazman et al., 2000, p. 5)

The ATAM is designed to elicit the data and information needed to adequately address the three previous questions. These questions, focused on requirements and quality attributes, are user-centric, and so the ATAM scenarios must be constructed by the user community (Naegle & Petross, 2007, p. 25). The methodology keys on scenario development in three main areas:

- **Use Case Scenarios.** As the name suggests, these scenarios describe how the system will be used and sustained in the harshest environments envisioned. It includes all interoperability requirements and duty cycles as well.
- **Growth Scenarios.** Growth scenarios focus on known and anticipated system change requirements over the intended life cycle. These scenarios include upgrades and technology refreshments planned; interoperability requirements, such as inclusion in future warfighting networks; changes in sustainment concepts, and other system changes expected to occur over time.
- **Exploratory Scenarios.** Exploratory scenarios focus on operations in unusual or stressful situations. These address user expectations when the system is degraded or operated beyond normal limitations due to emergency created by combat environments. These scenarios include Failure Modes and Effects Criticality Analyses (FMECA) to identify the essential functions that must not fail. As important to the software engineers, FMECA also identifies those enhancing functions that should *not* preclude the system from

functioning when that enhancing function is degraded or non-operational. The software engineers need that information to properly design the software.

Test cases are developed out of the scenarios, which firmly link the test program with the user requirements in the context of the scenarios. This methodology also helps to ensure that there are verification events for software and sustainment requirements, which are too often missing from the testing program.

As shown in Figure 1, the ATAM is an integrating function for many of the tools and techniques discussed here. It is designed to be an iterative process and would be most effective when started in early concept development, then continued through contract award, prototyping, and into the design review process.

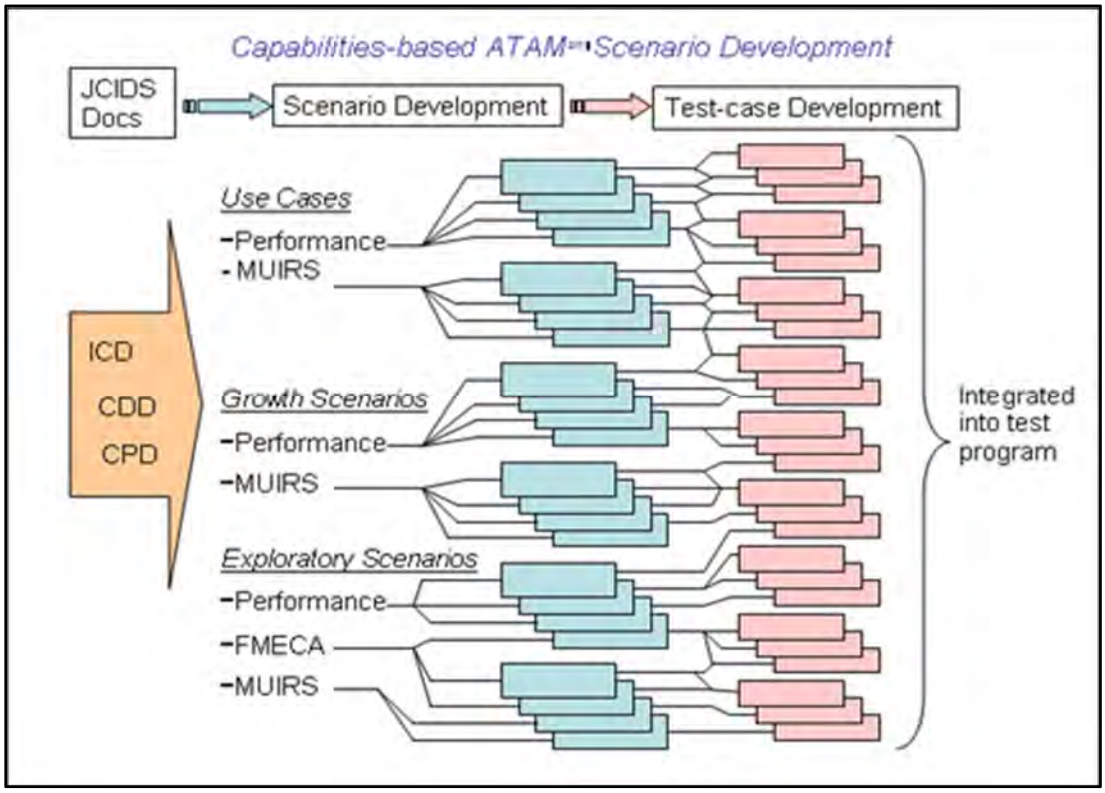

Figure 1. Quality Attribution Workshop and Architectural Tradeoff Analysis Methodology Integration Into Software Life-Cycle Management
(Naegle & Petross, 2007, p. 25)

Primary Software Acquisition Problem Areas Addressed

The ATAM process addresses four primary problem areas:

- The scenario development provides much more operational context than the typical OMS/MP provides. This level of detail helps to compensate for the immature software engineering environment and is critical for the proper design of the software architecture.
- The ATAM serves as a very effective software design metric function. With the software development team using 50% or more of the available resources for requirements analysis and software design before the Preliminary Design Review (PDR), it is critical to have an effective software design metrics function. Traditional software design metrics focus on the design complexity

and do not address whether the design is adequate or not. ATAM directly links the user requirements to the system architectural design.

- As the testing program is developed from the scenarios, it becomes difficult to omit any critical testing event. In addition, the software developer understands the tests or verification events that must be passed for user acceptance.
- By integrating the MUIRS analyses into the ATAM scenario development, sustainability and safety/security aspects cannot easily be omitted from the system design. As the testing plan flows from the scenarios, the MUIRS design elements will have corresponding test or verification events identified in the test plan.

Failure Modes and Effects Criticality Analysis

As the title indicates, this analysis methodology is designed to identify system failure modes and those failures' effects on the system, and ascertain the relative criticality of that type of failure. Blanchard (2004) described FMECA as follows:

> Given a description, both in *functional* and *physical* terms, the designer needs to be able to evaluate a system relative to possible failures, the anticipated modes and expected frequency of failure, their causes, their consequences and impact(s) on the system overall, and areas where preventative measures can be initiated to preclude such failures in the future. (p. 275)

He went on to state, "The FMECA is an excellent design tool, and it can be applied in the development or assessment of any product or process" (Blanchard, 2004, p. 276).

Including FMECA scenarios with the software systems and subsystems provides architectural design cues to software engineers. These scenarios provide analysis for designing redundant systems for mission-critical elements, provide "safe mode" operations for survivability- and safety-related systems, and drive the software engineer to conduct "what if" analyses with a superior understanding of failure-mode scenarios.

Primary Software Acquisition Problem Areas Addressed

The primary problem areas addressed by FMECA include requirements clarification and prioritization, and helping to ensure a sound software architecture design. This analysis also ensures that the most critical software systems are designed with the requisite reliability and will continue to function in degraded modes.

As previously stated, one of the main functions of performing FMECA is to identify those software functions that are not critical, and to ensure that failures or anomalies in those non-critical functions do not preclude or negatively affect system capabilities. Today's systems typically have numerous enhancing functions that improve performance but are not critical, and the software developers have no way to discern the difference between a critical system and an enhancing one without employing FMECA.

Integrating the Recommended Tools, Techniques, and Processes into the Defense Acquisition System

The tools, techniques, and processes were specifically selected for both their ability to address software-intensive systems development problems and their ability to integrate with the DAS. They are all SEP enablers designed to improve the critical DAS front-end processes, which are primarily the government's responsibility.

Figure 2 shows the processes applied at the latest possible developmental time to be effective. The earlier these tools, techniques, and processes occur, the more effective they

become. This representation also does not show the iterative cycles of QAW and ATAM or their overlapping nature.

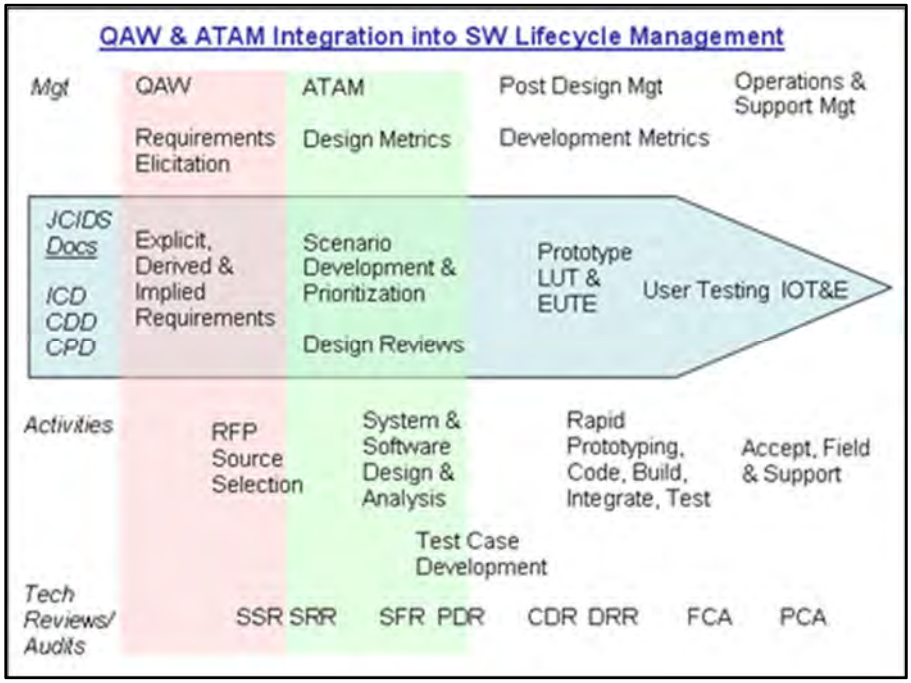

Figure 2. Quality Attribution Workshop and Architectural Tradeoff Analysis Methodology Integration Into Software Life Cycle Management
(Naegle & Petross, 2007)

As depicted in Figure 2, the QAW and ATAM are designed to address critical requirements and design front-end processes, where the government is primarily responsible for the process. The blue arrow shows how the warfighters and user community are continuously involved throughout the process, and are active participants in the QAW and ATAM processes. This is distinctly different than the traditional DAS where there is little formal user interaction between preparation of the JCIDS documents and the prototype limited user tests (LUT)/early user test and evaluation (EUT&E).

The user communities have a very significant role in driving the QAW and ATAM processes, which requires more user resources to support the system development. This user investment in the DAS is becoming more critical with the development of more software-intensive and complex systems of all kinds. This investment is absolutely necessary to avoid government to contractor misunderstanding of the system requirements and warfighter expectations, and would significantly reduce the cost and schedule delays associated with user dissatisfaction, user-test failure, and unnecessary system redesign.

Program Management Risk Reduction

These tools, techniques, and processes will not, of themselves, produce or guarantee anything. “An architecture analysis method, any architecture analysis method, is a garbage-in-garbage-out process. The ATAM is no different. It crucially relies on the active and willing participation of the stakeholders” (Kazman et al., 2000, p. 63). All of the tools and techniques described and recommended in this research are dependent on the team of professional stakeholders conscientiously performing their critical function in the development of the software-intensive system.

To effectively implement the recommended tools, techniques, and processes, the program management team must be professional, disciplined in their application of the SEP, and skilled in integrating the tools into the DAS. In a word used by the SEI, the team must be *mature*. The Defense Acquisition Workforce Improvement Act (DAWIA) mandates certain education and training levels for the professional workforce performing at various levels. The DoD invests significant resources in both education and training to help ensure the acquisition workforce competencies and comply with the DAWIA.

The DoD also evaluates the maturity of potential software developers by requiring an evaluation using SEI's Capability Maturity Model–Integrated (CMMI; or equivalent) for most software-intensive system acquisitions. The CMMI is a five-level model, and the software developer organization under evaluation must achieve at least a level three by an independent evaluation team to be eligible to be awarded the DoD contract.

As mentioned previously, the DoD does not currently require the PM offices managing software-intensive systems to achieve any maturity level on the Software Acquisition Capability Maturity Model (SA-CMM). The team effort between the government and the software developer strongly suggests that both the PM office and the software developer would reduce developmental risk by demonstrating an appropriate level of maturity.

Due in large part to the immature software engineering environment, each major DoD software design and build tends to be unique. That means that the software development in complex systems will act the same way as integrating a new technology would, and the resulting program risk is very high. The software TRLs have little meaning in this type of environment, so risk management is highly dependent on the government and software development teams' abilities to manage the system software development as a new technology with a low TRL.

A significant portion of the risk management is focused on the government and software development teams. As the software TRLs are mostly ineffective, I would recommend the further development of software Management Readiness Levels (MgtRLs) to mitigate the risks. Part of the management risk reduction is already in place with the DAWIA requirements and the software developer maturity levels that must be achieved.

References

Barbacci, M., Ellison, R., Lattanze, A., Stafford, J., Weinstock, C., & Wood, W. (2003, August). *Quality attribute workshops (QAWs)* (3rd ed.; CMU/SEI-2003-TR-016). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

Blanchard, B. S. (2004). *Logistics engineering and management* (6th ed.). Upper Saddle River, NJ: Pearson Prentice Hall.

Blanchette, S., Albert, C., & Garcia-Miller, S. (2010, December*). Beyond technology readiness levels for software: U.S. Army workshop report* (CMU/SEI-2010-TR-044). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

CJCS. (2012, January 10). *Joint capabilities integration and development system* (CJCS Instruction [CJCSI] 3170.01H). Washington, DC: Author.

Cooper, J., & Fisher, M. (2002, March). *Software Acquisition Capability Maturity Model® (SA-CMM®),* version 1.03 (CMU/SEI-2002-TR-010). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

DoD. (2005, July). Work breakdown structures for defense materiel items (MIL-HDBK-881A). *DoD Handbook*. Washington, DC: Author.

DoD. (2011, April). *Technology Readiness Assessment (TRA) guidance*. Washington, DC: Author.

DoD. (2012, January 12). *Operation Mode Summary & Mission Profile (OMS/MP) for the Joint Light Tactical Wheeled Vehicle, version 3.3.* Washington, DC: Author.

DoD. (2013a, November). *Operation of the Defense Acquisition System* (DoD Instruction 5000.02). Washington, DC: Author.

DoD. (2013b, December 3). *Defense acquisition guidebook*. Retrieved from https://dag.dau.mil

GAO. (2012, March 20). *Joint Strike Fighter: Restructuring added resources and reduced risk, but concurrency is still a major concern* (GAO-12-525T). Retrieved from http://www.gao.gov

Hagen, C., Hurt, S., & Sorenson, J. (2013, November/December). Effective approaches for delivering affordable military software. *Crosstalk Magazine,* 26–32.

Humphrey, W. (1990, August). *Managing the software process*. Reading, MA: Addison-Wesley.

Kazman, R., Klein, M., & Clements, P. (2000, August*). ATAM: Method for architecture evaluation* (CMU/SEI-2000-TR-004). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

Kruchten, P. (2005, March/April). Software design in a postmodern era. *IEEE Software, 18*(2), 17.

Naegle, B. R. (2006, September). *Developing software requirements supporting open architecture performance goals in critical DoD system-of-systems* (Acquisition Research Program Sponsored Report Series [NPS-AM-06-035]). Monterey, CA: Naval Postgraduate School.

Naegle, B. R., & Petross, D. (2007, September). *Software architecture: Managing design for achieving warfighter capability* (Acquisition Research Program Sponsored Report Series [NPS-AM-07-104]). Monterey, CA: Naval Postgraduate School.

Pietrasanta, A. M. (1998). *Current methodological research*. In ACM National Conference (USA) (pp. 341–346). New York, NY: ACM Press.

Software Engineering Institute/Carnegie Mellon Software Architecture. (2007). *The importance of software architecture*. Retrieved March 1, 2007, from http://www.sei.cmu.edu/architecture/index.html

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CA 93943

www.acquisitionresearch.net