NPS-CE-17-042

ACQUISITION RESEARCH PROGRAM SPONSORED REPORT SERIES

Total Ownership Cost—System Software Impacts

19 April 2017

Brad R. Naegle, Senior Lecturer

Graduate School of Business and Public Policy

Naval Postgraduate School

Approved for public release; distribution is unlimited.

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

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

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

Abstract

Department of Defense (DoD) software-intensive systems and the software content in other systems will continue to grow and may dominate total ownership costs (TOC) in the future. These costs are exacerbated by the fact that, in addition to contracted development costs, the bulk of software sustainment costs are also contracted. All of these factors indicate that DoD system software will continue to be a very expensive portion of TOC.

The software engineering environment remains immature, with few, if any, industry-wide standards for software development or sustainment. The Defense Acquisition System (DAS) is significantly dependent on mature engineering.

System software size and complexity are key indicators of both development costs and sustainment costs, so initial estimates are critical for predicting and controlling TOC. Unfortunately, the software size estimating processes require a significant amount of detailed understanding of the requirements and design that is typically not available when operating the DAS without supplementary analyses, tools, and techniques. Available parametric estimating tools require much of the same detailed information and are still too inaccurate to be relied upon. Similarly, understanding the potential software complexity requires in-depth understanding of the requirements and architectural design.

It is clear that the DoD must conduct much more thorough requirements analyses, provide significantly more detailed operational context, and drive the software architectural design well beyond the work breakdown structure (WBS) functional design typically provided. To accomplish this, the DAS must be supplemented with tools, techniques, and analyses that are currently not present.

Program managers for software-intensive systems must supplement the DAS processes to

- compensate for the immature software engineering environment
- gain sufficient detailed information to perform reasonable software size and complexity estimates critical to understanding and managing system TOC

- complete the inventory of derived and implied requirements, including the often neglected sustainability requirements, before the request for proposal (RFP) is issued
- provide more detailed system operational context, beyond what exists in most Operational Mode Summary/Mission Profile documents
- obtain more realistic contractor proposals in terms of cost and schedule associated with the software development and sustainment
- drive the software architecture for a more sustainable, less complex design
- monitor the software design process (metrics) to ensure the effort is progressing towards an effective, supportable, and testable design supporting the warfighter

The tools, techniques, and analyses presented in this research are designed to accomplish the tasks outlined above and are compatible with the Systems Engineering Process supporting the DAS. They also are designed to work together in a synergistic method to improve the software-intensive system development and sustainment performance influencing system TOC. Combined, the tools, techniques, and analyses provide a much improved understanding of the system and identify critical attributes that the software developers need to know to design an effective and supportable design. These tools help compensate for the immature software engineering environment, provide more detailed information needed to perform size and complexity estimates, and provide detailed operational context needed for proper software architectural design. They help produce superior RFPs and garner more realistic contractor proposals. They provide processes for monitoring critical software design activities and full test matrix crosswalks. All of these enhancements will help more accurately estimate and manage software TOC attributes.

Keywords: Total ownership cost (TOC), software, operating & support cost, sustainment cost, developmental cost, production cost, software supportability, post deployment software support (PDSS)

About the Author

Brad R. Naegle, Lieutenant Colonel, U.S. Army (Ret.), is a senior lecturer and academic associate for Program Management curricula at the Naval Postgraduate School. While an active duty Army officer, he was product manager for the Tactical Wheeled Vehicle Remanufacture Family of Programs from 1994 through 1996 and deputy project manager for the Light Tactical Vehicles Project Office from 1996 to 1997. He was the 7th Infantry Division (Light) Division Materiel officer from 1990 through 1993. Prior to that, he held numerous test and evaluation (T&E) and logistics leadership positions from 1977 to 1990. LTC Naegle is a distinguished graduate of the Naval Postgraduate School, with a Master of Science in Management with a Systems Acquisition specialty and a Bachelor of Science in Economics from Weber State University. He is a graduate of the U.S. Army Command and General Staff College, the Combined Arms and Services Staff School, and the Ordnance Corps Officer Basic and Advanced courses.

THIS PAGE INTENTIONALLY LEFT BLANK

NPS-CE-17-042

ACQUISITION RESEARCH PROGRAM
SPONSORED REPORT SERIES

Total Ownership Cost—System Software Impacts

19 April 2017

Brad R. Naegle, Senior Lecturer

Graduate School of Business and Public Policy

Naval Postgraduate School

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

Background

Significant technological advancements tend to have dramatic impacts on systems' life-cycle costs, sometimes reducing development and sustainment costs, and other times increasing those costs. For example, the development of electronic relays to replace physical relays vastly increased the maintenance and replacement intervals for systems so equipped. The resulting sustainment cost savings were significant. The advancements can have the opposite effect, as evidenced by the stealth aircraft coatings, which are difficult and costly to maintain:

> After a stealth aircraft flies, maintenance workers must recoat the skin repairing the tiny dings and burrs that increase the craft's radar signature. … The B-2's skin is so sensitive that maintenance on the plane must be carried out in environment-controlled hangars that currently exist only at Whiteman Air Force Base. The B-2's that participated in the Kosovo campaign thus had to fly more than 30 hours round-trip between Missouri and Yugoslavia. It then generally took from four to seven days to get them ready to return to combat. (Silverstein & Moag, 2000, p. 1)

Weapon system software functionality is another of those significant technological advancements and is changing the nature of a software-intensive system's total ownership cost (TOC). Understanding and influencing software developmental and sustainment costs is critical in predicting and controlling weapon system TOC in a fiscally constrained environment. Somewhere in the range of 85% to 90% of the F-22 Raptor's functions are wholly or partially software-controlled, and this trend is likely to continue through future weapon system development efforts.

This research is a continuation of TOC research efforts conducted by the Naval Postgraduate School Acquisition Research Program, and the focus of this effort is specifically exploring software sustainment effects on a software-intensive system TOC. It is no secret that Department of Defense (DoD) weapon systems have leveraged the advantages of software-controlled functions and that the amount of software has continually and radically increased, as indicated in Figure 1.

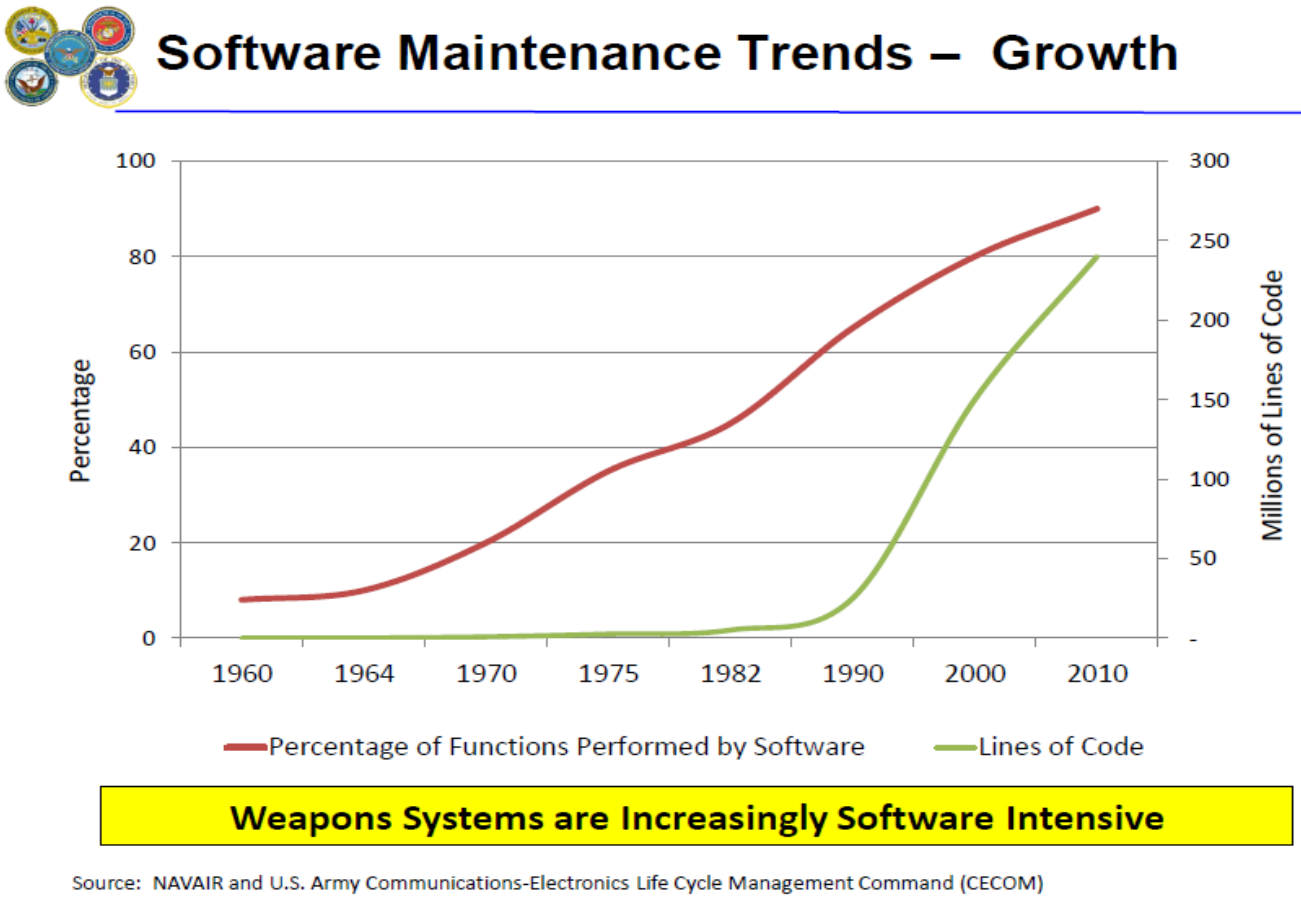


Figure 1. Software Functions and Lines of Code
(NAVAIR and U.S. Army CECOM Brief, 2012, p. 5)

Clearly, the DoD's desire for ever-increasing software-intensive systems will continue. In addition to the desired software-intensive weapon systems, tactical and strategic networks have only just begun to be developed, driving the amount of software in the DoD to unprecedented levels. What are the implications of software on development and sustainment costs? Is software different with regard to development? Is software sustainment different?

The cost of software development is not trivial, but like hardware-centric systems, 60–80% of the software life-cycle costs are typically incurred in the

sustainment phase, so software sustainment becomes a very important consideration when attempting to reduce the weapon systems' TOC.

Definitions (Boudreau & Naegle, 2003, p. 1)

Total ownership cost (TOC) has two definitions; the first is very broad, looking from the DoD or Service perspective.

> DoD TOC is the sum of all financial resources necessary to organize, equip, train, sustain, and operate military forces sufficient to meet national goals in compliance with all laws, all policies applicable to DoD, all standards in effect for readiness, safety, and quality of life, and all other official measures of performance for DoD and its Components. DoD TOC is comprised of costs to research, develop, acquire, own, operate, and dispose of weapon and support systems, other equipment and real property, the costs to recruit, train, retain, separate and otherwise support military and civilian personnel, and all other costs of business operations of the DoD. (Under Secretary of Defense for Acquisition, Technology, and Logistics [USD(AT&L)], 1998, p.5)

The second definition is deliberately written from the vantage point of the program manager (PM) of the warfighting system.

> Defense Systems TOC is defined as Life Cycle Cost (LCC). LCC (per DoD 5000.4M) includes not only acquisition program direct costs, but also the indirect costs attributable to the acquisition program (i.e., costs that would not occur if the program did not exist). For example, indirect costs would include the infrastructure that plans, manages, and executes a program over its full life and common support items and systems. The responsibility of program managers in support of reducing DoD TOC is the continuous reduction of LCC for their systems. (USD[AT&L], 1998, p. 2)

As Dr. Gansler said in his 1998 memorandum from which the above definitions were extracted, the PM's job in trying to reduce TOC is a very difficult one, and PMs should seek help wherever they can to reduce ownership costs. Because of the extreme amount of focus on the authorized and appropriated budget, it is easy for PMs to likewise focus on the near-term acquisition cost and make decisions that appear to be beneficial in reducing acquisition costs but that are detrimental to operations and support costs because they increase future budgets.

For example, if a program experiences a budget cut, which is a very typical occurrence, there is significant pressure to continue to deliver the same number of new systems, even though there has been a cut in funding. As a result, the PM looks for something else to cut out of the program. Logistics performance items are rarely deemed to be key performance parameters (KPPs), so they become easy targets for cutting during budget cut drills. So the PM is faced with a choice: Cut the number of systems to be acquired or reduce the logistics performance (eliminate built-in test [BIT] capability, onboard diagnostics/prognostics/autonomics, etc.), which will add significant operating and support (O&S) costs well after the PM has moved to a different position. Which choice do you suppose is most appealing to the PM?

Even the definition of *system* is changing as we move to system-of-systems (SoS) and net-centric system concepts. For example, developing the Single Integrated Air Picture (SIAP) as a system means that all the Services' manned and unmanned aircraft, many guided and unguided missile platforms, and a host of command and control systems are now at least a part of SIAP. Changes to any of the platforms (especially software changes) could result in changes to others, impacting the TOC of the individual weapon platforms and of the SIAP system. How do we account for SoS or net-centric system-driven changes/maintenance in forecasting or even attributing cost elements? For example, consider the networking software for Force XXI Battle Command Brigade and Below (FBCB2) as updated for the M1 Main Battle Tank and several other platforms using the network. The software update did not integrate perfectly with the M1's onboard software suite, causing uncommanded turret movement (Federation of American Scientists [FAS], 2011a). Because the other platforms in the system did not experience interoperability problems, the PM for M1 was assigned responsibility for the diagnosis and repair of M1 software to be compatible with the FBCB2 update. This begs the questions of which program should the TOC expense be attributed to (FBCB2 or M1), how would such TOC factors be forecasted, and who would budget for them?

TOC Processes: CAIV and R-TOC (Boudreau & Naegle, 2003, p. 2)

Pursuit of TOC reduction at the level of the warfighting system may be separated into two major approaches that are connected, end-to-end, along a life-cycle time line. During the developmental phases, the effort or process is called Cost As an Independent Variable (CAIV). For systems in the field or fleet, the process or goal becomes Reduction of Total Ownership Cost (R-TOC). The chart in Figure 2 is a typical depiction of the CAIV/R-TOC relationship.

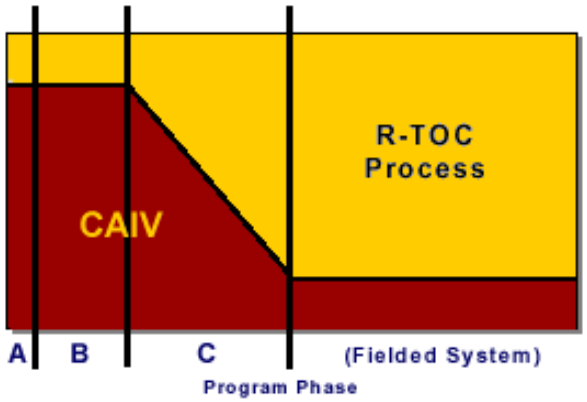


Figure 2. CAIV/R-TOC Relationship
(Kaye, Sobota, Graham, & Gotwald, 2000, p. 354)

The first approach, CAIV, addresses TOC during the warfighting system's developmental phases, beginning with the Concept Refinement phase. The focus of CAIV is to establish cost targets based on affordability and requirements and then to manage to those targets, thereby controlling TOC. CAIV includes consideration of costs for development, production, operations and support, and disposal. An example of the CAIV process would be to set specific cost and reliability targets for each subsystem or component of a weapon system in development such that the warfighting system would be able to achieve the required operational availability (A_O) at the specified cost.

Employing the CAIV concept early in the developmental process offers, potentially, the greatest opportunity for TOC reduction at the lowest possible investment cost. As an example, the TOC impact of using two different power plants

presents an opportunity to use the CAIV evaluation technique to estimate the TOC impact and make a best-value decision. For illustrative purposes, consider a standard internal combustion engine at a cost of $7,500 versus a hybrid-electric power plant costing $19,000. The impact to the acquisition cost is evident, but it excludes the cost savings associated with fuel consumption over the life of the system. If the system's operational mode indicates an average usage of 15,000 miles per year and an economic useful life (EUL) of 20 years, the total miles expected would be 300,000. If the standard engine in our comparison is estimated at 10 miles per gallon and the hybrid engine is estimated at 25 miles per gallon, the estimated fuel saved by the hybrid-powered system would be 18,000 gallons. At a current estimate of $1.25 per gallon, the operating and support impact is $22,500 per system (TOC improvement: $11,000 less expensive than the standard engine), and there are other reductions in fuel supply assets and attendant personnel that apply.

The second approach to TOC is the R-TOC, which focuses on the reduction of average procurement unit cost (APUC) and weapon system sustainment cost (i.e., O&S costs). R-TOC is employed as the warfighting system is produced and placed in service. Examples of R-TOC would be a value engineering change proposal (VECP) to reduce the cost of manufacturing a component by improving the process yield (the percentage of the manufactured items that are defect free) or a VECP to reduce the operating and support cost by improving the reliability of an expensive subsystem or component. Often there are the secondary benefits of enhanced performance (i.e., improved reliability and operational availability), but the forcing function is the reduction of operating and support costs, the largest constituent of TOC.

System software has become an ever-increasing TOC driver as more systems rely on software functions.

TOC Obstacles (Boudreau & Naegle, 2003, pp. 4–7)

Someone who is not involved with program management might wonder what is especially difficult about containing and controlling TOC. In truth, there are many difficulties. What follows is a description of some of the obstacles that get in the way of controlling or reducing weapon system TOC. All of these obstacles are well known but are entrenched and difficult to overcome.

The competing interests of users, developers, prime contractors, subcontractors, the Office of the Secretary of Defense (OSD), Service headquarters, maintainers, buying commands, and Congress may negatively impact TOC. The "user" who establishes requirements for a new system may be transfixed by the technical performance and may not clearly establish requirements for ownership cost to achieve specified system availability. Materiel developers may be too focused on acquisition cost and schedule (a typical complaint from the user community) and may ignore future logistics support issues. Prime contractors may concentrate on production costs, with less regard for system sustainment costs, particularly if their contract directs them toward reduction in production costs or if they sense that their customer is not interested in sustainment issues. The OSD and Service headquarters may encourage poor TOC decisions through funding instability and failure to demand life-cycle affordable solutions. Maintainers may contribute to poor R-TOC by failing to speak out loudly on lessons learned from previous systems. Buying commands may contribute to increased ownership costs by failing to look aggressively for cost drivers that need to be redesigned for lower cost of operation and improved reliability. Congress may restrict R-TOC by constraining the choices of cost-effective sustainment approaches.

Balancing Total Ownership Cost Goals That Are Conflicting. Successful program management includes the ability to achieve balance within a program. Indeed, PMs are directed by DoD Directive (DoDD) 5000.1 to manage their programs in a balanced way (USD[AT&L], 2003, Encl. 1, para. E1.29). Facets and perspectives that need to be balanced are manifold. Four elements of TOC that require balancing are development costs, procurement costs, operating and support

costs, and disposal costs. Development costs, the expenditure of resources during system development, may pay off in terms of reduced production and/or sustainment costs; producibility studies may save significant manufacturing costs; and reliability testing early in a program may allow for avoidance of sustainment costs over the service life of the weapon system. Occasionally, procurement or production cost constraints may conflict with sustainment cost targets; for example, heavy pressure to reduce production costs may lead to the selection of components that are inexpensive but not reliable. Such choices would reduce production cost but increase sustainment costs and very possibly result in an increase of TOC. When such cost goals conflict, a reasonable metric for maintaining balance would appear to be minimization of TOC (i.e., life-cycle cost, but often TOC is sub-optimized due to these competing pressures).

Balancing Cost, Schedule, System Performance, Sustainment, Quality, and Risk. In the same way that ownership cost goals must be balanced and harmonized, system solutions must be found that balance TOC against procurement cost goals, program schedule goals, system technical performance, equipment quality, supportability performance, and availability.

The DoD is relying on sophisticated, software-intensive systems to improve survivability and lethality, but software is susceptible to high TOC. Software "maintenance" is becoming a major TOC driver (Naegle, 2004, p. 1). Software is difficult to accurately estimate and sensitive to changing requirements. Its complexity, interface requirements, and relative ease in adding capability also tend to make it maintenance intensive (Humphrey, 1990, ch. 4). Software's negative influence on TOC is exacerbated by the fact that software support is most often provided by contractors, with very little opportunity to move software support to Government sources. For example, the 1980s vintage B1B Bomber budgets were approximately $100 million annually for software maintenance, and the 2010 budget was $227 million because several software-intensive systems were also being upgraded (Naegle & Petross, 2010, p. 25). The B1's software support is achieved through both contractor and Government software support organizations and is coordinated by an Air Force–supported Program Management Office (PMO).

During each life-cycle phase, the approach to TOC reduction and the methodology may change somewhat, while ownership cost goals and targets become more refined. For example, trade-off processes used in the Materiel Solution Analysis phase may be beneficial during that phase but may be inadequate for the Engineering and Manufacturing Development (EMD) phase without the inclusion of specific contractor incentives.

Materiel Developer Instability. Key members of the materiel developer team change over time. For example, the PM during the Integrated System Design phase would be unlikely to remain in that position through the Production and Deployment phase. As key personnel—PMs, chief engineers, product support managers, business-financial managers—change, program emphasis shifts, at least subtly. These personnel changes, which are a fact of life, may reflect in program missteps, including missed TOC targets.

Funding Instability. Resources tend to be unstable and subject to unanticipated, unexpected changes. Funding instability is also a fact of life in Government acquisition programs. Each time that funding is cut from a program, decision-makers adjust the program by postponing or eliminating some activity or system attribute. Decisions are made that will keep the program viable, and often the choice is to omit a system feature or a near-term activity that will reflect negatively on TOC—but not until later. Easing back on O&S cost targets is a tempting sacrifice when program funding gets cut. For example, reliability-centered maintenance studies cut to reduce cost during EMD would not affect the program noticeably until later on, when operational systems are in the field or fleet; the associated effect on TOC might be substantial. Eliminating onboard diagnostics/prognostics would certainly help meet funding cuts during the Procurement phase, but would likely be extremely costly in terms of maintainer training, diagnostics time, erroneous fault isolation, errant parts ordering, and associated maintenance man-hours for the life of the system.

Sticker Shock. The fact that a system's TOC "price tag" is extremely high when compared to its contract unit price may tend to keep stakeholders from

discussing TOC in any open forum, fearing that "sticker shock" might cause an adverse reaction from a decision-maker or politically powerful individual accustomed to seeing much lower cost figures. As an example, consider a system with an average procurement unit cost (APUC) of $1.5 million and a program acquisition cost of $2 million.[1] Typically, program acquisition cost would represent only about 28% of each individual system's TOC, with the remaining 72% representing O&S and disposal costs of about $5 million, for a TOC of $7 million per each weapon system. With an acquisition objective of 2,000 systems, the total procurement cost would be $3 billion, with a TOC estimate of approximately $14 billion. If unfamiliar with TOC estimates and without a readily available basis for comparison, a decision-maker might mistakenly conclude that the system would be unaffordable and cancel the program. Concern for such a scenario may create an impediment to widespread use of system life-cycle cost numbers, which would have the effect of refocusing decisions onto the acquisition "price tag," not the TOC "price tag."

Management of TOC

There is an increasing body of knowledge related to the control of TOC. In addition to specific congressional direction and ever more detailed DoD direction, very thoughtful articles have been published on the matter. Every PM tries different approaches to reduce costs. Additionally, commercial best practices have been recognized and suggested for use within the DoD (e.g., GAO-03-57 [GAO, 2003] in its entirety).

[1] APUC is the total procurement cost divided by the total procurement quantity. Program acquisition cost includes APUC, facilities, RDT&E, and other procurement costs. These terms are discussed in more detail, later in this research.

The Focus of This Paper—Software Development and Sustainment Impacts on TOC

Purpose

The purpose of this research is to examine DoD software development techniques to illustrate the impacts to sustainment costs, which is the main contributor to a system's TOC. The research also suggests tools, techniques, and analyses to assist PMs and others in addressing software-related TOC elements more effectively.

Scope of This Study

This study examines TOC from the perspective of software development management, the perspective of PM execution, and the perspective of available infrastructure support.

Introduction

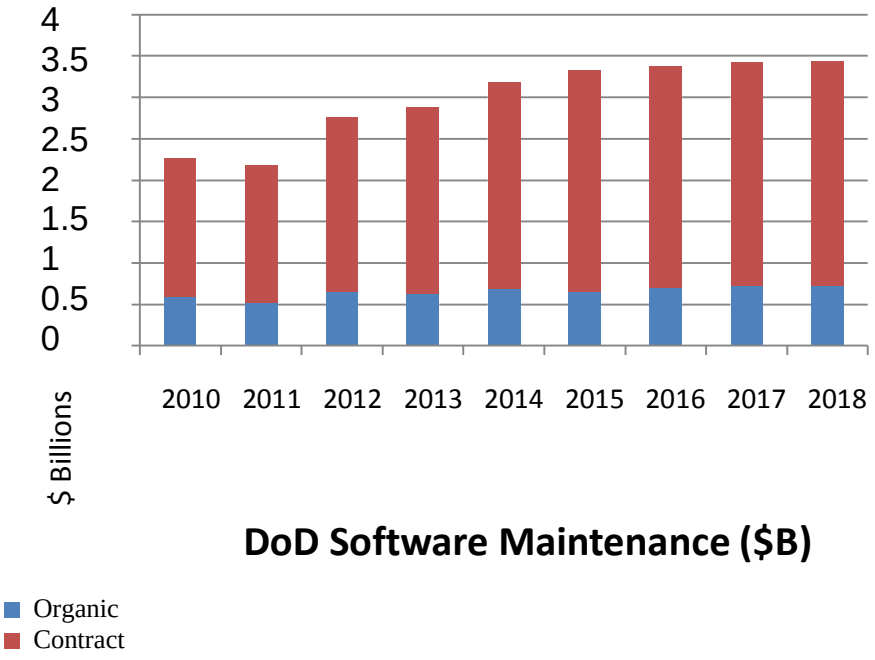
This report extends NPS research that was published in 2003 and 2011. The DoD initiatives on weapon system TOC is available on two websites:

> Much of the TOC and Reduction in TOC (R-TOC) efforts were captured in initiatives collected and shared on a TOC website maintained by the Institute for Defense Analyses (IDA; www.ida.org). The DAU Acquisition Community Connection website (https://acc.dau.mil/CommunityBrowser.aspx?id=22509&lang=en-US) also contains useful approaches to TOC and R-TOC. (Naegle & Boudreau, 2011)

Curiously, with all of the efforts to reduce TOC, there is little reference to reducing the software contribution to system TOC despite the fact that software functionality and software source lines of code (SLOC) continue to increase.

Software-related sustainment costs have been growing along with the growing amount of software developed for weapon systems. In addition to the software size (SLOC count), other factors can influence the system's supportability costs including architectural complexity, external interfaces, the engineering and

developmental discipline maintained. The bar chart in Figure 3 depicts the software sustainment growth that the DoD is experiencing, though the leveling after 2015 seems suspect. With the explosive software growth depicted in Figure 1 expected to continue, the resulting software maintenance costs will likely continue to grow.



Source: POM 2014 PB-45 SNaP Data

Figure 3. DoD Software Sustainment Growth

(Reynolds et al., 2012, p. 6)

The acquisition reforms initiated in the 1990s fundamentally changed how the DoD developed and procured its weapon systems. At the time, DoD weapon systems were hardware-oriented, and software was just beginning to be an important system development issue. A system's TOC performance is significantly defined during the developmental process, and these acquisition reforms had a dramatic effect—positive and negative—on TOC. The central idea behind the acquisition reforms was for the Government to become less prescriptive in the development of its weapon systems and to garner more innovation from industry. On the TOC side,

> acquisition programs have reduced their preparation for sustainment. MIL-STD-1388-2A and -2B, which became obsolete under the Acquisition Reform initiatives of the 1990s, were very detailed and for many years had guided acquisition logistics planning; they were mandatory until circa 1995.[2] These standards governed supportability analyses and served to inform sustainment planning, but they were onerous requirements and sometimes resulted in analyses that languished on the shelf and were never put to use. (Naegle & Boudreau, 2011, p. 15)

Transitioning from the military standard (MIL-STD)–based detailed specifications to the performance specifications essentially shifted the design priorities from the Government to the contractor, including those designs that critically impact TOC. The Government now had the opportunity to specify TOC performance, and implemented this ability with mixed results. For example, the F/A 18 Super Hornet aircraft uses F404 engines, which "connect to the airframe at only 10 points and can be replaced without special equipment, a four-person team can remove the engine in only 20 minutes" ("F/A 18," n.d.) Specifying TOC performance elements—in this example, the mean-time-to-repair (MTTR), which drove the F/A 18 design to accommodate the 20-minute removal time—creates the opportunity to improve TOC performance and leverage innovative techniques developed within industry. The downside is that the onus is on the Government to specify desired TOC performance or accept the TOC performance deemed appropriate by the contractor.

In 2003, the DoD user community also transformed how the warfighters communicate their requirements to the acquisition community. The Joint Capabilities Integration Development System (JCIDS), as indicated by its name, is a capabilities-based requirements development and communication platform, which does not focus on any particular solution for satisfying the identified capability need.

[2] In the mid-1990s, there were numerous acquisition reform initiatives intended to streamline acquisition processes and reduce cost. One of these initiatives was "specs and standards" reform. Many Government specs were rescinded to reduce the Government burden and cost of maintaining specs; in many cases, the Government switched to commercial specifications that were maintained by various technical societies or associations. Other mandatory specs were rescinded because they were thought unnecessary or provided insufficient benefit for the cost expended. MIL-STD 1388-2A and -2B were thought by some to fall into the latter category.

For example, an Initial Capabilities Document (ICD) requirement may be presented in language such as “The U.S. Marine Corps has a need for a man-portable anti-aircraft capability for effectively engaging low-flying aircraft.” If it is determined that a materiel solution is required, a Capabilities Design Document (CDD) is produced containing design cues such as “The anti-aircraft system must be supportable within the existing USMC personnel and logistics structure.” Typically, these capabilities-based requirements need significant interpretation to convert them to Performance Specification language. This interpretation includes the formulation of *derived* requirements; for instance, the “man-portable” capability derives into a weight limitation performance specification. More difficult yet, the capabilities-based language creates *implied* requirements. To develop a system “supportable within the existing USMC personnel and logistics structure” implies that no new personnel specialties are needed and that the new system can use existing storage and ammunition supply chains.

The system requirements’ operational context provided for the potential contractors is usually communicated through an Operational Mode Summary/Mission Profile (OMS/MP). The typical OMS/MP provides some important design cues, but like the performance-based requirements, depends on a very mature engineering environment to fill in the gaps and address the inherent vagueness in which the OMS/MP are prepared.

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.

The Joint Light Tactical Vehicle (JLTV), replacing the High Mobility Multi-purpose Wheeled Vehicle (HMMWV) family of vehicles, is a multi-mission platform. The JLTV program OMS/MP (version 3.3) dated January 12, 2012, paragraph 1.2, Document Overview, explains what the JLTV OMS/MP defines, including

- Expected operational modes
- Full spectrum operations, operational themes, and elements of the operational terms (offense, defense, and stability)
- Joint mission profiles and operational elements
- Terrain conditions in terms of mileage, speed, and roughness
- Environmental conditions (DoD, 2012, p. 3)

Obviously, a significant amount of mission and system information is omitted. The details regarding the following items are not provided:

- System mission configuration (JLTV is a multi-mission platform)
- Number of personnel
- Personnel equipment and supplies required
- Cargo/hauling capacity
- Interoperability requirements
    - Communications and network equipment
    - Situational awareness systems
    - Weapons
    - Trailers/towed systems
    - Special missions equipment (e.g., chemical weapons detection)
    - Electric power requirements for integrated equipment
- Crew maintenance tasks and frequency
- Survivability considerations

As with this JLTV OMS/MP, the documents typically do not provide much information on operations under stressful, degraded, or unusual conditions, which would provide critical design cues for the software engineers. 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 they need to adequately design the software and to predict the funding and schedule resources necessary to build the software the warfighter expects. The JLTV OMS/MP was specifically created for the Engineering and Manufacturing Development (EMD) phase. In paragraph 1.1, Purpose, it states,

> This OMS/MP describes system modes, mission profiles, and usage conditions for the JLTV during its operating life. When approved, it supersedes the OMS/MP published with the JLTV Request for Proposal (RFP) in February 2008, but will not take effect until JLTV EMD phase activities. The OMS/MP supports the basis for essential capabilities described in the JLTV Capability Development Document (CDD) documenting key usage factors directly applicable to design study, logistical analyses, O&S [operations and support] estimation, and reliability, availability, and maintainability (RAM) testing and analyses. (DoD, 2012, p. 3)
>
> It is clear that the typical OMS/MP provided to the contractor is a very high-level, vaguely articulated document. For example:
>
> 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. (Naegle, 2015, pp. 17 & 18)

With regard to the software component contribution to a weapon system's TOC, the following three research questions frame the focus of this paper:

First Research Question: With software's increasing influence in weapon system TOC calculations, are there more effective methods to positively influence and predict software sustainability costs?

Second Research Question: Are there aspects in the software environment that suggest the need for a supplemental DoD acquisition process driving software development toward improved TOC performance?

Third Research Question: What are the software-related TOC drivers and are there effective methods for predicting the software-related TOC costs?

Congressional Directives

Weapon systems' TOC growing price tag has not gone unnoticed by Congress, which has responded with considerable TOC-related language in the laws that it passes and guidance it provides. For example, the Weapon Systems Acquisition Reform Act of 2009 is all about controlling weapon systems TOC. In addition, TOC performance is what Congress is addressing in its changes to Nunn–McCurdy. It is what motivated Congress to require certificates at Milestones A and B (10 U.S.C. § 2366a, b). This appears to be the congressional motive in Public Laws 111-84 (National Defense Authorization Act [NDAA] for Fiscal Year 2010, 2009) and 113-291 (NDAA, 2015), which contain significant DoD TOC-related language, discussed in the following section. Congressional intervention into the DoD TOC situation has been driven by a perception that DoD has failed to adequately address and control weapon system TOC. "Having witnessed a lack of cost and process discipline spanning many years, particularly in the area of sustainment costs, Congress has acted to enforce discipline, instituting procedures with force of law to get weapon system costs under control" (Naegle & Boudreau, 2011, p. 17). This congressional interest and intervention continues today.

The Weapon Systems Acquisition Reform Act of 2009 (Naegle & Boudreau, 2011)

The Weapon Systems Acquisition Reform Act (WSARA) of 2009 is a congressional initiative to increase rigor in development of DoD Major Defense Acquisition Programs (MDAPs). The principal intent seems directed at controlling the ownership cost of the DoD's warfighting systems. The WSARA advances on a number of different fronts, a portion of which are described in the following paragraphs (WSARA, 2009).

Director of Cost Assessment and Program Evaluation ("Director CAPE"). The Director CAPE is a new appointive position, devised to give independent advice and analysis to the secretary of defense (SECDEF) and deputy secretary of defense (DEPSECDEF) on matters that fall within their areas of responsibility. Director CAPE

is responsible for functions formerly accomplished by Program Analysis & Evaluation, the Defense PA&E (WSARA, 2009, § 101).

Director CAPE has two deputies (WSARA, 2009, § 101):

- the Director for Cost Assessment (formed initially from the Cost Analysis Improvement Group [CAIG]), and
- the Director for Program Evaluation (formed from the remnants of PA&E, that is, PA&E less the CAIG).

Responsibilities. Director CAPE is responsible for cost estimation and cost analysis for MDAP acquisition programs, analysis, and advice in the planning and programming phases of Planning, Programming, Budgeting, and Execution (PPBE; WSARA, 2009, § 101). Additionally, Director CAPE provides analysis and advice to the Joint Requirements Oversight Council (JROC) and formulates study guidance used to conduct an Analysis of Alternatives of new major defense acquisition programs (WSARA, 2009, § 201). These responsibilities place Director CAPE in a position to provide advice and direction related to the accuracy of acquisition cost estimates and the affordability of acquisition programs. Quite apparently, Director CAPE is charged with advising the JROC to strengthen that body's role in issues of cost and affordability, as noted in the short JCIDS discussion.

Cost Estimation. Director CAPE is specifically charged by Congress with ensuring the accuracy of cost estimation and cost analysis by prescribing policies and procedures specifically related to acquisition programs (WSARA, 2009, § 101). In this capacity, Director CAPE is required to provide guidance to and consult with OSD leadership and the secretaries of the military departments regarding specific cost estimates and cost analyses to be conducted for a major MDAP or major automated information system (MAIS) program. This mandate includes specifics, such as the selection of statistical confidence levels of cost estimates in consideration of life-cycle costs of MDAP and MAIS programs. Director CAPE specifically reviews independent cost estimates (ICEs) for the Defense Acquisition Board (DAB) prior to certifications, low rate initial production (LRIP), or full rate production (FRP). Director CAPE is further charged to review cost analyses and

records for MDAP and DAIS and is given authority to participate in discussion of discrepancies between ICE and military department cost estimates. This includes disclosure of statistical confidence levels used by Director CAPE and the Services. Confidence levels below 80% must be justified and included in the next Selected Acquisition Report (SAR), which is sent from PMs, through their component and the OSD, to Congress. Director CAPE is required to report annually on cost-estimating accuracy and compliance with policy, along with consistent differences in cost-estimating methodology by the Services. This report goes to OSD leadership and congressional defense committees and is to be posted on the Internet for public review.

The WSARA specified that Director CAPE must report to the SECDEF on O&S costs for MDAPs within one year and to congressional defense committees within 30 days thereafter, followed by annual reports (WSARA, 2009, § 101). This represents another action that brings focus to the cost of operating and sustaining warfighting systems. This requirement has caused considerable consternation. See the discussion in the Government Accountability Office (GAO) Report 10-717 section later in this paper for additional perspective on the accuracy of O&S cost databases.

Director of Program Assessment and Root Cause Analysis (Director PARCA). The WSARA (2009, § 103) mandated that the SECDEF designate a senior official responsible for conducting program assessment and root cause analysis for MDAPs. Director PARCA is responsible for evaluating the utility of performance metrics used to measure cost, schedule, and performance of MDAPs and for making recommendations for improvement. This individual also advises on MDAP performance issues prior to certifications at Milestones A and B, prior to FRP, and in consideration of multi-year procurement decisions. Director PARCA accomplishes root cause analysis for MDAPs to determine causes for shortcomings in cost, schedule, or performance, including unrealistic performance expectations; unrealistic baseline estimates for cost or schedule; immature technologies; unanticipated design, engineering, manufacturing, or technology integration issues in program performance; changes in procurement quantities; inadequate

funding or funding instability; or poor PM performance (Government and/or contractor). Director PARCA must report annually (initially March 2010) on root cause analyses for MDAPs and submit to congressional defense committees a report of activities undertaken during the preceding year.

Director of Defense Research & Engineering (DDRE; WSARA, 2009, § 104). Together with the Director of Developmental Test and Evaluation (Director DT&E), DDRE reviews and assesses the technological maturity and integration risks of MDAPs. The WSARA requires annual reports (initially March 2010) to the SECDEF and the congressional defense committees. This strongly encourages MDAs, PEOs, and PMs to not permit programs to move forward until they are technologically ready.

Director of Systems Engineering (WSARA, 2009, § 102). This director is required to develop policy and guidance for systems engineering master plans for MDAPs in support of life-cycle management, sustainability, and reliability growth in contractor proposals. This directly relates to life-cycle cost (LCC) because of the focus placed on sustainability cost, typically the largest component of LCC. This is further discussed in the Other Documents section later in this paper, specifically RADM (R) Don Eaton's published perspective that poor reliability estimates distort true sustainment costs and that poor reliability is a large cost driver. If the contention is accurate that poor reliability estimates are a major cost driver, this should soon begin to appear in the root cause analyses that are mandated for programs that experience significant cost overruns.

Director DT&E and Director of Systems Engineering are required by WSARA to issue guidance on and detailed measureable performance criteria related to Systems Engineering Master Plans (SEMPs) and Developmental Test and Evaluation (DT&E) plans for MDAPS. Among these measureable performance criteria would likely be reliability, availability, maintainability, and related O&S costs; WSARA mandates establishment of a database to record and track weapon system performance data (WSARA, 2009, § 102).

JROC. The WSARA specifically charges the SECDEF to ensure that the JROC is engaged in consideration of trade-offs among cost, schedule, and performance objectives (§ 201). It was noted in our 2003 R-TOC report that the JROC was not focused on TOC and that the leadership was not "speaking with one voice" concerning the importance of TOC (Boudreau & Naegle, 2003, p. 49). This now appears to have been addressed as a matter of law.

Milestone Decision Authority (MDA). The WSARA mandates that MDA ensure appropriate trade-offs among cost, schedule, and performance objectives to increase confidence that the program is affordable (WSARA, 2009, § 201). The words are straightforward and unambiguous, but the interpretation of the "cost" element must be correctly applied to system life-cycle cost, not procurement cost.

Competition Throughout the Life Cycle (WSARA, 2009, § 202). The WSARA identifies 10 different approaches that may be incorporated into a MDAP acquisition strategy to ensure competition be used if cost effective. The list includes competitive prototyping; dual-sourcing; unbundling of contracts; use of modular, open architecture to enable competition for upgrades; use of build-to-print approaches; and acquisition of complete TDPs—along with several other approaches. These suggested measures involve competition among prime contractors and also among subcontractors at such tiers as appropriate. The WSARA views competition as extending into operations and sustainment of MDAPs.

Competitive Prototyping (WSARA, 2009, § 203). The WSARA mandates that MDAPs include competitive prototyping in their acquisition strategies prior to Milestone B (MS B) approval—that is, during the Technology Development Phase—unless waived by the Milestone Decision Authority. Waivers are specifically limited to cost effectiveness or failure to meet critical national security objectives. In the event of a waiver of competitive prototyping because it is not cost effective, non-competitive

prototyping of the system is still required prior to MS B, if benefits exceed cost and are consistent with national security objectives (WSARA, 2009, § 203). The importance of competitive prototyping is that contractors will feel competitive pressure earlier in the process. While generally considered good for the taxpayer, it is arguable that competitive prototyping has not resulted in cost reduction for the Joint Strike Fighter (JSF), which engaged in competitive prototypes but afterward experienced significant cost growth. While there may be multiple reasons for the cost growth experienced, the JSF program may reflect contractor "buy-in," long a problem in DoD acquisition programs.

Milestone Decision Authority Certifications and Follow-On Notifications (WSARA, 2009, § 204). Title 10 U.S.C. 2366a has been tightened, requiring in the MDA's Milestone A (MS A) certifications that Congress be notified if a program experiences or anticipates a cost slip of 25% or anticipates a schedule slip of 25% or more in meeting Initial Operational Capability (IOC). The MDA shall notify Congress within 30 days of identifying root causes and appropriate performance measures to guide the rest of the program development, and specifically addressing (a) the essentiality of the program to national security, (b) the lack of less costly alternatives, (c) new estimates of reasonable cost and schedule for the program, and (d) the adequacy of the program's management structure to control program cost and schedule.

Milestone B Certification Modification (WSARA, 2009, § 205). For programs that have gone through MS B decision, 10 U.S.C. 2366b certification has been amended by WSARA to require that Congress be notified of waivers by the MDA, in writing within 30 days, explaining the basis for a waiver. The MDA must review a troubled program at least annually to determine the extent to which the program is satisfying the certification terms, until such time as the MDA determines that the program has satisfied all the elements of the certification. Budget documents submitted to Congress or the President must be clearly marked as not fully satisfying the certification until the program has made the necessary corrections.

Nunn-McCurdy Cost Breach Modifications (WSARA, 2009, § 206). Title 10 U.S.C. 2433 (generally known as the Nunn-McCurdy Cost Breaches) has been amended by WSARA to require that the MDA consult with the JROC regarding program requirements. Then the MDA must determine the root cause or causes of the cost growth and, in consultation with Director CAPE, assess the following: the cost of completing the program with and without reasonable modification, the rough order of magnitude of proceeding with an alternative system or capability, and the need to shift funds from other programs due to the cost growth. There is a presumption of program termination unless the MDA notifies Congress of a waiver within 60 days. If the program is not terminated, the Secretary shall restructure the program, rescind the most recent milestone approval, require a new milestone approval, and require onerous additional program reviews.

The WSARA of 2009 Summary. There is no doubt that the demands made in WSARA increased the rigor and discipline required in acquisition and will be reflected in more careful cost estimation, increased caution in reviewing technological maturity before advancing programs to the next acquisition step or phase, better systems engineering and test planning, and renewed reliance on competition. All of these facets have the potential to better control life-cycle cost. Conversely, all the same facets introduce the potential for added bureaucracy and unnecessary delay. The WSARA initiatives address past shortcomings in MDAP acquisitions that have contributed to the increase of LCC. Whether these initiatives will reduce cost through better management or increase cost through additional bureaucracy remains to be seen.

National Defense Authorization Act for Fiscal Year 2010, Section 805 (Naegle & Boudreau, 2011)

The National Defense Authorization Act for Fiscal Year 2010 has special relevance to life-cycle cost, as will be explained. In this law, Congress mandated the product support manager (PSM) participation in MDAPs. The law emphasized that the PSM works for the PM, but is also specifically tasked to focus on product

sustainment (O&S) cost. This law increased the stature of the program's chief logistician, the individual who is responsible for developing a product support strategy for the warfighting system. Although a logistician, the PSM is responsible for conducting cost analyses to validate the product support (sustainment) strategy, including cost-benefit analyses that are described in OMB Circular A-94. The PSM is tasked to balance PBL support for optimization. He or she must review and revalidate product support strategies prior to a change in strategy or every five years (NDAA, 2010, § 805). The congressional conferees recognized that product support encompasses a wide range of logistics functions, including readiness, reliability, availability, and logistics burden (footprint) reduction—all of which explicitly or implicitly impact ownership cost (Kobren, 2010, p. 192). The NDAA for FY 2010 very apparently established a position within the MDAP PM office that is responsible for sustainment cost, to include reliability, which directly influences sustainment cost.

Duncan Hunter National Defense Authorization Act for Fiscal Year 2009, Section 814, Configuration Steering Boards for Cost Control Under Major Defense Acquisition Programs (Naegle & Boudreau, 2011)

DoD Instruction (DoDI) 5000.02 of December 2008 very succinctly promulgates the requirements of section 814 of the Duncan Hunter NDAA for FY 2009, specific to the formation of configuration steering boards (CSBs), as follows:

> The Acquisition Executive of each DoD Component shall establish and chair a CSB with broad executive membership including senior representatives from the Office of the USD (AT&L) and the Joint Staff. Additional executive members shall include representatives from the office of the chief of staff of the Armed Force concerned, other Armed Forces representatives where appropriate, the military deputy to the CAE and the Program Executive Officer (PEO) (section 814 of P.L. 110-417, Reference (w)).
>
> (1) The CSB shall meet at least annually to review all requirements changes and any significant technical configuration changes for ACAT I and IA programs in development that have the potential to result in cost and schedule impacts to the program. Such changes will generally be rejected, deferring them to future blocks or increments. Changes shall not be approved unless funds are identified and schedule impacts mitigated.

(2) The PM, in consultation with the PEO, shall, on a roughly annual basis, identify and propose a set of descoping [sic] options, with supporting rationale addressing operational implications, to the CSB that reduce program cost or moderate requirements. The CSB shall recommend to the MDA (if an ACAT ID or IAM program) which of these options should be implemented. Final decisions on descoping [sic] option implementation shall be coordinated with the Joint Staff and military department requirements officials. (USD[AT&L], 2008c, Enclosure 2, p. 30, para 9.d.)

This law introduces a strong bias toward limiting design changes to systems. Note the Service user representative is not named as a member of the CSB. The presumption may be that the user would tend to encourage requirements growth and costly changes. The CSB, for its part, will listen to the proposed change and make the board recommendations to the program MDA. In Part 2, the PM is *directed* to propose de-scoping options to reduce cost and requirements. The MDA is required to coordinate changes with the Joint Staff and component requirements officials (i.e., user representatives). The wording clearly indicates a bias against changes that will increase cost, or at the least deferring such changes to a future block or increment.

Carl Levin and Howard P. "Buck" McKeon National Defense Authorization Act for Fiscal Year 2015

The specific discussion referenced below is from Section 801, Modular Open Systems Approaches in Acquisition Programs; Section 831, Chief Information Officer Authority Enhancements; Section 832, Enhanced Transparency and Improved Risk Management in Information Technology Investments; Section 833, The Tenets of Federal Information Technology Acquisition Reform Act—FITARA

The 2015 NDAA included some specific language impacting software acquisition and software-related TOC, including the tenets of the Federal Information Technology Acquisition Reform Act Bill. Section 801 put into law the requirement for DoD software-intensive systems to implement an "open systems" structure to the maximum extent possible. Specifically,

> The Under Secretary for Acquisition, Technology, and Logistics shall review current acquisition guidance, and modify such guidance as

> necessary to (A) ensure that acquisition programs include open systems approaches in the product design and acquisition of information technology systems to the maximum extent practicable; and (B) for any information technology system not using an open systems approach, ensure that written justification is provided in the contract file for the system detailing why an open systems approach was not used. (NDAA, 2015, p. 136)

Sections 831 through 834 codified the tenets of the FITARA bill, as it failed to be passed into law separately. The FITARA was designed to apply only to non-tactical information technology (IT) systems, but DoD systems are rarely separated so cleanly. For example, a system for DoD contracting might appear to be non-tactical until contingency contracting teams are deployed in support of the theater commander in a tactical situation. Also, non-tactical IT systems are integrated onto tactical weapon systems such as the FAA air traffic control system, which is necessarily integrated into tactical aircraft.

The FITARA tenets include enhanced authority for chief information officers (CIOs), directing Governmental CIOs to have more authority in the acquisition of IT systems (Section 831). The DoD was provided specific exemption to this requirement, but with the examples provided above, there would be opportunities for overlap in acquisition authority.

Section 832 directs enhanced transparency and improved risk management for "covered" IT systems, again exempting DoD "national security" systems (NDAA, 2015, p. 150). While clearly exempting some DoD systems, the DoD remains subject to the improved risk management for many of their systems.

TOC Reports

GAO/T-NSIAD-98-123 and other GAO reports on Knowledge Point Management (Naegle & Boudreau, 2011)

Knowledge point management can be used to avoid program delays and the additional cost that accompanies schedule delays. For more than 12 years, the GAO has advocated the use of knowledge point management to guide development of warfighting systems and to control the advancement of programs until said systems have demonstrated their *readiness* to proceed to the next step in the development process (*Defense Acquisition: Improved Program Outcomes*, 1998). The three knowledge points recommended by the GAO are described in the following paragraphs.

Knowledge Point 1 occurs near Milestone B. The user's requirements must be synchronized with technology that is mature enough to support the endeavor, allow sufficient time scheduled to succeed, and provide sufficient funding to complete the development (GAO, 2003, p. 16). This knowledge point became relatively better understood when the *Technology Readiness Level Deskbook* was published in 2005 (Deputy Under Secretary of Defense for Science and Technology [DUSD(S&T)], 2005). Matching requirements against resources is a matter of discipline, and having the requisite knowledge before proceeding on is necessary because if any one of the several elements is absent (such as the application of required technologies while they are still immature), the program will likely be delayed and the impact on cost may be severe. Continuing GAO reviews have shown that Knowledge Point 1 demands enormous discipline that has, unfortunately, often been beyond the discipline demonstrated by DoD leadership over many years. In addition, software *development* (not reuse, commercial off-the-shelf [COTS], or government off-the-shelf [GOTS]) tends to behave as a new, immature technology, with each effort started from scratch. To combat this inherent problem, potential MDAP and MAIS software developers must undergo a maturity evaluation like the Software Engineering Institute's (SEI) Capability Maturity Model–Integrated (CMMI) and achieve a certain level of maturity through independent evaluation. For the

Capability Maturity Models, the potential software developer must achieve Level 3 or higher to be eligible to compete for the development (USD[AT&L], 2003]). It is also advisable for the PM office to be similarly evaluated using something like the SEI's CMMI-ACQ for acquisition activities, to help minimize the maturity risk with developing software. More often than not, programs are authorized to move into the Engineering and Manufacturing Development (EMD) phase before the technology is sufficiently mature to support a detailed design.

Knowledge Point 2 occurs when the design demonstrates that it is able to meet performance requirements. The design must be stable (i.e., 90% of the engineering drawings must be complete) and testing must show that the system performs at an acceptable level (GAO, 2003, p. 16). Although it would seem that completion of 90% of the engineering drawings is not a severe metric, this is quite demanding; failure to abide by this knowledge point is likely to slow down prototype build and result in prototype test failures caused by designs that were not quite ready for prime time.

Knowledge Point 3 occurs when the system can be manufactured within cost, schedule, and quality targets and it operates reliably (GAO, 2003, p. 16). In statistical process control terms, critical manufacturing processes are in control and consistently producing within quality standards and design tolerances. Reliability is demonstrated in iterative testing (i.e., comparison testing of manufactured product). This point should be demonstrated during LRIP, prior to the FRP decision. Failure to achieve this knowledge point will result in manufacturing delays, high costs of reworking or repairing manufacturing defects, and customers unhappy with the weapon system quality.

Knowledge Point Management is not new. The GAO did not invent the approach in 1998. It borrowed the idea from industry, recognizing that the technique should, and could, be applied to DoD acquisition. Recent changes to the Defense Acquisition System have largely embraced Knowledge Point Management. Getting acquisition programs synchronized with this approach has not happened overnight and is unlikely to happen for all programs if not strictly enforced by DoD leadership.

Evolutionary Acquisition. The use of evolutionary acquisition fits conveniently with Knowledge Point 1, discussed previously. Sometimes technology does not become mature as soon as hoped. Depending on the circumstances, technological immaturity might delay an MS B decision and the associated program new-start. In some cases, a technology that matures slower than needed may be substituted by an alternative technology that is mature and immediately available. Plainly, this hinges on whether or not the developing system can result in an increment of useful warfighting capability—as determined by the sponsor/user. Even when this happens, the program faces a difficult path that requires "extra" milestones that are exhausting to a program office staff. Such is the nature of evolutionary acquisition—avoiding one dilemma and replacing it with another. The evolutionary approach places heavy demands on a program office, which must prepare for a series of otherwise unnecessary milestones. Is it worth it?

The temptation might be to move the program ahead, betting that the required technology solution will miraculously arrive or mature just in time. While miracles sometimes happen, they should not be the anticipated substitute for a sound, well-planned, and executable strategy.

The logistics impact cannot be ignored, either. The result will either be multiple configurations or an expensive modification/upgrade program. Such impacts might play out for many years or even for the lifetime of the warfighting system. This may be associated training issues, repair parts configuration issues, software patches, and operational impacts. The cost of evolutionary acquisition could conceivably approach or even exceed the original cost of the program delay.

The right answer in acquisition depends on the circumstances. The effect on ownership cost should *always* be one of the metrics used to select the best course of action.

GAO Report 10-717 (Naegle & Boudreau, 2011)

In July 2010, the GAO published *Defense Management: DOD Needs Better Information and Guidance to More Effectively Manage and Reduce Operating and Support Costs of Major Weapon Systems* (GAO, 2010b). This report painted a dreary picture of relevant databases. The GAO found that important O&S cost-estimate documents had not been retained and that there were apparent gaps in the DoD's ability to capture actual O&S costs through the Services' Visibility and Maintenance of Operations and Support Costs databases (VAMOSC; GAO, 2010b, p. 16). Data in VAMOSC and other Service information systems or sources was inaccurate and incomplete (GAO, 2010b, pp. 16–20). The report stated that the important MDAP system life-cycle cost estimates were not being routinely retained or updated, nor was there a policy requiring that this be done. The GAO pointed out that there were no agreed-to O&S cost elements or metrics for tracking and assessing actual O&S cost performance for the various categories of weapon systems, but it noted that the Services should be required to collect and assess such data and maintain it in their VAMOSC databases. The GAO singled out the Army in particular as needing to develop and implement a strategy for improving its VAMOSC system. On August 19, 2010, Director CAPE was quoted by *Inside the Pentagon* as asking for relief from the requirement to establish O&S baselines for warfighting systems, ostensibly reporting that this initiative would be "infeasible and not advisable" (Mishory, 2010) because the VAMOSC database was severely flawed. Director CAPE appears to be in agreement with the GAO that sustainment data is flawed or missing and not suitable for rigorous analysis and assessment. A review of the GAO report suggests an array of difficulties in comparing actual data to baselines. Aircraft systems reviewed by the GAO appeared to cost less than expected because the quantities operating in the fleet were generally different (usually fewer) than expected (GAO, 2010b, p. 21). However, costs also were affected by unexpected changes in operational tempo (specifically, flying hours; GAO, 2010b, p. 22). Although both those factors might upset budget predictions, they need not upset performance predictions; rather, if shown as "cost per usage," reasonable comparisons might show the weapon system's performance against

baseline performance. Cost per mile or cost per flying hour or round fired could be compared to early cost estimates, as-tested costs, and changes in cost per year. Such comparisons would never be perfect, but they would suggest whether a weapon system was performing within the expected range.

VAMOSC data, by its very nature, is collected from many and varied locations—sometimes in garrison or home port, sometimes in operational and combat areas. There should never be an expectation of perfect or highly refined data, but, outside of combat areas, data ought to be collected that is "good enough" to support assessments as to whether equipment is operating in the expected performance range, if metrics are established for expected cost drivers. As suggested in our 2003 report, sample data collection (SDC), although expensive, provides a method for improving the accuracy of logistics and O&S cost data (Boudreau & Naegle, 2003).

Looking specifically at aviation systems across the Services, the GAO reported that most systems had no record of O&S cost estimates related to key milestone decisions. Two aircraft systems, the Air Force F-22A fighter and the Navy F/A 18F/G, did have some recorded O&S cost estimates (GAO, 2010b, pp. 24–26). The two cited examples suggest the seriousness of O&S cost-estimating inaccuracy and/or cost growth. F-22A actual cost per flight hour in 2007 was $55,783—67% higher than the $33,762 that had been projected in the 2007 President's Budget. Similarly, on a flight hour basis, the Navy F/A 18E/F cost $15,346 per flight hour of operation—40% higher than the $10,979 predicted in 1999.

GAO-08-1159T, *Defense Acquisitions: Fundamental Changes Are Needed to Improve Weapon Procurement Outcomes* (Naegle & Boudreau, 2011)

In his testimony, the GAO Director of Acquisition and Sourcing Management, Michael Sullivan, succinctly identified systemic problems that led to poor acquisition outcomes (GAO, 2008). His findings identified disconnects in the three systems that are essential to the acquisition of military weapons—the planning, programming, budgeting, and execution process (PPBE), the Joint Capabilities Integration and

Development System (JCIDS), and the Defense Acquisition System. He further characterized a breakdown of systems engineering at critical junctures, referred to as knowledge points. He also described a culture in the Services and the DoD that incentivizes overpromising system performance and underestimating cost and schedule, a pervasive problem across the DoD for many years.

DoD Weapon System Acquisition Reform Product Support Assessment (Naegle & Boudreau, 2011)

In the Product Support Assessment Team's (PSAT's) November 2009 report, *DoD Weapon System Acquisition Reform Product Support Assessment*, the team listed eight areas to improve product support. At least five of the areas impact system life-cycle cost (e.g., Product Support Business Model, Metrics, Operating and Support Costs, Analytical Tools, and Human Capital; PSAT, 2009, pp. 12–13).

Institute for Defense Analyses Study: *The Major Causes of Cost Growth in Defense Acquisition* (Naegle & Boudreau, 2011)

The 2009 Institute for Defense Analyses (IDA) study, led by Gene Porter, examined 11 MDAP systems that had exhibited significant cost growth between 1995 and 2006. The primary causes of cost growth stemmed from two defects: "weaknesses in management visibility, direction, and oversight" and "weaknesses in initial program definition and costing," neither of which was a new phenomenon (Porter et al., 2009, pp. ES-6—ES-14). Much of the blame for the first weakness was "a general lack of discipline" (Porter et al., 2009, p. ES-6).

Porter et al. (2009) make a series of recommendations that are intended to address the causes of cost growth reflected in their study; their recommendations are supportive of the goals of WSARA of 2009 (pp. ES-15–ES-18).

Other Documents (Naegle & Boudreau, 2011)

In his memorandum, *State of Reliability,* Dr. J. Michael Gilmore, the Director of Operational Test and Evaluation (DOT&E, 2010), made the link that poor reliability is a major contributor to LCC. The implication is that the long-held 28-72 LCC statistics could be altered by front-end attention to reliability growth. That is, investing more RDTE funding in reliability improvement at the front end could result in higher reliability components that would cost less to operate, malfunctioning less often. The remarkable thing here is that program leadership has tried to improve reliability in many, if not all, programs. Gilmore made reference to a recently published reliability standard, ANSI/GEIA-STD-0009, which should be employed. He quoted a May 2008 Defense Science Board (DSB) report, which stated that "high suitability (reliability) failure rates were caused by the lack of a disciplined systems engineering process, including a robust reliability growth program" (DSB, 2008, in the task force chairman's cover letter). The DSB further emphasized that the "single most important step … is to … execute a viable systems engineering strategy from the beginning, including a robust reliability, availability, and maintainability (RAM) program" (DOT&E, 2010, pp. 1–2)

Gilmore made his case further by stating,

> I understand that directing use of ANSI/GEIA STD-0009 is a change from business as usual. That change is urgently needed. Requiring the use of 0009 is appropriate for the following reasons:
>
> - 0009 is credible. To obtain an ANSI certification, 0009 was peer reviewed by 350 subject matter experts (SMEs) from all walks of the reliability community, including government, Services, academia, and industry.
> - 0009 is new, different, necessary. ANSI/GEIA STD-0009 is not similar to MIL-STD-785B. The two standards are quite different, and MIL-STD-785B will not suffice. MIL-STD-785B required a "level-of-effort" and discrete tasks, but not system engineering processes. MIL-STD-785B had no systematic processes to identify and mitigate failure modes throughout the product life cycle. 0009 corrects the failings of 785B.

- 0009 has become a model for others. Since publication of ANSI/GEIA STD-0009, major standards such as SAE JA 1000 and IEEE 1332 are now being rewritten to embrace the science-based, closed-looped approach of ANSI/GEIA STD-0009.

- 0009 has been formally adopted by DoD for use (August 20, 2009). ANSI/GEIA STD-0009 will ensure a systems level approach to identify and mitigate failure modes until requirements are met. (DOT&E, 2010, p. 3)

In his own words, Gilmore has publically entered into the reliability dialog because

> discussions that have occurred among our staffs participating in the re-convened Reliability Working Group indicate that there is some question as to whether reliability is an important issue, and there also appear to be questions about the merits of the reliability standard ANSI/GEIA-STD-0009. (DOT&E, 2010, p. 1)

Gilmore emphatically stated in the very next paragraph of his memo that there is no question about it. That is, defense acquisition systems completing research and development (R&D) are often not reliable, and he linked poor reliability to sustainment costs that are higher than necessary (DOT&E, 2010, p. 1). This is reflective of findings in RADM (R) Don Eaton's 2004 paper, discussed in this section.

RADM Don Eaton, retired Arthur Chair in Logistic Management at the Naval Postgraduate School, said in a July 24, 2010, e-mail, "If we thoughtfully analyzed the FOMs [figures of merit] of COST, SCHEDULE AND PERFORMANCE we would always conclude poor reliability is THE dominant cost driver as well as a key player in mission failure." In his August 2004 paper, *Improving the Management of Reliability*, he provided a stunning example from naval aviation, the trailing edge flap actuator for the F/A 18 A-D. He pointed out that the component reliability was set at 4,000 hours mean time between failure (MTBF). In operation, the demonstrated performance in MTBF was 138 hours, 3.45% of what it was supposed to be (Eaton, 2004, pp. 5–6). RADM Eaton did not attempt to calculate the impact to sustainment cost because that was not the purpose of his paper. Nevertheless, without calculating the impact in dollars, one can see that such poor performance reflects in significantly increased costs in maintenance man-hours for repair, repair parts

consumed, transportation of repair parts and/or replacement components, and required stockage levels that had to be maintained, not to mention the impact on the aircraft's mission availability rate. Such examples are not unique to aircraft, or to the Navy. Many, if not all, programs have reliability "bad actors" that need to be redesigned and replaced because of what they are costing in maintenance time, repair parts expense, and transportation. This situation could be improved by rigorous reliability improvement programs during system development, as described in the statements by Gilmore referred to previously. This would require disciplined leadership, PMOs determined to get the design right, and user insistence that reliability goals are set—and achieved—for warfighting systems.

Reliability improvement is bolstered by the involvement of product support managers as encouraged in the NDAA for Fiscal Year 2010, Section 805. The reliability improvement process can be enhanced by the use of collaborative tools to involve life-cycle logistics professionals and make available repair parts databases to sharpen design decisions. This effort can be further helped by Pareto analysis—that is, focus on the cost drivers, primarily the expensive items that break more often than predicted. This approach can be used early in the design process, too, by searching systems command and DLA databases to examine performance of similar or predecessor systems.

It is easy for field users, maintainers, and PMs to visualize cost databases that can be used to identify cost drivers in fielded, legacy systems. This is important work and a principal focus of VAMOSC databases, maintained by each of the Services. However, it must also be recognized that O&S databases are needed to support early O&S calculations of emerging systems, still in pre-acquisition. In her 2010 report, Marti A. Roper discussed the need for databases that support acquisition cost estimates—down to subsystem or component levels, showing cost ranges. Such a knowledge base is critical for the development of follow-on systems so that known cost drivers can be addressed for potentially significant life-cycle cost savings with deployment of the replacement system. Roper (2010) referred to this as capabilities-based parametric data analysis (pp. 71–73)

THIS PAGE INTENTIONALLY LEFT BLANK

DoD Policy

In response to the laws and guidance published by Congress, the OSD develops supporting policies and guidance to implement the intention of Congress. For the acquisition community, the USD(AT&L) provides the leadership and direction for the implementation.

The OSD implemented the 2009 version of WSARA on December 4, 2010, through the USD(AT&L) publication of Directive Type Memorandum (DTM) 09-027 (USD[AT&L], 2009b). About 10 months later, on October 21, 2010, the USD(AT&L) amended the original document, establishing a date by which the DoDI 5000.02 had to be revised (USD[AT&L], 2010c).

In addition to re-publishing the DoD Directives to reflect the new guidance, two consecutive USD(AT&L) executives published specific guidance in the Better Buying Power (BBP) series of implantation memoranda, detailed in the following section.

Better Buying Power (1.0) (Naegle & Boudreau, 2011)

Corollary to WSARA implementation, the USD(AT&L) published the *Implementation Directive for Better Buying Power—Obtaining Greater Efficiency and Productivity in Defense Spending* (USD[AT&L], 2010d). The intent of this implementation directive was to reach beyond WSARA mandates to obtain greater affordability-based decision-making in warfighting system programs. Pertinent specifics are as follows.

- Mandate affordability as a requirement. PMs are now required to treat affordability like a Key Performance Parameter (KPP) at Milestone A. The affordability target is to be stated in two metrics: average unit acquisition cost and average annual operating and support cost per unit. These metrics will be the basis for pre-MS B decision-making and systems engineering trade-off analysis to establish cost and schedule trade space. Such a mandate requires a database similar to the one Roper (2010) described (pp. 71–73). This will provide a basis for comparison against the applicable portfolio or mission area, and will reflect acquisition and O&S budget suitability to absorb the proposed program new start.

Analysis must address specific adjustments to fit new programs *affordably* into their portfolio or mission area (USD[AT&L], 2010d, p. 1).

- The MS B acquisition decision memorandum will include an affordability requirement for acquisition cost and O&S cost that will be the functional equivalent to a KPP and will be established as Acquisition Program Baseline (APB) metrics (USD[AT&L], 2010d, p. 2).
- Productivity growth through will cost/should cost. Should-cost targets will be set for all ACAT I, II, and III programs under consideration for major milestone decisions. Should-cost targets will be based on thoroughly scrubbed bottom-up assessments, assuming reasonable efficiency and productivity enhancement effort. Should-cost will be used as the basis of contract negotiation and incentives to track contractor and PEO/PM performance annually (USD[AT&L], 2010d, p. 2). Independent cost estimates will establish "forecasts of what a program *will cost* based on reasonable extrapolations from historical experience—to support budgeting and programming" (USD[AT&L], 2010a, p. 3). The motivation for industry is higher profit for better performance.
- Eliminate redundancy within warfighter portfolios. The DoD and the components have begun portfolio reviews to identify and eliminate system redundancy in warfighting systems. This function will be accomplished annually by the military departments and agencies (USD[AT&L], 2010d, p. 2).
- Make production rates economical and stable. This element is intended to synchronize production to portfolio affordability targets set at MS A, as adjusted at MS B, and economic order quantity (EOQ). Production rates will be part of the affordability analysis at MS A and MS B. MS C now requires a range of production rates, and deviation from that range without prior approval will lead to revocation of the milestone (USD[AT&L], 2010a, p. 4).
- Set shorter program timelines. Schedule slips are very expensive and delay the arrival of needed equipment into the hands of warfighters. Unfortunately, long developmental programs have been the norm for many years (USD[AT&L], 2010a, pp. 4–5). For future programs, the program schedule will be set at MS B, consistent with the cost trade-off analysis. This is logical because cost and schedule must be synchronized to meet affordability targets. Deviation from schedule without prior approval will lead to revocation of the milestone (USD[AT&L], 2010d, p. 2).
- Present a competitive acquisition strategy at each milestone. ACAT I, II, III, and IV are all required to include a competitive strategy prior to each milestone and to include reduction of single-bid competitions. The strategy will include discussion of market research, restricted specifications, and adequate time for proposal preparation. A 2% improvement goal of one-bid statistics has been established for 2011 (USD[AT&L], 2010d, p. 4).

- Remove obstacles to competition. Contract officers are required to conduct negotiations with all single-bid offerors, unless waived by the Head of Contracting agency (HCA), and the basis of negotiation shall be cost or price analysis, using non-certified data. Component or agency competition advocates are required to achieve an improvement rate of 10% per year in effective competition (USD[AT&L], 2010d, p. 4).
- Require open systems architecture and acquisition of tech data rights. Use of open system architecture and tech data rights will both be pursued to ensure the programs' lifetime consideration of competition. The results of these initiatives will be reported in the Acquisition Strategy Reports (USD[AT&L], 2010d, pp. 4–5).

Better Buying Power 2.0

Following the implementation memorandum detailed in the previous section, the USD(AT&L) published a second memorandum titled *Implementation Directive for Better Buying Power 2.0—Achieving Greater Efficiency and Productivity in Defense Spending* (USD[AT&L], 2013). This memorandum emphasized the continuation of Better Buying Power 1.0 in seven areas that were detailed above, with guidance for achieving the desired results shown here as sub-bullets:

- Achieve Affordable Programs
  - Mandate affordability as a requirement
  - Institute a system of investment planning to derive affordability
  - Enforce affordability caps
- Control Costs Throughout Product Lifecycle
  - Implement "should cost" based management
  - Eliminate redundancy within Warfighter portfolios
  - Institute a system to measure the cost performance of programs and institutions and to assess the effectiveness of acquisition policies
  - Build stronger partnerships with the requirements community to control costs
  - Increase the incorporation of defense exportability features in initial designs
- Incentivize Productivity and Innovation in Industry and Government
  - Align profitability more tightly with Department goals
  - Employ appropriate contract types
  - Increase use of Fixed Price Incentive contracts in Low Rate Initial Production
  - Better define value in "best value" competitions

- When Lowest Price Technically Acceptable is used, define Technically Acceptable to ensure needed quality
- Institute a superior supplier incentive program
- Increase effective use of Performance-Based Logistics
- Reduce backlog of DCAA Audits without compromising effectiveness
- Expand programs to leverage industry's IR&D

- Eliminate Unproductive Processes and Bureaucracy
  - Reduce frequency of higher headquarters level reviews
  - Re-emphasize Acquisition Executive, PEO and PM responsibility, authority, and accountability
  - Reduce cycle times while ensuring sound investment decisions

- Promote Effective Competition
  - Emphasize competition strategies and create and maintain competitive environments
  - Enforce open system architectures and effectively manage technical data rights
  - Increase small business roles and opportunities
  - Use the Technology Development phase for true risk reduction

- Improve Tradecraft in Acquisition of Services
  - Assign senior managers for acquisition of services
  - Measure productivity using the uniform services market segmentation
  - Improve requirements definition/prevent requirements creep
  - Increase small business participation, including through more effective use of market research
  - Strengthen contract management outside the normal acquisition chain—installations, etc.
  - Expand use of requirements review boards and tripwires

- Improve the Professionalism of the Total Acquisition Workforce
  - Establish higher standards for key leadership positions
  - Establish increased professional qualification requirements for all acquisition specialties
  - Increase the recognition and support of excellence in acquisition management
  - Continue to increase the cost consciousness of the acquisition workforce—change the culture (USD[AT&L], 2013, p. 3)

Better Buying Power 3.0

The third in the Better Buying Power series is BBP 3.0, with implementing memorandum titled *Implementation Directive for Better Buying Power 3.0—Achieving Dominant Capabilities through Technical Excellence and Innovation*, dated April 9, 2015. This version continued the tenets of the other BBP initiatives, with the Honorable Frank Kendall stating,

> Core initiatives focus on: ensuring that programs we pursue are affordable, mandating that our managers identify and pursue "should cost" savings opportunities, providing effective incentives to industry, emphasizing competition, reducing bureaucracy, improving our acquisition of contracted services, and building our professionalism. We will continue all of these efforts. (USD[AT&L], 2015, p. 1)

The major areas emphasized are bulleted as follows, with the implementation guidance specified as sub-bullets:

- Achieve Affordable Programs
  - Continue to set and enforce affordability caps
- Achieve Dominant Capabilities While Controlling Lifecycle Costs
  - Strengthen and expand "should cost" based cost management
  - Anticipate and plan for responsive and emerging threats by building stronger partnerships of acquisition, requirements and intelligence communities
  - Institutionalize stronger DoD level Long Range R&D Program Plans
  - Strengthen cybersecurity throughout the product lifecycle
- Incentivize Productivity in Industry and Government
  - Align profitability more tightly with Department goals
  - Employ appropriate contract types, but increase the use of incentive type contracts
  - Expand the superior supplier incentive program
  - Ensure effective use of Performance-Based Logistics
  - Remove barriers to commercial technology utilization
  - Improve the return on investment in DoD laboratories
  - Increase the productivity of corporate IRAD
- Incentivize Innovation in Industry and Government
  - Increase the use of prototyping and experimentation

- Emphasize technology insertion and refresh in program planning
- Use Modular Open Systems Architecture to stimulate innovation
- Increase the return on and access to small business research and development
- Provide draft technical requirements to industry early and involve industry in funded concept definition
- Provide clear and objective “best value” definitions to industry

- Eliminate Unproductive Processes and Bureaucracy
  - Emphasize acquisition chain of command responsibility, authority and accountability
  - Reduce cycle times while ensuring sound investments
  - Streamline documentation requirements and staff reviews
  - Remove unproductive requirements imposed on industry

- Promote Effective Competition
  - Create and maintain competitive environments
  - Improve DoD outreach for technology and products from global markets
  - Increase small business participation, including more effective use of market research

- Improve Tradecraft in Acquisition of Services
  - Strengthen contract management outside the normal acquisition chain—installations, etc.
  - Improve requirements definition for services
  - Improve the effectiveness and productivity of contracted engineering and technical services

- Improve the Professionalism of the Total Acquisition Workforce
  - Establish higher standards for key leadership positions
  - Establish stronger professional qualification requirements for all acquisition specialties
  - Strengthen organic engineering capabilities
  - Ensure development program leadership is technically qualified to manage R&D activities
  - Improve our leaders’ ability to understand and mitigate technical risk
  - Increase DoD support for STEM education (USD[AT&L], 2015, p. 2)

In summary, the three USD(AT&L) BBP initiatives are focused on improving the acquisition process, including the critical TOC performance from development through disposal. While very broadly stated, there are numerous tenets that would and should impact software TOC—both development and sustainment. Many of these policy directives will be analyzed against the software acquisition and sustainment environments to illustrate some specific challenges.

Software Acquisition Process Improvement Programs (Naegle & Boudreau, 2011)

On March 21, 2003, the OSD issued a memorandum to the secretaries of the military departments and other selected recipients, establishing the DoD's Software Acquisition Process Improvement Program and directing each Service to establish a similar program (OSD, 2003).

While clearly focused on the software acquisition process, this memorandum established the need for a more systemic approach that would include requirements development, configuration management, risk management, and test and evaluation, as well as all relevant stakeholders. These are all key tenets in designing systems with desirable TOC characteristics, and including logisticians as relevant stakeholders is necessary to help ensure that the Post Deployment Software Support (PDSS) planning produces a robust and supportable software architecture. As with any other system component, the software design architecture will determine the supportability performance that helps drive the system's TOC.

A Specific Navy Initiative: Gate Reviews (Naegle & Boudreau, 2011)

The Navy has instituted a series of reviews, termed "gate reviews" to better control program development cost. The Navy *Total Ownership Cost Guidebook* (Department of the Navy [DoN], 2010; published concurrently with SECNAVINST 5000.2E) depicts a series of 10 gate reviews that stretch across the pre-acquisition and acquisition phases and into the sustainment phase. Each gate review asks tailored cost questions relevant to the specific life-cycle event (DoN, 2010, pp. 4–32). The complete array of gate reviews is as follows:

- Gate 1—Initial Capabilities Document
- Gate 2—Analysis of Alternatives
- Gate 3—Capability Development Document
- Gate 4—System Design Specification
- Gate 5—RFP for Engineering and Manufacturing Development Contract
- Gate 6 Reviews
    - Integrated Baseline Review
    - Post Critical Design Review
    - Capability Production Document
    - Pre-Full Rate Production Decision Review
    - Sustainment Sufficiency Review(s)

At each gate review, formal design review, and assessment, programs must demonstrate progress toward their affordability initiatives, with strong consideration in mitigation or reduction of TOC. The Navy's intent is to change the culture from what the authors of this working paper perceive as a shortsighted goal of obtaining funds for development and procurement to the more complete perspective of total life-cycle cost affordability.

- Gate Review 1, which is intended to shape the Analysis of Alternatives (AoA) study analysis, requires consideration of O&S costs based on current or similar systems. AoA study TOC guidance is intended to be sufficiently detailed to inform and support the selection of a materiel solution from among the various AoA alternative candidates.
- Intermediate gate reviews are coupled to existing systems engineering and acquisition milestone review points. These reviews become a forum to assess whether program trade-offs and decisions are controlling life-cycle cost and whether the program is continuing on the correct affordability azimuth. Each of the gate reviews requires briefing of specific cost charts, making it unlikely that cost growth and schedule slippage can be obscured.

- The Gate 6 Sustainment Review(s), accomplished post-IOC, examine the warfighting system's actual performance data compared to the system's KPP thresholds and the warfighting system's actual life-cycle cost compared to its prior estimates of ownership cost.

In the aggregate, gate reviews provide for oversight and governance of MDAP system developments. In a wider sense, gate reviews provide a forum for lessons learned regarding TOC while controlling the affordability of individual systems—and, hence, the broader portfolios of warfighting systems—throughout the developmental, production, and sustainment phases of warfighting systems.

THIS PAGE INTENTIONALLY LEFT BLANK

System Software Development and Sustainment Environmental Challenges

While many of the TOC initiatives apply equally to hardware-oriented systems and software-oriented systems, there are some significant differences in both the software development and sustainment environments that need to be considered to gain better software-TOC performance. Understanding these differences in environments will help managers at all levels better manage the acquisition management system and provide the warfighter with systems that are easier and cheaper to sustain.

The Software Engineering Environment (Naegle, 2015)

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 remarks, “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 Technology Readiness Assessment (TRA) and the software Technology Readiness Levels (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 and operational environment cues 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, including TOC 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 TOC-oriented elements of maintainability, upgradability, interoperability, reliability, safety, and security. Most hardware-oriented engineering environments address these critical areas through widely accepted industry standards. For example, all DoD ground combat vehicles use a 24 volt, direct current, negative ground electrical system. Any current or future subsystem requiring vehicle power will automatically be designed to operate using those industry-wide electrical power standards.

The software engineering environment is in stark contrast to even our most advanced hardware-centric engineering environments. For example, in the automotive engineering field, a 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. This engineering maturity helps account for derived and implied requirements not explicitly stated in the performance specification. Most performance specifications do not explicitly address this capability because they would be automatically considered by any competent provider within the mature automotive engineering environment. A mature engineering environment includes design elements and industry-wide standards,

processes, materials, and techniques to which we have grown to expect. A significant problem will exist if we expect the software engineering environment to perform the same way as other, more mature engineering fields (Naegle & Petross, 2007).

As the example above illustrates, many system TOC elements are often standardized across hardware-oriented engineering environments due to the maturity of the sector's engineering maturity. Without the engineering maturity, software sustainability performance and expectations must be specified as part of the requirements generation process. The capabilities-based user requirements and performance-based acquisition requirements are specifically not designed to provide that level of specificity.

The Software Engineering Environment Challenge

The DoD's acquisition management system is designed to garner innovation from the commercial marketplace by leveraging the mature engineering environments present in most disciplines. The DoD develops its requirements beginning with the capabilities-based language provided by the users, then translating them into performance-based language for the RFP. This requirements generation system is purposely designed to allow the maximum contractor flexibility in satisfying the warfighter's needs.

Within the immature software engineering environment, this requirements generation process creates an opportunity for significant misinterpretation, and derived and implied requirements that are not addressed, all resulting in requirements creep that fuels cost increases and schedule slippage. Unlike mature 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 (Naegle, 2015, p. 13).

Addressing the Challenge

There are several necessary steps to effectively address the immature software engineering environment challenge:

1. The acquisition community must understand that the software engineering environment is different, and not mature. This must be an essential part of Knowledge Point 1 and of the Navy gate reviews 1 through 5, detailed earlier. The BBP memoranda help support this step by its direction to "improve the professionalism of the total acquisition workforce."

2. The acquisition community must take active steps to compensate for the software immature engineering environment.

    a. Requirements. Fully develop all requirements so that derived and implied requirements are specified. Sustainment performance including maintainability, upgradability, interoperability, reliability, and safety/security must be specified to improve TOC attributes. With software development, you get what you ask for and very little else.

    b. Operational context. Provide context for the requirements beyond what is provided in the typical OMS/MP. Software engineers need to understand how the system will be used and maintained, how it will be modified and interfaced in the future, which features are critical and which are non-critical enhancers, and how the user expects the system to operate under stressful conditions at the limits of the operational envelope. All of this required information is not available from any other source, and certainly not available in the software engineering environment.

3. The acquisition community must drive and monitor the software architectural design process to a much greater extent than what is needed for hardware-centric system. This is an essential function to reach Knowledge Point 2, and you literally could not achieve Knowledge Point 2 without the ability to drive the software architectural design. This would also be an essential function to effectively pass through the Navy gate reviews 4 through 6.

Estimating Software Size and Cost

Estimating the software size is essential to estimating development and sustainment costs. Unfortunately, estimating size is difficult for any software-intensive effort, and nearly impossible for unprecedented development efforts, including many DoD weapon systems. The DoD often seeks cutting-edge

technologies pursuing dominant capabilities, driving the need for developing unprecedented software development.

The Estimating Software Size and Cost Challenge

Estimating software size, especially for a cutting-edge weapon system, is challenging, at best. It is, however, essential for understanding both software developmental and sustainment costs, so is critical to understanding TOC.

> Software Size Estimating is an important activity in software engineering that is used to estimate the size of an application or component in order to be able to implement other program management activities such as cost estimation or schedule progress. The software engineer is responsible for generating independent estimates of the software size throughout the life cycle. These estimates are sometimes expressed as Software Lines of Code (SLOC), Function Points (FP), or Equivalent Software Lines of Code (ESLOC). An effective software estimate provides the information needed to design a workable Software Development Plan (SDP). This estimate is also input to the Cost Analysis Requirements Description (CARD) process. ("Software Management," n.d. p. 1)

The U.S. Air Force has published a guide for weapon system software development management and describes the software estimating challenge as follows:

> Weapon system acquisition programs routinely aim to develop and deliver unprecedented warfighting capability. This unprecedented capability is often realized by developing complex, SIS [software intensive system] or integrating existing systems and subsystems with other equally complex systems in new ways. Since acquisition programs are planned and estimated when only top-level performance requirements are available, it is extremely difficult to develop high confidence estimates and align expectations early in the program life cycle. Such early estimates are relatively subjective, involve numerous assumptions, and are almost always optimistic since the engineering activities that result in a complete understanding of the work to be accomplished have not been completed. This complete understanding typically does not mature until well into the design phase, and when it does, it usually confirms that initial estimates were optimistic, key assumptions (such as significant reuse) cannot be achieved, more work than planned needs to be done, and the amount of software that has to be developed and/or integrated is growing. (SecAF, 2008, p. 7)

Both the AcqNotes website and the *Air Force Guidebook* offer some guidance in estimating the amount of software that needs to be developed, which is not the only factor in the development cost, but certainly one of the most important.

The AcqNotes website recommends the following:

> There are various ways available to the software engineer to develop a size estimate. It is recommended that multiple techniques be used and the results combined to produce the final size estimate. Methods that can be used of estimating size are:
>
> - **Comparable to existing programs:** Compare the proposed functionality and other similarities to existing programs. If the proposed program has 20% more functionality than one program and 15% less than another, a fairly accurate estimate can be achieved using the actual sizes from the existing programs.
> - **Historical data:** Within a program, historical data of previous developments (estimates and actual) may exist. Since many of the parameters are usually the same (developer team, environment, platform, etc.) this is a good method to compare previous software builds and the proposed code. The more data that is used will increase the accuracy.
> - **Contractor estimate:** It is generally true the contractor has written software similar previously. They often maintain a database of past efforts (estimates and actual) and can produce a very accurate estimate. Since the contractor and the Government have different objectives, their estimate should never be relied on solely.
> - **Expert judgment (Delphi technique):** Engineers that have domain experience and knowledge can often accurately estimate the software size. Without extensive experience however, expert judgment is seldom more accurate than guessing.
> - **Level of effort or schedule:** This method does not really estimate the size to be developed, but rather defines the most that could be developed given unchangeable level of effort or schedule constraints. The software engineer uses productivity rates, integration time and software defect data from recently delivered programs to define the maximum size that could be developed. ("Software Management," n.d., p. 1)

The Air Force guidebook also has recommended considerations for estimating software size:

> The software estimating process consists of a series of activities that include estimating size of the software to be developed, modified, or reused; applying estimating models and techniques; and analyzing, crosschecking, and reporting the results. The following steps should be considered as part of any software estimating process:
>
> - Develop a notional architecture for the system, and identify program requirements likely to be satisfied by software.
> - Identify potential COTS, GOTS, and other sources of NDI software.
> - Identify existing software that will be modified, including the size of the overall software as well as the size of the expected modifications.
> - Identify software that will be newly developed for this program to provide functionality not available from existing software, or to adapt/integrate all the necessary software components.
> - Obtain software size information for all software elements, where size is carefully defined and measured in one of the two standard software size measures: non-comment source lines of code (SLOC) or function points.
> - Assess the uncertainty in the new and modified software sizes, based on historical data (if available) and engineering judgment.
> - Assess the uncertainty associated with the reusability of existing software (COTS, GOTS, and NDI) in the context of the program (see section 3.2.4). Estimate the trade studies, familiarization, and the integration and testing efforts required to accommodate the unmodified reused code.
> - Account for software complexity and the proposed development approach/processes, and assess any overlaps in software builds.
> - Be realistic about expected software productivity and any assumption of significantly higher than historical productivity due to applying the best people, improved/more efficient processes, or new and improved development tools. Past performance, where actual size, cost, and same program or a very analogous program, should be heavily

weighted. It is rare to have the A-team people for a long-duration embedded system development, and new processes and tools often fall short of expectations.

- Apply growth factors to new/modified and reuse software, based on past experience and the level of uncertainty.
- Account for all remaining uncertainties as estimate risks (see section 3.2.2).
- Ensure the estimate includes software support to systems engineering, system and sub-system requirements definition, configuration management, quality assurance, program management, system integration, and system test as appropriate.
- Address the software development life-cycle from software requirements analysis through software-related system integration and testing. The chosen modeling/estimation approach may not address the entire software effort since some commercial parametric models focus on the period starting with the baseline set of software requirements and ending with a fully integrated and tested subsystem/functional software product ready for software/hardware integration and test. Estimate and include any additional effort required to develop, allocate, and analyze the subsystem and software requirements; perform software to hardware (subsystem) integration and test; and perform system integration and test.
- Crosscheck estimate results with other methods such as other models, expert advice, rules of thumb, and historical productivity.
- Improve the estimate over time. (SecAF, 2008, pp. 27 & 28)

Both the AcqNotes and U.S. Air Force size estimating guidance suggest using multiple methodologies to form a more informed estimate of the likely software size of a developmental system. Nearly all of the guidance is dependent on an excellent understanding of the system requirements and operational context.

One common method to estimate the software size on a new developmental program is to use the analogy method, that is, to compare the new system to a similar system that was recently developed, assuming that the software will be

similar in overall size. The following is the first bullet in the AcqNotes software estimating guidance detailed previously in this section. It seems a logical approach, but has not proven particularly accurate in recent history:

> The premise is that the existing system's architecture, complexity, and functions are similar enough to fairly accurately predict the software development resources required for the new system. Unfortunately, this technique has proven to be ineffective as evidenced by the F-22 Raptor development and the follow-on F-35 Joint Strike Fighter (JSF) effort. The two high-performance, supersonic aircraft, have overlapping missions, are significantly similar, and are both developed by the same contractor. The F-22 would seem to be a very good predictor of the F-35 software development effort with the SwTRL [Software Technology Readiness Level] model, but it clearly was not:
>
> 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 would 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. (Naegle, 2015)

The second bullet guidance from AcqNotes indicates that the use of historical data may be useful in estimating a new system's software size. This is particularly challenging for the DoD as the new weapon systems the DoD often pursues have capabilities or features that are unprecedented (cutting-edge technologies). Certainly, there will be many subsystems in which historical data may be a good predictor for software size in existing, identical, or similar subsystems. However, the analogy method uses the historical data of a similar system as a surrogate for actual historical data, but suffers the challenges detailed previously.

The third AcqNotes bullet is “contractor estimates for software size.” The problem with contractor estimates is that the size estimate is needed far before a development contractor would be involved in the process. Of course, market research contractors could be used to garner “contractor estimates,” but this would require two essential preconditions. First, the market research contractor would need an extraordinary amount of requirements, operational context, and design detail on the proposed system to be able to provide to the marketplace to garner reasonably accurate software size estimates. Second, the market research would be conducted with industry members who can only respond to the information provided, so the estimates are only as accurate as the requirements-oriented information provided. In addition, the surveyed companies may be unwilling to provide much detail about their estimate as it could provide competitors with valuable competitive information.

The expert judgement, or Delphi Method (AcqNotes bullet 4), depends on the level of expertise of the engineers providing the estimate and their total understanding of the system to be developed. The DoD may gain access to expert software engineers that are inside the Government or through contracting for such expertise, but the level of understanding is dependent on the requirements generations system and the operational context provided.

There are also numerous parametric models, like Barry Boehm’s Constructive Cost Model (COCOMO), that may be used in an attempt to estimate effort and cost (USC, 2002). COCOMO, like other estimating models, requires a software size estimate to be used. One of the inputs to the model is the Annual Change Traffic (ACT), or the percentage of the software that needs to be accessed for sustainability purposes. Obviously, the model would need to know the software size to perform the percentage calculations.

Because of all of the variables that are needed for the models, they can be quite misleading. For example, the University of Southern California (USC) used the models and then compared actual results to those estimated. They found that COCOMO “demonstrates an accuracy of within 20% of actuals 46% of the time for

effort, and within 20% of actuals 48% of the time for a nonincremental development schedule" (USC, 2002). They found that, with more initial data input, the model accuracy improved to 30% of actuals 75% of the time. Boehm himself stated that "a software cost estimation model is doing well if it can estimate software development costs within 20% of the actual costs, 70% of the time, and on its home turf (that is, within the class of projects to which it is calibrated)" (SecAF, 2008, p. 21).

Obviously, using the results of parametric models alone would not result in the accurate estimates required by the DoD. The BBP memoranda specify "would cost" and "should cost" estimates that the models simply could not accurately produce. The software development cost and schedule estimate would necessarily need to be sufficiently accurate to avoid a Nunn-McCurdy violation in a software-intensive system development program.

Addressing the Challenge

Obviously, a fairly accurate software size estimate is necessary to predict both developmental and sustainment costs on a new system, and it is clear that obtaining an accurate size estimate is significantly challenging. The necessary precursor to software estimation is described earlier in this paper as compensating for the immature software engineering environment. Without more clearly defined requirements and operational context, accurately estimating software size is nearly impossible.

As suggested in both the AcqNotes and U.S. Air Force software estimating guidelines, a multi-faceted approach is needed. To be successful, each approach must be completed with significant discipline and rigorous systems analysis that goes beyond the current practices. If successful, the software size estimate will help predict both developmental and sustainment software costs.

Software Sustainability Architecture

A system's architecture and sustainability performance are strongly linked. As the F/A 18 engine removal example provided earlier illustrates, the airframe

architecture was carefully designed to allow the 20-minute engine removal. Likely, this feature was engineered in response to some specific requirements language.

As introduced earlier, much of the design priority has been delegated to the contractor as the requirements language is capabilities-based on the user side and performance-based on the program management side. The DoD is responsible for driving the architectural design through the performance-based specification language, which requires a very in-depth understanding and development of the requirements passed on to the contractor.

The Software Architecture Challenge

Driving the software architectural design towards improved system TOC performance has numerous and complex challenges. The DoD requirements generation process is designed around the premise that the commercial marketplace has solutions for achieving the system performance specified by the DoD. This philosophy came from the acquisition reforms of the ’90s, when systems were much more hardware oriented, and the associated engineering environments were mature. As the DoD has moved to software-oriented systems, the philosophy did not change, even though the software engineering environment is not mature. This has created a significant mismatch in what the DoD communicates and what it expects to be delivered. Much of the mismatch can be linked to the software engineering immaturity:

> 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 accustom 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. (Naegle, 2014, p. 14)

The DoD also provides the top levels of the work breakdown structure (WBS) to provide cues to the necessary design structures, but like the requirements generation process, the communication through the WBS is often too vague or lacking in necessary detail for the software engineers to understand important aspects of the design.

> 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. (Naegle, 2014, p. 8)

The design metrics are very important to ensure that the software architecture is meeting the warfighter needs and expectations for the new system, including the TOC performance. Too often, this process serves to identify missing requirements or clarify vague requirements, causing significant requirements creep impacting the cost and schedule.

Addressing the Challenge

Again, step one for addressing the software architecture challenge is to understand that the software engineering environment is immature and that the DoD front-end processes must help compensate for that immaturity. The requirements generation process, the Operational Mode Summary/Mission Profile (OMS/MP), the WBS, and the resulting performance specification and Government-specified functional architecture (top levels of the WBS) must drive the software engineer to develop the detailed system architecture to the total needs of the warfighter. The software engineering environment will not compensate for vague or missing requirements and there are virtually no industry-wide standards for sustainability.

Processes to both drive the software architecture and monitor the design activities is unlike the contractor's hardware architecture activities and significantly more critical. Fifty percent or more of the software effort is expended in requirements and architectural design, which is far greater than typical hardware-oriented systems. This means that half or more of the software development resources have been used by the Preliminary Design Review (PDR), which occurs quite early in the developmental process. Requirements creep and software changes after the PDR are significantly disruptive to the design process and are costly in both funding and schedule. In addition, changes occurring after the design is complete are typically accommodated through the use of software patches. While these patches may function adequately, they typically weaken the software structure and add difficulty to the sustainment effort as they add lines of code, are not generally well documented, and add complexity to problem analyses in the deployed system.

Software Sustainment Activities

The Post Deployment Software Support (PDSS) structure—maintainers, software engineering tools, documentation, licenses, and so forth—must all be funded and in place at the initial deployment as software maintenance will likely be required immediately due to the complexity. As demonstrated in Figure 3, most of the DoD software sustainment effort is accomplished through Contracted Logistics Support (CLS) strategies, so the support contracts are critical to system deployment.

As with hardware-oriented systems, the software sustainability performance is significantly defined by the system architecture. The software engineering immaturity means that there are no industry-wide standards for software sustainability, so the DoD must drive the desired sustainability performance into the software design.

The two major components that help determine a system's software sustainment cost are software size (SLOC count) and complexity. Many of the effort estimating tools need the software size to estimate the number of software maintainers that need to be dedicated to the sustainment effort. Complexity factors are then added into the calculations.

The Software Sustainment Challenge

The DoD system acquisition process is driven through the performance-based specifications, program WBS functional architectural cues, and high-level OMS/MP and, therefore, relies heavily on the contractor's expertise backed by the industry's mature engineering environments. This process is not adequate for driving the software architecture to a sustainable design as the immature software engineering environment has no industry-wide standards for sustainability, so software sustainability performance must be totally driven through the DoD front-end processes.

Unlike even the most sophisticated hardware system, the software maintainers must have the same skill sets as the design engineers, and so the DoD is typically contracting for software engineers to maintain the software. The software

sustainment cost factors include maintainers, software tools, license fees, and associated contract costs for most DoD systems. While the non-personnel costs can be considerable, the cost of the maintainers is usually the largest part of the sustainment cost because the DoD is typically contracting for software engineers to maintain the software components.

The events driving the need for software maintenance are not always within the control of the system's PM, as demonstrated with the M1 Abrams tank example, provided earlier. As the DoD continues to network platforms into Systems of Systems (SoSs), each platform is subject to the network's complexities and interoperability requirements.

Addressing the Challenge

The solutions for addressing the software sustainability challenge are rooted in solving the other issues presented in this section, as they all tend to build on one another. The DoD needs to recognize that the software engineering environment is immature, significantly different than the hardware-oriented engineering environments. That immaturity renders much of the DoD front-end processes ineffective for software-intensive systems, so active steps augmenting the standard acquisition processes must be taken to compensate.

TOC performance is being influenced by the ever increasing software functionality of DoD systems, so improving TOC performance means effectively addressing software development and sustainability costs. The software costs and performance are dependent on how effective the acquisition front-end processes address them, and the standard DoD acquisition management system appears to be insufficient for the software components.

Software Initiatives Addressing TOC

The software TOC issues presented, and their underlying causes, call for supplementary Systems Engineering Process (SEP) tools, techniques, and analyses to be applied to the DoD acquisition process. The following sections describe recommended tools, techniques, and analyses that would help address the issues presented. All of these are designed to work within the Defense Acquisition System (DAS).

Controls on Software Development (Naegle & Boudreau, 2011)

1. Driving the Software Requirements and Architectures for System Supportability

While the tools and techniques described in this section were designed for the software components, they would be just as effective for any non-software component because they are Systems Engineering (SE) oriented. The SEP focus used does not attempt to separate software from other components, so all system components would benefit from using these tools and techniques.

a. Software Supportability Analysis

As with hardware system components, software supportability attributes must be designed into the system architecture. Many hardware-oriented engineering fields are now quite mature, so that a number of supportability attributes would be automatically included in any competent design, even if they were not specified by the user community. For example, the state of maturity for the automotive engineering field means that, in any automotive-related program, there would be supportability designs allowing for routine maintenance of system filters, lubricants, tires, brakes, batteries, and other normal wear-out items. There are few, if any, corresponding supportability design attributes that would be automatically included in even the best software construct. Virtually all of the software supportability attributes required must be explicitly specified because they would not likely be included in the design architecture without clearly stated requirements. With

software, you get what you specify and very little else. So how does one ensure that required software supportability attributes are not overlooked?

Logistics Supportability Analysis (LSA), performed extremely early, is one of the keys for developing the system supportability attributes needed and expected by the warfighter. The F/A 18 Super Hornet aircraft was designed for higher reliability and improved ease of maintenance compared to its predecessors ("F/A 18," n.d.) because of warfighter needs for generating combat power in the form of aircraft sorties available. The LSA performed on the F/A 18 determined that a design fostering higher reliability and faster maintenance turnaround time (the engines are attached to the airframe at 10 locations and can be changed in about 20 minutes by a four-man team) would result in more aircraft being available to the commander when needed. The concept for software LSA is no different, but implementing sound supportability analyses on the software components has been spotty, at best, and completely lacking, at worst.

To assist in effective software LSA, a focus on these elements is key: Maintainability, Upgradeability, Interoperability/Interfaces, Reliability, and Safety & Security—MUIRS.

Maintainability

The amount of elapsed time between initial fielding and the first required software maintenance action can probably be measured in hours, not days. The effectiveness and efficiency of these required maintenance actions is dependent on several factors, but the software architecture that was developed from the performance specifications provided is critical. The DoD must influence the software architecture through the performance specification process to minimize the cost and time required to perform essential maintenance tasks.

Maintenance is one area in which software is fundamentally different from hardware. Software is one of the very few components in which we know that the fielded product has shortcomings, and we field it anyway. There are a number of reasons why this happens; for instance, there is typically not enough time, funding,

or resources to find and correct every error, glitch, or bug, and not all of these are worth the effort of correcting. Knowing this, there must be a sound plan and resources immediately available to quickly correct those shortcomings that do surface during testing and especially those that arise during warfighting operations. Even when the system software is operating well, changes and upgrades in other interfaced hardware and software systems will drive some sort of software maintenance action to the system software. In other words, there will be a continuous need for software maintenance in the planned complex SoS architecture envisioned for net-centric warfare.

Because the frequency of required software maintenance actions is going to be much higher than in other systems, the cost to perform these tasks is likely to be higher as well. One of the reasons for this is that software is not maintained by ”maintainers,” as are most hardware systems, but is maintained by the same type of people that originally developed it—software engineers. These engineers will be needed immediately upon fielding, and a number will be needed throughout the lifespan of the system to perform maintenance, add capabilities, and upgrade the system. There are several models available to estimate the number of software engineers that will be needed for support; planning for funding these resources must begin very early in the process. Because the DoD has a very limited capability for supporting software internally, early software support is typically provided by the original developer and is included in the RFP and proposal for inclusion into the contract or as a follow-on Contractor Logistics Support (CLS) contract.

Upgradeability

A net-centric environment composed of numerous systems developed in an evolutionary acquisition model will create an environment of almost continuous change as each system upgrades its capabilities over time. System software will have to accommodate the changes and will have to, in turn, be upgraded to leverage the consistently added capabilities. The software architecture design will play a major role in how effective and efficient capabilities upgrades are implemented, so

communicating the known, anticipated, and likely system upgrades will impact how the software developer designs the software for known and unknown upgrades.

Trying to anticipate upgrade requirements for long-lived systems is extremely challenging to materiel developers, but is well worth their effort. Unanticipated software changes in the operational support phase cost 50 to 200 times the cost in early design, so any software designed to accommodate an upgrade that is never realized costs virtually nothing when compared to changing software later for a capability that could have been anticipated. For example, the Army Tactical Missile System (ATACMS) Unitary was a requirement to modify the missile from warhead air delivery to surface detonation—that is, flying the warhead to the ground. The contract award for the modification was $119 million. The warhead was not new technology, nor particularly challenging to integrate with the missile body. The vast majority of this cost was to reengineer the software to guide the missile to the surface. Had there been an upgrade requirement for this type of mission in the original performance specification, this original cost (including potential upgrades, even if there were 10 other upgrade requirements that were never applied) would have been a fraction of this modification cost.

Interfaces/Interoperability

OA design focuses on the strict control of interfaces to ensure the maximum flexibility in adding or changing system modules, whether they are hardware or software in nature. This presupposes that the system modules are known—which seems logical, as most hardware modules are well-defined and bounded by both physics and mature engineering standards. In sharp contrast to hardware, software modularity is not bounded by physics, and there are very few software industry standards for the modular architecture in software components. This is yet another area in which the software developer needs much more information about operational, maintenance, reliability, safety, and security performance requirements, as well as current, planned, and potential system upgrades. These requirements, once well-defined and clearly communicated, will drive the developer to design a software modular architecture supporting OA performance goals. For example, if a

system uses a Global Positioning System (GPS) signal, it is likely that the GPS will change over the life of the system. Knowing this, the software developer creates a corresponding discrete software module that is much easier and less expensive to interface, change, and upgrade as the GPS system does so.

With the system software modular architecture developed, the focus returns to the interfaces between hardware and software modules, as well as to the external interfaces needed for the desired interoperability of the net-centric force. Software is, of course, one of the essential enablers for interoperability and provides a powerful tool for interfacing systems, including systems that were not designed to work together. Software performing the function of "middleware" allows legacy and other dissimilar systems to interoperate. Obviously, this interoperation provides a significant advantage, but it comes with a cost in the form of maintainability, resources, and system complexity. As software interfaces with other components and actually performs the interface function, controlling it and ensuring the interfaces provide the desired OA capability becomes a major software-management and software-discipline challenge.

One method being employed by the DoD attempts to control the critical interfaces through a set of parameters or protocols rather than through active management of the network and network environment. This method falls short on several levels. It fails to understand and control the effects of aggregating all of the systems in a net-centric scheme. For instance, each individual system may meet all protocols for bandwidth, but when all systems are engaged on the network, all bandwidth requirements are aggregated on the network—overloading the total bandwidth available for all systems. In addition, members of the Software Engineering Institute (SEI) noted,

> While these standards may present a step in the right direction, they are limited in the extent to which they facilitate interoperability. At best, they define a minimal infrastructure that consists of products and other standards on which systems can be based. They do not define the common message semantics, operational protocols, and system execution scenarios that are needed for interoperation. They should not be considered system architectures. For example, the C4ISR domain-specific information (within the JTA) identifies acceptable

> standards for fiber channels and radio transmission interfaces, but does not specify the common semantics of messages to be communicated between C4ISR systems, nor does it define an architecture for a specific C4ISR system or set of systems. (Morris, Levine, Meyers, Place, & Plakosh, 2004, p. 38)

Clearly, understanding and controlling the interfaces is critical for effective interoperation at both the system and SoS levels. The individual PM must actively manage all systems' interfaces impacting OA performance, and a network PM must do the same for the critical network interfaces. Due to this necessity of constant management, a parameters-and-protocols approach to net-centric OA performance is unlikely to produce the capabilities and functionality expected by the warfighter.

Understanding the software interfaces begins with the software architecture; controlling the interfaces is a unique challenge encompassing the need to integrate legacy and dissimilar systems and the lack of software interface standards within the existing software engineering environment. As stated earlier, the architecture needs to be driven through detailed performance specifications, which will help define the interfaces to be controlled. An effective method for controlling the interfaces is to intensely manage a well-defined Interface Control Document (ICD), which should be a Contract Data Requirements List (CDRL) deliverable on any software-intensive or networked system.

Reliability

While the need for highly reliable weapon systems is obvious, the impact on total system reliability of integrating complex software components is not so obvious. Typically, as system complexity increases, maintaining system reliability becomes more of a challenge. Add the complexity of effectively networking an SoS (all of which are individually complex) to a critical warfighting capability that is constantly evolving over time, and reliability becomes daunting.

Once again, the software developer must have an understanding of reliability requirements before crafting the software architecture and developing the software applications. Highly reliable systems often require redundant capability, and this holds true for software components as well. In addition, software problems tend to

propagate, resulting in a degradation of system reliability over time. For example, a Malaysian Airlines Boeing 777 suffered several flight control problems resulting in the following: a near-stall situation, contradicting instrument indications, false warnings, and difficulty controlling the aircraft in both autopilot and manual flight modes. The problems were traced to software in an air data inertial reference unit that was feeding erroneous data to the aircraft's primary flight computer (PFC), which is used in both autopilot and manual flight modes. The PFC continued to try to correct for the erroneous data received, adjusting flight control surfaces in all modes of flight, displaying indications that the aircraft was approaching stall speed and overspeed limits simultaneously, and causing wind shear alarms to sound close to landing (Dornheim, 2005, p. 46). It is critical for system reliability that the software developers understand how outputs from software applications are used by interfaced systems so that appropriate reliability safeguards can be engineered into the developed software.

Software that freezes or shuts down the system when an anomaly occurs is certainly not reliable nor acceptable for critical weapon systems, yet these characteristics are prevalent in commercially based software systems. Mission reliability is a function of the aggregation of the system's subcomponent reliability, so every software subcomponent is contributing to or detracting from that reliability. The complexity of software makes understanding all failure modes nearly impossible, but there are many techniques that software developers can employ when designing the architecture and engineering the applications to improve the software component reliability. Once requirements are clearly communicated to the developers, the software can be engineered with redundancy or "safe mode" capabilities to vastly improve mission reliability when anomalies occur. The key is identifying the reliability requirements and making them clear to the software developers.

Safety & Security

Very few software applications have the required safety margins associated with critical weapon systems used by warfighters in combat situations—where they are depending on these margins for their survival. Typically, the software developers have only a vague idea of what their software is doing and how critical that function is to the warfighter employing the weapon system. Safety performance must be communicated to the software developers from the beginning of development so they understand the link between software functionality and systems safety. For example, suppose a smart munition senses that it does not have control of a critical directional component, and it calculates that it cannot hit the intended target. The next set of instructions the software provides to the malfunctioning system may well be critical to the safety of friendly troops, so software developers must have the necessary understanding of operational safety to decide how to code the software for what will happen next.

Software safety is clearly linked with reliability since software that is more reliable is inherently safer. It is critical that the software developer understands how the warfighter expects the software to operate in abnormal situations, in degraded modes, and when inputs are outside of expected values. Much commercially based software simply ceases to function under these conditions or gives error messages that supersede whatever function was being performed, none of which are acceptable in combat operations.

With software performing so many critical functions, there is little doubt that software applications are a prime target for anyone opposing U.S. and Allied forces. Critical weapon system and networking software must be resistant to hacking, spoofing, mimicking, and all other manner of attack. There must be capabilities for isolating attacks and portions of networks that have been compromised without losing the ability to continue operations in critical combat situations. The software developer must know that all of these capabilities are essential before he or she constructs software architectures and software programs, as this knowledge will be very influential for the software design and application development. The Software

Engineering Institute's *Quality Attribute Workshop* states, "As an example, consider security. It is difficult, maybe even impossible, to add effective security to a system as an afterthought. Component as well as communication mechanisms and paths must be designed or selected early in the lifecycle to satisfy security requirements" (Barbacci et al., 2003, p. 2).

Interoperability challenges are increased when the SoS has the type of security requirements needed by the DoD. Legacy systems and existing security protocols will likely need to be considered before other security architecture can be effectively designed. OA capabilities will be hampered by the critical need for security; both must be carefully balanced to optimize system performance and security. This balance of OA and security must be managed by the DoD and not the software developer.

Physical security schemes and operating procedures will also have an impact on the software architecture. For example, many communication security (COMSEC) devices need only routine security until the keys, usually software programs, are applied; then, much more stringent security procedures are implemented. Knowledge of this security feature would be a key requirement of the developer; he or she must understand how and when the critical software pieces are uploaded to the COMSEC device. The same holds true for weapon systems that upload sensitive mission data just prior to launch.

Residual software on equipment or munitions that could fall into enemy hands presents another type of security challenge that needs to be addressed during the application development. For example, the ATACMS missile air-delivers some of its warheads, leaving the missile body to freefall to the surface. It is very conceivable that the body could be intact and, of course, unsecured. If critical mission software was still within the body and found by enemy forces, valuable information might be gleaned from knowing how the system finds its targets. The Government would certainly want the developer to design the applications in a way that would make anything recovered useless to the enemy, but this is a capability that is not intuitive to the software developers (Naegle, 2006, pp. 17–25).

Effective Software Development Tools Supporting System TOC Analyses

1. Software Engineering Institute's Quality Attribute Workshop

The Quality Attribute Workshop (QAW) is designed to help identify a complete (or as complete as possible) inventory of system software requirements through analysis of system quality attributes. One of the intents is to develop the derived and implied requirements from the user-stated requirements, which is a necessary step when user-stated requirements are provided in terms of capabilities needed as prescribed by the Joint Capabilities Integration Development System (JCIDS) process. A system's TOC, and those elements that contribute to TOC, are system quality attributes. Although obviously important to the warfighter, the associated operations and support, training/education, and facility costs are rarely addressed in much detail and need to be derived from stated requirements or augmented with implied requirements through the QAW process, or something similar.

The QAW helps provide a facilitating framework and process designed to more fully develop the derived and implied requirements that are critical to clearly communicate to potential contractors and software developers. Including a robust LSA process using the MUIRS focus elements, described previously, within the QAW process will likely significantly improve requirements analysis for those associated TOC elements and vastly improve the accuracy of system TOC projections. While improving the system requirements development, QAW is designed to work with another SEI process called the Architectural Tradeoff Analysis Methodology^{SM} (ATAM^{SM}) to further improve the understanding of the system for potential contractors and software developers.

SEI's Architectural Tradeoff Analysis Methodology[SM]

The Software Engineering Institute's Architectural Tradeoff Analysis Methodology[SM] (ATAM[SM]) 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, including TOC 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, Klein, & Clements, 2000, p. 1).

Within the Systems Engineering Process (SEP), the ATAM[SM] provides the critical requirements loop process, tracing each requirement or quality attribute to corresponding functions reflected in the software architectural design. Whether ATAM[SM] 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 will be 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 warfighters' JCIDS stated, derived, or implied requirements.

The ATAM[SM] 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[SM] scenarios are a key to providing the necessary information to answer the first two questions, driving the software engineer to design the architecture to answer the third. This is a critical point at which all of the MUIRS elements need to be considered and appropriate scenarios developed.

The ATAM[SM] uses three types of scenarios: *Use-case scenarios* involve typical uses of the system to help understand quality attributes in the operational context; *growth scenarios* involve anticipated design requirements, including upgrades, added interfaces supporting SoS development, and other maturity needs; and *exploratory scenarios* involve extreme conditions and system stressors, including Failure Modes and Effects Criticality Analysis (FMECA) scenarios (Kazman et al., 2000, pp. 13–15). As depicted in Figure 4, the scenarios build on the basis provided in the JCIDS documents and requirements developed through the QAW process. These processes lend themselves to development in an Integrated Product Team (IPT) environment led by the user/combat developer and including all of the system's stakeholders. The IPT products will include a set of scenarios, prioritized by the needs of the warfighter for system capability. The prioritization process provides a basis for architecture trade-off analyses. When fully developed and prioritized, the scenarios provide a more complete understanding of requirements and quality attributes in context with the operation and support (including all of the MUIRS elements) of the system over its life cycle. A more complete understanding of the system's TOC elements should emerge from this type of analysis.

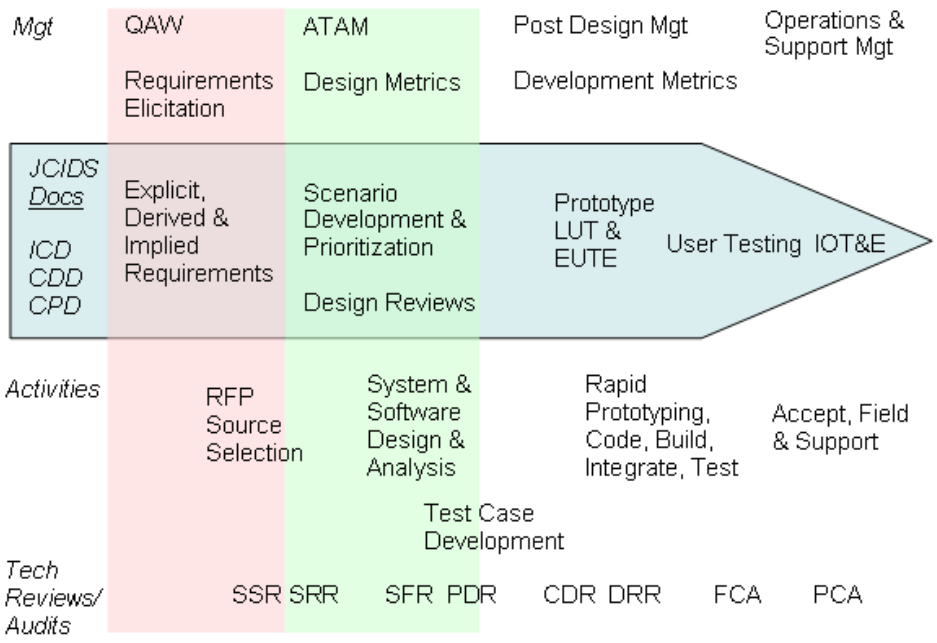


Figure 4. QAW & ATAM[SM] Integration Into Software Life-Cycle Management

Just as the QAW process provides a methodology supporting RFP, source-selection activities, and the Software Specification and System Requirements Reviews (SSR and SRR), the ATAM[SM] provides a methodology supporting design analyses, test program activities, and the System Functional and Preliminary Design Reviews (SFR and PDR). The QAW and ATAM[SM] methodologies are probably not the only effective methods supporting software development efforts, but they fit particularly well with the DoD's goals, models, and SEP emphasis. The user/combat developer (blue arrow block in Figure 4) is kept actively involved throughout the development process—providing key insights the software developer needs to successfully develop warfighter capabilities in a sustainable design for long-term effectiveness and suitability. The system development activities are conducted with superior understanding and clarity, reducing scrap and rework, and saving cost and schedule. The technical reviews and audits (part of the DoD overarching SEP) are supported with methodologies that enhance both the visibility of the necessary development work as well as the progress toward completing it.

One of the main goals in analyzing the scenarios is to discover key architectural decision points that pose risks for meeting quality requirements. Sensitivity points are determined, such as real-time latency performance shortfalls in target tracking. Trade-off points are also examined so that TOC impacts resulting from proposed trade-offs can be analyzed. The Software Engineering Institute explained, "Trade-off points are the most critical decisions that one can make in an architecture, which is why we focus on them so carefully" (Kazman et al., 2000, p. 23).

The ATAM[SM] provides an analysis methodology that complements and enhances many of the key DoD acquisition processes. It provides the requirements loop analysis in the SEP, extends the user/stakeholder JCIDS involvement through scenario development, provides informed architectural trade-off analyses, and vastly improves the software developer's understanding of the quality requirements in context. Architectural risk is significantly reduced, and the software architecture presented at the Preliminary Design Review (PDR) is likely to have a much higher probability of meeting the warfighters' need for capability, including TOC elements.

Together, the QAW and ATAM[SM] provide effective tools for addressing problem areas common in many DoD software-intensive system developments: missing or vaguely articulated performance requirements, significantly underestimated software development efforts (resulting in severely underestimated schedules and budgets), and poor communication between the software developer and the Government (both user and PM). Both tools provide frameworks for more detailed requirements development and more effective communication, but they are just tools—by themselves, they will not replace the need for sound planning, management techniques, and effort. Both QAW and ATAM[SM] provide methodologies for executing SEP Requirements Analysis and Requirements Loop functions, effective architectural design transition from user to developer, and SEP design loop and verification loop functions within the test-case development.

A significant product resulting from the ATAM[SM] is the development of test cases correlating to the use case, growth, and exploratory scenarios developed and

prioritized. Figure 5 depicts the progression from user-stated capability requirements in the JCIDS documents to the ATAM[SM] scenario development, and finally to the corresponding test cases developed. The linkage to the user requirements defined in the JCIDS documents is very strong as those documents drive the development of the three types of scenarios, and, in turn, the scenarios drive the development of the use cases. The prioritization of the scenarios from user-stated Key Performance Parameters (KPPs), Critical Operational Issues (COIs), and FMECA analysis flows to the test cases, helping to create a system test program designed to focus on effectiveness and suitability tests—culminating in the system Operational Test and Evaluation (OT&E). FMECA is one of the focus areas that will have a dynamic impact on TOC analysis because it will help identify software components that need higher reliability and back-up capability. The MUIRS focus helps ensure that TOC elements are addressed in design and test.

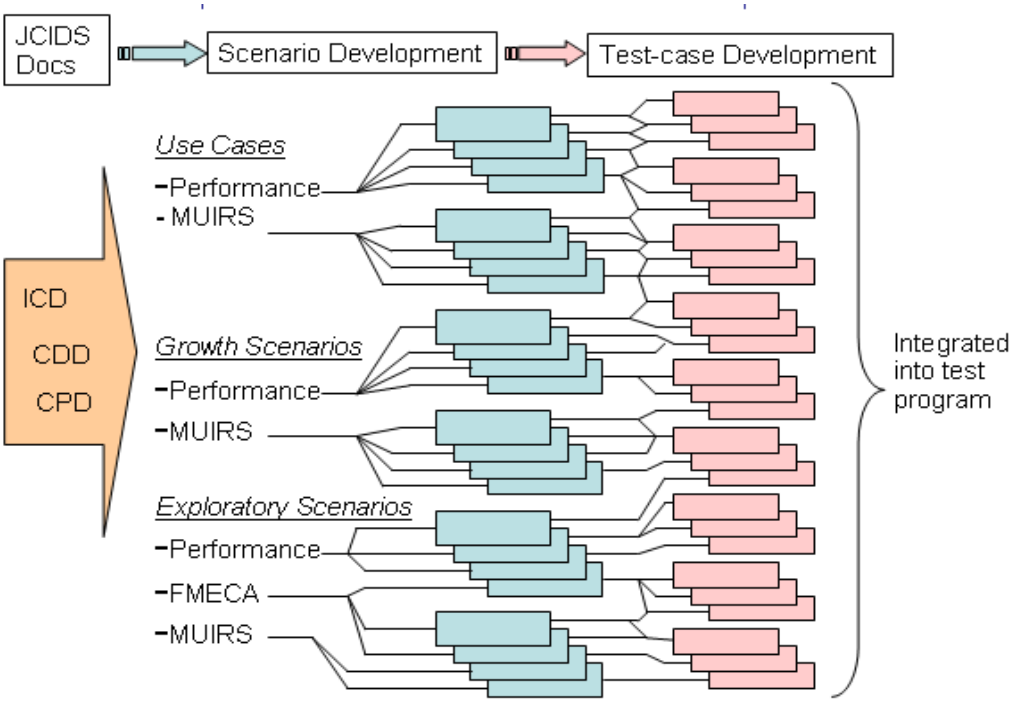


Figure 5. Capabilities-Based ATAM[SM] Scenario Development

The traceability from user-stated requirements through scenario development to test-case development provides a powerful communication and assessment methodology. The growth scenarios and resulting test cases are particularly suited for addressing and evaluating TOC design requirements because the system evolves over its life cycle, which is often overlooked in current system development efforts.

The software developer's understanding of the eventual performance required in order to be considered successful guides the design of the architecture and every step of the software development, coding, and testing through to the Full Operational Capability (FOC) delivery and OT&E. Coding and early testing of software units and configuration items is much more purposeful due to this level of understanding. The MUIRS and FMECA focus will help the design process for better TOC performance.

The resulting test program is very comprehensive as each prioritized scenario requires testing or other verification methodologies to demonstrate how the software performs in each related scenario and satisfies the quality attributes borne of the user requirements. The testing supports the SEP design loop by verifying that the software performs the functions allocated to it and, in aggregate, performs the verification loop process by demonstrating that the final product produces the capability identified in the user requirements through operational testing.

Both QAW and ATAM[SM] require the capturing of essential data supporting decision-making and documenting decisions made. These databases would be best used in a collaborative IT system, as described in the next section.

Collaborative IT Systems

Collaborative IT tools are being used today in the private sector to connect various stakeholders—designers, logisticians, cost analysts, field service representatives, system users—who have the need to communicate. Such tools could be used to support current and emerging warfighting systems. Collaborative tools could be adapted to address reliability and ownership cost concerns related to warfighting systems. Tools that facilitate improved communications would likely

have immediate payoff in being able to speed up solutions to problems. For example, field service representatives (FSRs) and users could quickly raise problems to technical staff for resolution. Cost analysts could more quickly identify emerging cost drivers and initiate business case analyses. Production and quality technicians could rapidly learn of field defects that are the result of production defects. Other FSRs and users could be alerted to emerging problems and be armed with advance knowledge that might avert impending failures.

The reliability improvement process could be enhanced by the use of collaborative tools, because of the ease with which LCL professionals could bring repair parts databases to bear on design decisions. This would be helped by Pareto, that is, a focus on the cost drivers or reliability drivers, especially the expensive items that fail more often than predicted. This approach could be used up-front in pre-acquisition phases, too, by tying in legacy databases that contain performance information of similar or predecessor systems.

Think of the impact to business case analysis (BCA). Cost estimates depend on solid cost databases that are continually updated by current systems in order to identify major cost drivers that might be candidates for redesign or improved manufacturing processes to achieve better reliability and reduced life-cycle cost. Collaborative IT could contribute to the accuracy and completeness of cost estimates.

Component improvements that result from collaborative databases would pay off in legacy systems, but might deliver a second payoff in reduced ownership cost of future systems as well. Collaborative databases could be cross-referenced in an architecture that would arrange cost and reliability information in system, subsystem, or component databases, enabling better cost estimating of emerging systems.

An example of the potential value of collaborative efforts in improving reliability and reducing TOC is the microwave tube on the Aegis program, developed in the early 1980s. The tubes were expensive to maintain (an estimated $8.20 per operating hour), ubiquitous (nearly 30,000 units in 2010), and initial reliability numbers were lower than expected (as low as 1,300 hours MTBF). Through a

collaborative effort between the program manager, NAVSEA, and several commercial vendors, design and manufacturing improvements increased the MTBF to 40,000–45,000 hours, drastically reducing the associated TOC from $8.20 to $0.45 per operating hour for all associated naval combat systems (Apte & Dutkowski, 2006, pp. 3–21).

Collaborative IT tools could potentially be implemented through apps to smart handheld devices, such as iPhones, Androids, or Blackberries. These devices, which are ubiquitous at systems commands and contractor design and logistics facilities, could be very valuable and convenient for field service representatives, military maintenance personnel, and even users in some environments.

Conclusions and Recommendations: Major Thrusts to Control Software Component TOC

Conclusions

DoD software-intensive systems and the software content in other systems will continue to grow and may dominate the TOC costs in the future. These costs are exacerbated by the fact that, in addition to contracted development costs, the bulk of the software sustainment costs are also contracted. In addition, the skill sets needed tor software sustainment are the same as for software development, so the DoD is contracting for software engineers to perform maintenance functions. All of these factors indicate that DoD system software will continue to be a very expensive portion of TOC.

The software engineering environment remains immature, with few, if any, industry-wide standards for software development or sustainment. The Defense Acquisition System (DAS) is significantly dependent on mature engineering environments to compensate for the gaps and interpretation requirements presented with the performance-based specifications, vague Operational Mission Summary/Mission Profiles, and high-level work breakdown structures (WBSs) that the DoD provides during the request for proposal (RFP) process.

The developer software engineers will consume 50% or more of their contract resources analyzing requirements and developing the architectural design. This effort is expended before the Preliminary Design Review (PDR) and requirement additions (requirements creep), or changes beyond that point have disastrous effects on the software design and can even cause a complete redesign at extreme cost in funding and schedule.

The system software size and complexity are key indicators of both the development costs and the sustainment costs, so the initial estimates are critical for predicting and controlling TOC. Unfortunately, the software size estimating processes require a significant amount of detailed understanding of the requirements and design that is typically not available when operating the DAS

without supplementary analyses, tools, and techniques. Available parametric estimating tools require much of the same detailed information and are still too inaccurate to be relied upon. Similarly, understanding the potential software complexity requires in-depth understanding of the requirements and architectural design.

It is clear that the DoD must conduct much more thorough requirements analyses, provide significantly more detailed operational context, and drive the software architectural design well beyond the WBS functional design typically provided. To accomplish this, the DAS must be supplemented with tools, techniques, and analyses that are currently not present.

Recommendations

Program managers for software-intensive systems must supplement the DAS processes to

- compensate for the immature software engineering environment
- gain sufficient detailed information to perform reasonable software size and complexity estimates critical to understanding and managing system TOC
- complete the inventory of derived and implied requirements, including the often neglected sustainability requirements, before the RFP is issued
- provide more detailed system operational context, beyond what exists in most OMS/MP documents
- obtain more realistic contractor proposals in terms of cost and schedule associated with the software development and sustainment
- drive the software architecture for a more sustainable, less complex design
- monitor the software design process (metrics) to ensure the effort is progressing towards an effective, supportable, and testable design supporting the warfighter

The tools, techniques, and analyses presented in this research are designed to accomplish the tasks outlined above, and are compatible with the Systems Engineering Process (SEP) supporting the DAS. They also are designed to work together in a synergistic method to improve the software-intensive system development and sustainment performance influencing system TOC. They are

certainly not the only tools, techniques, and analyses available to improve the process, and others may be as effective, as long as they can address the bulleted items above.

The maintainability, upgradability, interoperability, reliability, and safety/security (MUIRS) analysis technique is designed to help identify derived and implied requirements that need to be more fully articulated to ensure that the software engineer adequately considers these critical system attributes. These were selected because they are often missing from the user's capability-based requirements documents and the resulting performance specification, yet they are critical for the warfighter and are significant TOC drivers.

The Quality Attribute Workshop (QAW) is a technique to help more fully detail all requirements, including derived and implied. It is often used with the system WBS to more fully develop the desired functional design, especially when combined with the MUIRS analyses.

The Architectural Tradeoff Analysis Methodology^{SM} (ATAM^{SM}) is designed to be used with the QAW and provides detailed operational context through the scenario development, providing critical design cues to the software development engineers. The scenarios include Use Cases (how the system will be used and maintained if fielded today), Growth Cases (how the system will likely change over its life cycle, including future networking), and Exploratory Scenarios (how the system is to operate under unusual or stressful conditions). This research recommends including the MUIRS analyses in the ATAM, as well as Failure Modes and Effects Criticality Analyses (FMECA) to identify critical functionality requirements.

Combined, the tools, techniques, and analyses provide a much improved understanding of the system and identify critical attributes that the software developers need to know to design an effective and supportable design. These tools help compensate for the immature software engineering environment, provide more detailed information needed to perform size and complexity estimates, and provide detailed operational context needed for proper software architectural design.

They help produce superior RFPs and garner more realistic contractor proposals. They provide processes for monitoring critical software design activities and full test matrix crosswalks. All of these enhancements will help more accurately estimate and manage software TOC attributes.

References

Apte, A., & Dutkowski, E. (2006, May 31*). Total ownership cost reduction case study: AEGIS microwave power tubes* (NPS-AM-06-008). Retrieved from Naval Postgraduate School, Acquisition Research Program website: http://www.acquisitionresearch.net

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.

Boudreau, M. W., & Naegle, B. R. (2003, September 30). *Reduction of total ownership cost* (NPS-AM-03-004). Retrieved from Naval Postgraduate School, Acquisition Research Program website: http://www.acquisitionresearch.net

Brodsky, R. (2010, May 19). Pentagon reports progress on anniversary of procurement reform law. *Government Executive*. Retrieved from http://www.GovExec.com

Business Executives for National Security (BENS), Task Force on Defense Acquisition Law and Oversight. (2009, July). *Getting to best: Reforming the defense acquisition enterprise*. Washington, DC: BENS.

Carl Levin and Howard P. "Buck" McKeon National Defense Authorization Act for Fiscal Year 2015, Pub. L. No. 113-291, § 831–833, 128 Stat. 3292 (2014, December 19).

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

Chairman of the Defense Science Board (DSB). (2008, May). *Report of the Defense Science Board Task Force on developmental test & evaluation.* Washington, DC: Author.

Chairman of the Joint Chiefs of Staff (CJCS). (2009a, March 1). *Joint capabilities integration and development system* (CJCS Instruction 3170.01G). Washington, DC: Author.

Chairman of the Joint Chiefs of Staff (CJCS). (2009b, July 31). *Manual for the operation of the joint capabilities integration and development system* (CJCSM 3170.01). Washington, DC: Author.

Defense acquisition: Improved program outcomes are possible: Written testimony of GAO's Louis J. Rodrigues before the Subcommittee on Acquisition and Technology of the Senate Committee on Armed Services (GAO/T-NSIAD-98-123), 105th Cong. (1998). Retrieved from GAO website: http://www.gao.gov

Defense acquisitions: Fundamental changes are needed to improve weapon program outcomes: Written testimony of GAO's Michael J. Sullivan before the Subcommittee on Federal Financial Management, Government Information, Federal Services, and International Security of the Senate Committee on Homeland Security and Governmental Affairs (GAO-08-1159T), 110th Cong. (2008). Retrieved from GAO website: http://www.gao.gov

Defense Acquisition University (DAU). (2005, March). *Performance based logistics: A program manager's product support guide*. Fort Belvoir, VA: Author.

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

Department of Defense (DoD). (2012).

Department of the Navy (DoN). (2010, February 16). *Total ownership cost guidebook* [Concurrent with SECNAVIST 5000.2E]. Washington, DC: Author.

Deputy Under Secretary of Defense for Science and Technology (DUSD[S&T]). (2005, May). *Technology readiness assessment (TRA) deskbook*. Washington, DC: Author.

Director of Operational Test and Evaluation (DOT&E) [J. Michael Gilmore]. (2010, June 30). *State of reliability* [Memorandum]. Washington, DC: Author.

Dornheim, M. A. (2005, September). A wild ride. *Aviation Week & Space Technology, 163*, 46.

Duncan Hunter National Defense Authorization Act for Fiscal Year 2009, Pub. L. No. 110-417, § 814, 122 Stat. 4356 (2008, October 14).

Eaton, D. R. (2004, August 1). *Improving the management of reliability* (NPS-LM-04-009). Retrieved from Naval Postgraduate School, Acquisition Research Program website: http://www.acquisitionresearch.net

F/A 18. (n.d.). In *Wikipedia*. Retrieved from http://www.wikipedia.org/wiki/McDonnell_Douglas_F/A-18_Hornet

Fast, W. R. (2011, January–February). WSARA one year later. *Defense AT&L*, *40*, 4–8.

Federation of American Scientists (FAS). (2011a, March 10). M1 Abrams main battle tank. Retrieved from http://www.fas.org/man/dod-101/sys/land/m1.htm

Federation of American Scientists (FAS). (2011b, March 11). F/A 18 Hornet. Retrieved from http://www.fas.org/programs/ssp/man/uswpns/air/fighter/f18.html

Fowler, R. (2010, March–April). The future of product support. *Defense AT&L*, 16–20.

General Accounting Office (GAO). (2003, February 11). *Best practices: Setting requirements differently could reduce weapon systems' total ownership costs: Report to the Subcommittee on Readiness and Management Support, Committee on Armed Services* (GAO-03-57). Retrieved from http://www.gao.gov

Government Accountability Office (GAO). (2008, December). *Defense logistics: Improved analysis and cost data needed to evaluate the cost-effectiveness of performance based logistics (*GAO-09-41). Retrieved from http://www.gao.gov

Government Accountability Office (GAO). (2010a, May). *Defense acquisitions: Strong leadership is key to planning and executing stable weapon programs* (GAO 10-522). Retrieved from http://www.gao.gov

Government Accountability Office (GAO). (2010b, July 20). *Defense management: DoD needs better information and guidance to more effectively manage and reduce operating and support costs of major weapon systems* (GAO-10-717). Retrieved from http://www.gao.gov

Government Accountability Office (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

Hill, S. (2006, September). A winning strategy: Collaborating on product designs speeds time-to-market, improves quality, boosts revenues. *Manufacturing Business Technology*, *24*, 18–21.

Humphrey, W. S. (1990). *Managing the software process*. New York, NY: Addison-Wesley.

Institute for Defense Analyses (IDA). (n.d.). Reduction of total ownership costs. Retrieved from http://rtoc.ida.org/rtoc/rtoc.html

Kaye, M. A., Sobota, M. S., Graham, D. R., & Gotwald, A. L. (2000, Fall). Cost as an independent variable. *Defense Acquisition Review Quarterly*, *24*, 353–371.

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

Kobren, B. (2010). The product support manager: Achieving success in executing life cycle management responsibilities. *Defense Acquisition Review Journal*, *54*, 182–204.

Kratz, L., & Buckingham, B. A. (2010). Achieving outcomes-based life cycle management. *Defense Acquisition Review Journal*, *53*, 45–66.

Kreisher, O. (2010, May 6). Navy secretary outlines how he'll reduce weapons costs. *Government Executive*. Retrieved from http://www.GovExec.com

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

Levin & McKeon. (2015).

Major defense acquisition programs: Certification required before Milestone A or Key Decision Point A approval, 10 U.S.C. § 2366a (2009).

Major defense acquisition programs: Certification required before Milestone B or Key Decision Point B approval, 10 U.S.C. § 2366b (2009).

Microsoft Corporation. (2009, October). Microsoft Architecture Application Guide 2.0. Retrieved from https://msdn.microsoft.com/en-us/library/ff650706.aspx

Mishory, J. (2010, August 19). CAPE report finds: DoD cannot create baselines to track operating and support costs. *Inside the Pentagon*. Retrieved from http://defensenewsstand.com/Inside-the-Pentagon/

Morris, E., Levine, L., Meyers, C., Place, P., & Plakosh, D. (2004, April). *System of systems interoperability (SOSI): Final report*. Pittsburg, PA: Carnegie Mellon University, Software Engineering Institute.

Naegle, B. R. (2004, July). *The impact of software support on total ownership cost* (NPS-AM-04-007). Monterey, CA: Naval Postgraduate School.

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

Naegle, B. R. (2015, February 4), *Gaining control and predictability of software-intensive systems development and sustainment* (NPS-AM-14-194). Monterey, CA: Naval Postgraduate School.

Naegle, B. R., & Boudreau, M. W. (2011, March 14). *Total ownership cost—Tools and discipline* (NPS-CE-11-014). Monterey, CA: Naval Postgraduate School.

Naegle, B. R., & Petross, D. (2007, October). *Developing software requirements supporting open architecture performance goals in critical DoD system-of-systems* (NPS-AM-07-104). Monterey, CA: Naval Postgraduate School.

Naegle, B. R., & Petross, D. (2010, January). *P-8A Poseidon multi-mission maritime aircraft (MMA) software maintenance organization concept analysis* (NPS-LM-10-006). Monterey, CA: Naval Postgraduate School.

National Defense Authorization Act for Fiscal Year 2010, Pub. L. No. 111-84, § 805, 123 Stat. 2190 (2009, October 28).

Naval Air Systems Command (NAVAIR) and U.S. Army. (2012). CECOM brief.

Navy Logistics Functional IPT. (2010, April 30). *DON acquisition governance (Gate Reviews): Establishing a post-IOC sustainment review* [Briefing]. Washington, DC: DoN.

Office of the Secretary of Defense (OSD). (2003, March 21). [Aldridge, E. C., & Stenbit, J. P.]. *Software acquisition process improvement programs*. [Memorandum]. Washington, DC: Author.

Peters, K. M. (2010, February 2). Defense budget tackles major management issues. *Government Executive*. Retrieved from http://www.GovExec.com

Porter, G., Gladstone, B., Gordon, C. V., Karvonides, N., Kneece, R. R., Jr., Mandelbaum, J., & O'Neil, W. D. (2009, December). *The major causes of cost growth in defense acquisition: Executive summary* (Vol. 1). Alexandria, VA: Institute for Defense Analyses.

Principal Deputy Under Secretary of Defense for Acquisition, Technology, and Logistics (PDUSD[AT&L]) [Frank Kendall]. (2010, April 23). *Preparation for Defense Acquisition Board (DAB) meetings, DAB readiness meetings (DRM), and DAB planning meetings (DPM)* [Memorandum]. Washington, DC: USD(AT&L).

Product Support Assessment Team (PSAT). (2009, November). *DoD weapon system acquisition reform product support assessment*. Washington, DC: Author.

Reynolds et al. (2012).

Roper, M. A. (2010). Pre-Milestone A cost analysis: Progress, challenges, and change. *Defense Acquisition Review Journal*, *53*, 67–75.

Scully, M. (2010, February 2). Gates sacks F-35 manager, withholds Lockheed payments. *Government Executive*. Retrieved from http://www.GovExec.com

Secretary of the Air Force (SecAF). (2008, August). *Weapon system software management guidebook.* Washington, DC: Author.

Siemens. (2010). *Knowledge-centric MRO transformation* (PLM Software White Paper). Retrieved from http://www.siemens.com/plm

Silverstein, K., & Moag, J. (2000, January–February). The Pentagon's 300-billion-dollar bomb. Mother Jones. Retrieved from www.motherjones.com/politics/2000/01/pentagons-300-billion-dollar-bomb

Software management: Software size estimate. (n.d.). In *AcqNotes*. Retrieved March 9, 2017, from http://www.acqnotes.com/acqnote/careerfields/software-size-estimate

Tremaine, R. L., & Seligman, D. (2010). It's time to take the chill out of cost containment and re-energize a key acquisition practice. *Defense Acquisition Review Journal*, *54*, 242–267.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [Jacques S. Gansler]. (1998, November 13). *Definition of total ownership cost (TOC), life cycle cost (LCC), and the responsibilities of program managers* [Memorandum]. Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [Jacques S. Gansler]. (1999, October 26). *Software evaluations for ACAT I programs* [Memorandum]. Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [Edward C. Aldridge]. (2002, February 13). *Performance based logistics* [Memorandum]. Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [Edward C. Aldridge]. (2003, May 12). *The defense acquisition system* (DoD Directive 5000.1). Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [John J. Young]. (2007a, September 19). *Prototyping and competition* [Memorandum]. Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]). (2007b, November 20). *The defense acquisition system* (DoD Directive 5000.01). Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [John J. Young]. (2008a, July 21). *Implementing reliability, availability and maintainability policy* [Memorandum]. Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [John J. Young]. (2008b, July 31). *Implementing a life cycle management framework* [Memorandum]. Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]). (2008c, December 8). *Operation of the defense acquisition system* (DoD Instruction 5000.02). Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [Ashton B. Carter]. (2009a, November). *DoD weapon system acquisition reform product support assessment.* Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [Ashton B. Carter]. (2009b, December 4). *Implementation of the Weapon Systems Acquisition Reform Act of 2009* (Directive Type Memorandum 09-027). Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [Ashton B. Carter]. (2010a, September 14). *Better buying power: Guidance for obtaining greater efficiency and productivity in defense spending* [Memorandum]. Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [Ashton B. Carter]. (2010b, October 7). *Requirements for life cycle management and product support* (Directive-Type Memorandum [DTM] 10-015). Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [Ashton B. Carter]. (2010c, October 21). *Implementation of the Weapon Systems Acquisition Reform Act of 2009* (Change 1 to December 4, 2009, version; Directive-Type Memorandum [DTM] 09-027). Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [Ashton B. Carter]. (2010d, November 3). *Implementation directive for Better Buying Power—Obtaining greater efficiency and productivity in defense spending* [Memorandum]. Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [Frank Kendall]. (2013, April 24). *Implementation directive for Better Buying Power 2.0—Achieving greater efficiency and productivity in defense spending* [Memorandum]. Washington, DC: Author.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) [Frank Kendall]. (2015, April 9). *Implementation directive for Better Buying Power 3.0—Achieving dominant capabilities through technical excellence and innovation* [Memorandum]. Washington, DC: Author.

University of Southern California. (2002, September). COCOMO. Retrieved March 9, 2017, from http://sunset.usc.edu/cse/pub/research/COCOMOII/cocomo_main.html

Weapon System Acquisition Reform Act of 2009, Pub. L. No. 111-23, 123 Stat. 1704 (2009).

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

www.acquisitionresearch.net