NPS-AM-06-070

EXCERPT FROM THE PROCEEDINGS

OF THE

THIRD ANNUAL ACQUISITION RESEARCH SYMPOSIUM

DEVELOPING PERFORMANCE BASED REQUIREMENTS FOR OPEN ARCHITECTURE DESIGN

Published: 30 April 2006

by

Brad Naegle

3rd Annual Acquisition Research Symposium of the Naval Postgraduate School:

Acquisition Research: Creating Synergy for Informed Change

May 17-18, 2006

The research presented at the symposium was supported by the Acquisition Chair of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request Defense Acquisition Research or to become a research sponsor, please contact:

NPS Acquisition Research Program
Attn: James B. Greene, RADM, USN, (Ret)
Acquisition Chair
Graduate School of Business and Public Policy
Naval Postgraduate School
555 Dyer Road, Room 332
Monterey, CA 93943-5103
Tel: (831) 656-2092
Fax: (831) 656-2253
E-mail: jbgreene@nps.edu

Copies of the Acquisition Sponsored Research Reports may be printed from our website www.acquisitionresearch.org

Conference Website:
www.researchsymposium.org

Proceedings of the Annual Acquisition Research Program

THIS PAGE INTENTIONALLY LEFT BLANK

Developing Performance Based Requirements for Open Architecture Design

Presenter: Brad Naegle, Lieutenant Colonel, US Army (Ret), is a Lecturer and Academic Associate at the Naval Postgraduate School, Monterey, California. While on active duty, LTC (ret.) Naegle was assigned as the Product Manager for the US Army 2½-Ton Extended Service Program (ESP) and the USMC Medium Tactical Vehicle Replacement (MTVR) from 1994 to 1996, and the Deputy Project Manager for Light Tactical Vehicles from 1996 to 1997. He was the 7th Infantry Division (Light) Division Materiel Officer from 1990 to 1993 and the 34th Support Group Director of Security, Plans and Operations from 1987 to 1988. Prior to that, Naegle held positions in Test and Evaluations and Logistics fields. He earned a Master's Degree in Systems Acquisition Management (with Distinction) from the Naval Postgraduate School and a Bachelor of Science degree from Weber State University in Economics. He is a graduate of the Command and General Staff College, Combined Arms and Services Staff School, and Ordnance Corps Advanced and Basic Courses.

Brad Naegle
Lecturer, Naval Postgraduate School
Ph: 831-656-3620
E-mail: **bnaegle@nps.edu**

Abstract

To implement the capabilities conceptualized in *Joint Vision 2020*, complex, secure networks of weapon systems, intelligence platforms, and command and control mechanisms must be seamlessly integrated and maintained over time. Accurate and timely information will enable *Joint Vision 2020* key tenets: Dominant Maneuver, Precision Engagement, Focused Logistics, and Full Dimensional Protection. These networks are central warfighting platforms in the information age.

As these capabilities are developed over time in an evolutionary manner, interoperability on the Net-Centric Warfare (NCW) networks is essential, and both hardware and software systems must be designed in an Open-systems Architecture (OA) fashion to accommodate the vast number of changes anticipated. Professional Program Management will be needed to successfully develop these key warfighting platforms.

Materiel Developers will need to recognize the relatively immature nature of the software engineering domains and actively compensate for this immaturity. System software performance capabilities must be much more detailed than typical hardware-centric systems, as the current state of software engineering disciplines is unlikely to satisfy implied, yet critical performance requirements. Essential OA performance characteristics including Maintainability, Upgradability, Interfaces/Interoperability, Reliability, Safety and Security (MUIRSS) must be fully analyzed and clearly communicated to the software developer to ensure the DoD obtains the flexibility and longevity desired from NCW systems.

Keywords: Net-Centric Warfare, Interoperability, Open Systems Architecture, Software Requirements, System of Systems, Family of Systems

Introduction

Joint Vision 2020 is the Chairman of the Joint Chiefs of Staff's guiding document for development of the future force and warfighting capabilities. It states, "If our Armed Forces are to be faster, more lethal, and more precise in 2020 than they are today, we must continue to invest in and develop new military capabilities." It continues, dictating, "The overall focus of this vision is full spectrum dominance—achieved through the interdependent application of dominant maneuver, precision engagement, focused logistics, and full dimensional protection" (CJCS, 2000, pp. 1-2). The key word is "interdependent," as it prescribes interoperability requirements to a level never before achieved. Flexible networks of complex system-of-systems must be successfully developed to realize this vision.

To implement the concepts presented in *Joint Vision 2020*, the Director of Force Transformation anticipates a new era:

> As the world enters a new millennium, our military simultaneously enters a new era in warfare—an era in which warfare is affected by a changing strategic environment and rapid technological change. The United States and our multinational partners are experiencing a transition from the Industrial Age to the Information Age. Simultaneously, we are fully engaged in a global war on terrorism set in a new period of globalization. These changes, as well as the experiences gained during recent and ongoing military operations, have resulted in the current drive to transform the force with network-centric warfare (NCW) as the centerpiece of this effort. (2005, p. 3)

This quote from *The Implementation of Network-centric Warfare* clearly indicates the direction that the DoD is taking in developing the next generation's warfighting capabilities. The success of the initial NCW systems deployed since Desert Storm, as limited as they were, revealed the potential battlespace domination offered through networked systems providing situational and information superiority. One major challenge in constructing effective NCW systems is designing the network to seamlessly integrate existing, planned and future platforms and systems into a secure, fully interoperable, near real-time information system. The network will need to accommodate complex systems that may or may not have been designed to interoperate. The networked systems themselves are extremely complex and will have been developed decades apart. The network design must be open, flexible and able to adapt to this wide disparity of system-of-systems.

It is well understood that an Open-systems Architecture (OA) design is required to meet both current and future warfighting needs and is a critical element in net-centric warfare systems-of-systems concepts. These highly integrated systems are increasingly dependent on software solutions for integration into the net-centric scheme; therefore, software interfaces are one of the main keys for achieving the tactical and strategic synergies of the net-centric system. This paper will focus on the challenges presented when the Department of Defense (DoD) conducts capabilities analysis and derives performance specifications for a software-intensive, net-centric, system-of-systems architecture that meets OA needs throughout the life of the system.

You got to be careful if you don't know where you're going, because you might not get there! – Yogi Berra

The DoD Performance Specification development process transforms the warfighter requirements into terms that are more understandable for the system developer, usually the prime contractor. Typically, the system performance requirements are decomposed through at least three levels using the Work Breakdown Structure (WBS) methodology. The concept is to provide the contractor sufficient detail with regard to performance, constraints, and intended environments without stifling innovative solutions to meeting those requirements. The number of WBS levels developed by the DoD is dependent on the complexity of the system and the engineering domain maturity. For example, the automotive engineering discipline is very mature, and a level three WBS for a tactical truck system would most probably be sufficient. To determine whether the WBS is ready to hand off to a contractor, the Materiel Developer must continue WBS development to a point where either the contractor has enough information to develop the system needed by the warfighter, or any contractor derived solution that meets the stated performance requirements is acceptable. While easily stated, this presents a daunting challenge in complex systems, especially those that are software-intensive.

Software engineering is not mature, and there are few industry-wide standards for languages, tools, architectures, reuse, or procedures. Software developed for complex weapon systems is typically started from scratch with each new system; very little existing software code is reused. In addition, new languages and associated tools are introduced every few years. For this and other reasons, software programs grow exponentially in size and complexity, expanding desired capabilities but limiting the maturation process. The DoD Materiel Developer must recognize the relative immaturity of software engineering when developing the WBS for software-intensive systems and, more importantly, compensate for that immaturity.

The current state of software engineering maturity drastically impacts an area of extreme DoD concern—Supportability. Hardware-centric performance specifications rely heavily on mature engineering environments to account for a significant portion of the system's supportability performance. Using the automotive engineering example, there is little need of specifying supportability requirements such as features for oil, filter, tire and coolant replacement as they are industry-standard features that would be included in any competent design. There are few corresponding software engineering standards for supportability features, and most commercially based software is not designed for long-term use as is typically the requirement for DoD systems. There are literally hundreds of ways to build the architecture and construct the code for even the most basic software function. Without physical or established engineering techniques, the software developer is bounded only by his or her imagination and creativity in satisfying broad specifications. The resulting software may function correctly, but may not possess the OA design needed to effectively maintain, upgrade, or interface it with the constantly changing net-centric systems and environment.

DoD acquisition professionals must recognize that the warfighter capabilities needed require software development techniques that differ significantly when compared to their commercially based counterparts. The software engineering techniques used in short-lived software products may not prove effective in developing long-lived DoD software-intensive, warfighting systems. DoD systems are designed to have a very long life span, including software-intensive systems, in direct contravention with most commercially based software designs. The need for OA design—upgradeable, flexible, and highly reliable software that is maintainable over a long life span—is paramount to DoD's warfighting systems, but industry-standard software engineering techniques do not necessarily incorporate those features.

What this means to the DoD is that the capabilities analysis and resulting system performance specifications must be completed in significantly more detail to achieve software performance that meets warfighter's needs. The software developer needs to be driven to OA design by the performance specifications because software engineering discipline and state of the practice are unlikely to provide sufficient architectural designs without explicit performance requirements clearly communicated. Providing more detailed performance specifications seems to run counter to acquisition reforms implemented to allow industry flexibility and innovation in achieving performance thresholds and goals, but that is not the intent. The detailed performance specifications provide the software developer much more information about areas that the customer—the DoD—sees as critical to the overall system performance. This will have a significant impact on the system software design supporting OA performance and will provide the basis for a much more accurate cost and schedule estimate in the proposal received.

Near-term Challenges

The net-centric warfare concepts feature system-of-systems in an elaborate network requiring a significant number of critical interfaces. As each system is added or later upgrades its capabilities, it likely drives an interface change with other interfaced systems, necessitating the need for flexibility in accommodating interface changes from affected interoperating or networked systems. It is easy to visualize dozens of software changes driven by upgrades in the interfaced components of the network and the critical need for effective OA designs to quickly and economically accommodate change over a long life span. Again, this level of design flexibility is not a software industry norm for most commercially designed systems.

Safety and Security requirements for DoD weapon system software have few commercial counterparts. Obviously, commercially based critical medical equipment, aviation systems, and banking systems would also require a high degree of safety and security, but the combat environment weapon systems are intended to operate within, and the military lives that are always at stake adds to criticality of the need. The net-centric warfare environment will necessarily require unprecedented security measures. Software must be designed to continue to operate critical weapon systems in degraded modes, reject spurious input without freezing or failing, and resist intrusion, viruses and other attacks. Anything short of that will put military members and the critical missions they perform at risk. Most commercially based software engineering disciplines do not consider such stringent safety and security requirements. The system's OA design must allow for the flexibility needed while simultaneously ensuring safety and security requirements. These two forces are rarely in concert and usually are in conflict.

Considering the state of immature software engineering that exists today, it is clear that the DoD will not achieve the level of software-intensive system performance necessary if the WBS and performance specification are not developed more fully before hand-off to the developer or contractor. Due to the pressure to shorten the acquisition timeline, there is a tendency to rush the Request for Proposal (RFP) to the prospective contractors without developing the WBS below level three or including the performance specification with sufficient detail. This approach works with systems based in mature engineering environments as the contractor understands that all of those unstated requirements will be satisfied through the established engineering standards; thus, the proposed schedule and cost estimates will be fairly accurate. With a software-intensive system, this is not the case due to many of the reasons presented earlier. The most diligent contractor can only provide cost and schedule estimates based on what is presented in the RFP. If a significant portion of the software development effort is not evident in the RFP, the contractor estimates may be grossly understated, causing substantial—and avoidable—funding shortfalls and schedule overruns that plague the development effort throughout the acquisition phase and well into the system's lifecycle.

A Methodology for Software OA Capabilities Analysis

For DoD software-intensive systems to attain the broad spectrum of warfighter performance and long-term supportability with predictable costs and schedules, the Materiel Developer must provide performance specifications in the RFP that are detailed in areas that hardware-centric systems with mature engineering environments need not be. In addition to the system's software performance issues, the OA areas of Maintainability, Upgradeability, Interfaces/Interoperability, Reliability, Safety, and Security (MUIRSS) must be carefully analyzed to ensure that the potential contractors understand the Government requirements and constraints in each of these areas. It is likely that the WBS will have to be developed several more levels in order to capture essential requirements; potential contractors would need to see such WBS development to form a realistic proposal with an executable schedule and an accurate cost estimate.

The Systems Engineering Process (SEP) is the preferred technique for analysis within each of the MUIRSS categories as it provides a highly structured and comprehensive methodology for developing the WBS. This will be a key tool for the DoD Materiel Developer in developing capabilities requirements and communicating them to the software developer via the performance specifications. Recognizing the existing shortfalls in software engineering maturity, this methodology will greatly assist the software developer in understanding OA-related performance requirements; this, in turn, will significantly influence the software architecture design and the level of effort estimated to build the desired system. The alternative leaves the software developer estimating these requirements without the background or experience to do so, or worse yet, discovering the extent of the actual requirements after the work has begun.

The capabilities analysis process must capture the OA performance needed for supporting the system throughout its lifecycle. This analysis should drive a robust Post Production Software Support (PPSS) plan addressing the MUIRSS elements of the OA design. The MUIRSS elements are interdependent and tend to apply across the system and software architecture. Each MUIRSS element is discussed in the following paragraphs to provide a basis for analyzing capability requirements within the area and capturing performance characteristics that are essential to the DoD.

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 where software is fundamentally different from hardware. Software is one of the very few components where 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 typically is not enough time, funding or resources to find and correct every error, glitch, or bug, and not every one is 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 system-of-systems 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. As the DoD has a very limited capability for supporting software internally, typically, early software support is 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 was for $119 million for the modification. 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 ten 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 where 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 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 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 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 system-of-systems level. The individual program manager 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 a system-of-systems (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: a near stall situation, contradicting instrument indications, false warnings, and difficulty controlling the aircraft in both autopilot and manual flight modes. The problem was 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

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 have 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, as 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, 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 supercede whatever function was being performed, none of which are acceptable in combat operations.

Security

With software performing so many critical functions, there is little doubt that software applications are a prime target for anyone opposing US 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 of 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 all these capabilities are essential before he/she constructs software architectures and software programs, as this knowledge will be very influential for the software design and application development.

Interoperability challenges are increased when the system-of-systems have 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/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 may be gleaned from knowing how the system finds its targets. We 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.

Network Development

The network is a lynchpin for the combat effectiveness of NCW architecture, and as such, should be developed under a professional Program Management (PM) organization. The US Navy has achieved optimal results by assigning a PM for the Link 16 Program as noted by SEI: "The Navy created a PMO and funded it with money from affected programs. These monies were returned to programs specifically to work toward Link 16 capability" (Morris et al., 2004, p. 33). SEI goes on to describe the need for professional program management by stating, "What is needed are processes that help to reach agreements, blinders that avoid getting distracted by things that are not related (e.g., portability), and to be agnostic about specific technologies (e.g., CORBA or Message Oriented Middleware)" (p. 34). A network PM would help facilitate and broker those agreements to the benefit of the network, vastly increasing the probability that the NCW asset will provide the warfighter the capability and advantage visualized by DoD.

Summary

To get the needed Open Architecture performance the DoD is seeking for software components, the Material developer will have to specify it in the RFP and Performance Specification. Unlike many hardware-centric engineering environments, the immature software engineering environment is unlikely to compensate for essential performance that is not specified. With the Materiel Developer performing the capabilities analysis using the MUIRSS approach outlined above, the potential software developers will be provided a much more detailed understanding of critical capabilities the DoD expects from its software components.

This same technique should result in significantly more accurate proposals as much more of the software development work can be estimated from the RFP and Performance Specification provided. Yes, proposals will likely continue to be overly optimistic, especially in a competitive environment. And yes, changes and details will still be revealed after the contract is signed—but the cost growth should be in the range of ten percent of the cost, not the current average of one-hundred percent of the original proposal. Schedule estimates will also be much more accurate as the scope of the software work is better understood by the contractors, keeping schedule slippage to under fifteen percent of the original proposal estimate.

Conducting this analysis will be as challenging as it is time-consuming, especially since it is applied in the early stages of the acquisition process when there is great pressure to "get the RFP on the street." The enormous potential time and cost savings realized throughout the remaining development and the system's lifecycle by completing the thorough MUIRSS capability analysis warrants the needed analysis time. There is an old carpenter's adage that applies well in this case: "measure twice, cut once."

List of References

Army RDT&E. (2005, February). Army RDT&E budget item justification (R2a Exhibit). Other missile product improvement programs. PE Number 0203802A.

Chairman of the Joint Chiefs of Staff (CJCS). (2000, June). *Joint Vision 2020.*

Chaos: A Recipe for Success. (1999). The Standish Group International.

Director, Force Transformation. (2005, January). *The implementation of net-centric warfare*. Washington, DC: Office of the Secretary of Defense.

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

Forsberg, K., Mooz, H., & Cotterman, H. (2000). *Visualizing project management: A model for business and technical success* (2nd ed.). New York: John Wiley & Sons.

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

Lewis, G. A., Morris, E. J. & Wrage, L. (2004, December). *Promising technologies for future systems*. Carnegie Mellon Software Engineering Institute.

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

Pracchia, L. (2004, April). Improving the DoD software acquisition process. *Crosstalk*, 4-7.

US Air Force Software Technology Support Center (STSC). (2000, May). *Guidelines for successful acquisition and management of software intensive systems (GSAM*) (Version 3).

US General Accountability Office. *Defense acquisitions: Stronger management practices are needed to improve DoD's software-intensive weapon acquisition*. Report to the Committee on Armed Services, US Senate. GAO 04-393. Washington, DC: author.

THIS PAGE INTENTIONALLY LEFT BLANK

2003 - 2006 Sponsored Acquisition Research Topics

Acquisition Management

- Software Requirements for OA
- Managing Services Supply Chain
- Acquiring Combat Capability via Public-Private Partnerships (PPPs)
- Knowledge Value Added (KVA) + Real Options (RO) Applied to Shipyard Planning Processes
- Portfolio Optimization via KVA + RO
- MOSA Contracting Implications
- Strategy for Defense Acquisition Research
- Spiral Development
- BCA: Contractor vs. Organic Growth

Contract Management

- USAF IT Commodity Council
- Contractors in 21st Century Combat Zone
- Joint Contingency Contracting
- Navy Contract Writing Guide
- Commodity Sourcing Strategies
- Past Performance in Source Selection
- USMC Contingency Contracting
- Transforming DoD Contract Closeout
- Model for Optimizing Contingency Contracting Planning and Execution

Financial Management

- PPPs and Government Financing
- Energy Saving Contracts/DoD Mobile Assets
- Capital Budgeting for DoD
- Financing DoD Budget via PPPs
- ROI of Information Warfare Systems
- Acquisitions via leasing: MPS case
- Special Termination Liability in MDAPs

Logistics Management

- R-TOC Aegis Microwave Power Tubes

- Privatization-NOSL/NAWCI
- Army LOG MOD
- PBL (4)
- Contractors Supporting Military Operations
- RFID (4)
- Strategic Sourcing
- ASDS Product Support Analysis
- Analysis of LAV Depot Maintenance
- Diffusion/Variability on Vendor Performance Evaluation
- Optimizing CIWS Life Cycle Support (LCS)

Program Management

- Building Collaborative Capacity
- Knowledge, Responsibilities and Decision Rights in MDAPs
- KVA Applied to Aegis and SSDS
- Business Process Reengineering (BPR) for LCS Mission Module Acquisition
- Terminating Your Own Program
- Collaborative IT Tools Leveraging Competence

A complete listing and electronic copies of published research within the Acquisition Research Program are available on our website: www.acquisitionresearch.org