





2007-04-01

# Software Architecture: Managing Design for Achieving Warfighter Capability

Naegle, Brad



NPS-AM-07-017

# EXCERPT FROM THE

# PROCEEDINGS

**OF THE**

# FOURTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM WEDNESDAY SESSIONS

**Software Architecture: Managing Design for Achieving Warfighter Capability**

**Published: 30 April 2007**

**by**

**Brad Naegle, Naval Postgraduate School**

**4th Annual Acquisition Research Symposium of the Naval Postgraduate School:**

**Acquisition Research: Creating Synergy for Informed Change**



Prepared for: Naval Postgraduate School, Monterey, California 93943

**Acquisition Research program**
**GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY**
NAVAL POSTGRADUATE SCHOOL

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

The following article is taken as an excerpt from the proceedings of the annual Acquisition Research Program. This annual event showcases the research projects funded through the Acquisition Research Program at the Graduate School of Business and Public Policy at the Naval Postgraduate School. Featuring keynote speakers, plenary panels, multiple panel sessions, a student research poster show and social events, the Annual Acquisition Research Symposium offers a candid environment where high-ranking Department of Defense (DoD) officials, industry officials, accomplished faculty and military students are encouraged to collaborate on finding applicable solutions to the challenges facing acquisition policies and processes within the DoD today. By jointly and publicly questioning the norms of industry and academia, the resulting research benefits from myriad perspectives and collaborations which can identify better solutions and practices in acquisition, contract, financial, logistics and program management.

For further information regarding the Acquisition Research Program, electronic copies of additional research, or to learn more about becoming a sponsor, please visit our program website at:

www.acquistionresearch.org

For further information on or to register for the next Acquisition Research Symposium during the third week of May, please visit our conference website at:

www.researchsymposium.org

THIS PAGE INTENTIONALLY LEFT BLANK

# Software Architecture: Managing Design for Achieving Warfighter Capability

**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
(831) 656-3620
bnaegle@nps.edu

> "Software architecture forms the backbone for any successful software-intensive system. An architecture is the primary carrier of a software system's quality attributes such as performance or reliability. The right architecture—correctly designed to meet its quality requirements, clearly documented, and conscientiously evaluated—is the linchpin for software project success. The wrong one is a recipe for guaranteed disaster" (Software Engineering Institute/Carnegie Mellon, 2007).

## Introduction

Software engineers will typically spend 50% or more of the total software development time designing software architecture, and that architecture may provide up to 80% of a modern weapon system's functionality. Increasingly, these systems will operate within a network or other system-of-systems architecture. Obviously, the requirements driving that architectural design effort and the process for tracing requirement to functions, insight into the process, and control of the effort are critical for the successful development of the capability needed by the warfighter.

The DoD typically monitors and controls system technical development through implementation of the Baselines, Audits and Technical Reviews within an overarching Systems Engineering Process (SEP) (Defense Acquisition University, 2004, December, chap. 4). Because of the relatively immature software engineering environment, significantly more analysis and development of the requirements is required. In addition, the software architectural design effort is dependent on in-depth requirements analysis, is resource intensive, and must occur very early in the process. Effective management and implementation of design metrics is essential in developing software that meets the warfighters' needs. This management and metrics effort supplements and supports the system technical development through the Baselines, Audits and Technical Reviews.

There are numerous variations and models of the Systems Engineering Process (SEP). This research uses the model depicted in Figure 1 (below), which illustrates the systems engineering functions described throughout this paper. The concepts are transferable to the SEP "V" model currently used by the DoD.

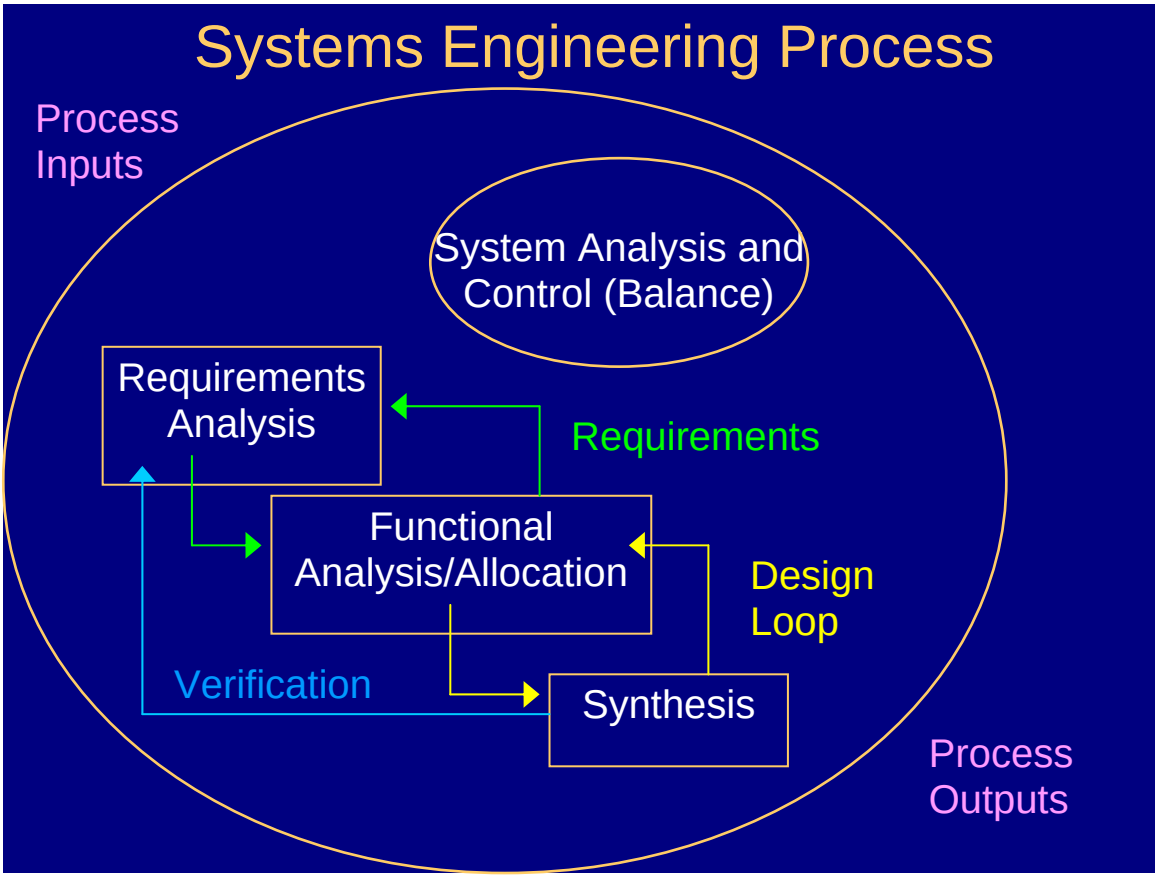


**Figure 1. Systems Engineering Process**

## Software Requirements Impact

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

Using the SEP approach, the explicit user capabilities requirements specified in the Joint Capabilities Integration and Development System (JCIDS) provides the Input for system Requirements Analyses. These analyses are intended to illuminate all system-stated, derived and implied requirements and quality attributes necessary to achieve the capabilities needed by the warfighter. The Work Breakdown Structure (WBS) is a methodology for defining ever-increasing levels of performance specificity—using the SEP

to guide the development of each successive layer (Department of Defense, 2005, July, pp. 1-5).

## Software Engineering Environment

The software engineering environment is not mature, especially when compared to hardware-centric engineering environments. Dr. Philippe Kruchten 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" (Kruchten, 2005, p. 17). Software engineering is significantly unbounded as there are no physical laws that help define environments; and to date, no industry-wide dominant language, set of engineering tools, techniques, reusable assets, or processes have emerged.

This lack of engineering maturity impacts both requirements development and the subject for this research, design of the architecture, which will be discussed later. 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.

### Performance Specifications and the Work Breakdown Structure (WBS)

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

In stark comparison, the software engineering environment offers little assistance in compensating for derived and implied requirements, and developers are limited to respond, almost exclusively, to the explicit requirements provided. The *DoD Handbook 881A*, "Work Breakdown Structures for Defense Materiel Items," recommends a minimum of three levels be developed before handoff to a contractor. 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 (Department of Defense, 2005, July, p. 3). Complex weapon systems are nearly always high-cost, and the complex software development needed almost always means that it 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 level of software development effort is discovered only after contract award.

The WBS provides the basis for the performance specification and is a powerful communications medium with potential contractors as the upper levels provide a functional system breakdown structure from the DoD's perspective. The same WBS continues to be developed by the contractor, eventually providing the detailed breakdown structure, which is the basis for the cost and scheduling estimates provided in the proposals and used in the Earned Value Management (EVM) metrics during execution.

### Software Quality Attributes

As the system requirements are developed, software quality attributes are identified and become the basis for designing the software architecture. One methodology for fully developing the software attributes is to use the Software Engineering Institute's Quality Attribute Workshop (QAW), which is implemented before the software architecture has been created and is intended to provide stakeholder input about their needs and expectations from the software (Barbacci, Ellison, Lattanze, Stafford, Weinstock, & Wood, 2003, August, p. 1).

While the QAW would certainly be useful after contract award, conducting the workshop between combat developers/users and the program management office before issuance of the Request for Proposal (RFP) would provide an improved understanding of the requirements, enhance the performance-specification preparation, and improve the ability of the prospective contractors to accurately propose the cost and schedule. This approach would support the goals of the System Requirements Review (SRR), which is designed to ascertain whether all derived and implied requirements have been defined.

The QAW process provides a vehicle for keeping the combat developer and user community involved in the DoD acquisition process, which is a key goal of that process. In addition, the QAW includes scenario-building processes that are essential for the software developer in designing the software system architecture (Barbacci, Ellison, Lattanze, Stafford, Weinstock, & Wood, 2003, August, pp. 9-11). These scenarios will continue to be developed and prioritized after contract award to provide context to the quality attribute. Specific recommendations for this process will be discussed later.

### Maintainability, Upgradability, Interoperability/Interfaces, Reliability, and Safety/Security (MUIRS) Analytic Technique

The QAW provides the "how," and the performance requirements (with Maintainability, Upgradability, Interoperability/Interfaces, Reliability, and Safety/Security (MUIRS) analytic technique) provides the "what"—or at least a significant portion of it. The MUIRS elements also help capture the need for Open Architecture (OA), especially in the Maintainability, Upgradability, and Interoperability/Interfaces elements. Much of the software performance that typically lacks consideration and is not routinely addressed in the software engineering environment can be captured through development and analysis of the MUIRS elements. Analyzing the warfighter requirements in a QAW framework for performance in each MUIRS area will help identify software quality attributes that need to be communicated to potential software contractors (Naegle, 2006, September, pp. 17-24). While this technique would be effective within any system, it is especially effective in compensating for the lack of software engineering maturity and in conveying a more

complete understanding of the potential software-development effort, resulting in more realistic proposals.

The MUIRS analytical approach provides a framework to capture, develop, and document derived and implied requirements—which may be vaguely alluded to or missing from the user/combat developer's requirements documents. For example, a user requirement might be simply presented in terms like, "The network must be secure in all modes within the intended environment." Without further development, the software engineer may interpret that requirement in many different ways, planning for authentication and encryption/decryption routines. Applying the Safety/Security analytic approach in a QAW format, the derived and implied requirements are likely to elucidate the following requirements:

- Ability to constantly monitor the network to detect and counteract active or passive intrusion or attacks
- Ability to provide details of attacks to Intelligence/Counter Intelligence personnel
- Ability to conduct passive measures to ensure that all node operations are conducted with authorized personnel exclusively
- Ability to quarantine a suspect node without impacting the rest of the network. Ability to lift the quarantine when properly authenticated.
- Ability to identify information provided to, or requested by the quarantined node for Intelligence/Counter Intelligence analysis
- Passive ability to authenticate information sources
- Ability to interoperate with other secure devices and networks without the risk of compromise
- Ability to accommodate network system changes and upgrades
- Ability to accommodate a wide array of users and organizations, formed into the secure network task force as missions dictate

The difference in the level of requirement development is significant, and the more complete information provides necessary performance thresholds that must be accommodated by the software design and development effort. The software architecture would likely be vastly different the implied and derived security requirements are considered. The amount of work required to meet the actual software security-performance attributes is revealed to the contractor prior to proposal preparation—which should vastly improve the cost and schedule accuracy of the proposal submitted. In addition, the software engineer gains a much more in-depth understanding of the system being developed, thereby improving the design effort described later.

Similar analyses of all MUIRS elements provide a much more complete understanding of requirements and insight into the operational environment envisioned by the warfighter. This level of understanding is absolutely crucial for effective design of the software architecture. If the design effort is started without this level of understanding of the requirement attributes, significant architectural design rework or outright scrapping of early design attempts is inevitable—resulting in increased costs and lengthened schedules.

## Software Architecture Characteristics

### Software Developer Effort

In past acquisition programs, software development was considered something that could be fielded and then "fixed" after the weapon systems were deployed. The complexity of software, interface problems and the cost for rework were grossly underestimated; the result was costly schedule slips and less-than-desired performance.

When software development was in its infancy in 1968, Alfred M. Peitrasanta at IBM Systems Research Institute wrote:

> Anyone who expects a quick and easy solution to the multi-faced problem of resource estimation is going to be disappointed. The reason is clear; computer program system development is a complex process; the process itself is poorly understood by its practitioners; the phases and functions which comprise the process are influenced by dozens of ill-defined variables; most of the activities within the process are still primarily human rather than mechanical, and therefore prone to all the subjective factors which affect human performance. (Pietrasanta, 1968, pp. 341-346)

After numerous, costly software disasters, we have learned that software development must be a parallel effort with system development within the acquisition framework to ensure that requirements are being met and usable products are being delivered to the warfighter. As the system requirements are defined, the requirements for the software should also be developed concurrently. One critical factor in the software development effort is applying systems engineering discipline to the process and ensuring that discipline is continuous and rigorous throughout the development. Software development has the highest degree of program risk and tends to evolve into a state of turmoil, which is detrimental to the goal of mission-ready software and has a negative impact on cost, schedule and performance.

### Software Functionality and Design Architecture

The design of the architecture begins with the description of the system and identifies the functions required for the system to provide the capabilities desired. The required functions will drive the design of the system architecture. System functionality and performance requirements are documented in the Government's Request for Proposal (RFP). The potential contractor must break down those functions and performance requirements and consider Maintainability, Upgradeability, Interfaces/Interoperability, Reliability, Safety, and Security (MUIRSS) in the design-decision process. The MUIRSS analysis will ensure the contractor understands the requirement and will also identify any limiting factors in the system requirements tradeoffs. The desired functionality and the analysis will drive the system architecture. For software-intensive acquisition programs, it is even more critical that the performance requirements be communicated and understood by the software developer.

## Work Breakdown Structure

The Government's requirements and specifications for a new weapon system are detailed in the RFP; this includes a Government-produced Work Breakdown Structure (WBS) to at least three levels. This is known as the Program WBS and is handed off to the contractor to develop a WBS that defines the level of detail required for product development. This contractor-generated product will ensure the system developer understands the program objectives and the products to be delivered in performance of the contract. The WBS details the functionality and performance of the system and provides a baseline to track performance against cost and schedule. For most hardware-centric programs, a WBS for the top three levels of the system under development is usually enough detail to manage the program. Because of the volatile nature of software development, immature software engineering environment, and the potential impact to cost, schedule and risk, the WBS for software intensive programs need to be developed down to Level 5 or lower for a software-intensive program—including system-of-systems (SOS) and net-centric systems development.

Level 1 of the WBS describes the entire project. If the program is a Systems of Systems (SOS) project, Level I becomes that overarching system. The Army Future Combat System (FCS) has a number of platforms that are segments of the total system. Each platform becomes a major segment of that product (Level 2); the software development would then be broken down to Level 6, which identifies software-configuration items.

Using the FCS as an example, Level 1 describes the overall FCS concept and environment. Level 2 details the major product segments of the SOS. With our example of FCS, the Level 2 would be the manned systems, i.e., infantry-carrier vehicles, command-and-controlled vehicles, mounted combat systems, etc.

Level 3 defines the major components or subsets of Level 2. For software development, decomposing the software WBS to the lowest component is critical for the developer to fully comprehend the detailed level of effort required to design and develop effective systems. Under the FCS scenario, Level 3 would be one of the subsystems on board the manned systems, e.g., the fire-control systems and environmental-control systems. It is clear that WBS definition to this level provides only a very top-level insight to the system being developed; thus, for the software-intensive system, the WBS fails to convey enough information for the contractor to propose a realistic cost and schedule estimate. Too much of the software development work is hidden at this level.

Level 4 becomes a breakout of the component parts of the subsystem. Using a manned vehicle in the FCS program, Level 5 of the WBS would identify the component functions for the fire-control system: for example, detect the target, aim at the target and fire the munitions. The software build set would support the functionality of that component within the subsystem. Again, using FCS as the overarching program, Level 6 is a sum of software items (SI's) which satisfy a required function and are designated for configuration management. If the software requirements or attributes are well defined, the result is a product that is properly designed to functionally perform to the users' requirements. Further development below Level 6 may be necessary to adequately convey the derived and implied requirements needed by the software developer.

## Systems Engineering Process

Just as it supports hardware development, the Systems Engineering Process (SEP) is essential in the development of software design. In software development, good quality and predictable results are paramount goals in creating the specified warfighter capabilities within cost and schedule constraints. To accomplish those goals, we examine the methods, tools and processes that the software developer uses in building the software with the intent of attaining a product that provides all of the necessary functionality and is supportable, efficient, reliable and easy to upgrade.

The SEP also helps identify and manage program risk. How mature is the processes of the software developer? One cause for delays and cost overruns in the C-17 Globemaster program was the contractor's lack of software experience, which is an element of the developer's maturity. To address developer maturity, SEI developed an evaluation tool in the mid-1980s known as the Capability Maturity Model (CMM) which rates software developers on key elements of maturity including experience, processes, management and demonstrated predictability. This gives the DoD insight into the maturity of potential developers as a means of risk reduction.

The system requirements, stated in the RFP, detail the software's functions, what it must do and how well, under what conditions, and identifies interfaces and interoperability requirements. The performance requirements are also analyzed for required response time, maintainability and modularity, open-architecture requirements and transportability. This phase of the SEP also addresses any restricting factors—for example, interface with legacy systems, any required operating systems—and also identifies issues such as data and software rights constraints.

The developer then identifies software attributes and decomposes functions to the lowest level, ensuring that each performance specification in the RFP has, as a minimum, one function. The functional architecture, the block diagrams and software interfaces are described during this step.

These functions are then combined into a system that describes the architecture, defines all interfaces, explains operating parameters, produces the SI's and develops the documentation, technical manuals, and any deliverable media (Kazman, Klein & Clements, 2000, August, p. vii).

## Attribute-driven Design

*"Quality attribute goals, by themselves, are not definitive enough either for design or for evaluation"* (Barbacci, Ellison, Lattanze, Stafford, Weinstock, & Wood, 2003, August, p. 3)

The design of the system architecture will be driven by the quality attributes requirements. The performance goals of the system must be defined not only in attributes or qualities, but also in how those attributes interact or interface with the system and subsystems. If those attributes are poorly communicated, the architectural design will fail to meet the performance goals and could potentially impact the overall program cost and schedule. Those critical attributes or qualities must be carefully documented and articulated to the software designer. To evaluate the architecture, the designer must receive a detailed description of the desired attributes with the overall proposed design of the system.

However, in the evaluation of the design, an analysis of the attributes may not be enough detail for the developer. The RFP or performance specification needs to address any operational requirements or constraints. Clearly, understanding the attributes in the context of how they are used is critical for the software designer.

## Software Architecture Analysis

> If a software architecture is a key business asset for an organization, the architectural analysis must also be a key practice for that organization. Why? Because architectures are complex and involve many design tradeoffs. Without undertaking a formal analysis process, the organization cannot ensure that the architectural decisions made—particularly those which affect the achievement of quality attributes such as performance, availability, security, and modifiability—are advisable ones that appropriately mitigate risks. (Kazman, Klein & Clements, 2000, August, p. vii)

This quote from the Software Engineering Institute illustrates the importance of performing architectural analysis in developing software-intensive systems.

After thorough requirements development and elicitation, architectural analysis is the next necessary step in managing the software development and serves as the SEP functional allocation step. Defining the requirements and software quality attributes is a critical first step to any program development and provides the basis for architectural analysis. One of the main functions of the architectural analyses is to understand how the quality attribute is being achieved by the design architecture and, just as importantly, is to gain insight into how those attributes interact with each other. For example, it is crucial to understand how security is ensured while the open-system architecture the DoD requires is maintained.

### Understanding Quality Attributes in Context

It is not sufficient to understand a quality attribute without understanding the context in which it will be used and sustained by the warfighter. One method of gaining the needed context is to develop operational scenarios that would place all software quality attributes into system-use cases spanning key effectiveness and suitability issues. The development and prioritization of the operational scenarios must be accomplished by the user, combat developer, warfighter, and other stakeholders—keeping them actively engaged in the developmental process.

The context in which the attributes function provides significant design cues to the software engineer. For example, the M1A2 Abrams main battle tank uses numerous inputs for precisely engaging threat targets. Several such inputs are essential for any acceptable probability of hitting the desired target, including target acquisition (finding the target), location (azimuth and range), aiming/tracking, and firing the projectile. To increase accuracy, several other systems are employed that enhance one or more of the essential functions, including cross-wind sensor, temperature sensor, muzzle-reference system, and others. The tank main-gun engagement scenario separates the essential functions from the enhancing functions, allowing the software engineer to design the software to permit an engagement when all of the essential functions are operational—even when an enhancing function, like the temperature sensor, is not working. The warfighter can continue to fight effectively using the system, increasing mission reliability. Without development of these

scenarios, every requirement and quality attribute appear to be in the "essential" category, which may result in a design that precludes critical operations when a non-essential enhancing system is not working.

### Operational Scenario Development

A scenario is a short statement describing an interaction of one of the stakeholders with the system (Kazman, Klein & Clements, 2000, August, p. 13). A warfighter would describe using the system to perform a task or mission in a range of environments (dark, cold hot, contaminated, etc.). A leader would describe system employment in concert with other joint and allied systems in a system-of-systems approach. A system maintainer would describe preventative or restorative maintenance tasks and procedures. A trainer would describe programs of instruction to task, condition and standard.

Much of the necessary operational scenario development work has been accomplished through implementation of the Joint Capabilities Integration and Development System (JCIDS) (Chairman of the Joint Chiefs of Staff, 2005, May). JCIDS is the user's capability-based requirements generation process, providing a top-down baseline for identifying future capabilities. It uses a Concept of Operations (CONOPS) analysis technique to assess current systems' and programs' abilities to provide the warfighter with capabilities to accomplish missions envisioned in the applicable CONOPS. These CONOPS provide the basis for operational scenario development.

Two of the JCIDS key documents, the Capabilities Design Document (CDD) and Capabilities Production Document (CPD):

state the operational and support-related performance attributes of a system that provide the desired capability required by the warfighter, attributes so significant that they must be verified by testing and evaluation. The documents shall designate the specific attributes considered essential to the development of an effective military capability and those attributes that make significant contribution to the key characteristics as defined in the [Joint Operations Concepts] JOpsC as [Key Performance Parameters] KPPs. (Chairman of the Joint Chiefs of Staff, 2005, May, p. A-17)

Key system attributes within the context of the CONOPS are the genesis of scenario building and will help guide the user in developing a prioritized set of operational scenarios considered essential in designing the software architecture.

### Failure Modes and Effects Criticality Analysis (FMECA)

Failure Modes and Effects Criticality Analysis (FMECA) is a type of exploratory scenario analysis designed to expose potential failure modes and their impact on the system functionality and mission accomplishment. Scenarios are developed that explore system operations in likely or critical subsystem failure modes; then, the criticality of those failures is analyzed. Operations in degraded modes are also analyzed to gain insight into graceful degradation capabilities as subsystems fail and the system is reduced to ever-decreasing levels of basic functionality. With up to 80% of weapon-system functionality in the system software, it is critical for the design engineer to understand warfighter needs and expectations in these failure modes.

FMECA scenarios with the software systems and subsystems provide architectural design cues to software engineers. These scenarios provide analysis for designing

redundant systems for mission-critical elements, “safe mode” operations for survivability- and safety-related systems, and drive the software engineer to conduct “what if” analyses with a superior understanding of failure-mode scenarios. For example, nearly all military aircraft are “fly-by-wire,” with no physical connection between the pilot controls and the aircraft-control surfaces, so basic software avionic functions must be provided in the event of damage or power-loss situations to give the pilot the ability to perform basic flight and navigation functions. Obviously, this would be a major design driver for the software architect.

### Architectural Trade-off Analysis SM

The Software Engineering Institute’s Architectural Trade-off Analysis Methodology SM (ATAM) is an architectural analysis tool designed to evaluate design decisions based on the quality attribute requirements of the system being developed. The methodology is a process for determining whether the quality attributes are achievable by the architecture as it has been conceived before enormous resources have been committed to that design. One of the main goals is to gain insight into how the quality attributes trade off against each other (Kazman, Klein & Clements, 2000, August, p. 1).

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

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

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

The 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.

The ATAM 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 upgrades, added interfaces supporting system-of-systems development, and other maturity needs; and *exploratory scenarios* involve extreme conditions and system stressors, including FMECA scenarios (2000, pp. 13-15). As depicted in Figure 2, below, 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 capability. The prioritization process provides a basis for architecture tradeoff 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 of the system over its lifecycle.

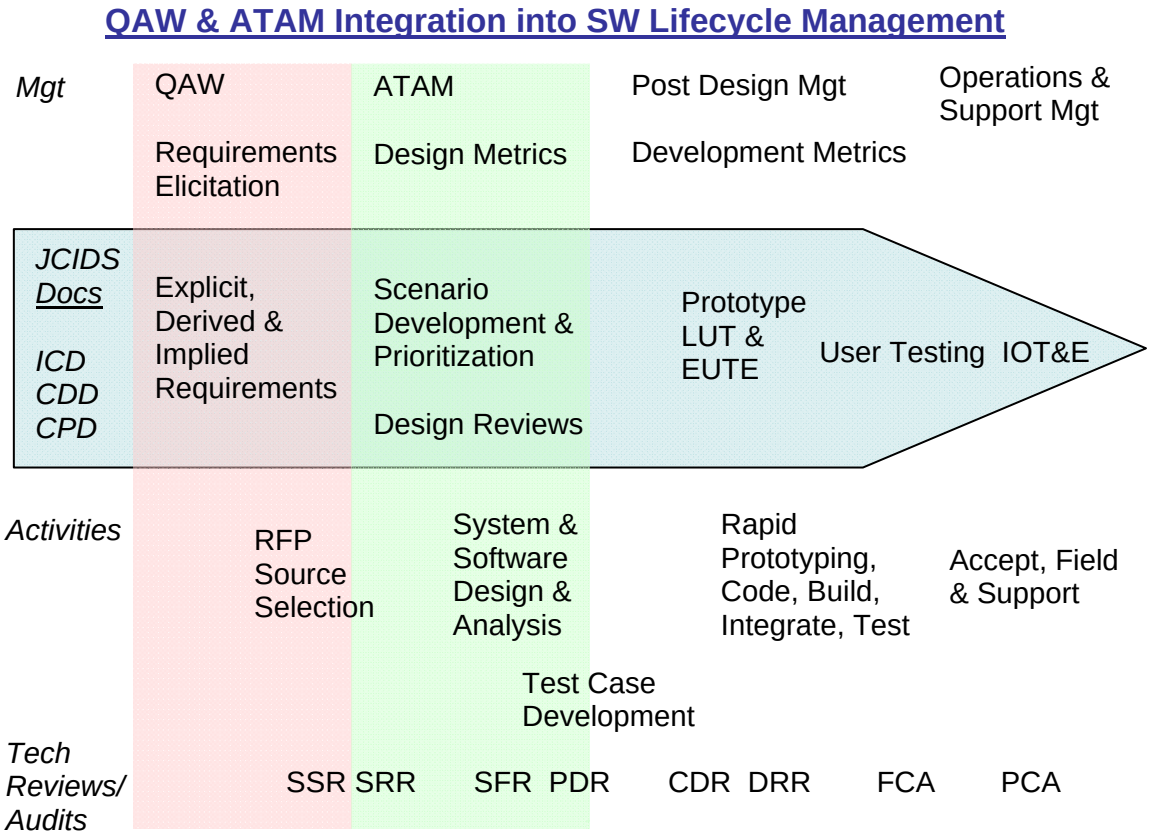


**Figure 2. QAW & ATAM Integration into Software Lifecycle Management**

Just as the QAW process provides a methodology supporting RFP and Source-selection activities, the Software Specification and System Requirements Reviews (SSR and SRR), the ATAM provides a methodology supporting design analyses, test program activities, the System Functional and Preliminary Design Reviews (SFR and PDR). The QAW and ATAM 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 2, above) 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 the visibility of the development work that is needed and the progress toward completing it.

One of the main goals in analyzing the scenarios is to find key architectural decision points that pose risk for meeting quality requirements. Sensitivity points are determined, such as real-time latency performance shortfalls in target tracking. Tradeoff points are also examined, such as level of encryption and message-processing time. The Software Engineering Institute explains, "Tradeoff points are the most critical decisions that one can make in an architecture, which is why we focus on them so carefully" (Kazman, Klein & Clements, 2000, August, p. 23).

The ATAM provides an analysis methodology that compliments 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 tradeoff 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.

### Test-case Development

A significant product resulting from the ATAM is the development of test cases correlating to the use case, growth, and exploratory scenarios developed and prioritized. Figure 3, below, depicts the progression from user-stated capability requirements in the JCIDS documents to the ATAM scenario development, and finally to the corresponding test cases developed. The linkage to the user requirements is very strong as the user 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).

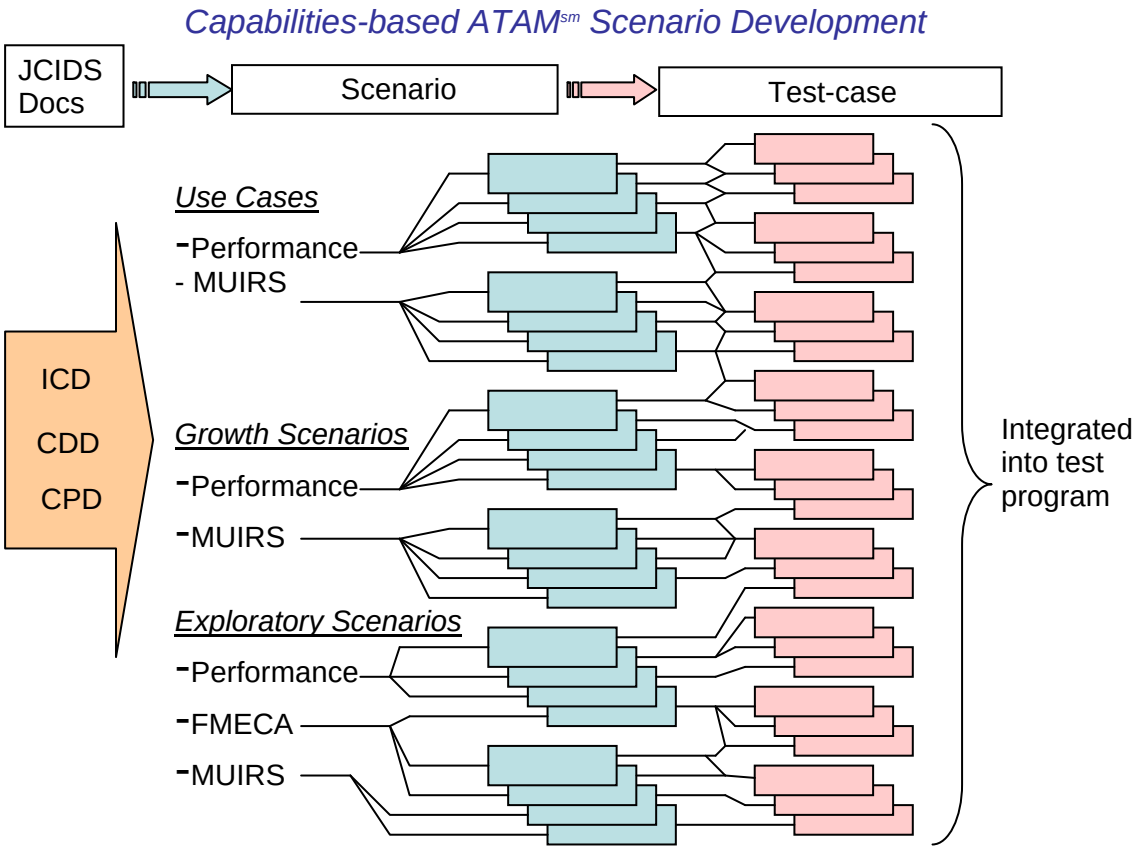


**Figure 3. Capabilities-based ATAM Scenario Development**

The software developer's understanding of the eventual performance required 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 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.

## Architectural Analysis Products

### Architecture Documentation and the Preliminary Design Review (PDR)

One of the main purposes of the PDR is to evaluate the system architectural design before committing significant resources to the construction of the system. It is a key review

in the SEP as it provides traceability from the requirements to the functional allocation of the proposed design.

It is critical to have a complete functional- or object-oriented Software Design Document reviewed at the PDR. Given that, the software developer would likely have spent 50% or more of the effort at the time of the PDR for a software-intensive system. Discovering that the proposed software design is insufficient at this point in the development cycle can be disastrous to the budget and schedule for the entire program, especially if the proposed design must be scrapped or if there is significant redesign required.

### Architecture Documentation

Documenting the process decisions in designing the software architecture provides a record of design decisions, tradeoffs made, and priorities implemented throughout the design effort and design reviews. The active involvement of the user and all system stakeholders throughout this process is one of the keys to achieving a robust design that provides warfighter capabilities and long-term, cost-effective sustainability. The ATAM provides methodologies that formalize the stakeholder participation in the architectural design.

The ATAM would help drive documentation from quality attributes to both the three types of prioritized scenarios as well as the test cases needed to demonstrate or verify performance. The quality attributes are understood in the context of the user-prioritized scenarios, so design decisions have strong linkage to user priorities. The test cases help guide the design effort as the software engineer has a very clear understanding of what the software must do, under what conditions, and to what standard. Design reviews each have a clearly defined focus, with the ATAM products providing a common understanding of what is to be accomplished.

### Scenario Inventory

One of the main products resulting from the ATAM is the prioritized inventory of use case, growth, and exploratory scenarios that drive the architectural design. As the user (along with other stakeholders) is the primary source for scenario development, the resulting design is user-oriented, not engineer-oriented.

The prioritization of the scenarios provides the basis for tradeoff analyses and design decisions, placing tradeoff decisions where they should be—with the warfighter. With the user involved throughout the design process, the resulting system is much more likely to satisfy warfighter capability requirements.

### Software and System Test Program

The development of test cases from the scenarios, as depicted in Figure 3 above, provides the Design Loop function of the SEP by ensuring that the software developed performs the functions defined by the scenarios, which represent the quality attribute requirements in context. The inventories of test cases are developed from the user-defined scenarios so that there is one or more test case for every scenario. The test cases will tend to satisfy both technical issues (as the software developed will be tested against its intended function) as well as operational issues, as each function is borne of the users' scenarios.

The aggregated test cases are part of the system's overall test program and contribute to readiness for the Initial Operational Test and Evaluation (IOT&E). The IOT&E is the defining event in the SEP Verification Loop, ensuring that the software developed satisfies user effectiveness and suitability requirements and meets warfighter capability needs specified in the JCIDS documents.

### Software Design Metrics

From the DoD's point of view, gaining insight and control of the software design process is crucial to delivering the warfighter capabilities required. In addition, metrics provide a means to monitor and control the process. The metrics chosen must provide the DoD insight into how the software architecture is designed to satisfy quality attributes and requirements across a broad spectrum of functionality and long-term sustainability performance. In addition, technically oriented design metrics such as complexity are also important, but are not the focus of this research.

The system architectural design is very much a shared responsibility between the DoD and the software developer, so metrics must also reflect developmental measures spanning both. For instance, designating the completed set of prioritized scenarios as a design metric involves measuring the build of the scenarios in a collaborative user/stakeholder/developer environment.

Using the completion of the ATAM products as metrics is logical as they are measurable, are key processes in the architectural design, and serve as indicators to the progress towards successfully completing the design process. Useful ATAM-based metrics would include:

- Business Drivers Developed
- Prioritized Scenario Sets Developed
- Attribute Utility Tree
- Sensitivity Points & Tradeoff Points
- Architecture Approach Document

## Summary

The main goal of the DoD acquisition process is to develop identified warfighter capabilities within predicted and controlled timelines and cost targets; yet, many software-intensive systems developed have experienced significant cost and schedule growth due, at least in part, to the software development component. There are many factors that contribute to the problem—including how and when the DoD conveys the needed quality attribute requirements.

The DoD acquisition model uses the Systems Engineering Process (SEP) as the central process for controlling the developmental process of its systems. The SEP is an integrated process with the DoD and the contractors selected, thereby urging shared responsibility for effective systems development. The process begins and ends with the user or combat developer responsible for providing the capabilities-based requirements, which are further developed and decomposed by the Program Manager and contractors responsible for building the system. The system components are constructed, integrated and continually tested, culminating in the User's acceptance testing, usually the Initial Operational Test and Evaluation (IOT&E).

A key to the SEP implementation is effective and complete development and communication of the system requirements. This must happen at some point for any system to be successfully developed; but *when* it happens is extremely important to the cost and schedule estimate accuracy. When the contractor has a good understanding of the work to be completed from the requirements presented, more accurate estimates are offered in the contractor's proposal before the program schedule is locked in with a contract. If a significant portion of the work is discovered through requirements decomposition *after* the contract is in place (typical of software components), the estimates provided in the proposal are severely understated, and the program schedule and budgets are no longer appropriate.

One reason the software component is more sensitive to the requirements development is that the software engineering environment is immature when compared to most hardware-centric environments. Vague or missing requirements for a hardware item may be compensated by a mature engineering environment that accommodates implied requirements. For instance, the automotive industry would provide the ability to easily replace normal wear-out items like filters and tires, whether or not such provisions were specified. The software engineering environment does not offer that level of maturity.

The MUIRS analytical technique helps capture software performance requirements that are routinely overlooked in the immature software engineering environment. The MUIRS analysis helps capture and convey Open Architecture needs, safety and security considerations, and long-term supportability performance needed by the warfighter.

In addition to simply understanding the breadth of system requirements, the software engineer needs to understand them in context of the operations, supportability, and environments to design a software architecture that is effective. It is not enough to understand what the software must do; the engineer must understand under what circumstances, in what environments, and to what standard the function must be performed.

What the DoD needs to improve the acquisition of software-intensive systems are methodologies that capture and convey quality attribute requirements in an operational context, within a Systems Engineering Process environment. The Software Engineering Institute's Quality Attribute Workshop (QAW) and Architecture Tradeoff Analysis Methodology $^{SM}$ (ATAM) provide well-suited techniques for developing requirements in context. The QAW process before contracting helps provide enough requirements elicitation for more accurate contractor proposals; likewise, the ATAM helps provide the operational context through scenario and test-case development before the software design effort. Both products support the SEP, providing methodologies for performing critical SEP functions.

DoD personnel (user/combat developer and Program Manager/materiel developer) are key and integral to the development of effective and suitable warfighter capabilities within predictable cost and schedule parameters. Improving the processes that develop and convey system quality attribute requirements in context will improve the cost, schedule and performance predictability of software-intensive systems and will reduce the supportability costs over the life of the system.

## References


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

Chairman of the Joint Chiefs of Staff. (2005, May). *Joint capabilities integration and development system.* (Chairman of the Joint Chiefs of Staff Instruction (CJCSI) 3170.01E).

Defense Acquisition University. (2004, December). *Defense Acquisition Guidebook*. Retrieved March 1, 2007, from http://akss.dau.mil/dag/DoD500.asp?view=document.

Department of Defense. (2005, July). Work breakdown structures for defense materiel items (MIL-HDBK-881A). In *Department of Defense handbook*. Washington, DC: author.

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

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

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

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

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

# 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 Lifecycle 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

H6f


**Acquisition research Program**
**Graduate school of business & public policy**
**Naval postgraduate school**
**555 DYER ROAD, INGERSOLL HALL**
**MONTEREY, CALIFORNIA 93943**

www.acquisitionresearch.org