2009

Application of model based systems engineering methods to development of combat system architectures

Banner-Bacin, Linda

Monterey, California. Naval Postgraduate School

NPS-SE-09-002

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

Application of Model Based Systems Engineering Methods to Development of Combat System Architectures

Prepared by

MSSE / MSSEM Cohort 6

Naval Surface Warfare Center, Port Hueneme Division

March 2009

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL

Monterey, California 93943-5000

Daniel T. Oliver
President

Leonard A. Ferrari
Executive Vice President and Provost

This report was prepared for the Chairman of the Systems Engineering Department in partial fulfillment of the requirements for the degree of Master of Science in Systems Engineering (MSSE) and Master of Science in Systems Engineering Management (MSSEM). Reproduction of all or part of this report is authorized.

Linda Banner-Bacin	Tim Carpenter	David Chacon	James Chandler
James Childs	Tuyen Hoang	Robert Howard	James Isaian
Seung Kang	Michael Kinberg	James Kong	Jeremy Manz
Ruth Matela	Jonathan Mendiola	John O'Neill	Leonard Ortiz
Tan Pham	Jamal Rayshouny	Eric Sarabia	Kihoon Sung
Heng Sysavath	Caleb Vajdos	Armando Valdez	Eric Vasquez
Alan Wellesley	Mindy Wentland	Paul Wheeler	

Reviewed by:

John Mike Green
Project Advisor

Raymond Madachy, Ph.D.
Second Reader

Released by:

David H. Olwell, Ph.D.
Department of Systems Engineering

Karl A. van Bibber, Ph.D.
Vice President and Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE**	**3. REPORT TYPE AND DATES COVERED**
	March 2009	Technical Report

4. TITLE AND SUBTITLE: Application of Model Based Systems Engineering Methods to Development of Combat System Architectures	**5. FUNDING NUMBERS**
6. AUTHOR(S): MSSE/MSSEM Cohort 6, NSWC PHD	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER** NPS-SE-09-002
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES: The views expressed in this report are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT: This report was prepared for the Chairman of the Systems Engineering Department in partial fulfillment of the requirements for the degree of MSSE and MSSEM. Reproduction of all or part of this report is authorized.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT: Navy acquisition activities frequently produce combat system architectures based on existing systems rather than stakeholder requirements. This approach limits software component reuse which, in turn, limits potential application to other platforms. The objective of this Capstone project was to develop a methodology for creating complex combat system architectures that emphasize the use of Software Product Lines (SPLs), requirements traceability, integrated supportability and Modeling and Simulation (M&S) early and throughout the approach. To address this objective, an integrated methodology that utilizes Model Based Systems Engineering (MBSE) to create open, supportable combat system architectures was developed. The methodology was evaluated by applying it to a naval surface combatant Anti-Air Warfare (AAW) mission area. Application of the methodology led to the following major findings: (1) Proven systems engineering practices, languages and tools can be integrated with the MBSE approach for developing complex architectures, (2) Creation of domain centered SPLs facilitates planned reuse and allows for assessment to candidate architectures, (3) Requirements traceability can be achieved by using a combination of modeling languages and tools, (4) M&S application can extend beyond operational scenarios to address life cycle cost, and (5) Engineers and logisticians can effectively use MBSE to integrate supportability into design. Overall, this project demonstrated the benefits of an MBSE approach tailored to developing affordable and supportable combat system architectures that meet mission requirements.

14. SUBJECT TERMS: Open Architecture, Supportability Requirements and Reliability Theory, Service Oriented Architecture, Department of Defense Architecture Framework, Domain Analysis, Software Product Lines, Model Based Systems Engineering, Process for Systems Architecture and Requirements Engineering (Hatley-Pirbhai method, et al.)	**15. NUMBER OF PAGES** 737 **16. PRICE CODE**

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Navy acquisition activities frequently produce combat system architectures based on existing systems rather than stakeholder requirements. This approach limits software component reuse which, in turn, limits potential application to other platforms. The objective of this Capstone project was to develop a methodology for creating complex combat system architectures that emphasize the use of Software Product Lines (SPLs), requirements traceability, integrated supportability and Modeling and Simulation (M&S) early and throughout the approach. To address this objective, an integrated methodology that utilizes Model Based Systems Engineering (MBSE) to create open, supportable combat system architectures was developed. The methodology was evaluated by applying it to a naval surface combatant Anti-Air Warfare (AAW) mission area. Application of the methodology led to the following major findings: (1) Proven systems engineering practices, languages and tools can be integrated with the MBSE approach for developing complex architectures, (2) Creation of domain centered SPLs facilitates planned reuse and allows for assessment to candidate architectures, (3) Requirements traceability can be achieved by using a combination of modeling languages and tools, (4) M&S application can extend beyond operational scenarios to address life cycle cost, and (5) Engineers and logisticians can effectively use MBSE to integrate supportability into design. Overall, this project demonstrated the benefits of an MBSE approach tailored to developing affordable and supportable combat system architectures that meet mission requirements.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Navy acquisition activities frequently produce combat system architectures based on existing systems rather than stakeholder requirements. This approach limits software component reuse which, in turn, limits potential application to other platforms. Additionally, Department of Defense Instruction (DoDI) 5000.02 prescribes the early integration of supportability requirements. Methodologies currently in use identify supportability as a requirement but tend not to maintain it as a priority throughout the development process.

In response to these issues, an integrated methodology that utilizes Model Based Systems Engineering (MBSE) and the Agile process was defined to create open and supportable system architectures. This methodology incorporates a common modeling language, utilizes domain analysis to support Software Product Line (SPL) reuse, maintains traceability of requirements and architecture functionality, and integrates supportability, sustainment and life cycle cost considerations. Also described is a system engineering process that outlines requirements generation analysis, functional analysis and allocation, architecture definition, and Verification and Validation (V&V).

The methodology was evaluated by applying it to the Anti-Air Warfare (AAW) mission thread, in particular Anti-Ship Missile Defense (ASMD). The AAW implementation included the development of a systems architecture and design artifacts, including Department of Defense Architecture Framework (DoDAF) views. The project demonstrated the benefits of an MBSE approach tailored to developing architectures that support Open Architecture (OA), SPL, and integrating supportability early in the system development process. Technical conclusions resulting from the research, development and application of the methodology are summarized in the following paragraphs.

Proven systems engineering practices, languages and tools can be integrated with the MBSE approach for developing complex architectures. Decomposition of the objectives and associated research identified many solutions and methodologies available to support a top-down or bottom-up approach. Based on tenets from multiple authors, a new end-to-end methodology for system design was developed to include key aspects in requirements generation, architecture development, and Modeling and Simulation (M&S).

Requirements traceability can be achieved by using a combination of modeling languages and tools. Traceability is critical on large complex systems due to the sheer volume of technical data and the likelihood of human error when trying to conduct V&V manually using engineering artifacts. Requirements generation and traceability were achieved using the Systems Modeling Language (SysML) as the modeling language and CORE as the architecture tool. Manual V&V errors are reduced given that SysML contains methods based on the allocation relationship depicted in the artifacts for verifying traceability. Sample test criteria and events were used to successfully verify CORE could be used to assess demonstration of requirements.

M&S can provide significant value in conducting tradeoffs during design. However, the majority of M&S is focused on verifying operational parameters within scenarios vice optimizing system design. M&S was applied using a top-down approach to verify system operational behavior and validate initial operational requirements. The software tool Extend was used to perform the simulation of a raid scenario. Through multiple variations of models and simulations, it was found that there could be anomalies or elements that need adjustment in the architecture. The unexpected results from the raw data led to more extensive research of the initial inputs, which led to additional simulation runs. Defining objectives, processes and model development were all key milestones in building the Extend model.

Engineers and logisticians can effectively use MBSE to integrate supportability into system design. The Navy advocates the integration of supportability early in the concept

development and design phases, but very little training or guidance is provided on how to effectively do this. Many logisticians are not equipped with the knowledge or experience to adequately support initial system concept and architecture development. Similarly, many design engineers lack the training and experience of considering supportability during concept exploration, design and development. On this project, engineers and logisticians collaborated to meet an expressed objective of integrating supportability into design as depicted in the resulting artifacts. Supportability was considered during requirements generation, functional analysis and architecture composition. Integrating supportability early in design provided the maintenance concept and planning with a solid foundation for conducting tradeoff decisions between operational enhancements and life cycle sustainment considerations.

THIS PAGE INTENTIONALLY LEFT BLANK

1 INTRODUCTION

1.1 PROBLEM STATEMENT

Acquisition of systems with open software architecture designs is a stated priority for the Office of the Chief of Naval Operations (OPNAV), Naval Sea Systems Command (NAVSEA) and Program Executive Office Integrated Warfare Systems (PEO IWS). This prioritization is consistent with Department of Defense (DoD) objectives to acquire and field systems with architectures that are mission capable and have the appropriate support concept that is consistent with the objective of reducing support infrastructure and life cycle cost.

The current processes for developing combat system architectures are too often defined by existing systems which inhibits the incorporation of supportability requirements up front in design and limits the integration of stakeholder requirements. Additionally, the systems engineering process currently prescribed by the DoDI 5000.02 provides guidance on the early integration of supportability requirements, but the methodologies in use do not consider supportability as a high level requirement in system architecture development.

The United States Navy (USN) (hereinafter called "Navy") currently develops software for a specific purpose (e.g., the DDG-1000 combat system) with the intent that it can be reused on other platforms. This intent has not always been achieved, resulting in a lack of commonality which increases the logistics footprint, including training, supply chain, technical documentation and other life cycle cost drivers. Stakeholders require the ability to integrate and sustain future capabilities with constrained fiscal resources. Experience has demonstrated that the management of large complex systems and technologies in pursuit of meeting Navy requirements demands a structured, repeatable method for evaluating alternatives, which can be enhanced by integrating supportability requirements using a Model Based Systems Engineering (MBSE) methodology. A barrier to effective software reuse is the lack of a common Software Product Line (SPL) with appropriately

defined domains agreed upon by all stakeholders. Implementation of SPLs supports Modular Open Systems Architectures (MOSA), which can maximize software reusability and system commonality while reducing software maintenance, testing and Commercial-Off-The-Shelf (COTS) hardware obsolescence replacement costs. Further, the SPL approach includes supportability analysis as an integral continuous activity throughout the systems engineering process. The domain modeling and analysis process should construct architectures that are best implemented by SPLs providing a framework for driving the systems engineering methodology.

Development of the SPL methodology is consistent with the Department of Defense Architecture Framework (DoDAF). The DoDAF is intended to ensure that architecture descriptions can be compared and related across programs, mission areas, and ultimately, the enterprise; thus, establishing the foundation for analyses that supports decision-making processes throughout the DoD. (DoD 2007) Modeling and Simulation (M&S) usage in support of MBSE was assessed. The goal was to understand the implications that this information has on new initiatives, such as Simulation-based Acquisition, where M&S is more fully integrated into the acquisition process.

Clearly, a shift in the current acquisition paradigm is required to achieve the goals of fielding open, supportable and interoperable combat system architectures. Instead of developing systems with potential reuse that do not fully meet interoperability and supportability requirements, new combat system architectures need to be developed based on stakeholder requirements.

1.2 CAPSTONE PROJECT DESCRIPTION

This Capstone project was performed in accordance with the Statement of Work (SOW) (Appendix A) and the Program Management Plan (Appendix B). This Capstone project developed a methodology for creating complex combat system architectures that use DoDAF artifacts as a system requirements specification and implements open,

supportable designs. In developing this methodology, the project addressed the following SOW objectives:

1. Show the viability of SPLs in developing domain specific systems built from components that allow software reuse across diverse platforms.
2. Demonstrate the viability of integrating top level supportability requirements into the overall system level requirements and system architecture development.
3. Develop an Anti-Air Warfare (AAW) mission architecture using the project methodology, including M&S, to support development of an integrated operational and support system structure.
4. Develop a management and organizational approach to support the methodology development and the first three objectives.

The following key concepts were identified as areas which could contribute to the methodology:

1. Open Architecture (OA)
2. Supportability Requirements and Reliability Theory
3. Service Oriented Architecture (SOA)
4. DoD Architecture Framework (DoDAF)
5. Domain Analysis
6. Software Product Lines (SPLs)
7. Model Based Systems Engineering (MBSE)
8. Process for Systems Architecture and Requirements Engineering (Hatley-Hrushka-Pirbhai methods, et al.)
9. The Systems Engineering "VEE" model
10. Net-centric architectures

Chapter 3 reviews and discusses the application of these concepts as applied to the methodology.

This project provides the Navy a notional methodology for complying with Open Architecture (OA) and life cycle cost reduction mandates. The methodology complies with requirements to use a simulation-based approach, and supports the development of a repository of reusable combat system software.

1.3 BACKGROUND INFORMATION

This section stresses the importance of addressing the issues discussed and amplifies its importance to the future of combat system acquisition. After providing a historical background, an overview of some of the terminologies and key concepts listed in section 1.2 will also be covered.

1.3.1 Historical Background

The DoD has been working for decades to develop and field systems that are interoperable with other systems within the U.S. military services, between the services and with allied forces. Improved technologies and increased capability requirements has increased the challenge of fielding interoperable System of Systems (SoS) and interoperable Family of Systems (FoS). System designers and technical agents have had to create complex architectures to describe and define these systems. With no standards in place for depicting these architectures, inconsistencies arose with the way architectures were developed and documented. These inconsistencies with requirements and design documentation exacerbated the incompatibility problems of the underlying systems.

It has long been recognized within the DoD that better acquisition outcomes are needed to accomplish DoD transformation objectives in current fiscal environment. Without improved acquisition outcomes, achieving DoD's transformation objectives will be difficult given the current fiscal environment. DoD is currently investing in weapon systems that it is depending on to transform military operations. While these weapon systems are expected to provide unprecedented capabilities, the cost and complexity to develop these new systems will be exceptional. However, the nation's long-term fiscal

imbalances will likely place pressure on the affordability of DoD's planned investments. Without better acquisition outcomes, there is greater risk that DoD will not be able to achieve its transformation objectives. (GAO 2007)

1.3.1.1 Department of Defense Architectural Framework

In the early 1990's, the DoD began to address the problems with acquisition by developing a document that provided definitions, standards and frameworks for the architectures developed for DoD systems. The Command, Control, Communications, Computers, Surveillance and Reconnaissance (C4ISR) Architecture Framework, approved in 1996, was the first of these documents and eventually revised into the DoDAF Guidance Document. The most recent DoDAF document, version 1.5, was approved in April 2007. The importance of DoDAF will be revisited in later sections.

1.3.1.2 Systems Engineering Methodologies

The efficient and effective development of complex SoS architectures requires sound systems engineering processes and an effective overarching methodology. Many related but differing systems engineering processes have been developed over the years to support the development of complex systems. MBSE, which incorporates the use of M&S to support the development of system design, is one process found to be particularly effective for complex systems. In addition, the Hatley-Pirbhai (HP) method for systems architecture and requirements engineering has also been recognized as an effective systems engineering process.

The Navy and private industry recognize that reuse of software can decrease life cycle costs and enhance the ability of systems engineers to field new technologies and capabilities that meet current and future requirements. A key area that has not been fully explored and incorporated into MBSE is supportability, which is vital to the evolution of MBSE and will also be addressed herein. This project takes an academic approach to proposing how to increase MBSE functionality and applicability to meet current and

future Navy systems development, modernization and life cycle support requirements. The Navy has long recognized the value of OA for combat systems, but has been limited in its ability to take full advantage of it by rapid advancements in computer technology and the acquisition community's processes. The key OA principles espoused by the Navy include the following (Green 2008a):

1. Modular design and design disclosure
2. Reusable application software
3. Interoperable joint warfighting applications and secure information exchange
4. Life cycle affordability
5. Encouraging competition and collaboration through development of alternative solutions and sources

As systems become increasingly complex and the use of OA becomes more common, system architectures must be designed to accommodate efficient and rapid insertion of new technologies, allow maximum reuse of software, as well as early consideration of supportability in the system architecture development phase. The DoD and private industry currently accept MBSE as a method for proving concepts, reusing architectures and incorporating technical performance attributes into the system's architecture and throughout the entire systems engineering process. (Green 2008a)

1.3.1.3 Open Architecture and Software Product Lines

A big part of ongoing DoD Acquisition Policy initiatives is focused on the high cost of software development, testing and maintenance. The Navy recognized that many of its tactical and non-tactical military systems had common functional software requirements. Accordingly, this meant that it should be possible to reuse the software developed for similar functions in different applications and systems, thus reducing the overall cost of software development. Unfortunately, previously developed and fielded systems were found to have unique, tightly integrated closed architectures that do not facilitate software reuse and low-cost modification. It was determined that limiting the cost of future

software acquisition and maintenance required the development of open software architectures with modular integration structures that facilitated the reuse of software modules, as well as the integration of new or upgraded software modules and software maintenance activities. The ultimate implementation of OA acquisition was the development of SPLs with open, compatible architectures and common services and applications capable of supporting reuse, interoperability and supportability in diverse military systems. (Fein 2008)

1.3.1.4 Supportability Benefits of Common OAs and SPLs

In 2004, then Assistant Secretary of the Navy for Research, Development and Acquisition (ASN RDA) John Young established the Open Architecture Enterprise Team (OAET) for the following reasons.

> To oversee the development and implementation of enterprise-wide OA processes, business strategies and technical solutions,…computing architectures in the fleet that are performance-limited and expensive to upgrade [and that] implementing OA across the Navy will increase processing capacity, provide system growth potential, reduce cycle time for future upgrades, and enable common, interoperable war fighting capabilities to be fielded at reduced cost. (Guertin 2008)

Programs such as Sea Athena and Common Command and Decision were predecessors in developing the current approach to OA systems for the Navy. (Green 2008a) Studies have indicated that commonality significantly reduces subsystem ownership costs by reducing the costs of acquisition, operations and support. Subsystem commonality also increases mission effectiveness by reducing cycle time, improving reliability and availability, and mitigating obsolescence. (Nuffort 2001)

1.3.1.5 Early Integration of System Supportability Requirements

Supportability is commonly the last element to be considered in the design development process, even though it is the most significant determinant of life cycle operation and support costs, which are typically the largest contributors to total ownership cost. (Van

Bennekom and Goffin 2002; Miles 2008) Design engineers may lack the requisite training and experience to develop systems that meet supportability parameters; thereby, utilizing a common methodology which incorporates supportability parameters can close the gap on addressing supportability considerations in design.

Department of the Navy (DoN) program executives and managers are typically incentivized to ensure systems and products are developed and delivered that meet fleet functional requirements on schedule and within budget. As a result, decisions are often made that sacrifice long term supportability and life cycle cost savings in favor of short term considerations, such as production. On the LPD 17 shipbuilding program, for example, decisions were made to defer logistics; supportability products and services, such as technical manuals, training materials, spare parts, Planned Maintenance System (PMS), computer and Information Technology (IT) support, etc., were provided after the delivery of the systems and equipment they were required to support. In many cases, late or deferred development of supportability design items contributed to a recent finding that two-thirds of DoD systems completing operational testing over a ten-year period (1997-2006) failed to meet reliability requirements. (DSB 2008)

1.3.2 Key Concepts Overview

Different definitions were found in the various reference documents for some commonly used terms. In order to effectively describe the relationships between the key concepts addressed by this project, it is important that we establish the definitions for the terminologies associated with them. An Integrated Dictionary (Appendix C) provides a list of the key concept definitions used throughout the report. The following are some of the key terms:

- **Department of Defense Architecture Framework (DoDAF):** The DoDAF defines a common approach for DoD architecture description development, presentation and integration. The Framework is intended to ensure that

architecture descriptions can be compared and related across organizational boundaries, including Joint and multinational boundaries. (DoD 2003)

- **Domain Analysis:** Domain analysis is "the process of identifying, collecting, organizing, and representing the relevant information in a domain, based upon the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory, and emerging technology within a domain." (Kang et al. 2007)
- **Hatley-Hruschka-Pirbhai (H-H-P) Methods:** Methods that support the *Process for System Architecture and Requirements Engineering*, described in the book of the same name written by Hatley, Hruschka and Pirbhai. (Hatley et al. 2000)
- **Interoperability:** The ability of systems, units or forces to provide data, information, material and services to, and accept the same from, other systems, units or forces, and to use the data, information, material and services so exchanged to enable them to operate effectively together. (DoD 2004)
- **Model Based Systems Engineering (MBSE):** The discipline of systems engineering in a *model-based* or *model-driven* context. (Estefan 2008)
- **Open Architecture (OA):** OA systems implement sufficient non-proprietary specifications for interfaces, services and supporting formats to enable properly engineered components to be utilized across a wide range of systems with minimal changes, to interoperate with other components on local and remote systems, and to interact with users in a style that facilitates portability. (NSWCDD 2004)
- **Service Oriented Architecture (SOA):** A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations. (Nelson 2007)
- **Software Product Lines (SPLs):** An SPL is a set of software intensive product systems that share a common, managed feature set satisfying a particular market segment's specific needs. (Bosch 2000)

- **Supportability of Systems:** Provision of maintenance, training, test equipment, technical documentation, supply support, facilities, transportability, human systems interfaces and other non-functional requirements to ensure a system is usable, reliable and maintainable throughout the planned system life cycle.
- **System of Systems (SoS):** A set or arrangement of individual systems that are related or connected to provide a capability. The loss of any part of the system will degrade the performance or capabilities of the whole.
- **Systems Engineering "VEE" Model:** A design and integration process that emphasizes engineering activities. A variant is "Incremental Development", in which a useful subset of a system is produced initially, followed by the sequential enhancement of the initial subset with expanded versions, until the full system is operational. (Buede 2000)

1.4 PROJECT SUMMARY

The goal of this project was to develop a methodology for creating complex, supportable system architectures that integrate DoDAF artifacts with the acquisition requirements process, while implementing OA and SPLs. The Capstone Project Team successfully developed an MBSE methodology that can be used to create complex combat system architectures. Existing approaches and related concepts were researched to support the development of a methodology that meets the objectives listed above and in the SOW. The methodology was demonstrated by applying it to an AAW mission thread. M&S was used to verify and validate the requirements, generate data and design artifacts, and to provide the documentation required to demonstrate the process.

1.4.1 Project Report Description

This Capstone Project Report presents the project results using a standard thesis paper format. Chapter 1 introduces the problem, describes the project objectives and discusses its importance to future combat system architecture development. It also provides the historical background and summary of key concepts. Chapter 2 summarizes the literature

research conducted, emphasizing research findings on the key concept areas. Chapter 3 describes the developed methodology and its corresponding products. Consequently, Chapter 4 describes the application of the methodology to the AAW mission thread, and provides details on the computer models and tools used, the artifacts and products generated and the degree of process validation that was achieved. Finally, Chapter 5 summarizes the project report and provides conclusions, lessons learned and recommendations for follow-on work.

THIS PAGE INTENTIONALLY LEFT BLANK

2 RESEARCH SUMMARY

2.1 OVERVIEW

One of the most important aspects of any project is the upfront research. This helps to describe what has been examined and analyzed in the past and creates a path forward to accomplish overall project goals. The goal of this research was to gain knowledge and understanding of Model Based Systems Engineering (MBSE), Open Architecture (OA), Software Product Lines (SPLs), Modeling and Simulation (M&S), supportability and other key concepts, as well as existing methods and practices in use today to develop the methodology recommended in this report. Figure 2-1 depicts the process used to focus research in areas considered critical to achieving the stated objectives. Key focus areas were assigned to address the specific questions discussed in Chapter 1. General research was also conducted to assist in developing the methodology used for the development of complex system architectures.

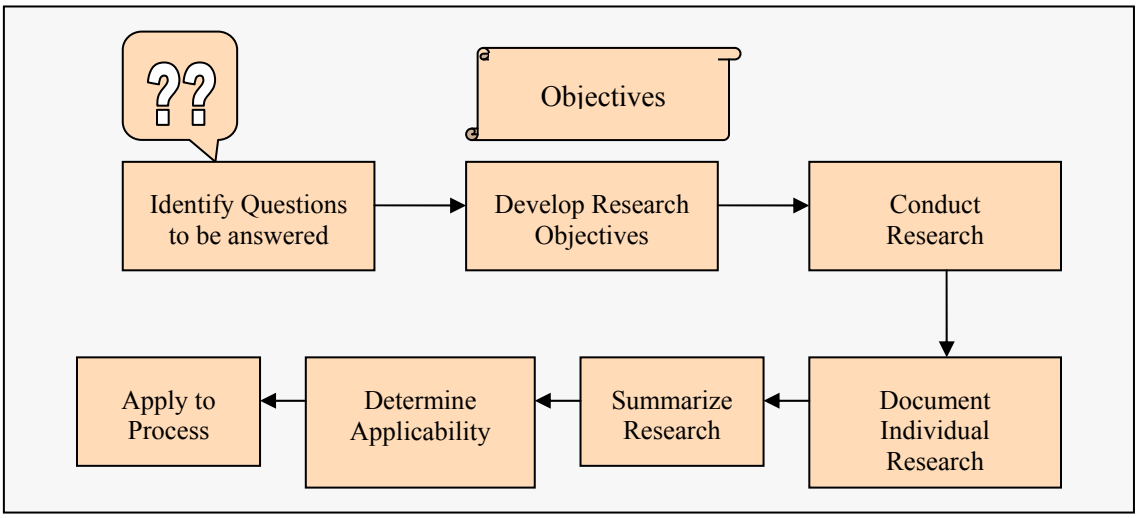


Figure depicts the research process used to focus on the critical areas needed to achieve the Capstone objectives.

Figure 2-1: Research Process

Questions were generated and allocated to various team members to facilitate individual research. A summary format was used to facilitate the sharing of information among team members and to ease the use and understanding of the researched material.

Additional research was conducted to answer specific methodology questions and to resolve identified technical issues.

2.2 RESEARCH SCOPE AND DEPTH

Appendix D provides a compilation of the research summaries conducted during the first phase of this project. Research material included white papers, published articles, textbooks, interviews, presentations, and information obtained during graduate studies. Table 2-1 summarizes the quantity of research documents organized by specific areas. The scope of information ranged from supportability-specific to theoretical models and processes for developing architectures, technical policies and instructions. Research was conducted to define the appropriate role of M&S in MBSE, and to identify existing processes that could be implemented to develop combat system SPLs that support software reuse and apply the DoD Architecture Framework (DoDAF) as a specification.

Table 2-1: Summary of Research Artifacts

Note: Some research artifacts contained information on more than one key concept.

Research Areas	Research Artifacts Quantity
Open Architecture	14
Service Oriented Architecture	2
DoD Architecture Framework	8
Domain Analysis	6
Software Product Lines	8
Model Based Systems Engineering	23
Systems Engineering "VEE"	3
Software Reuse	6
Process System Architecture & Requirements Engineering	3

Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

Table 2-1: Summary of Research Artifacts (cont.)

Research Areas	Research Artifacts Quantity
Concept of Operations	1
Software Architecture Types	7
Modeling & Simulation	3
Systems Modeling Language	13
ExtendSim Tools & Discrete Event Modeling	2
CORE	4
Reliability Theory	3
Supportability	7
Anti-Air Warfare (P_{RA}, etc.)	10
Total = 123	

2.3 KEY CONCEPTS AND ANALYSIS

Numerous resources were available to conduct research in each of the topic areas. The topic areas provided significant lessons learned and proposed benefits and limitations relative to the specific subject. In summary, it was found that there was limited information to support integration of the different aspects of systems engineering, such as integration of M&S with the Hatley-Pirbhai process.

Decomposition of the objectives and associated research identified many solutions and methodologies available to support a top-down or bottom-up approach. Numerous languages, tools and processes were found to provide solutions supporting an MBSE approach. Reusability was addressed from academia, and DoDAF artifact development was well-documented in terms of descriptions and expected information. Requirements traceability was also addressed in the majority of processes. The challenge in sorting

through the research was differentiating between theory and an implementable application. Additionally, the majority of authors focused on a specific aspect of a systems engineering process vice an end-to-end approach that would provide a closed-loop systems engineering process. The research topics were described and captured as independent subjects and included benefits and limitations. The following sections summarize the limitations that were found and outline the cornerstones of a proposed approach that will be described in Chapter 3.

2.3.1 Open Architecture

The research concluded that OA is a viable way to reduce life cycle cost by decreasing the investment cost of Research, Development, Test and Evaluation (RDT&E), and allowing for a more Agile system by leveraging off commercial standards. OA was found to be critical in meeting modern battlefield requirements by providing significant flexibility in design architecture to improve affordability, interoperability, supportability and performance. OA recommendations include consideration of small businesses, including the warfighter in all phases of acquisition and development, maintaining open design standards, use of business case analyses to determine OA priorities, emphasis on interface management, and use of proper contract types to influence and incentivize. (Guertin 2008)

- Benefits: Key benefits include improved affordability and increases in adaptability and capability with a decrease in development time. Use of COTS shortens the development timeline, and therefore the total time needed for developing and fielding a new capability.
- Limitations: Research uncovered little to no information regarding changes that could be implemented during the Concept and Design phase from either an analysis or requirements basis. Minimal data was found to identify how the use of COTS could change the way supportability of systems is currently executed. Numerous articles address the potential benefits, but little information was found on how COTS can change the traditional processes or how it can be used to improve sustainment capabilities.

2.3.2 Service Oriented Architecture

SOA was researched for potential application to an architectural style supporting loosely linked services that are interoperable and technology–agnostic. An SOA platform is comprised of three functional components: a universal data access layer, a metadata repository, and an enterprise data bus. This type of platform was cited as a viable way to incrementally increase capabilities from a SoS perspective by providing an ideal framework for data integration technology. (Levis 2008; Informatica Corp. 2005)

- Benefits: Key benefits include use of open standards for integrating COTS and OA, allowing for flexible use of developmental methods, leveraging a common infrastructure to optimize reuse, and supporting a flexible information environment with a net-centric focus.
- Limitations: Few limitations were found in the reference material regarding the use of SOA. However, since it primarily benefits SoS, SOA is considered beyond the scope of this project. While SOA has demonstrated value in the business world, it has yet to be proven in the combat systems or Command, Control, Communications, Computers, Combat Systems, Intelligence, Surveillance and Reconnaissance (C5ISR) realms. The emphasis on this paradigm can force the architecture to take on a form that may not be suitable when real-time system response is required, such as supporting a weapons system error budget. (Green 2008a)

2.3.3 Department of Defense Architecture Framework

DoDAF is a collection of required artifacts for new acquisition programs (ACAT I or II based on CJCSI 3170.01G), and was one of the most heavily researched topics. DoDAF is defined as a "common approach for DoD architecture description development, presentation and integration." (DoD 2007) The framework is intended to ensure that architecture descriptions can be compared and related across organizational boundaries, including joint and multinational boundaries. DoDAF depicts the system in various views to define system attributes, including All Views (depicting overarching aspects of the architecture), Operational Views (depicting tasks and activities, operational elements,

and information exchanges), System Views (descriptions of systems and interconnections) and Technical Standards Views (minimal sets of rules governing the arrangement, interaction, and interdependence of the system's parts or elements). Research found that there is some variance in the order in which the views should be developed. (Dam 2006)

- Benefits: DoDAF benefits include standardization of products and artifacts, and facilitation for use of MBSE and a top-down approach, which allows traceability from mission to system components. As described in Appendix E, the DoDAF can be viewed as a specification model for architecting complex, interoperable systems. DoDAF enhances the ability to determine reusability of SPLs based on common views. When combined with DoDAF, the Systems Modeling Language (SysML) provides the modeling notation, backed with the formal semantics of its meta-model, while the DoDAF provides classification and is useful for presenting the operational and system descriptions.
- Limitations: Information on how to integrate DoDAF with other systems engineering processes was limited. For example, DoDAF guidance did not address how to integrate or design supportability into the system, nor did it discuss use of M&S to optimize system design. The framework guidance does not provide a methodology for development; it only provides defined products or artifacts. Additionally, DoDAF is limited in the use of standardization for producing artifacts with a common approach.

Compliance with DoDAF, based on CJCSI 3170.01G, enhances the methodology for developing programs to achieve net-centricity and interoperability. It provides a basis for minimum architecture views in a model-based graphic to improve communication and understanding of requirements. Key shortfalls included descriptions of interrelationships among the different systems engineering functions when using DoDAF. Research did not provide significant insight into integrating supportability parameters when developing DoDAF artifacts, using DoDAF for managing SPLs and domains, applying M&S to optimize design, or using systems engineering processes to go from one view to another. Very little information was found on how early logistics planning can be accomplished

when following DoDAF. Correlation to logistics tasks, such as supportability analysis, was also not clearly articulated. Additionally, a process was not found for applying Reliability, Availability and Maintainability (RAM) considerations using DoDAF to allow a top-down approach for a mission-level SoS Operational Availability (A_O), which could be used to assess domains and SPLs for suitability.

Research also indicated numerous methods and sequences for developing the associated views along with processes that follow a systems engineering methodology. M&S application when developing DoDAF views could provide benefit to optimize allocations of functions to systems and provide quantitative data for assessing and assigning risk at a mission or system level that could be used for planning future T&E requirements based on technology forecast. Additionally, the majority of information assumes an inherent systems engineering process to develop the views, which is not clearly defined. An example would be transitioning from an operational view, which is based on activities, to a systems view, which is based on interconnection functionality. The researched literature did not identify systems engineering practices, such as attribute optimization, for allocating functions or provide a recommended approach, such as Hatley-Pirbhai, to allocate functions into physical domains. Although DoDAF provides a method for a top-down approach to developing weapon systems based on mission, numerous areas are not fully defined to provide a total systems approach.

2.3.4 Domain Analysis

Domain analysis was researched to identify existing approaches being used to develop weapon systems. According to Kang, domain analysis is:

> The process of identifying, collecting, organizing, and representing the relevant information in a domain, based upon the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory, and emerging technology within a domain. (Kang et al. 2007)

Nilson further states that domain analysis should:

> carefully bound the domain being considered, consider commonalities and differences of the systems in the domain, organize an understanding of the relationships between the various elements in the domain, and represent this understanding in a useful way. (Nilson et al. 1994)

Data from the PEO IWS, Space and Naval Warfare Systems Command (SPAWAR), and the Navy Technical Reference Model were assessed to determine differences in domain structure and attributes. Additional work is still required to enable software reuse at an enterprise level across multiple systems and platforms. Processes for optimizing software domains and for standardizing domain functionality are not likely to be solved in the short term. (Larish 2008) Numerous sources provide approaches for conducting domain analysis based on architecture construct to optimize for a specific set of attributes, which may be performance, cost or sustainment based. Other sources also showed how to structure data within a repository for analysis and reuse of domains on a recurring basis. Although numerous architecture development methods include steps or provisions for conducting domain analysis, very little was found outside of the Bosch literature on how to conduct analysis by a quantitative method that facilitates predicting life cycle cost and sustainment requirements based on alternative architecture constructs. (Bosch 2000)

- Benefits: Significant benefits can be achieved by conducting domain analysis in identifying life cycle requirements and cost. When a proposed architecture construct is modeled and simulated based on the Concept of Operations (ConOps) or life cycle scenarios, it provides valuable insight to determine the optimal construct based on the quality attributes assigned. Numerous references and proposed constructs for domain structure and analysis are available for leveraging best practices.
- Limitations: Domain structures vary significantly across platforms, missions and stakeholders, creating confusion in domain definitions and development of common terminology or structures and associated functionality.

2.3.5 Software Product Lines

SPLs were researched to develop a methodology that can support planned and unplanned software reuse. Many references are available on methods for constructing SPLs, which are defined at various levels depending on the authors (e.g., Eriksson 2003 and Krueger 2008). SPLs and the relationship to architecture and establishment of a domain library are discussed in numerous documents. Software architecture styles, such as Object-Oriented (OO), event-based and layered systems, were investigated to determine which architecture would best support an SPL structured for reuse capability using OA and COTS.

- Benefits: Establishing SPLs that have defined architectural functions and are traceable or identifiable given defining parameters, such as inputs and outputs, could provide significant value in terms of schedule and cost efficiencies. By having a systematic method for reusing SPLs, the developer is able to decrease development requirements and potentially decrease the associated cost of not only requirements generation, but also of other areas such as integration, M&S, T&E and supportability. At a macro level, having ready-for-use SPLs could create common or standard practices that could lead to an increased level of standardization as applied to human factors engineering for the Graphical User Interface (GUI). (Eriksson 2003; Krueger 2008)
- Limitations: Key limitations include significant initial investment and development of a repository and associated standards to provide common guidance for developing SPLs. Historically, most programs are under cost, performance and schedule constraints, and will therefore be limited in their abilities to apply additional resources to meet future capabilities, reduce cost and shorten schedules. No formal requirements exist at the DoN or DoD levels to mandate a minimum level of architecture guidelines to establish or maintain an SPL repository that could be used beyond a specific program. SPL literature was often correlated to the use of DoDAF and SOA, although overall guidance was not found on how to select and define the appropriate construct for the SPL based on use of an SOA, federated or OA approach.

The focus of research on MBSE, SOA frameworks, SPL architecture and systems engineering processes was on demonstrating the viability of SPLs in developing warfare domain-specific systems built from components that allow software reuse across multiple and diverse platforms (e.g. submarines, surface ships, and others). Although numerous authors provided pieces of the solution, no single author has pulled all the elements together to resolve the problems. Gomaa and Farrukh's portrayal of a structure to develop software in a product line approach, coupled with Bosch's methodology for assessing software architecture against quality attributes (including sustainment criteria) goes a long way toward meeting the objective. In addition, the Hatley-Pirbhai method provides a basis for structuring the software to optimize performance and component reuse. However, several issues still remain unresolved.

Issues included a lack of common criteria or definitions that hampers the ability to develop a common library for use in developing SPL architectures. A formal or standardized set of criteria is required to assess the SPL against requirements and interfaces to determine the ability to reuse. Maintaining a relational database becomes increasingly more difficult within warfare domains as the assessment for capability of reuse moves from a high-level, such as the detection function in AAW, down to a reuse *code level*, such as the track trajectory algorithm. Establishment of standardized criteria to maintain as part of a library system would significantly improve the capability to assess a SPL for reuse. Compounding the issue is the lack of an oversight person across all warfare domains and components, creating developer-focused solutions and criteria.

Additional issues include how SPL assessment can be integrated into the M&S and T&E processes to reduce development and testing costs while maintaining a high level of SPL capability. M&S and T&E references did not provide clear guidance on how concepts for software reuse could be applied to T&E; they simply assumed that it would reduce T&E costs. A structure based on criteria, which could be traced to T&E efforts, could not only support software assessment, but also facilitate early T&E planning, thereby allowing a program to better plan total integration, estimate testing costs and develop

schedules. Finally, integration of logistics and supportability was poorly defined in virtually all SPL development processes. The majority of examples found during research was focused on operational aspects and were not related to system design and development.

2.3.6 Model Based Systems Engineering

MBSE research focused on tools, languages and other areas, such as relationships to DoDAF. MBSE can support development of SPLs that can be reused and artifacts that would minimize ambiguity. MBSE is heavily supported by developmental and system architecture tools, such as CORE and other Computer-Aided Systems Engineering (CASE) tools. Key elements of MBSE include use of behavior models that identify functions, states and controls of a system. The language selected for the project was SysML, which was developed to support MBSE. SysML is a general purpose modeling language for systems engineering applications. It is a dialect of the Unified Modeling Language (UML), the industry standard for modeling software-intensive systems. It supports the specification, analysis, design, verification and validation of a broad range of systems and SoS. (SysML Partners 2008) SysML provides three types of behavior constructs: interactions, state machines and activities. The majority of SPL research referred to the use of an MBSE approach to optimize SPL reuse and lower costs through reduced design time.

An MBSE approach used for developing requirements for weapon systems is not a new concept for the Navy. Oliver stated that:

> Modeling is used to reduce the time and effort expended by engineers shortening the design cycle time. It is used to check the information for consistency and completeness reducing the error rate. It is used to preserve the current engineering results for use during later maintenance, product upgrade, or product replacement efforts. It is used to describe unambiguously; every symbol and number such that each has one and only one meaning. The models ensure that at the end of the process all necessary information is available and correct. Modeling in no way substitutes for creative engineering thinking and problem solving.

> Creativity and new solutions come from the engineers. Modeling reduces their manual work and improves accuracy. (Oliver et al. 1997)

In today's environment, systems engineers capture requirements and specifications of behavior and structure in a text/graphics/tabular method. While this process persists, each downstream engineering discipline will continue interpreting language specifications instead of receiving data in a more knowledge-based approach. The manual interpretation efforts are costly, error prone and waste valuable resources (time of skilled engineers/logisticians used to solve other problems). Automated tools will ensure correct information is conveyed both up and down the systems engineering spectrum.

Models improve clarity and improve communication of concepts and ideas between project personnel. MBSE, in conjunction with modeling tools, can also bridge the divide between engineering disciplines and logistics by maintaining *meta-data*. SysML, DoDAF and AP233 are key standards and enablers to support model driven systems engineering, although there are some limitations, including lack of ability to show timing, interaction overview and communications diagram. Some estimates indicate that the use of model based product development in large products can reduce cost by as much twenty to one. MBSE is more applicable to supporting inclusion of COTS and reusable components. Use of graphical tools have matured and increased with application of MBSE. In summary, MBSE as a methodology can be characterized as the collection of related processes, methods and tools to support the discipline of systems engineering in a model based or model driven context. (Oliver 1998)

- Benefits: Key benefits include clarity of requirements and the ability to decompose structures and functions. Additional benefits include the ability to move data from one artifact to a corresponding artifact that may support a different function or requirement. MBSE is considered a critical part of Simulation-based Acquisition, which is a key element of designing supportability into the system. (King 2007) Many references indicated that MBSE results in higher quality and lower cost, and supports deployment of requirements to multiple languages due to the graphical representation of the requirements.

(Oliver 1998; Hardy 2006) It also enhances the ability to conduct design analysis through runtime models and simulations, and makes requirements and rationale accessible to all designers. MBSE significantly enhances traceability between architecture and requirements, and supports a Modular Open Systems Architecture (MOSA) approach, which is used heavily for optimizing quality and reducing supportability costs throughout the life cycle. (Oliver et al. 1997; Fisher 1998)

- Limitations: MBSE can require additional effort during development of systems engineering artifacts to include data that allows traceability between artifacts and flow of relationships. MBSE may require use of multiple languages to allow domain-specific model languages to represent "aspects of interest". The use of MBSE can be difficult when upgrading legacy systems, subsystems or components due to the requirement to reverse engineer. Minimal information was found on how to design and integrate supportability into MBSE, and on how to integrate M&S in early phases to optimize design and meet acquisition objectives. (Schmidt 2006)

2.3.7 System Architecture Approaches

System architecture was researched to identify various approaches to develop architectures. Focus included comparisons of structured methods versus OO methodology. Both methods are capable of being applied in an MBSE approach using SysML. When OO is combined with SysML, it can provide bidirectional verification between requirements and architecture and can readily support the use of M&S during early development. The various software architectures have unique benefits and limitations that are described in numerous research sources, several of which indicated that use of layered structures within the OO approach provides for the optimal structure to minimize life cycle costs in a COTS/OA environment.

- Benefits: OO more readily supports development and deployment of systems in COTS and OA environments, and is more suitable to development of SPLs intended for reuse capability. (Rickman 2000)

- Limitations: Similar to MBSE, a considerable amount of information regarding the construct of the architecture is available, but little is provided on how to integrate supportability into the process. Few references address what, if any, changes to other areas, such as M&S, T&E and integration, are required or would be beneficial. Research indicated that OO might not support large system design due to immaturity while structured analysis is very mature and used traditionally.

2.3.8 Modeling and Simulation

A significant amount of information was found on developing a methodology to optimize the use of M&S as an integration tool, citing how it could be used to improve decision-making early in the acquisition process. Additional research of M&S in Navy acquisition can be found in Appendix F.

- Benefits: M&S enhances the ability to optimize domain structure and forecast life cycle costs of the associated architecture. A variety of tools and capabilities exist to accomplish M&S for both operational and supportability objectives and decision-making. Support exists within the Navy infrastructure to provide information, guidance and awareness, but there appears to be funding limitations to establish an overall capability for applying M&S as an enterprise solution due to program-unique requirements. Established criteria exist to support the validation of M&S tools and models for use in programs in addition to guidance providing recommended structures and processes. While processes and initiatives are being implemented by the Navy, several issues were also cited.
- Limitations: Key issues include lack of requirements to apply M&S as a mandatory effort during domain and SPL development. Formal milestones, by functional areas, to mandate M&S requirements are not directly related to domain architecture, nor are there common standards that integrate program-level direction and documentation. M&S efforts applied to DoDAF are not clearly documented. Researched literature primarily focused on development of the views, but not on integration of other functional disciplines such as M&S or supportability. A clear understanding of data required to support M&S efforts for

suitability and mandatory sustainment Key Performance Parameters (KPPs) is not well understood or articulated. Although M&S can play a significant role, the majority of systems engineering references (Buede 2000; Blanchard and Fabrycky 2006) do not provide a clear understanding of how M&S can be integrated into an SPL approach or how M&S can be applied using DoDAF to optimize design and validate that the planned architecture meets the associated requirements.

2.3.9 Reliability Theory and Supportability

Reliability theory and supportability were researched extensively, with majority of the findings focused on the production and sustainment life cycle phases. Few recommendations for improving supportability during system concept or design with MBSE, SysML, SPL or SOA were found. Most of the research on supportability provided lessons learned, emphasized its effects on life cycle costs, and showed how supportability functions affect the systems engineering process. A key exception was research conducted on simulation-based acquisition, which establishes an information environment in conjunction with the systems engineering artifacts, leverages data from both, and uses the environment to provide supportability inputs into concept and design. References on reliability theory tended to focus on hardware. Policies mandating sustainment as a mandatory KPP and requiring assessment of RAM throughout the program will continue to increase focus and visibility on sustainment. (King 2007; Iyer 2007)

Achieving the Chief of Naval Operations' (CNO) vision of supportability in design requires the systems engineering methodology to incorporate supportability concepts and design throughout the acquisition process. Jones described Integrated Logistics Support (ILS) as a "disciplined and unified management of all activities necessary to produce a supportable system design and reasonable support capability to achieve a predetermined set of measurable objectives within an acceptable cost of ownership." (Jones 2006)

CNO strategic planning has emphasized supportability as a key objective, and academia has put in place processes to address it. Yet, as identified in the May 2008 *Report of the Defense Science Board Task Force on Developmental Test & Evaluation*, "early in the study, it became obvious that the high suitability failure rates were the result of systematic changes that had been made to the acquisition process." (DSB 2008) This report went on to identify the following findings:

- The high suitability failure rates were caused by the lack of a disciplined systems engineering process, including a robust reliability growth program during system development.
- RAM shortfalls are frequently identified during Developmental Testing (DT), but program constraints (schedule and funding) often preclude incorporating fixes and delaying Initial Operational Test and Evaluation (IOT&E).
- Sequential workforce cuts in the last ten years had a significant adverse impact on the Navy acquisition capability.
- Acquisition personnel reductions combined with loss of guidance documents and retirement of experienced senior industry and government personnel have exacerbated the adverse impact.

The same DSB Report also indicated a continuing issue with acquisition of systems that demonstrated unacceptable supportability, reporting that only 75 programs of 228 met reliability objectives. The majority of researched supportability literature, such as *OA in Naval Combat System Computing of the 21st Century* by Capt. Strei and *Naval Open Architecture – Overview on OA* by Capt. Shannon, addressed post-development actions, such as sustainment and planning for common hardware. (Strei 2003; Shannon 2006) A considerable amount of literature is available on managing COTS hardware with a focus on creating common hardware solutions to optimize support.

In a 30 January 2009 memo to U.S. Secretary of Defense (SECDEF) Robert Gates, Pentagon Acquisition Chief John Young offered a candid assessment of the causes of cost growth in over 40 major weapons programs, and offered the following statement:

> I think this data sample highlights that some defense acquisition program performance, and costs to the taxpayer, are driven by churn in requirements and budgets which are beyond a program manager's control in the current DoD process. (Young 2009)

When a program experiences budget or schedule challenges and constraints, a common result is the deferment of non-critical capabilities that are not required for meeting operational performance objectives, which often negatively impacts the program's suitability performance. This provides the basis for a scenario in which supportability and sustainment efforts may be deferred to later in the acquisition process. Recent reviews of programs, such as Aegis, DDG 1000, LPD17 and LCS, all indicate deferral of supportability requirements, such as software to perform maintenance or the acquisition of technical data. Furthermore, a lack of understanding of engineering to supportability artifact relationships brought about by the loss of experienced personnel in recent years and exacerbated by a lack of requirements traceability, results in a lack of ability to articulate impacts to supportability when engineering tradeoffs are being performed.

In summary, supportability research has indicated there is a need to integrate supportability and suitability within the systems engineering process throughout the developmental phases. The basic processes exist but are resulting in systems that are not suitable due to factors external to logistics development. Only by integrating logistics within the systems engineering process during requirements development, functional allocation and other key engineering activities, can the experienced logistician expect to characterize and articulate the logistics impact and execute risk management or aversion efforts to reduce the impact later on during sustainment.

- Benefits: A significant amount of information is available from resources, including Defense Acquisition University (DAU), training courses and literature, on developing logistics products and providing detailed information on logistics

processes during the different acquisition phases. Systems engineering literature and references describe the need to plan for suitability factors such as reliability, maintainability and human factors engineering. Furthermore, numerous papers are available on improving supportability through reduction of hardware variations and a movement to common hardware architecture for enterprise planning. Mandatory KPPs that include material readiness availability will require new acquisition programs to include and assess non-functional requirements, such as reliability, in their programs during early acquisition. Models for military development and analysis methods to determine the optimal product support plans are in place to support Military Standard (MIL-STD) development.

- Limitations: Reliability theory and the development of tools and analysis methods lag behind the development of approaches to meet future operational capabilities. The focus of sources that were researched tended to be from an operational context, which limits the ability of logisticians and reliability engineers to have a blueprint by which to invoke key sustainment requirements. No research material was found documenting practices that integrate supportability into systems engineering artifacts or support traceability from requirements to logistics products. Nor were any processes documented that describe how logisticians assess and characterize suitability issues when schedule, technical and cost constraints impact the engineering processes.

Very little literature was found on how to integrate logistics into the systems engineering process or the role of the logistician during early design and development. The majority of researched references focused on life cycle considerations, such as technical refresh planning or managing obsolescence. Additionally, research found that with the introduction of COTS, some programs, such as the submarine Acoustic-Rapid COTS Insertion (A-RCI), have significantly changed how business is done. (Stevens 2008) Strategies included information on how to apply Performance-Based Logistics (PBLs), minimizing churn, acquiring data rights and other sustainment issues. Very little information was found on how to integrate logistics during design, with the majority of

support strategies being based on the systems engineering "VEE" model. Some of the problems cited include the following: (1) logistics efforts that do not directly correlate to the proposed engineering processes, (2) engineering data and processes that are not reviewed by the logistician, and (3) significant changes made during the acquisition process that are not assessed for their logistics impact. Efforts are ongoing in other countries to establish and link logistics requirements planning, based on a common information tool, to engineering artifacts and products, with numerous successes documented; however, the cost and infrastructure to establish these types of processes remain a concern and is an issue for most programs. Similar to a common information source for SPLs, establishing a common method or process based on an automated information system continues to challenge programs.

The majority of supportability information was found when researching engineering topics and subjects such as SysML, which showed examples of how supportability requirements can be depicted from a design perspective. Bosch explained how supportability can be a design consideration or driver. The traditional logistician often lacks the knowledge, skill and ability to function well in an engineering environment, and may not understand how requirements can be tied to logistics products and how the relationship between SPL and logistics products is defined. Conversely, most engineers are unaccustomed to designing products with supportability considerations in mind. Significant issues remain on how supportability can be integrated, characterized and correlated during concept exploration and design. Often, suitability issues found during operational testing are not the result of logistics planning shortfalls or other logistics issues, but rather on funding shortfalls and faulty engineering. When estimating life cycle costs, programs rarely include worst-case scenarios, which usually results in the estimates being far lower than actual costs.

2.3.10 General Topics

General topics included research on tools utilized by the Capstone team, architecture reuse concepts, modeling languages, current approaches being applied to weapon systems development, and lessons learned from previous programs. In general, modeling languages were deemed critical elements by which the team could optimize the methodology and establish a vehicle that allows traceability within requirements and to artifacts generated outside the requirements process.

2.4 SUMMARY

Research indicated that no single solution is capable of solving all the issues identified in Chapter 1. There are numerous ways to achieve a system design that supports traceability from requirements to functionality within a system. Approaches can be either top-down or bottom-up and can be requirements-driven or model-driven. Knowledge gaps exist between traceability of requirements and engineering artifacts that may require new engineering processes to be employed to meet acquisition needs. Challenges remain to the supportability and engineering communities due to variances in approaches, languages, formats and acquisition processes used. A significant number of the issues impacting programs today are the result of fiscal and/or schedule constraints. Very little documentation is available on executing recommended concepts in such constrained environments. Most authors based their approaches on fully funded and adequately scheduled projects that allowed the optimal solution to be defined and executed. Table 2-2 provides a summary of the areas researched.

A successful methodology must support tradeoff decisions and accurate forecasting of requirements that can be applied to resources. A language must support the capability to decompose requirements in a traceable manner without ambiguity. The systems engineering process must allow communication of requirements across engineering, acquisition and supportability disciplines without misrepresentation, and allow assessment of existing or planned capabilities to optimize reuse and minimize acquisition

costs. The functional allocation process must allow a systems approach that optimizes design based on mission capability, and not from an existing weapon system based on MIL-STD processes. Artifacts must address how joint and interoperability requirements are being met without sacrificing performance. Supportability must be integrated throughout the acquisition process in a method that allows risk identification and awareness when constrained by cost and schedule.

Table 2-2: Summary of Areas Researched

Key Concept	Areas Researched
Open Architecture	Software, Hardware Asset Reuse Enterprise Repository; Naval Open Architecture
Service Oriented Architecture	Net-Centric Warfare; Universal Data Access; Metadata-Driven Architecture; Enterprise Data Bus
DoDAF	Representations for the DoDAF products such as tables, IDEF, UML and SysML; Repositories to hold products such as Core Architecture Data Model 2.0 and the DoD Architecture Repository System
Domain Analysis	Domain definition and concepts; Domain Specific Modeling Languages; Domain Modeling
Software Product Lines	Family-Oriented Abstraction; Specification, and Translation; Application Engineering, Domain Engineering; Feature-Oriented Domain Analysis; Generative Programming
Model Based Systems Engineering	Information Models, Entity-Relationship-Attribute Language, Software Tool Interoperability, Enhanced Functional Flow Block Diagram, CASE Tools, CORE, NetViz, Proforma, UML, SysML, Rhapsody, MOSA
System Architecture Approaches	Ship Defense Analysis Process; Real-Time Decision Support for AAW; Process Standards and Architecture; Federation Development & Execution Process; Threat Evaluation and Weapons Assignment; Data Fusion, Network Centric Warfare; Situational Awareness; Pipes and Filters; Implicit Invocation; Client-Server; Layered; Service-Oriented.
Modeling & Simulation	DoD Acquisition M&S Master Plan; Navy Modeling & Simulation Office; Navy Modeling and Simulation Information Service; M&S Governance Board; DoD Directive 5000.59 (4 Jan 94); SECNAVINST 5200.38A, 28 Feb 2002; OPNAVINST 5200.34 (28 May 2002)
Reliability Theory & Supportability	DOD Instruction 5000.02 Dec 2008; Reliability Engineering and Analysis (Systems Engineering and Analysis 4th Edition, Blanchard & Fabrycky, 2006) (Integrated Logistics Support Handbook 3rd Edition, James V Jones, 2006), Life Cycle Sustainment, Technical Data Rights.

THIS PAGE INTENTIONALLY LEFT BLANK

3 PROJECT METHODOLOGY

The primary objective of the project was to document a methodology for creating complex system architectures that employs a Model Based Systems Engineering (MBSE) approach. The MBSE approach facilitates the integration of concepts such as Open Architecture (OA), the definition of the role of Modeling and Simulation (M&S), the integration of supportability requirements, and the production of required artifacts. The key concepts described in the previous chapters are prioritized in this chapter in order to explain how and why the selected methodology was developed.

Research led to a methodology that captured the following outcomes:

- The tenets of MBSE and the Agile process promote a management approach that encourages frequent inspection and adaptation, teamwork, self-organization and accountability.
- Creation of domain centered Software Product Lines (SPLs) facilitate planned reuse and allow for the assessment of candidate architectures.
- SPLs incorporate supportability and requirements traceability early and throughout the above approach.
- Development of engineering and acquisition artifacts, including DoD Architecture Framework (DoDAF) views, facilitates the use of M&S to evaluate design prior to development.

These concepts form the basis for the proposed methodology, and describe how MBSE and DoDAF are integrated. Also described are the overall process and sub-processes used to execute the methodology. The process includes the major functions and products related to requirements development, functional allocation, architecture composition, and validation and verification consistent with developing an architecture that is optimal for software reuse.

The resulting methodology addresses the following:

- Identifies and establishes a common language.
- Utilizes domain analysis to support SPL reuse.
- Maintains traceability of requirements and architecture functionality.
- Integrates supportability, sustainment and life cycle cost considerations.

3.1 OVERALL APPROACH

3.1.1 Background

The International Council on Systems Engineering (INCOSE), in its October 2006 version of Systems Engineering Vision 2020, defines MBSE as:

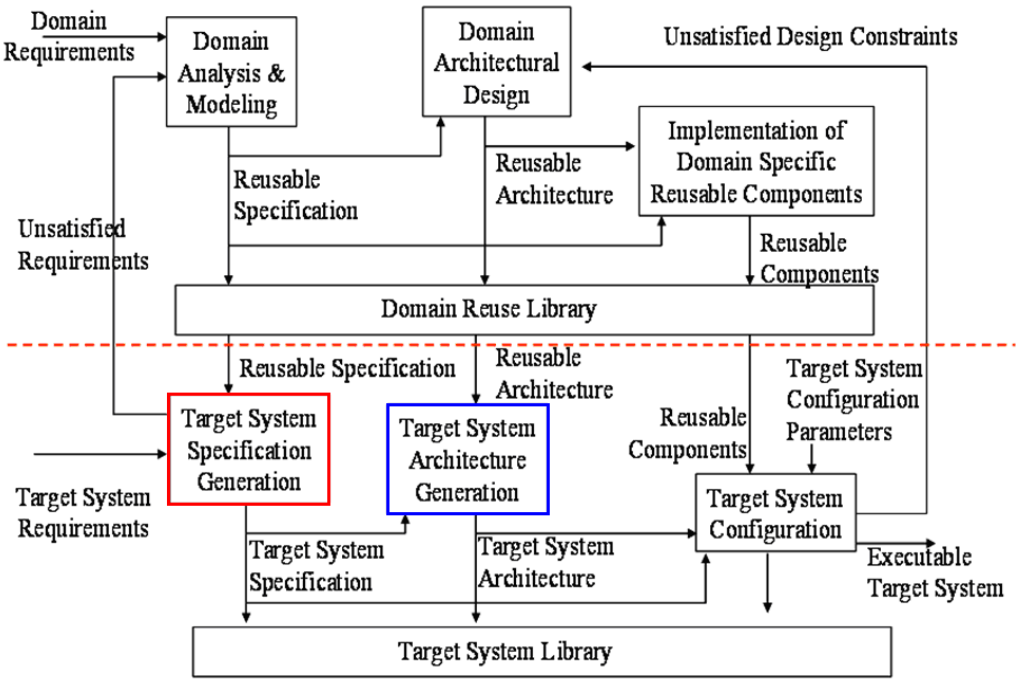
> the formalized application of modeling to support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases. (Crisp 2006)

A model is an approximation, representation or idealization of selected aspects of the structure, behavior, operation or other characteristics of a real-world process, concept or system. A model usually offers different views to serve different purposes. A view is a representation of a system from the perspective of related concerns or issues. The development of a model requires several iterations to arrive at a usable level of detail to meet systems acquisition requirements.

Agile systems engineering principles were applied to the MBSE approach to facilitate short, iterative cycles with rapid deployment and evolutionary capability. An Agile approach allows for repetition to provide greater definition to meet changing Navy requirements. Benefits include improved coordination, enhanced requirement definition, and the endorsement of stakeholders. The proposed methodology is intended to utilize an Agile approach, similar to what was pioneered by the Aegis shipbuilding program: build a little, test a little, learn a lot.

3.1.2 Domain Modeling & Analysis

The Navy currently architects systems using a domain view; however, the Navy's usage is inconsistent among stakeholders and generally does not have a mission focus as defined in the relevant literature. Subsystems are captured to some degree but there is no agreed method for developing domains from a SoS approach. The Navy has previously fielded complex weapon systems, such as Aegis and the Ship Self Defense System (SSDS). However, commonality between these systems continues to be a challenge because of the lack of a common domain view. Domain engineering would allow for the capture of artifacts and reuse of models and software. As today's combat systems become more software intensive, MBSE becomes more applicable to system development. Traditional systems engineering begins with requirements generation, as depicted in Figure 3-1. Capturing requirements using a domain engineering approach would enable the ability to understand what the Navy has today and what it wants to build tomorrow. A Navy library concept would contain all domain artifacts for a given problem set. The library would have to be engineered to combine requirements, architectures, M&S and components.



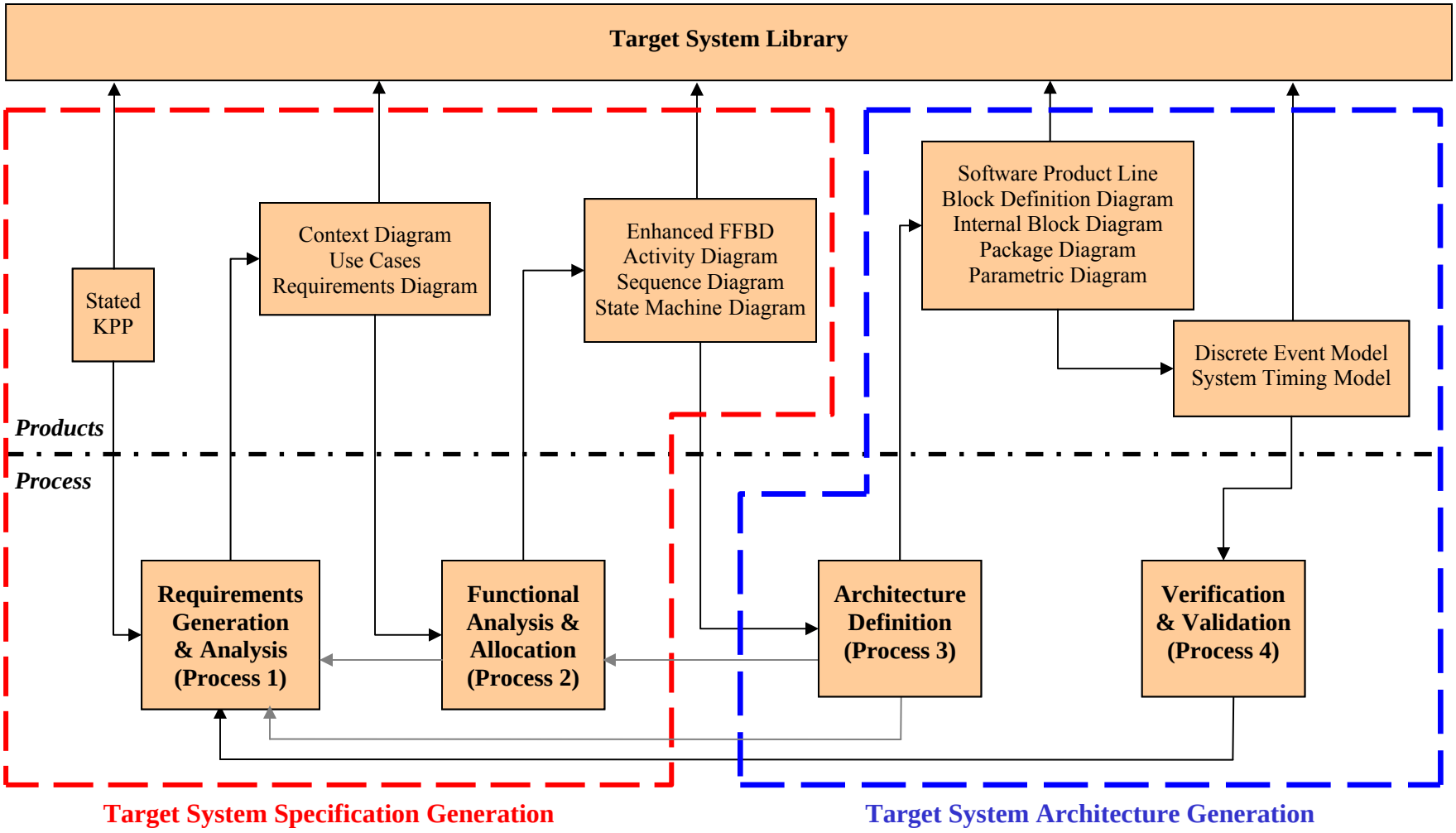
Illustrates the concept of predictive reuse through the use of a domain life cycle model.

Figure 3-1: Evolutionary Domain Life Cycle Model (Gomaa and Farrukh 1999)

3.1.3 Target System Architecture Development

Figure 3-2 illustrates the selected methodology, which can be used in an iterative manner to develop an executable system that is documented in a system library. From an SPL reuse perspective, the *Target System Library* represents an automated storage medium for the various products that describe the target system. The left side of the figure represents the system requirements generation, while the right side represents the system architecture generation. The lower half represents traditional systems engineering processes, while the upper half represents its products. The following is a description of the four main processes depicted in the figure:

- **Requirements Generation and Analysis (Process 1):** Process 1 uses a provided KPP (for this project, P_{RA}) from the stakeholders, or previous requirement(s) as an input. The outputs include Context Diagrams, Use Cases and Requirements Diagrams. These products are stored in the target library.
- **Functional Analysis and Allocation (Process 2):** Process 2 uses products from Process 1 as inputs. The outputs include Enhanced Functional Flow Block Diagrams (EFFBDs), Activity Diagrams, Sequence Diagrams and State Machine Diagrams. These products are stored in the target library. Feedback is provided from Process 2 to Process 1 to ensure the artifacts are technically representative of the requirements.
- **Architecture Definition (Process 3):** Process 3 uses products from Process 2 as inputs. The outputs include SPLs, Block Definition Diagrams, Internal Block Diagrams, Package Diagrams and Parametric Diagrams, which are used to develop Discrete Event and System Timing Models. Feedback is provided from Process 3 to Processes 1 and 2, and is assessed against the original inputs to verify technical traceability of requirements to functionality.
- **Verification and Validation (Process 4):** Process 4 uses products from the first 3 processes (Requirements, Functional and Architecture sub processes). It uses originating requirements to establish performance measures, functional models to develop the model, and architecture values and allocation to develop the expected values or variables.



Illustrates the selected methodology graphically, which can be used in an iterative manner to develop an executable system.

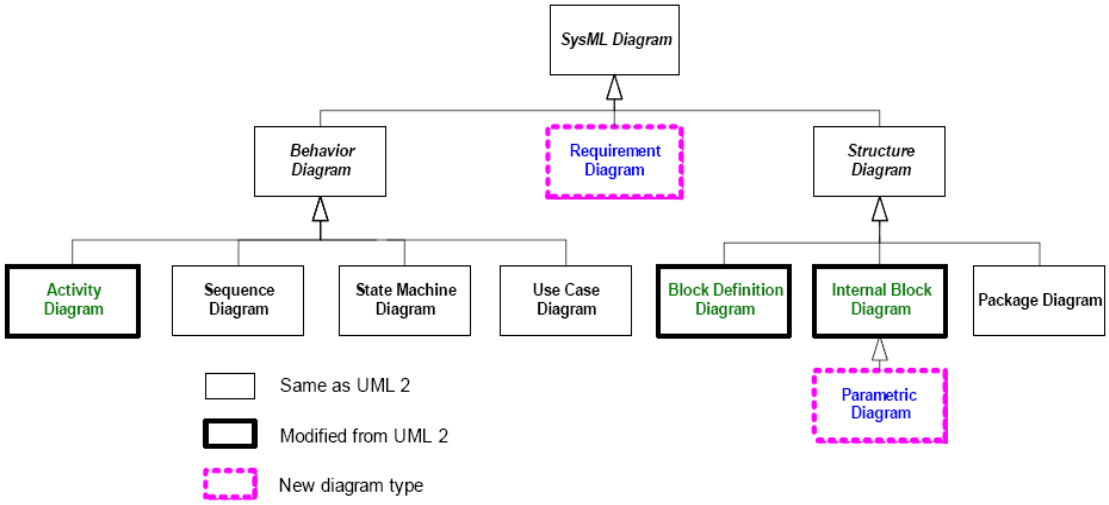
Figure 3-2: Top Level Methodology

3.1.4 Integration of Best Practices

The proposed methodology provides an MBSE approach, proven systems engineering practices, a common language for developing and documenting artifacts, and is capable of supporting changes in the acquisition process. To meet the requirements and objectives stated in Chapter 1 and based on the research discussed in Chapter 2, the proposed methodology provides an integrated approach for developing SPLs capable of being assessed and planned for reuse. It provides a basis for establishing a domain-based approach that can apply M&S as an iterative and recurring process to support business decision-making and technical tradeoffs. It does not rely on any one methodology, but incorporates best practices from multiple authors and sources.

3.1.5 SysML Application in Overall Methodology

The proposed methodology supports the MBSE approach by selecting Systems Modeling Language (SysML) as a standard modeling language to construct requirements and architecture artifacts while maintaining traceability. SysML is a general purpose graphical modeling language for specifying, analyzing, designing and verifying complex systems that can include hardware, software, information, personnel, procedures and facilities. Specifically, SysML provides graphical representations with a semantic foundation for modeling system requirements, behavior, structure and parametrics as shown in Figure 3-3. (OMG 20008) SysML is an enabler for MBSE, and can show relationships to represent various types of allocations, including functions to components, logic to physical components and software to hardware. (Friedenthal et al. 2008) SysML modeling provides the ability to assess the impact of changing requirements to a system's architecture. It is a precise language capable of supporting constraints and parametric analysis that allows models to be analyzed and simulated. Furthermore, SysML is an open standard and supports information interchange to other systems engineering tools. (Kunstar 2009)



Illustrates the taxonomy of SysML diagrams.

Figure 3-3: SysML Diagram

3.1.6 DoDAF in Overall Methodology

The DoDAF views were used to depict the proposed architecture, communicate it to stakeholders and comply with mandatory requirements for new acquisition programs. The operational views were used to validate primary activities required to execute mission warfare threads while the system views were used to show the allocation of functionality to the various systems required to execute the mission. Technical views would be used to document existing and plan future standards for use in acquisition and contract development. System Views would be developed to document future programs of record, assisting in test and evaluation efforts and in addition to providing artifacts for developing the models used in simulation to validate requirements.

3.1.7 Hatley-Pirbhai Method in Overall Methodology

The Hatley-Pirbhai (HP) methodology was used as the system engineering process to support development of operational views and allocate functionality to systems. It provides a methodology to develop the views based on an MBSE approach to observe the behaviors, functions, controls and data flows necessary to meet mission requirements.

3.1.8 Assessment of Architectures in Overall Methodology

A process for assessing quantitative and qualitative measures of software architectural styles was used based on Bosch's literature. (Bosch 2000) The result is a methodology to assess the architecture against non-functional requirements, such as maintainability and supportability. The following briefly describes how the four assessment methods were applied to the proposed methodology:

- **Scenario-Based Evaluation:** The ConOps was used to derive a scenario-based evaluation to assess the architecture suitability given frequency of hardware and software use. It is used as a basis for brainstorming, defining concepts and using the concepts to assess the proposed solution.
- **Simulation:** The performance model was simulated against requirements to determine if the intended architecture met the proposed Probability of Raid Annihilation (P_{RA}) given by the stakeholders. If the project timeline is expanded, simulation could be used to assess the reliability of the given architecture and system availability and other non-functional requirements, in addition to assessing primary mission threads.
- **Mathematical Modeling:** Mathematical modeling supports the physical assumptions and variables used in simulation. It provides a basis for the inputs, constraints and variables, and facilitates predictions of outcomes based on limited data within given confidence levels.
- **Experienced-Based Assessment:** Experienced-based assessments are the most subjective form of assessment methods. However, the value of experience cannot be understated. Experience-based assessment was used to compare architecture attributes, such as layered and event-based architectures, against the quality attribute, which did not have supporting data, such as latency, performance, interchangeability, etc.

3.1.9 Domain and Artifact Storage and Production in Overall Methodology

CORE was the tool selected to provide an additional method of verification and provides traceability between artifacts, and can be utilized to support software reuse assessments for future systems. Updating of CORE is a continual process as artifacts are developed. CORE captures artifact data and establishes relationships to improve traceability.

3.2 REQUIREMENTS GENERATION AND ANALYSIS

Process 1, depicted in Figure 3-4, is based on Martin's approach (Martin 1997) and provides a more detailed explanation of the Requirements Generation and Analysis process:

- **1.1 Collect Stakeholder Requirements:** Customers/stakeholders are identified and their requirements are collected, organized and categorized. These requirements are then ranked according to their relative importance to one another. Frequent contact and discussion with stakeholders is encouraged throughout the requirements generation and analysis process.
- The next three steps may be performed in parallel.
    - **1.2.1 Define Mission/System Objective:** Develop a complete definition of the system's mission and objectives based on stakeholders' requirements.
    - **1.2.2 Define System Scenarios:** Define the inputs and stimuli planned or expected and the response for each. Prioritize scenarios according to the likelihood of occurrence and severity of strain on the system. These scenarios are drivers for the system test philosophy and approach, and test cases and functional failure mode analysis will be based on these scenarios.
    - **1.2.3 Define System Boundary:** Determine the internal and external elements and subsystems required to achieve the system purpose. Define the system in terms of both space and time, and include physical and operational boundaries. Define the intended service life of the system, which is when will the system start performing its mission or objective and when will it be disposed.
- Upon completion of the previous steps, proceed to the next three steps, which may also be performed in parallel.
    - **1.3.1 Define Environmental and Design Constraints:** Identify and document the constraints that will limit or define the system's

performance or design. Design constraints should include such non-functional requirements as power, volume, weight, dimensions, etc. Environmental constraints should be defined for all primary functions and system scenarios.

- **1.3.2 Define Operations and Support Concept:** Identify and document the support concept that will influence design. The support concept development is part of the supportability planning process.
- **1.3.3 Define Measures of Effectiveness (MOEs):** Identify and document the KPPs required to meet operational requirements. Develop the relationships between the parameters that push design. MOEs are used to assess the utility of the system.

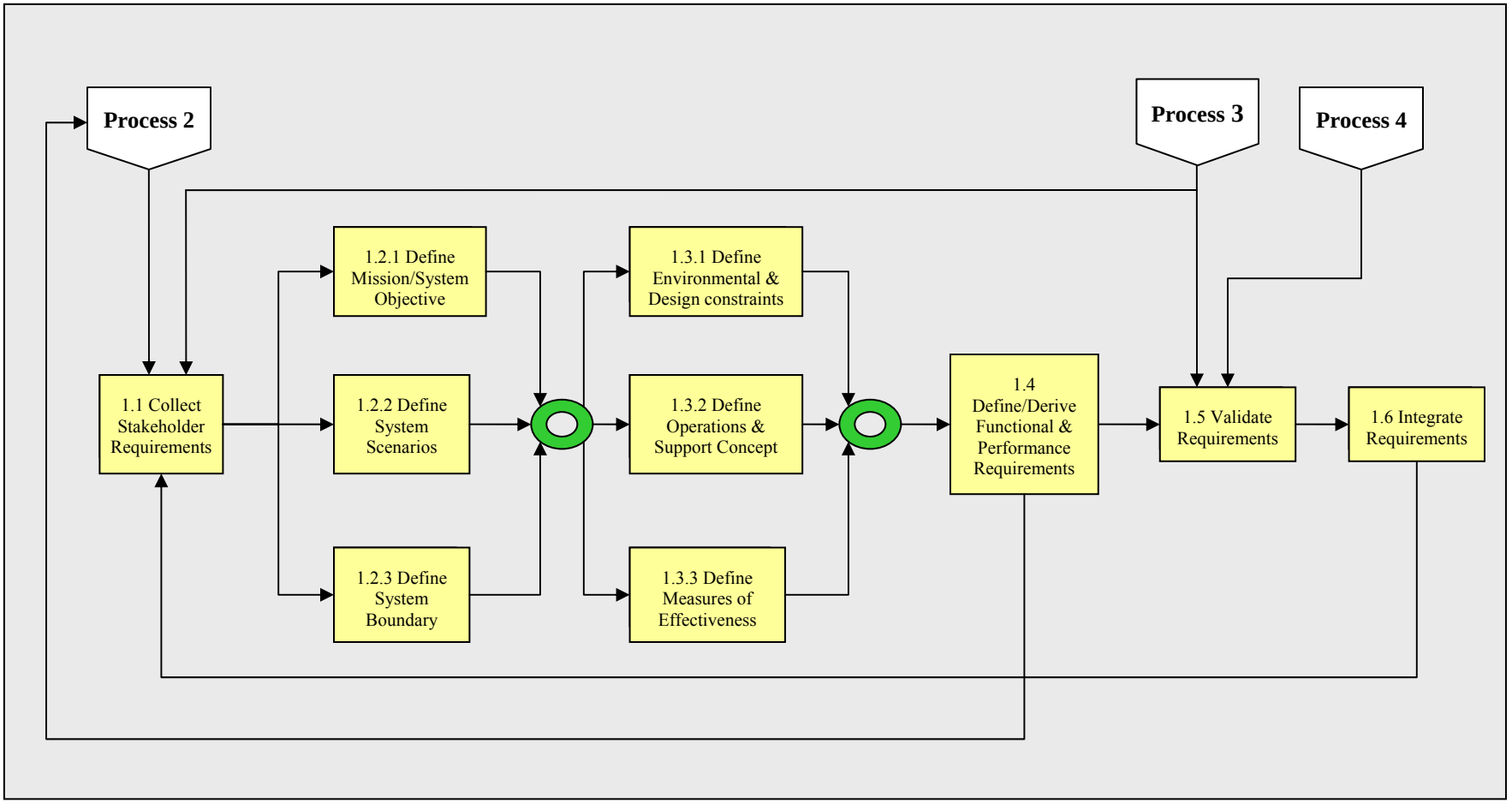
- **1.4 Define and Derive Requirements:** After a thorough understanding of the above scenarios, boundaries, requirements and objectives, define and derive the functional and performance requirements.
- **1.5 Validate Requirements:** Ensure technical requirements are complete and consistent with user and stakeholder requirements and specifications.
- **1.6 Integrate Requirements:** Confirm that requirements trace to the system performance and design. Repeat any prior tasks to refine the requirements as needed before proceeding to the Functional Analysis and Allocation activity.

3.2.1 SysML Application in Requirements Development

One of the principal extensions to SysML is to support requirements (see Figure 3-3). The requirement stereotype extends class to specify the "shall" statement and captures the requirement identification number. The requirements diagram is used to integrate the system models with text-based requirements that are typically captured in requirements management tools. The Unified Modeling Language (UML) containment relationship is used to decompose a requirement into its constituent requirements. A requirement is related to other key modeling artifacts via a set of stereotyped dependencies. The

"derive" requirement and "satisfy" dependencies describe the derivation of requirements from other requirements and the satisfaction of requirements by design, respectively. The "verify" dependency shows the link from a test case to the requirement or to the requirements it verifies. In addition, the UML "refine" dependency is used to indicate that a SysML model element is a refinement of a textual requirement, and a "copy" relationship is used to show reuse of a requirement within a different requirement hierarchy. The "rationale" concept can be used to annotate any model element to identify supporting rationale, including analysis and trade studies for a derived requirement, design or some other decision. Only the most basic attributes of a text-based requirement are included in the SysML specification. More specialized requirement types can be designated using specialization of the requirement stereotype. Typical examples are operational, functional, interface, control, performance, physical and storage requirements. (SysML Partners 2008)

These stereotypes may restrict the types of model elements that can satisfy or refine the requirement. For example, a performance requirement can only be satisfied by a set of constraints in a parametric diagram along with an associated tolerance and/or probability distribution, or a functional requirement might be satisfied by an activity or operation of a block. Requirements and artifacts represent those attributes necessary to define system architectures, including structure, behavior, requirements and parametrics. The requirements model can be shown in a graphical, tree structure or tabular format. The graphical format is called a requirements diagram. The activities are linked to the requirements they satisfy and the relationship is shown in an attached note. The requirements model is not meant to replace external requirements management tools, but rather to be used in conjunction with them to increase traceability within UML models. Its primary advantage is that it allows for the modeling of the requirements, the system and the traceability between them to be performed in a single model. The requirements diagram captures requirements hierarchies and requirements derivation. The "satisfy" and "verify" relationships allow a modeler to relate a requirement to a model element that satisfies or verifies the requirements. The requirement diagram provides a bridge between the typical requirements management tools and the system models.



Illustrates the requirements generation and analysis process from stakeholder needs through requirements validation and integration. (Martin 1997)

Figure 3-4: Requirements Generation and Analysis (Process 1)

3.3 FUNCTIONAL ANALYSIS AND ALLOCATION

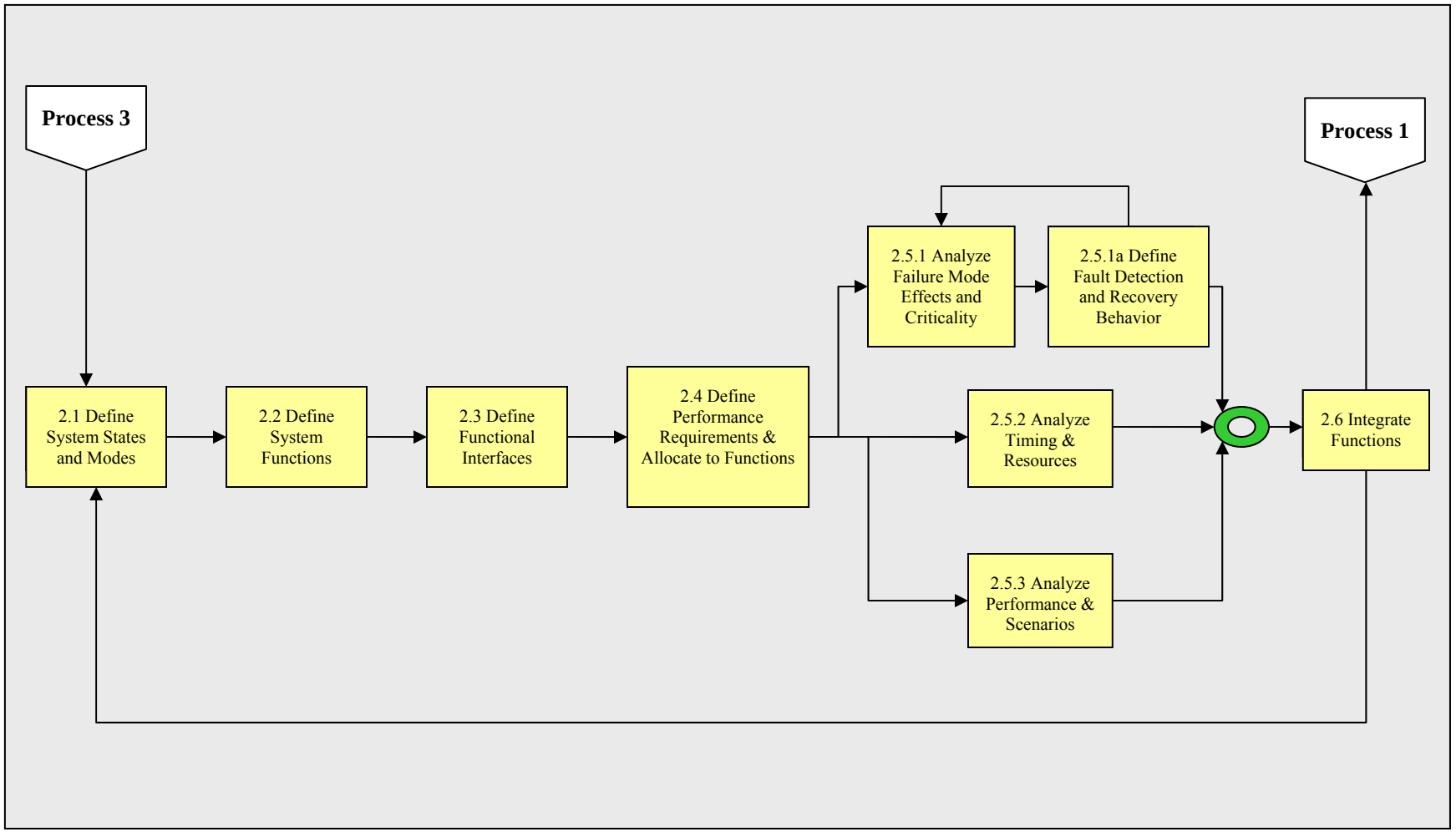
Process 2 can commence upon completion of Process 1. Process 2, depicted in Figure 3-5, is based on Martin's approach (Martin 1997) and provides a more detailed explanation of the Functional Analysis and Allocation process:

- **2.1 Define System States and Modes:** Define the states and modes that the system will experience through consideration and analysis of the system's expected environment and intended uses. The states and modes should be consistent with the mission profile and ConOps.
- **2.2 Define System Functions:** Derive the operational and support behavior of the system in terms of the functions the system must perform. This task includes some form of functional, control and/or data flow analysis. The functions and functional interfaces may be documented on Functional Flow Block Diagram (FFBD) function lists, Enhanced FFBD (EFFBD) or function trees. Describe functions using a verb to define the required task or activity.
- **2.3 Define Functional Interfaces:** Define the inputs/outputs and start/end states for each function. This ensures that all state transitions are completely defined within functions, and required inputs/outputs are provided. Identify function triggers and their associated interface items. Before performance is allocated, review system functionality for completeness and consistency. Special interfaces may be required to support interaction of the system with other equipment in the customer environment or organizational structure and for transportation and handling equipment.
- **2.4 Define Performance Requirements and Allocate to Functions:** Define and determine how well each function must be performed. Individual functional performance should be driven by the required system performance. Establish technical budgets where needed. Model the performance allocations as necessary to assist in the allocation process. Record the decisions and trade studies performed during this task to ensure traceability is maintained.

- After completion of the previous steps, proceed to the next three steps, which may be performed in parallel.
    - **2.5.1 Analyze Failure Mode Effects and Criticality:** Analyze the functional consequences of any specific failure. This should be performed in conjunction with the Failure Mode Effects and Criticality Analysis (FMECA) performed by reliability and maintainability engineers. Document results of this task in fault trees or FMECA tables.
        - **2.5.1a Define Fault Detection and Recovery Behavior:** Provide changes to the functional definition in response to irregular conditions. Recovery from faults due to operational failures may lead to the need for maintenance functions. Analyze the new behavior again for failure mode effects and criticality.
    - **2.5.2 Analyze Timing and Resources:** Analyze the system behavior for compliance with timing requirements and internal operations using the functional description and system constraints.
    - **2.5.3 Analyze Performance and Scenarios:** Through the use of the functional description, analyze the modeled behavior for static and dynamic consistency and ability to execute. Use M&S tools as much as possible to help assess expected performance of individual system elements and the system as a whole. Analyze system behavior for each system scenario and for each system stimulus. Define execution *threads* within the functional architecture, which can be used during verification to establish that the proper functions are being performed.
- **2.6 Integrate Functions:** Ensure that system and subsystem interfaces are adequately defined. Ensure all functions collectively meet system requirements and provide optimal system performance according to the defined MOEs. Repeat the above tasks for lower level functions.

3.3.1 SysML Application in Artifacts Development

The SysML behavioral diagrams include the activity diagram, sequence diagram, state machine diagram and use case diagram. State machines are used to specify state-based behavior in terms of system states and activities, which have been significantly extended from UML 2.0 activities, and represent the basic unit of behavior that is used in activity, sequence and state machine diagrams. The activity diagram is used to describe the flow of control, inputs and outputs among actions.



Illustrates the functional analysis and allocation process from the definition of system states through function integration. (Martin 1997)

Figure 3-5: Functional Analysis and Allocation (Process 2)

3.4 ARCHITECTURE DEFINITION

Process 3, depicted in Figure 3-6, is based on Martin's approach (Martin 1997) and provides a more detailed explanation of the Architecture Definition process:

- The first two steps may be performed in parallel.
  - **3.1.1 Assess Technology Alternatives:** Evaluate technologies that can be applied to solve the problem. Identify possible system concepts and options. Examine technology trends to determine appropriate level of technology at time of deployment.
  - **3.1.2 Synthesize System Element Alternatives:** Define and refine system element alternatives for each relevant set of functional requirements using a bottom-up approach.
- **3.2 Allocate Functions to System Elements:** Identify which functions will be performed by which system elements or subsystems. Apportion the associated performance of each function to the appropriate system element.
- **3.3 Allocate Constraints to System Elements:** Identify constraints that pertain directly to system elements and that do not apply to behavioral functions. These constraints should include non-functional requirements.
- **3.4 Define Physical Interfaces:** Define electrical, data, mechanical and other interfaces between elements of the system. Identify all interfaces between the system and the external world, including the supportability domain. Document these interfaces in interface control documents.
- After completion of the previous steps, proceed to the next three steps, which may be performed in parallel.
  - **3.5.1 Define Platform and Architecture:** Delineate the platform(s) upon which the system/product will be installed. Define the architectures in terms of product structures and interactions between the products and with elements in the environment. Map scenarios to various configurations of the system. The hierarchical relationship of the system elements should be documented in an Architecture Block Diagram (ABD) or another suitable artifact.

- **3.5.2 Refine Work Breakdown Structure (WBS):** Translate the selected architecture and its decomposition into a WBS format for work planning and cost/schedule tracking and control.
- **3.5.3 Develop Lifecycle Techniques and Procedures:** Develop appropriate models and parameters to support life cycle cost analysis. Define how the system will be manufactured, verified, deployed, operated, maintained and disposed of. Define training and other supportability products and procedures. Identify all required enabling products and their key characteristics.

- **3.6 Check Requirements Compliance:** Ensure all functional and performance requirements have been mapped to the system elements and subsystems. Ensure that the system elements at each level in the architecture satisfy all requirements and constraints. Compliance may be checked using System Effectiveness Analyses, simulations, demonstrations, inspections and/or tests. Models and prototypes may also be used.
- **3.7 Integrate System Elements:** Progressively integrate the system elements into items that provide an end-use function (bottom-up). At each level, the resulting design requirements, physical configuration and physical interfaces should be verified to ensure that functional and performance requirements are satisfied.

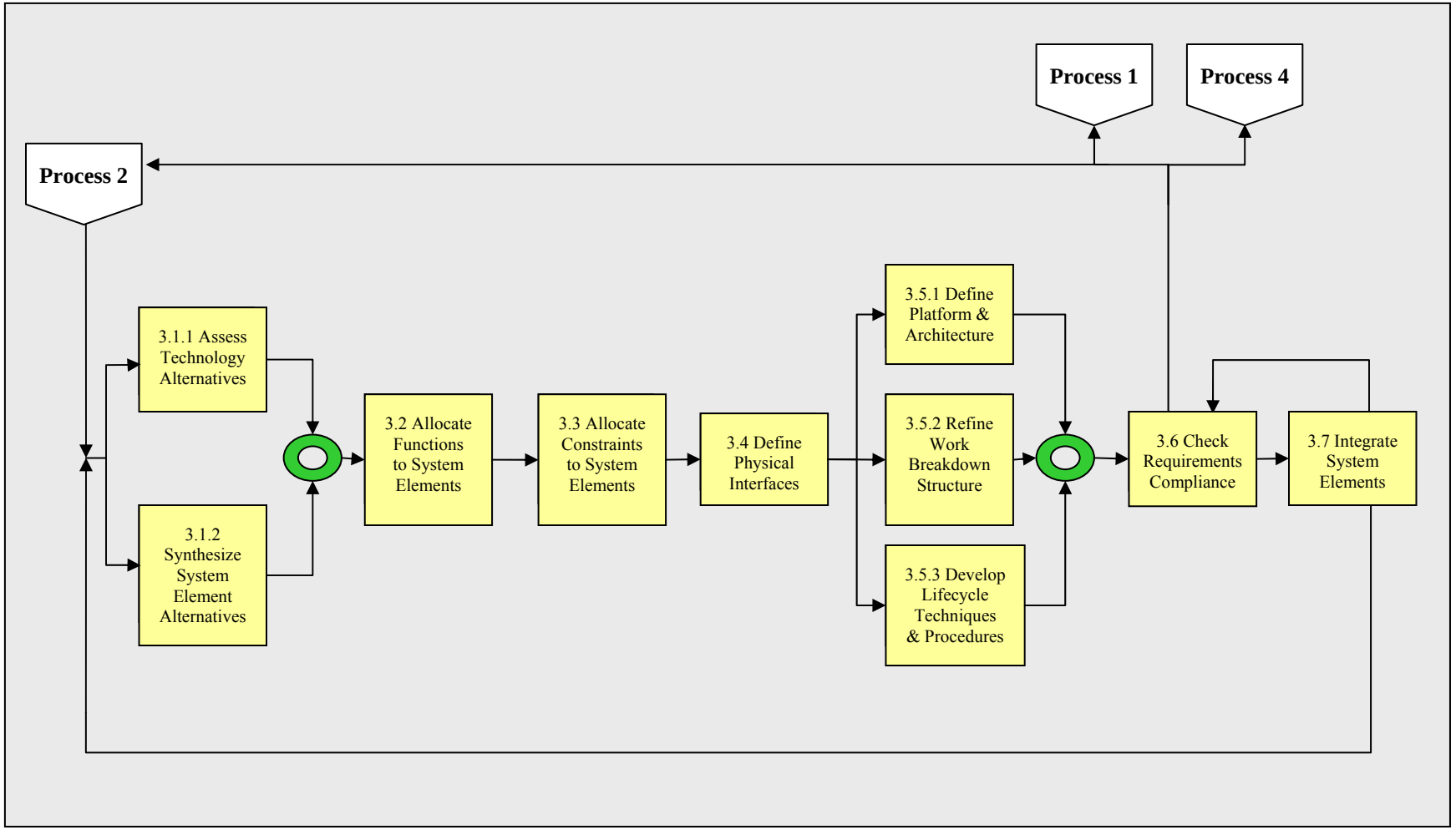
3.4.1 SysML Application in Architecture Development

The major structural extension in SysML is the block, which extends the UML-structured class. It is a general-purpose hierarchical structuring mechanism that abstracts away much of the software-specific detail implicit in UML-structured classes. Blocks can represent any level of the system hierarchy, including the top-level system, a subsystem, or logical or physical component of a system or environment. A SysML block describes a system as a collection of parts and connections between them that enable communication and other forms of interaction. Ports provide access to the internal structure of a block for use when the object is used within the context of a larger

structure. SysML provides standard ports that support client-server communication (e.g., required and provided interfaces) and FlowPorts that define flows in or out of a block. Two diagrams are used to describe block relationships. The Block Definition Diagram (BDD), similar to a traditional class diagram, is used to describe relationships that exist between blocks. The Internal Block Diagram (IBD) describes the internal structure of a system in terms of its parts, ports and connectors. (OMG 2009)

3.4.2 Assessment of Software Architectures

Bosch identified four methods for assessing the architecture using quality attributes during design. This project's approach utilized all four methods based on the maturity of design. The four basic methods included scenario-based evaluation, simulation, mathematical modeling and experience-based assessment. Scenario-based evaluation utilizes the ConOps sustainment scenarios to depict the operational and sustainment concepts. This provides data points and assumptions, such as the frequency of technical refreshes for both hardware and software, which are significant drivers in estimating life cycle costs. The proposed or alternative system architectures are also evaluated using simulation. Simulation is applied to operational considerations, such as response and Human Systems Integration (HSI) considerations. Values are derived from the model to establish baseline assumptions that are input into the appropriate models, which are used in simulations. This approach also supports varying development timelines and independent development of systems and subsystems. As systems are developed, the actual values can be used to update the model supporting the simulation to determine the confidence of meeting mission requirements. Mathematical modeling is initially captured in the SysML as parametric data and provides the basis for operational factors, such as detection ranges, processing performance and other factors that can be used to compare alternative approaches. Although it is the least quantitative and explicit, experience-based assessments may provide the greatest insight based on experience vice theory. An example that illustrates this is the learning curve theory. It has been proven that experience results in increased efficiency, and this can be applied to developing architecture. (Bosch 2000)



Illustrates the architecture definition process in which functions from Process 2 are allocated to system elements. (Martin 1997)

Figure 3-6: Architecture Definition (Process 3)

3.5 VERIFICATION AND VALIDATION

Process 4 is depicted in Figure 3-7 and provides a more detailed explanation of the Verification and Validation process:

- **4.1 Define the M&S Objectives:** Primary concern is with establishing the specific objectives for M&S within the context and scope of the system architecture. This step requires a detailed review of the ConOps, system requirements, functional architecture, behavioral analysis, supportability functions and/or other applicable documents and artifacts to assist in defining the objectives.
- **4.2 Characterize the Model:** Define the critical measures of the system, establish the MOEs and identify the inputs for the models. The MOEs are used to evaluate the primary objectives for M&S with regard to critical measures. Inputs include assumptions, constraints, variables, parameters and analysis data.
- **4.3 Identify Tools:** Research and define the tools needed to perform M&S. It is important to ensure the appropriate tool is utilized. For example, if the model is simple, computer spreadsheets will suffice for minor to moderate calculations. If the model is more intricate, a higher-level modeling tool, such as Extend or Arena, is recommended.
- **4.4 Build a Parametric Diagram:** After characterizing the model and identifying the necessary tools, the next step is to review the critical measures, input variables and required calculations for the system to facilitate the development of a parametric diagram. The parametric diagram is used to structure the characteristics of the model into a high-level mathematical diagram that will aid in depicting the flow of calculations for the simulation.
- **4.5 Build a High Level Model:** When designing the high level model, it is important to start from the known requirements and the functional architecture.
- **4.6 Construct Main Functions:** After generating the high level model, the process begins for constructing the main functions of the model. It is helpful to take one main function of the model and separate it from its interfacing counterparts to focus on the particular logical aspects of the function.

- **4.7 Incorporate Supportability:** Modeling is the creation of abstractions or representations of the system to predict and analyze performance, cost, schedules and risks, and to provide guidelines for systems research, development, design, manufacture and management. (Maier and Rechtin 2002) It is imperative that the model represents the supportability factors and incorporates the elements into the model from the beginning. Supportability factors are critical in predicting availability, reliability, readiness and life cycle costs associated with the system being modeled, and should be integrated throughout the requirements hierarchies and functional architecture design. It is important for the engineering and supportability team to examine the main functions of the model and employ changes at a high level before starting the detailed system modeling.
- **4.8 Build, Execute and Analyze the Model:** The last step in the methodology may either be the least or most difficult to implement, depending on how well the upfront planning and/or design for the model has proceeded. The key is to utilize all the previously identified techniques, processes and tools to help construct the model.

3.5.1 SysML in Support of M&S

A key advantage in the use of SysML is the ability to represent data required to develop models and simulations. It is used to express constraints (equations) between value properties, provide support for engineering analysis (e.g., performance, reliability), and facilitate identification of critical performance properties. The following describes the use of SysML artifacts in relation to M&S:

- **Requirements:** Requirements depiction, covered in section 5.2, provides the basis of M&S focus. The results of the M&S should use the requirements to develop the Measures of Performance (MOPs) and MOEs developed for the M&S performance measures. The requirements will be used to determine M&S requirements and will also be used to understand the decomposition of the model. For example, when constructing an M&S for P_{RA}, the M&S will follow the requirements flow down to the next requirements level hierarchy such as

Probability of Kill (P_K), Command and Control (C2), and Engage. This creates a top down approach for modeling that allows for further iterations and models developed at a lower level to be traced to the top level requirements. Additionally, it allows improved understanding of the ability to tradeoff between sub functions.

- **Behavior:** Behavior diagrams include sequence, state machine and use cases which serve as the basis for the model structure. The activity diagrams developed by the architect and team are used to model the system behavior modules and tasks. Additionally, it allows the sequence of events to be modeled and traced back to expected operation and performance.
- **Structure:** Structure diagrams provide the basis for the performance variation used internal to the system performance such as processing time. Based on the allocation of functionality within the architecture, the processing time and functionality will be impacted. The architecture provides a basis for expectations, and when used in conjunction with an architecture tool such as CORE, allows a baseline variable to be input. The baseline variable is traceable directly to the proposed architecture.
- **Parametric:** The parametric diagram represents constraints on system property values such as performance, reliability, and mass properties. This serves as a means to integrate the specification and design models with engineering analysis models. SysML also defines a model of value types that can have units and dimensions and probability distributions. The value types are used to type properties of blocks. The parametric diagram is a specialized variant of an internal block diagram that restricts diagram elements to represent constraint blocks, their parameters and the block properties that they bind A constraint block supports M&S by providing and defining a set of parameters and one or more constraints on the parameters. By default, these parameters are non-directional and have no notion of causality. These constraint blocks are used in a parametric diagram to constrain system properties. Constraint blocks may be used to express mathematical equations, statistical values and utility functions that could be used in trade studies to facilitate identification of critical performance properties.

> Parametric diagram represents the usage of the constraints in an analysis context by the binding of constraint parameters to value properties of blocks.

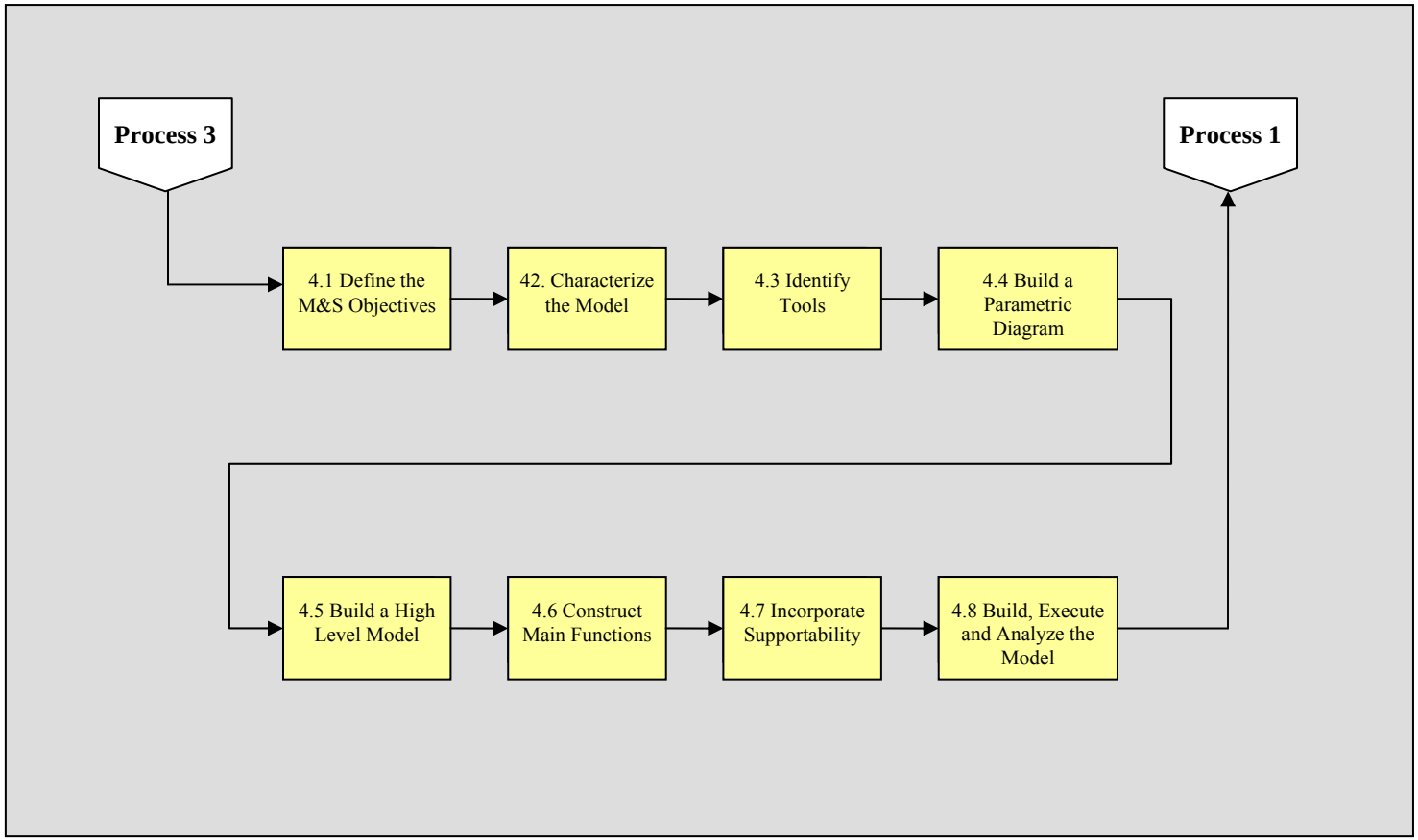
3.5.2 M&S in Support of T&E

DoDI 5000.02 requires that T&E programs be structured to provide accurate, timely and essential information to decision makers for programs in all acquisition categories throughout the system life cycle. The program manager should develop a robust, integrated T&E strategy for Developmental Test and Evaluation (DT&E), Operational Test and Evaluation (OT&E) and Live Fire Test and Evaluation (LFT&E). This is done in order to validate system performance, and to ensure that the product provides measurable improvement to existing operational capabilities. However, this integrated approach should not compromise DT&E, OT&E or LFT&E objectives. The program manager, in concert with the user and test communities, is required to integrate M&S activities with government and contractor DT&E, OT&E, LFT&E, SoS interoperability and performance testing into an efficient continuum. (DAG 2004)

It should also be noted that by law (10 US Code 2399), the initial OT&E of a major defense program's system effectiveness and suitability must not be based exclusively on computer modeling, simulation or analysis of system requirements and design specifications. While modeling can never entirely replace T&E in a live environment, it does play an important role in T&E. DoDI 5000.02 states the following:

> Successful developmental test and evaluation (DT&E) to assess technical progress against critical technical parameters, early operational assessments, and, where proven capabilities exist, the use of modeling and simulation to demonstrate system/system-of-systems integration are critical during this effort. (DoD 2008)

The use of M&S has been touted in recent years primarily because of the ease of manipulation of the parameters of the system under test and the lower cost associated with M&S when compared to live testing.



Illustrates the process in which the system architecture is verified and validated through modeling and simulation.

Figure 3-7: Verification and Validation (Process 4)

3.6 TOOL USAGE AND DODAF PRODUCTION

3.6.1 Methodology Overview

An automated tool was selected for use that was capable of supporting requirements, functional analysis, architecture, M&S and T&E functions, as well as to improve response time to changes, minimizes manual efforts to update and sustain engineering artifacts and improve verification and validation efforts through traceability of data. Key characteristics for selection of a tool include aspects such as compatibility with programming language, artifact production, technical usability, and portability to receive and output from other tool sets. As mentioned previously, CORE was the tool provided for this project.

3.6.2 CORE

The process for conducting analysis as a total integrated approach was based on evaluation and traceability of artifacts and their respective data within CORE. Each functional area developed artifacts, which were inputted into CORE based on the overall approach. Analysis was conducted to determine if the artifacts satisfied data input requirements. For example, an initial set of requirements was developed using SysML. During review and analysis, it was identified that a scenario was required to develop the EFFBD artifact. The requirements generation process was then reviewed and modified allow development of data using existing or new artifacts, based on the specific data required. During the initial development of the EFFBD, it was identified that a ConOps document would be required resulting in an additional first pass deliverable to develop the OV-1, which was the end state artifact identified for Level 0.

The originating set of documents used to develop the planned use of CORE was based on CORE system user guidelines and experimentation. The major CORE steps followed the top-level process described in section 3.1.

3.6.3 DoDAF – Use of Common Artifacts to Represent Architecture

The DoDAF was established as a guide for the development of architectures and as a way of improving communications among all stakeholders during the design of complex systems.

> The DoDAF provides the guidance and rules for developing, representing, and understanding architectures based on a common denominator across DoD, Joint, and multinational boundaries. It provides insight for external stakeholders into how the DoD develops architectures. The DoDAF is intended to ensure that architecture descriptions can be compared and related across programs, mission areas, and, ultimately, the enterprise, thus, establishing the foundation for analyses that supports decision-making processes throughout the DoD. (DoD 2007)

Dam describes the use of DoDAF as a methodology to develop and define architecture requirements from an integrated aspect. A select set of DoDAF views is required for new acquisition programs, which include All Views (AV), Operational Views (OV), System Views (SV) and Technical Views (TV). Although this approach has been used to create Navy Network Warfare Command (NETWAR) and C4I centric systems, it can also be used to provide a basis for combat systems. The primary consideration for selecting a model approach was a standard set of models and an understanding of data relationships, which allow traceability from a top-down or bottom-up approach. The DoDAF views are developed in an iterative manner from a mission need to a detailed component need. The views are capable of being supported when using SysML and provide a graphic depiction. The views are also capable of supporting an iterative or agile approach that allows decomposition to the level needed.

The process for development was based on review of existing models from external Programs of Record, papers written by Systems Engineering Institute (SEI) and literature by S. Dam, PhD.

3.7 SUMMARY

In this chapter, we have described the planned MBSE methodology to be used in developing complex system architectures. The methodology uses an agile approach to achieve high iteration/low duration development sequences, and integrates M&S to optimize design throughout development. It further integrates and provides a method for integrating supportability requirements into the system architecture. Finally, it uses a tool set to improve the validation and verification effort throughout the engineering developmental stage, which reduces manual efforts related to generation of engineering and programmatic artifacts. The method is based on using SysML, which will provide traceability in a top-down approach using MBSE principles regardless of tool selection.

In the following chapter, the methodology will be demonstrated using a P_{RA} requirement for developing an AAW combat system. It will include artifacts developed to meet engineering and architecture development processes. It will provide a baseline of expected versus actual results, and capture lessons learned which occurred during the learning phase.

4 METHODOLOGY APPLICATION AND VALIDATION

This chapter summarizes efforts executed in support of the proposed methodology described in Chapter 3. It documents and examines the results of applying the process to develop an Anti-Air Warfare (AAW) architecture to meet stakeholder requirements. The following sections provide detailed findings associated with the major process steps and products.

Current Navy acquisition efforts use a variety of low and high fidelity models to design and modernize systems. The collaboration of users, owners, design engineers/logisticians, marketing organizations and acquisition offices produce system requirements. Requirements data is normally provided in the form of text files, graphics, tables, etc. Systems engineering uses the requirements data to generate a set of specifications, which need to be captured in different views that the community of interest can use with minimal misinterpretation.

Naval AAW systems have three primary functions: detect, control and engage (also referred to as Detect to Engage (DTE)). The AAW mission is complex and multi-faceted. AAW mission requirements were scoped down to address Surveillance, Self Defense (SD) and Limited Area Defense (LAD). A first order engineering analysis was performed to define the mission scope. Using the requirements and constraints, a functional architecture was developed using SysML, and a CORE model was developed to perform a proof of concept of the methodology.

Application of the methodology was affected by both the schedule and learning curves for the computer tools that were used. Due to schedule constraints, there were parallel efforts to develop SysML products, a proposed architecture, and DoDAF artifacts in CORE. This led to inconsistencies in the artifacts produced. Despite these

inconsistencies, the nodes of the system are consistent in that DTE is completed through Search, Detect, Command and Control (C2) and Engage functions.

4.1 EXECUTION OF OVERALL APPROACH

The following sections detail the approach used to develop a top level AAW architecture that meets Surveillance, SD and LAD mission requirements. An Agile approach encouraged frequent inspection and adaptation, as well as teamwork, self-organization and accountability. The methodology produced clear requirements products for the stakeholders.

4.1.1 Chapter Overview

Section 4.2 reviews the requirements developed from the Concept of Operations (ConOps) (Appendix G). The ConOps described the relationships and boundaries of the system and documented scenarios used to ensure stakeholder expectations were met. Requirements were developed using the Systems Modeling Language (SysML). SysML provides a common notation that facilitates the connection and parsing of requirements. References were consulted (Buede 2000; Berk et al. 1989; Wood 2001) to support the development of a logical-based AAW architecture, further discussed in section 4.3. The term *logical* denotes that the architecture was decomposed to a level sufficient to illustrate that a design was achievable. A complete physical architecture was not developed; a functional architecture was implemented in CORE to show that a system could be developed using a Model Based Systems Engineering (MBSE) approach. In section 4.4, use of a layered software architecture was explored to support Software Product Lines (SPLs) that could be reused. The Hatley-Pirbhai (HP) development process, described in section 4.4.3, was also used to uncover behaviors, data flow, control flow and state relationships from which a specification could be developed. This process helped in allocating and processing the requirements of the system into software modules.

Section 4.5 illustrates the use of Modeling and Simulation (M&S) in acquisition engineering. An AAW model was developed using the Extend simulation program to verify the architecture meets AAW requirements and validate the methodology. Section 4.6 describes how CORE was used to produce DoDAF artifacts and captures relationships that can reduce the time required to transfer artifacts between collaborating design teams. CORE was also used to generate systems engineering review artifacts, such as programmatic and Test and Evaluation (T&E) documents. Section 4.7 provides a chapter summary that shows MBSE is the best approach for systems architecture development.

4.2 REQUIREMENTS GENERATION AND ANALYSIS

4.2.1 Requirements Introduction

Requirements generation has a major impact on both the life cycle and the success of a program. It is important that requirements show the "what" and the "how well" the system will perform. According to Buede, "requirements do not provide solutions but rather define the problem to be solved." (Buede 2000) Research within the government and industry has shown that laying a solid foundation of requirements for a program ensures the system produced performs its mission, is supportable, maintainable and reliable, and satisfies cost and schedule constraints.

Just as there is a hierarchy associated with the physical components of a system, there is also a hierarchy of requirements. At the top of the hierarchy are mission requirements, which relate the mission and activity needs that are important to stakeholders. Mission requirements typically involve the interaction of several systems and people, and are often described as System of Systems (SoS). (Buede 2000)

Requirements were developed for an AAW mission thread. The three mission requirements within the AAW mission were Surveillance, SD and LAD. Supportability

requirements were derived to support the AAW mission thread. Stakeholders provided the following high level mission requirements from which constraints and performance parameters for the system were derived:

- **Surveillance:** System shall detect, track and identify all air objects within a specified surveillance volume.
- **SD:** System shall defend ownship against a stream-raid attack consisting of 8 Anti-Ship Missile (ASM) threats at a Probability of Raid Annihilation (P_{RA}) of 0.99.
- **LAD:** System shall defend a High Value Unit (HVU) attack consisting of 6 ASM threats at a P_{RA} of 0.99.
- **Supportability:** The Operational Availability (A_O) of the system was identified as 0.90 to support the operational P_{RA}. This indicates that system reliability and availability will have the appropriate Mean Time between Failure (MTBF), Mean Time to Repair (MTTR) and Mean Logistics Delay Time (MLDT). Supportability requirements, such as training, manning and Human Systems Integration (HSI), were derived from the high level A_O.

Research was conducted on how to follow an MBSE approach to develop traceable requirements. The methodology demonstrated traceability of requirements throughout sub-processes 1 through 4. Requirements traceability was achieved through the use of modeling languages and tools, such as SysML and CORE. System artifacts were generated using the methodology outlined in Chapter 3. The artifacts produced in requirements generation eventually supported follow-on artifact development in the system design and verification phases of the project. The following sections will detail the scope of the requirements generation process, the resources and tools that were used, and resulting artifacts that were produced.

4.2.2 Requirements Incorporation into the Sub Process

Requirements were refined using the AAW ConOps (Appendix G). The ConOps was used to bridge the gap between the stakeholders, the requirements generation process and the resulting system architecture. It describes the environment in which the system was intended to operate. Additionally, supportability requirements were defined early in the developmental process to ensure their incorporation in the system design. Supportability requirements, such as A_O and HSI, were incorporated in the requirements models. Measures of Effectiveness (MOE) were defined for the system.

The following is an example of the process used to develop and analyze requirements. Using the SD requirement as a model, the question to be answered was what would be a suitable LAD against what threat raid density? The following mathematical approach was used based upon an outer area battle analysis. An Excel based approach was used for analysis purposes. According to Wagner,

> the number of shots that the SAM (Surface-to-Air Missile) ship can fire during the time the target is within range depends on maximum SAM range r_{max}, target speed u, SAM speed v, and the range to CPA (Closest Point of Approach), r_{CPA}. Assume first that the threat aircraft is detected early enough so that the first intercept can be made at the edge of the SAM envelope. The first target angle, theta, is the angle between the line of flight of the threat aircraft and the line from the point A where the aircraft enters the SAM envelope and the point Z where the SAM ship is located. The first lead angle, alpha, is the angle with vertex at the SAM ship and rays determined by the point A and the point B where the second intercept is made. (Wagner et al. 1999)

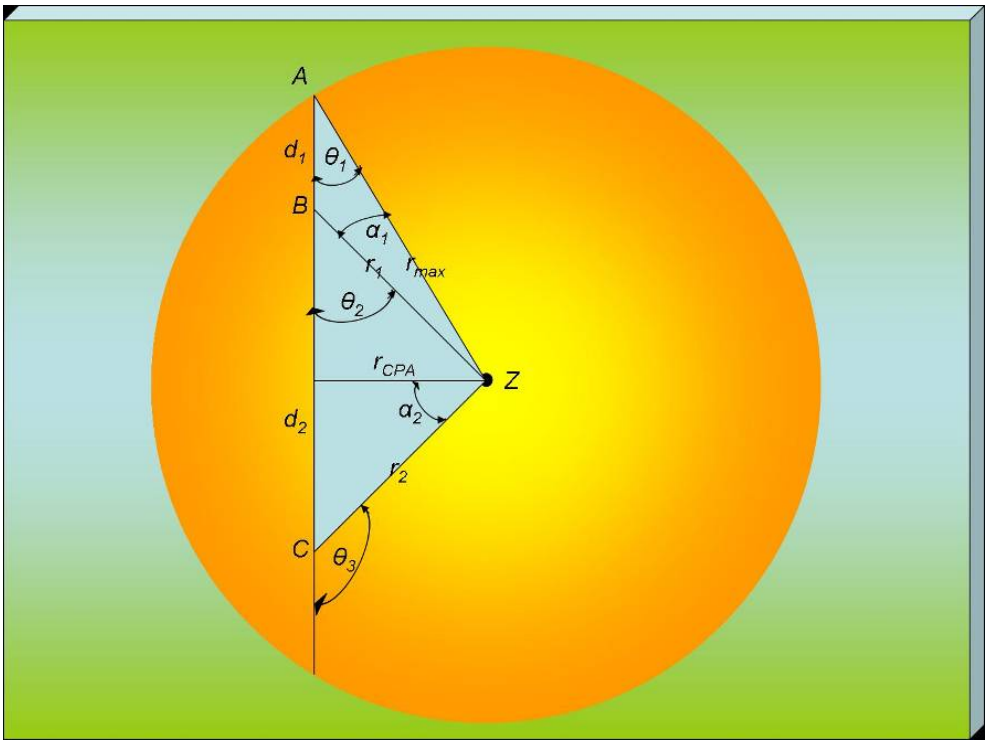
The following equations were used to determine the values shown in Table 4-1 for the LAD requirement (Figure 4-1 illustrates the corresponding geometry):

$$\theta = \arcsin(r_{CPA} / r_{\max})$$

$$\alpha = \arcsin(u \sin(\theta) / v)$$

$$r = \frac{r_{\max} \sin(\theta)}{\sin(\alpha + \theta)}$$

$$d = \frac{r_{\max} \sin(\alpha)}{\sin(\alpha + \theta)}$$



Describes the target track geometry associated with the weapons analysis equations.

Figure 4-1: Target Track Geometry (Wagner et al. 1999)

Table 4-1 and Figure 4-2 together show the analysis for Weapon 0, the long-range guided missile that is capable of LAD. In Table 4-1, an r_{CPA} of 30 km was chosen to fly within the ship's engagement envelope for Weapon 0. The Boolean column "Intercept Possible?" shows whether or not the missile is physically able to reach the target based on threat and missile distances and velocities. The maximum intercepts possible for this threat is nine; that is, if the missile had a P_K of 0 but flew the entire distance to the target, the ship would be able to fire a maximum of nine missiles at the threat.

Note that this simple analysis does not account for system reaction delays, kill evaluation times, number of illuminators, reduced P_K for crossing targets, etc. The actual number of possible intercepts will likely be smaller than nine if these constraints had been included in the analysis. Figure 4-2 depicts this same analysis across the entire spectrum of the missile's range capability.

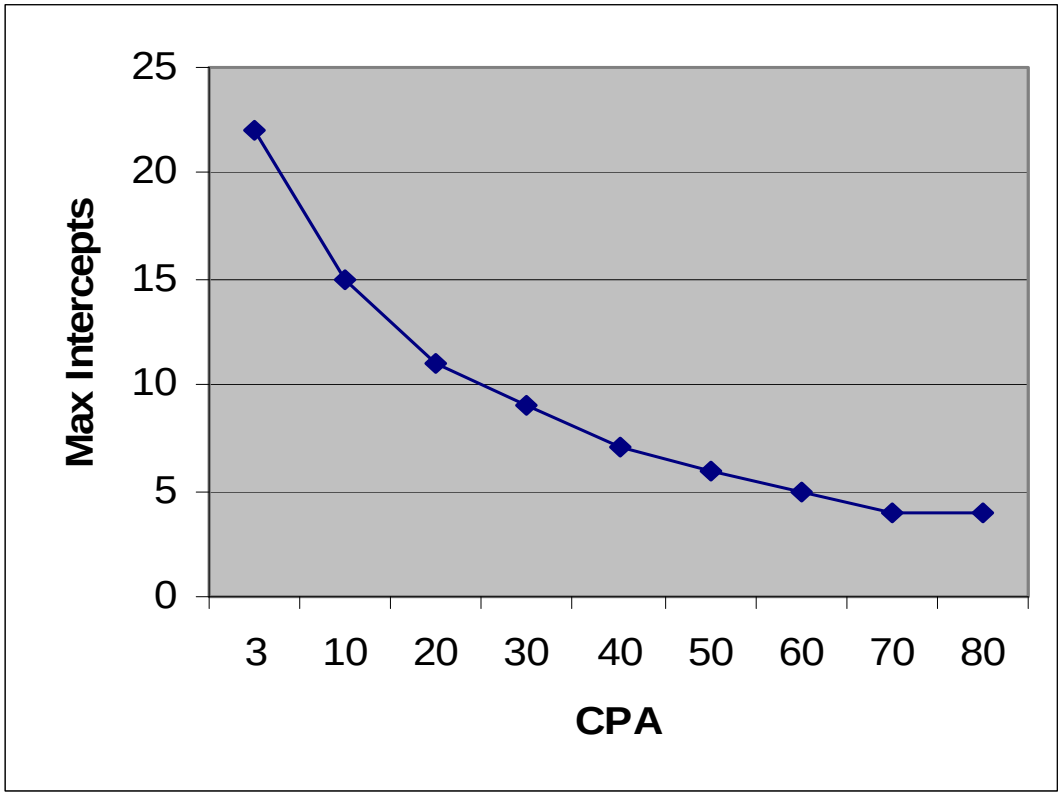
It is also important to note that this is the maximum number of engagements possible against a single threat that is not killed. Since the weapons chosen for our AAW engagements will have relatively high P_K, the ship will likely kill any threat within the raid with two missiles. Given the above analysis, the derived requirement that the "system shall defend a HVU attack consisting of 6 ASM threats at a P_{RA} of 0.99" seems to be reasonably achievable. Only further analysis with M&S can prove whether this requirement is unachievable, in which case the requirement must be adjusted according to stakeholder needs and managed expectations.

Table 4-1: Weapon Analysis

Inputs:	r_{CPA} =	30	km					
	r_{max} =	80	km					
	v =	0.85	km/s					
	u =	0.306	km/s					
Outputs:	r_{total}=	149.36	km					
	m =	9	maximum number of intercepts possible					
i	θ		α		r_i (km)	d_i (km)	$\sum d_i$ (km)	Intercept Possible?
	degrees	radians	degrees	radians				
1	21.0	0.37	7.4	0.13	60.3	21.6	21.6	1
2	28.4	0.50	9.8	0.17	46.4	16.6	38.2	1
3	38.1	0.67	12.6	0.22	37.0	13.1	51.3	1
4	50.8	0.89	15.8	0.28	31.3	11.0	62.2	1
5	66.6	1.16	18.6	0.32	28.8	10.0	72.3	1
6	85.2	1.49	20.1	0.35	29.7	10.3	82.5	1
7	105.3	1.84	19.5	0.34	34.9	12.1	94.6	1
8	124.8	2.18	16.7	0.29	46.1	16.1	110.8	1

Table 4-1: Weapon Analysis (cont.)

i	θ		α		r_i (km)	d_i (km)	$\sum d_i$ (km)	Intercept Possible?
	degrees	radians	degrees	radians				
9	141.5	2.47	12.7	0.22	66.0	23.4	134.1	1
10	154.2	2.69	8.9	0.16	99.1	35.4	169.5	0
11	163.2	2.85	6.0	0.10	152.1	54.6	224.1	0
12	169.1	2.95	3.9	0.07	236.0	84.8	308.9	0
13	173.0	3.02	2.5	0.04	367.6	132.2	441.1	0
14	175.5	3.06	1.6	0.03	573.6	206.5	647.6	0
15	177.1	3.09	1.0	0.02	895.8	322.5	970.1	0
16	178.2	3.11	0.7	0.01	1399.5	503.8	1473.8	0
17	178.8	3.12	0.4	0.01	2186.5	787.1	2261.0	0
18	179.2	3.13	0.3	0.00	3416.2	1229.8	3490.8	0
19	179.5	3.13	0.2	0.00	5337.8	1921.6	5412.4	0
20	179.7	3.14	0.1	0.00	8340.2	3002.5	8414.9	0
21	179.8	3.14	0.1	0.00	13031.6	4691.4	13106.2	0
22	179.9	3.14	0.0	0.00	20361.8	7330.3	20436.5	0
23	179.9	3.14	0.0	0.00	31815.3	11453.5	31890.0	0
24	179.9	3.14	0.0	0.00	49711.5	17896.1	49786.1	0
25	180.0	3.14	0.0	0.00	77674.2	27962.7	77748.8	0
26	180.0	3.14	0.0	0.00	121365.9	43691.7	121440.5	0
27	180.0	3.14	0.0	0.00	189634.2	68268.3	189708.8	0
28	180.0	3.14	0.0	0.00	296303.4	106669.2	296378.1	0
29	180.0	3.14	0.0	0.00	462974.0	166670.6	463048.7	0



Illustrates the maximum number of long-range missiles fired against a threat at a given CPA. This was used to derive the LAD requirement.

Figure 4-2: Max Number of Intercepts at a Given CPA

The functional and performance requirements were derived and defined from the mission requirements, and documented in the SysML diagrams in the next section.

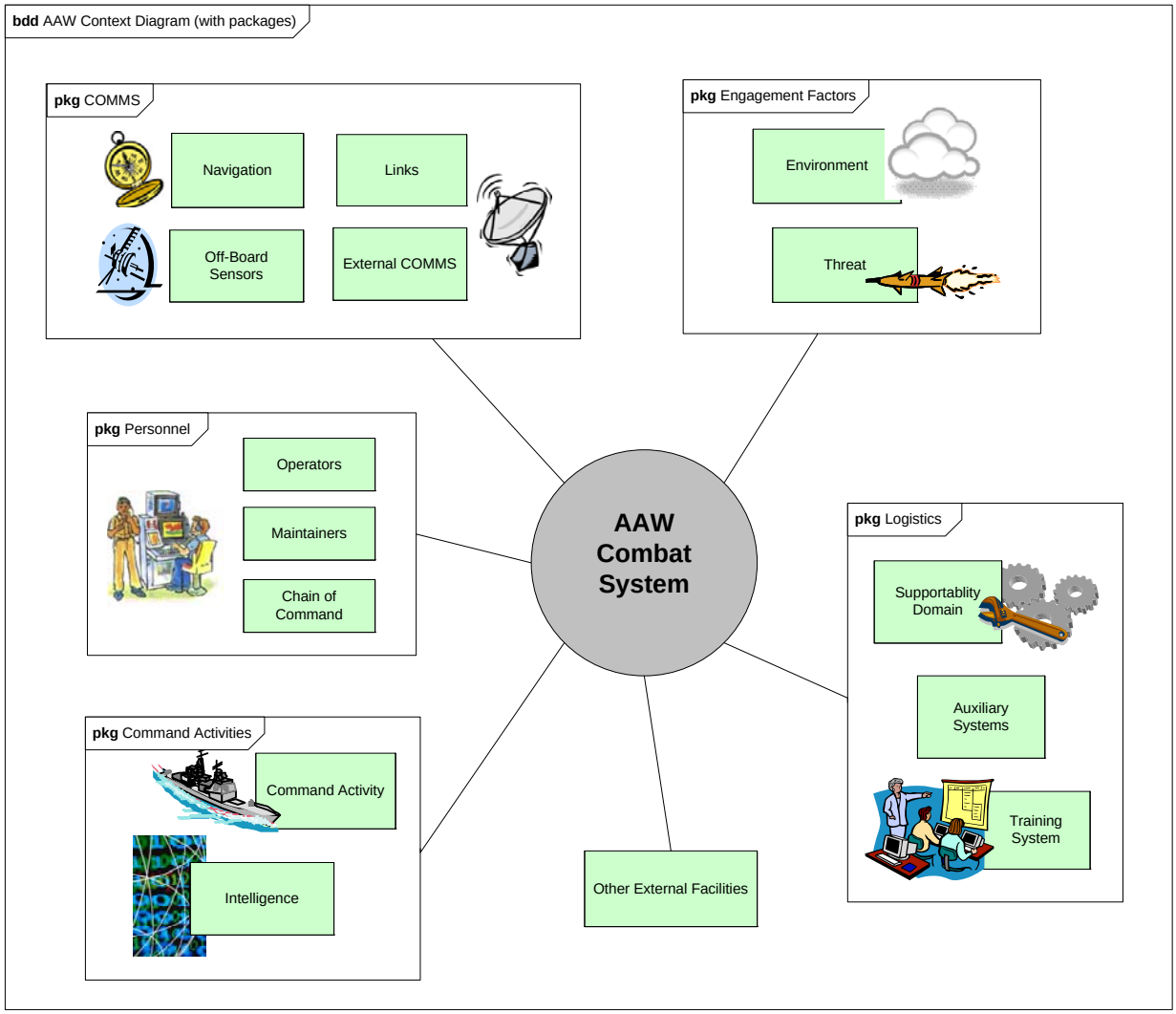
The next step in the process was to validate and verify the requirements using math and logic functions. Geometry was used to determine Visual and Radar Horizon Range (as described in section 4.4); this determined the maximum target detection range. Self defense requirements were verified using M&S as described in section 4.5. SysML requirements data was used to develop CORE design artifacts. CORE design artifacts, described in section 4.6 provided traceability back to the originating requirements.

4.2.3 SysML Products Generation and Analysis

As described in section 3.1.5, SysML was used to create artifacts documenting system requirements and design features. Although several SysML automation tools are used in industry, none were available for this project due to funding constraints. Instead, a SysML 1.0 template for Microsoft Visio was used to represent system design concepts in the modeling language. A summary of each diagram developed in the requirements process is given below.

4.2.3.1 Context Diagram

The context diagram was developed to define system boundaries and identify the actors and external systems that interact with the system. Figure 4-3 depicts a context diagram for the AAW system. External systems include personnel, communications, logistics, command activities, and engagement factors. The context diagram depicts the system and identifies interfaces that must be developed for successful interoperability. This diagram was developed in accordance with the scenarios in the AAW ConOps.



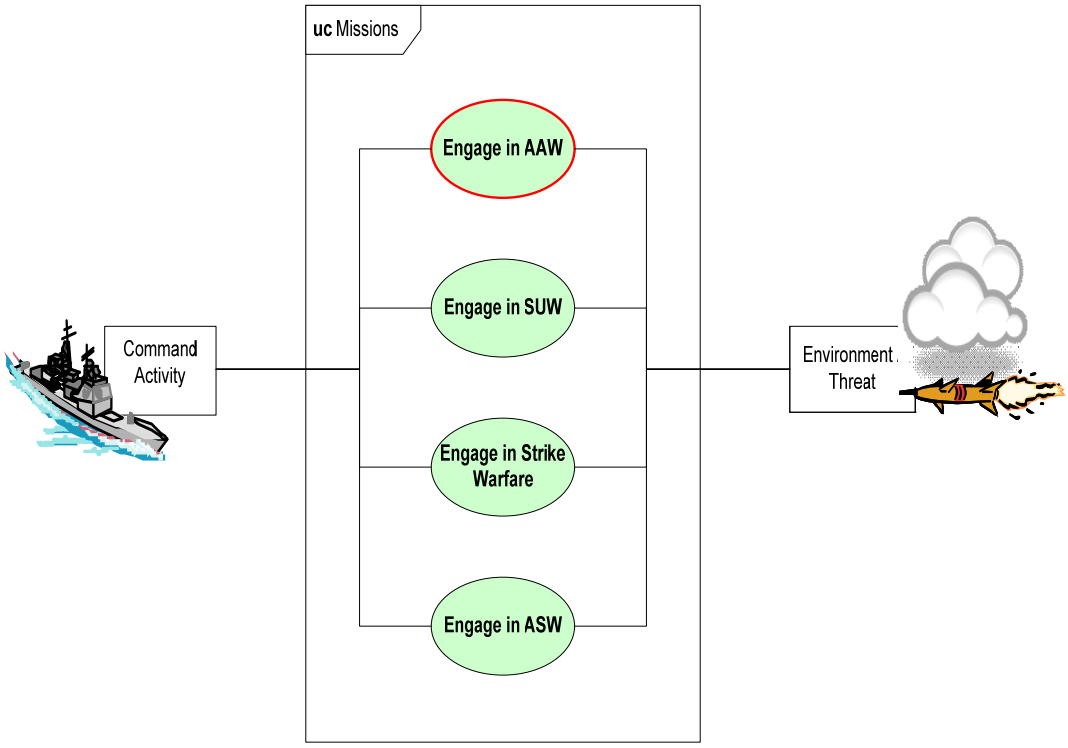
Depicts the AAW system and the following external influences to the system: personnel, communications, logistics, command activities, and engagement factors.

Figure 4-3: AAW System Context Diagram

4.2.3.2 Use Case Diagrams

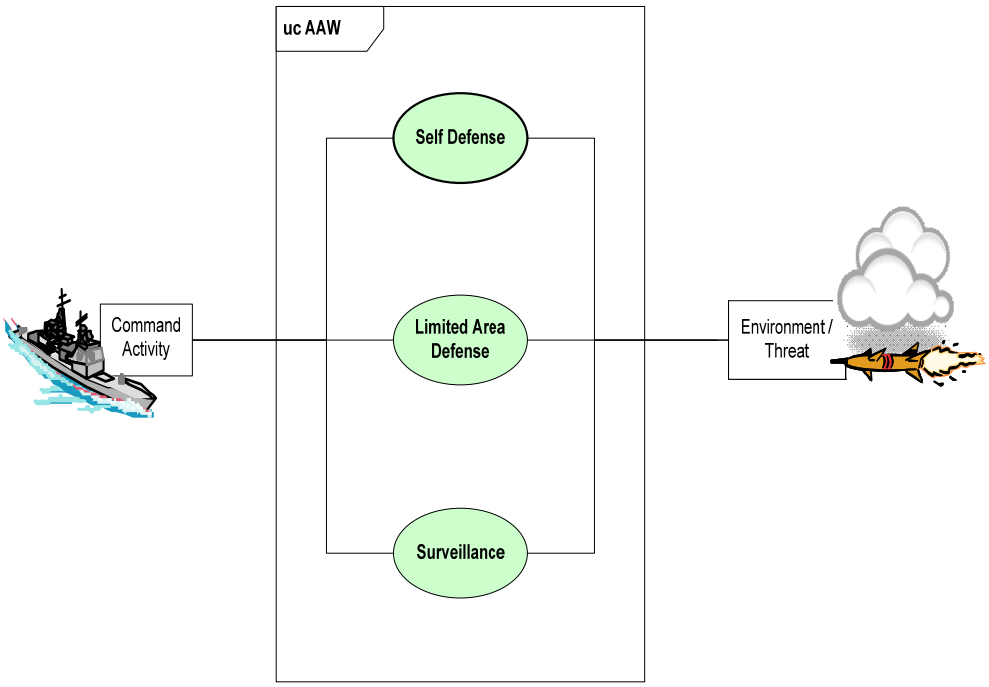
The Unified Modeling Language (UML) use case diagram describes the major tasks the system must accomplish to achieve its overall goal. Figures 4-4 through 4-7 are the use case diagrams developed for the AAW system. The illustrations provide a portion of what would be provided to develop a complete use case. The use case descriptions provided in the following paragraphs would be included in a fully dressed use case. The use case diagram identifies the actors that use the system and captures the system behavior by showing internal functionalities, capabilities and dependencies. Use cases are goal-oriented and describe the workflow of a process instead of interactions among system components. The use case diagrams help system analysts understand the underlying problems to be solved and the objectives to be accomplished by the perceived systems.

For the combat system, use cases were used to form mission threads. Figure 4-4 is a multi-mission use case diagram that represents the many functions a warship command activity may engage in, such as AAW, Surface Warfare (SUW), Strike Warfare and Anti-Submarine Warfare (ASW). The diagram depicts a system with four possible mission capabilities: Engage in AAW, Engage in SUW, Engage in Strike Warfare, and Engage in ASW. The actors in these mission threads are identified as the Command Activity or the Environment/Threat. Each oval in Figure 4-4 describes a scenario depicting the interactions between the actors and the overall mission. The AAW mission threads (Surveillance, Self Defense and Limited Area Defense) are depicted in the AAW Mission Use Case diagram (Figure 4-5).



Depicts the possible mission roles in the form of a use case diagram.

Figure 4-4: Multi-Mission Use Case Diagram

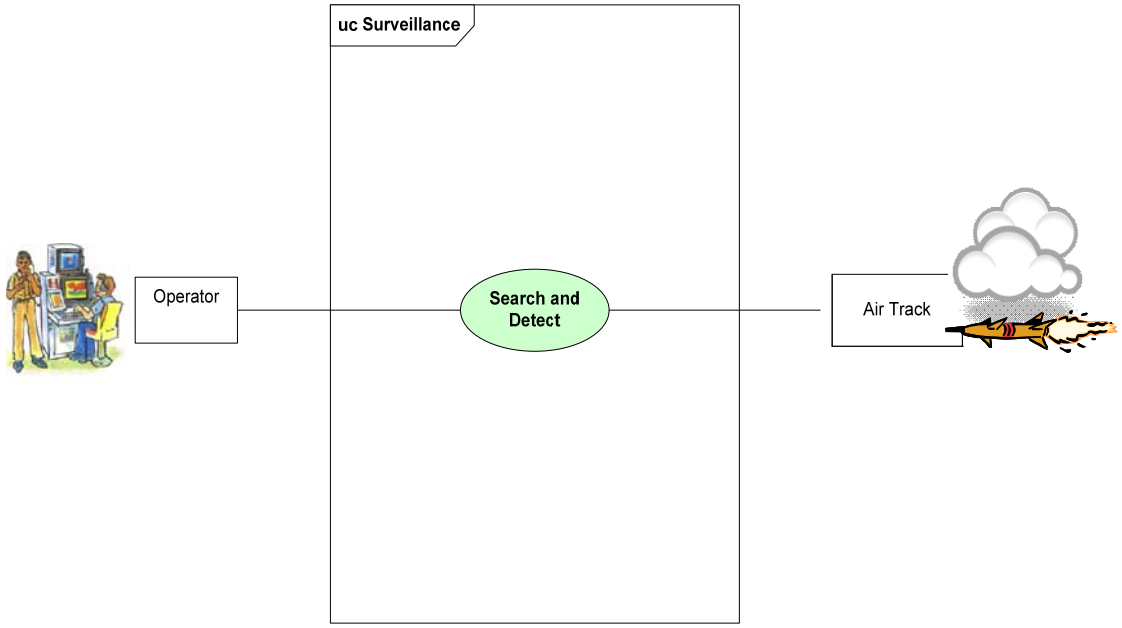


Depicts the decomposition of the AAW mission in the form of a use case diagram.

Figure 4-5: AAW Mission Use Case Diagram

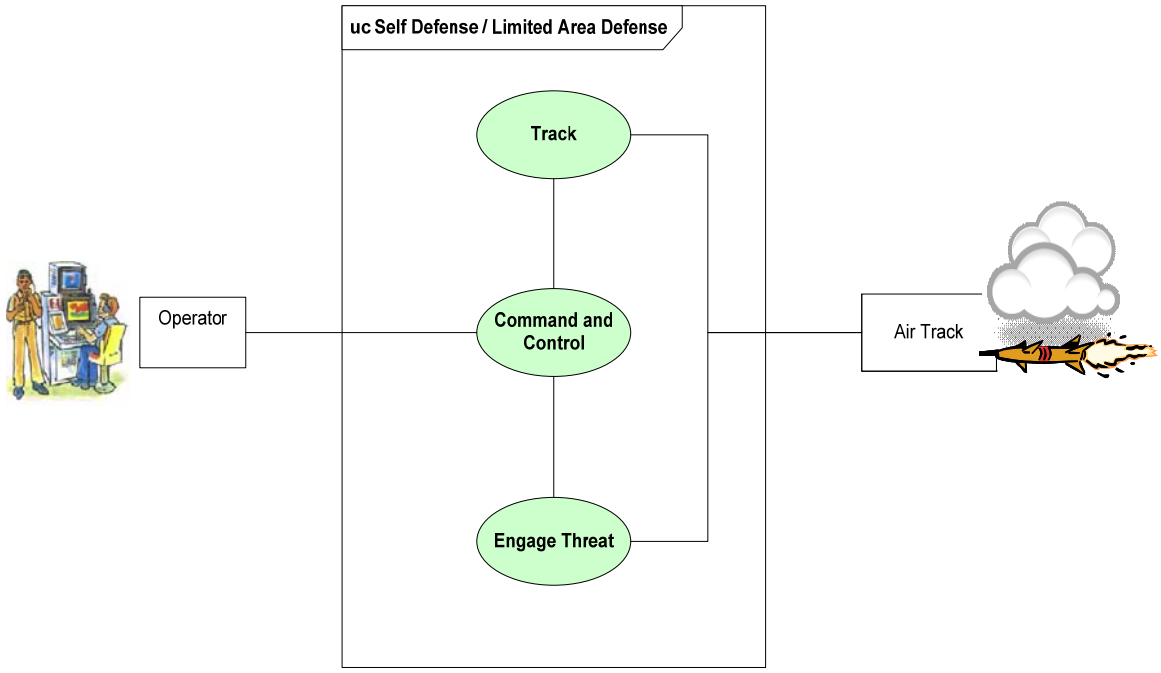
Figure 4-6 is the Surveillance use case diagram and depicts the high-level search, detect and track functions required to perform the surveillance mission. The surveillance system could be a multi-function phased-array radar capable of search, automatic detection, transition to track, tracking of air and surface targets, and missile engagement support.

Figure 4-7 is the Self Defense and Limited Area Defense use case diagram. The high-level functions are Track, Command and Control (C2), and Engage functions. The functions can be performed by a combination of personnel, equipment, communications and facilities based on the level of automation required. The tracking function continuously determines and updates the location (bearing, range and elevation) and direction of a target which is then fed to C2. The C2 functions consist of planning, coordinating, directing, and controlling forces and operations to accomplish the mission. C2 additionally provides the human interface for control of system operation. C2 provides engagement orders and data for directing engagement. The output of the tracking system can be sent to a fire control system, which stores the information and derives the target's motion and its future position. The engage function provides the capability for decisions to be made by automatically based on doctrine. The actors in these mission threads are identified as the Operator and Air Track.



Depicts the decomposition of the surveillance mission in the form of a use case diagram.

Figure 4-6: Surveillance Use Case Diagram



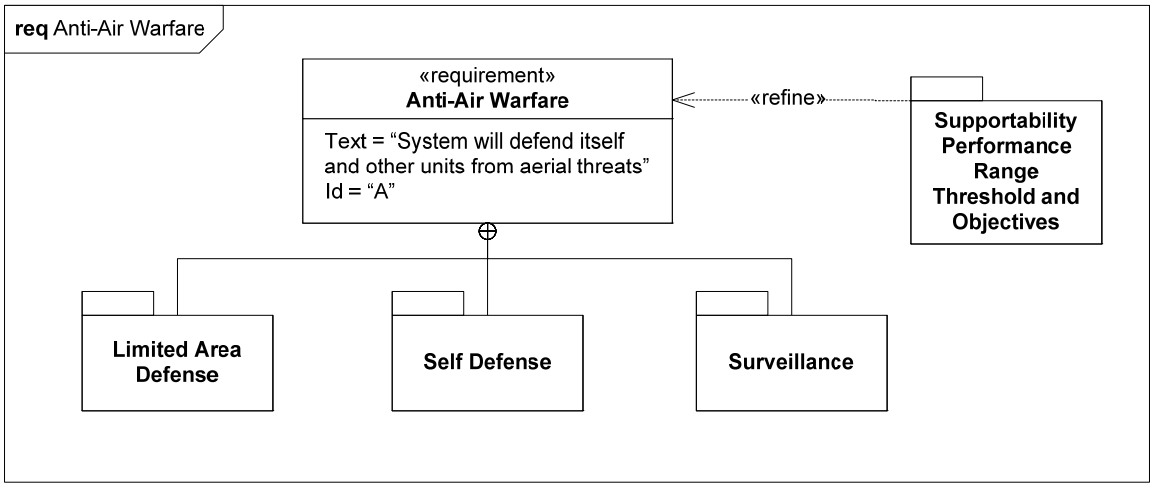
Depicts the decomposition of the self defense/limited area defense mission in the form of a use case diagram.

Figure 4-7: Self Defense/Limited Area Defense Use Case Diagram

4.2.3.3 Requirements Diagrams

The requirements process involves the elicitation, specification, prioritization and management of requirements. Traditionally, these requirements have been captured in natural language, which can be ambiguous and open to interpretation. SysML provides a method for representing functional and non-functional requirements, and their relationships with one another, in a hierarchical graphical form to reduce or eliminate ambiguity. SysML provides a bridge between natural language-based requirements and methods of functional allocation, such as activity and sequence diagrams. SysML requirements diagrams also support traceability of requirements during system modeling and specification. "Requirements traceability helps in identifying the sources, destinations, and links between requirements and models created during system development." (Soares and Vrancken 2007) The requirements diagram is the first model in which effectiveness measures are identified and assigned values that allow designers to bound the solution space, evaluate the alternatives and discriminate the solution from competitor solutions (Friedenthal 2008).

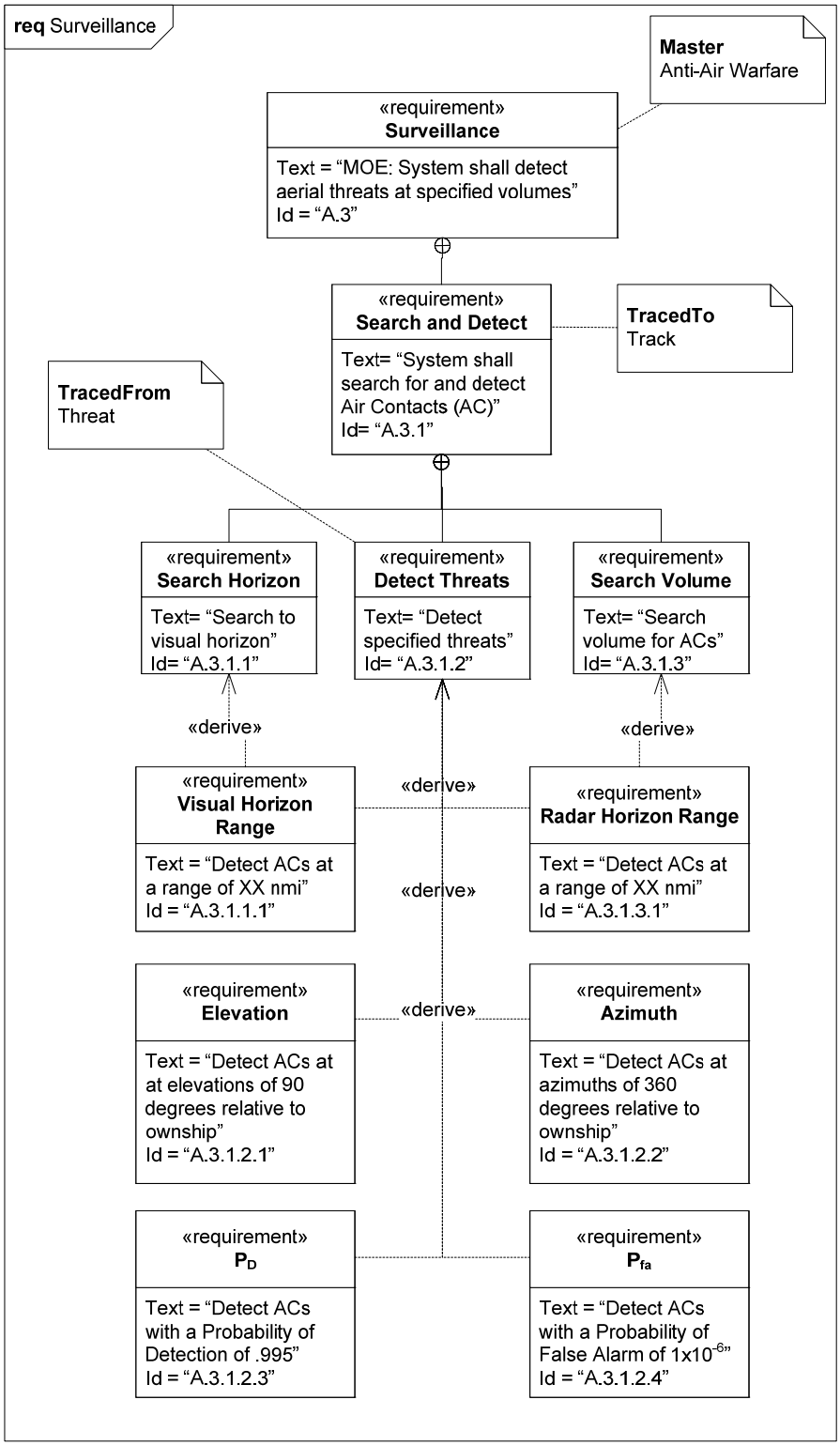
Figure 4-8 is the AAW Requirements Diagram. In this diagram, the AAW mission is decomposed to display a parent-child relationship between AAW and its three functional areas: Surveillance, SD and LAD. Furthermore, the entire AAW mission is refined and supported by non-functional Supportability Performance Range Thresholds and Objectives. These mission areas and supportability concepts were packaged for further decomposition in follow-on diagrams.



Depicts the packages that support the AAW requirements in the form of a requirements diagram.

Figure 4-8: AAW Requirements Diagram

Figure 4-9, the Surveillance Requirements Diagram, illustrates the surveillance mission: to search for and detect air contacts to be further investigated by other functions in the system. The surveillance system performs a horizon or volume search and detects air contacts in a marine environment. Additional requirements were derived from these higher-level requirements, shown by the SysML "derive" relationship.



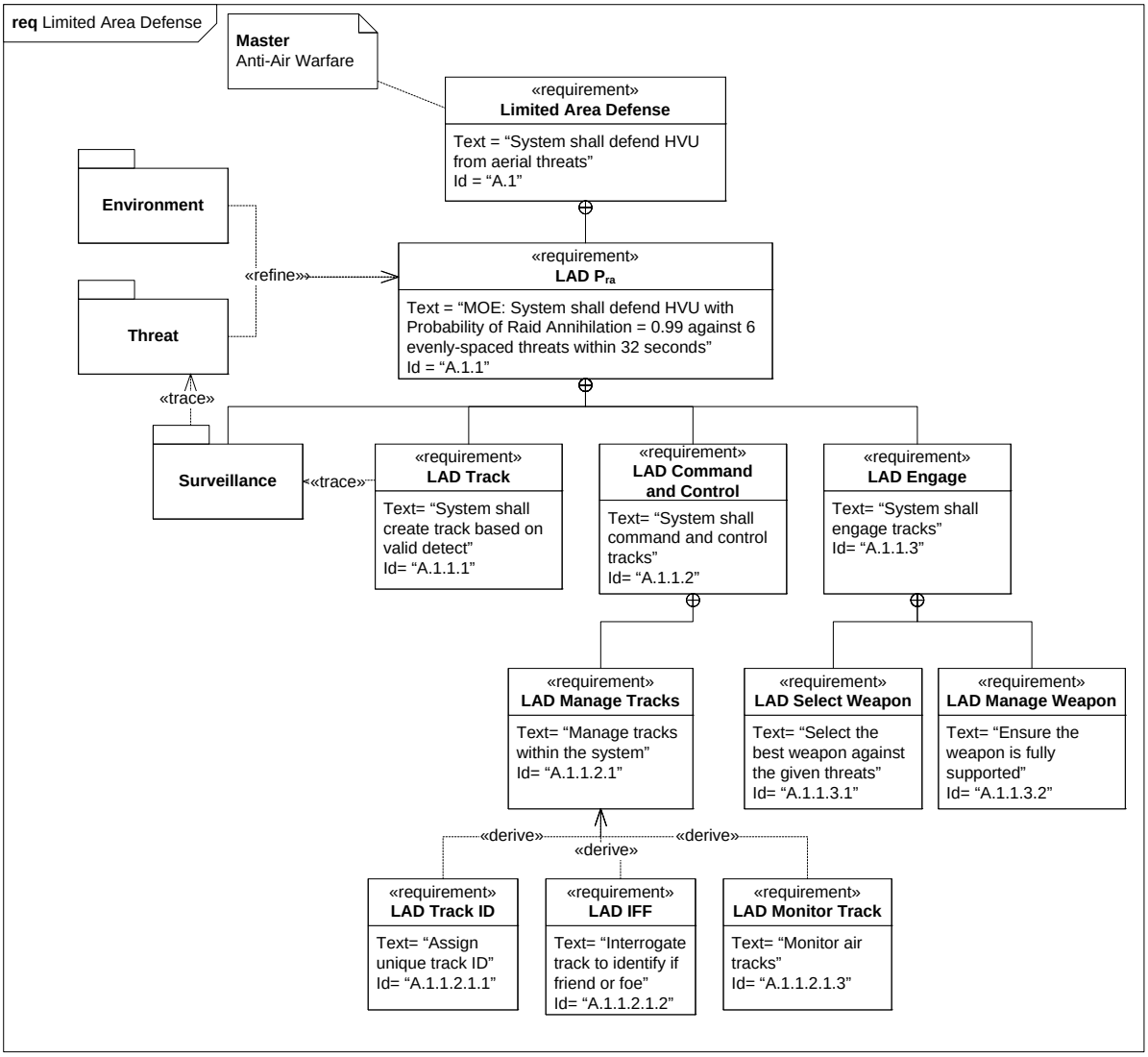
Depicts the requirements traceability for the surveillance package.

Figure 4-9: Surveillance Requirements Diagram

The SD and LAD mission areas are similar in functionality. The differences lie in the ability of the AAW system to perform point defense (defend against threats coming at ownship) and area defense (defend or cover another ship (a high value unit) operating in the vicinity of ownship). Both mission areas use surveillance air contact detections to track, command, control and engage targets determined to match threat profiles. First order analysis determined what geometries could be covered with the weapons and reaction times limitations identified in the ConOps. (Wagner et al. 1999) The following parameters were identified: detection ranges, reaction time of the system (including the speed of the weapons chosen), and potentially reduced P_K of weapons based on aspect angles of the threat.

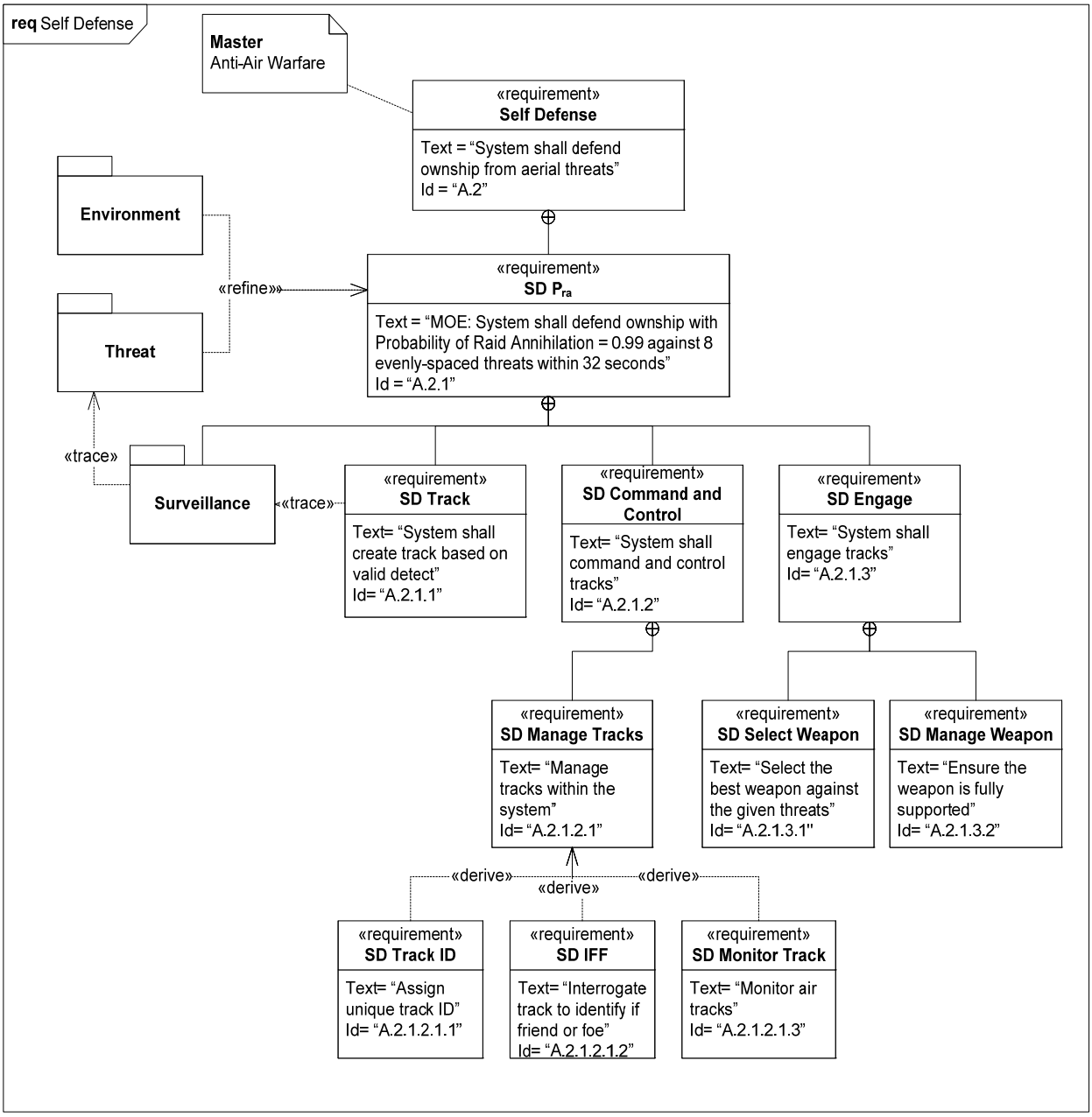
Figures 4-10 and 4-11 decompose the LAD and SD mission areas, respectively. Functionally in a LAD role, the system shall defend HVUs from aerial threats with a specified P_{RA}. Similarly, the system shall defend ownship from aerial threats in the SD role.

Figures 4-12 and 4-13 define the threats and environments, respectively, in which the system will operate. To limit the scope of the mission, a single type of threat in a single type of environment was chosen. Because some lower-level requirements may be shared among many different threat types, the "copy" relationship is used in Figure 4-12. Both diagrams provide placeholders for additional types of threats and environments.



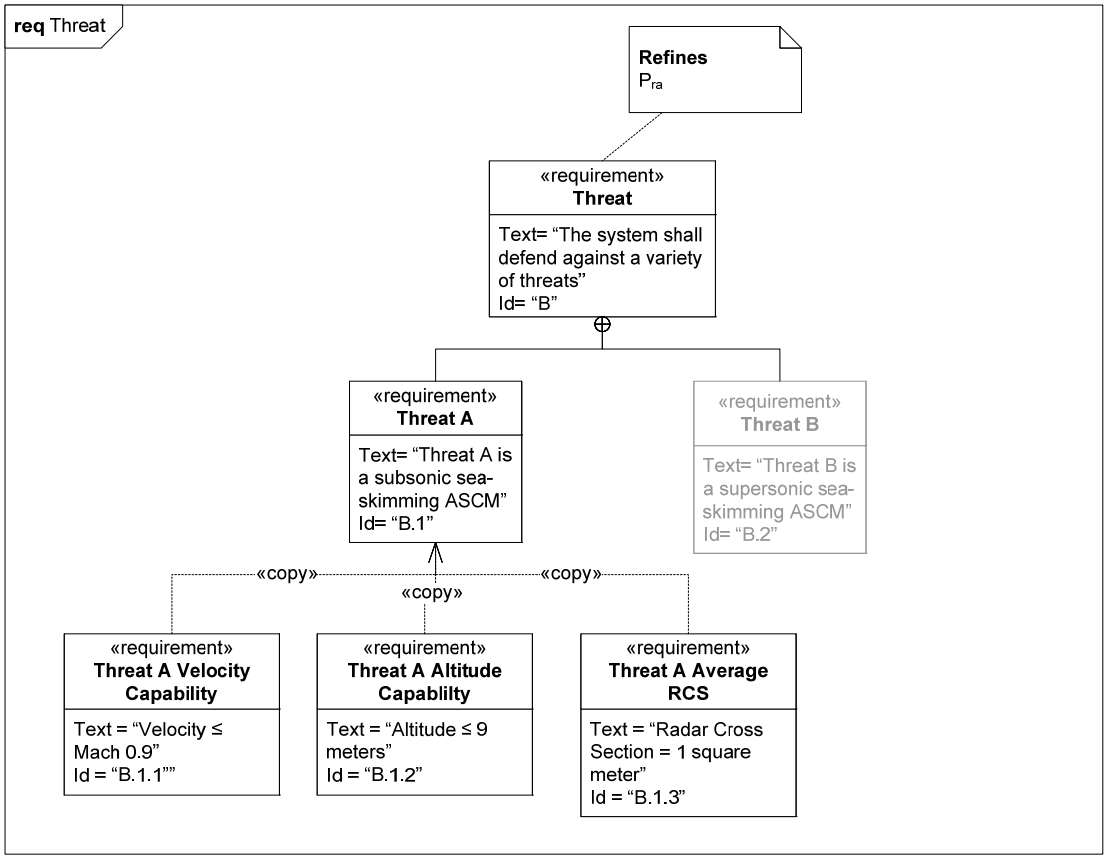
Depicts the requirements traceability for the limited area defense package.

Figure 4-10: Limited Area Defense Requirements Diagram



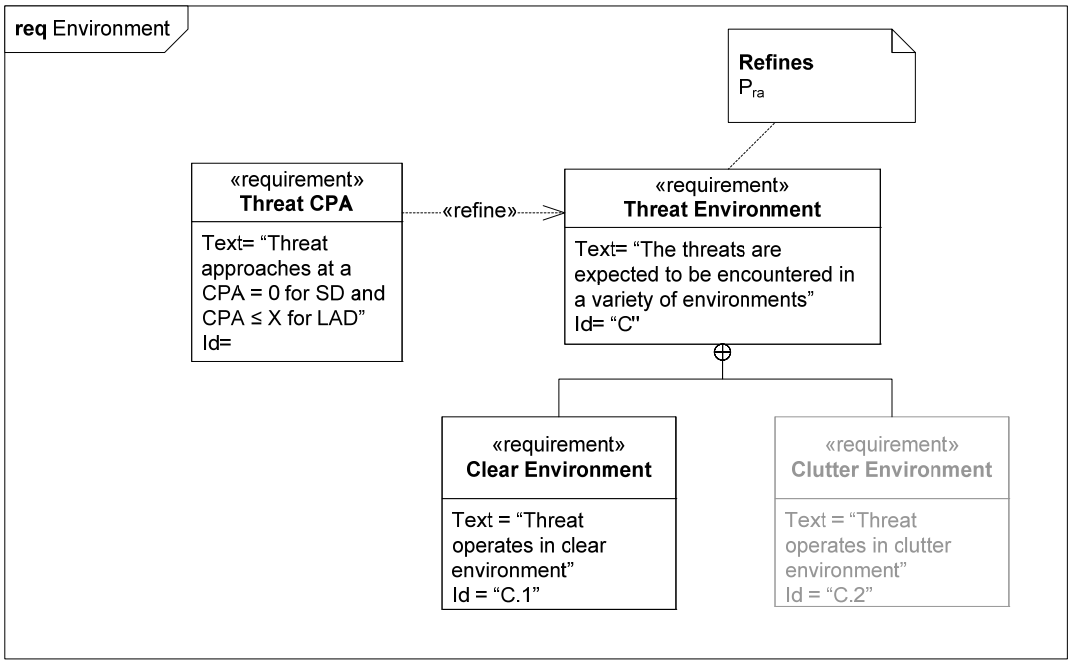
Depicts the requirements traceability for the self defense package.

Figure 4-11: Self Defense Requirements Diagram



Depicts the requirements traceability for the threat package.

Figure 4-12: Threat Package Requirements Diagram



Depicts the requirements traceability for the environment package.

Figure 4-13: Environment Package Requirements Diagram

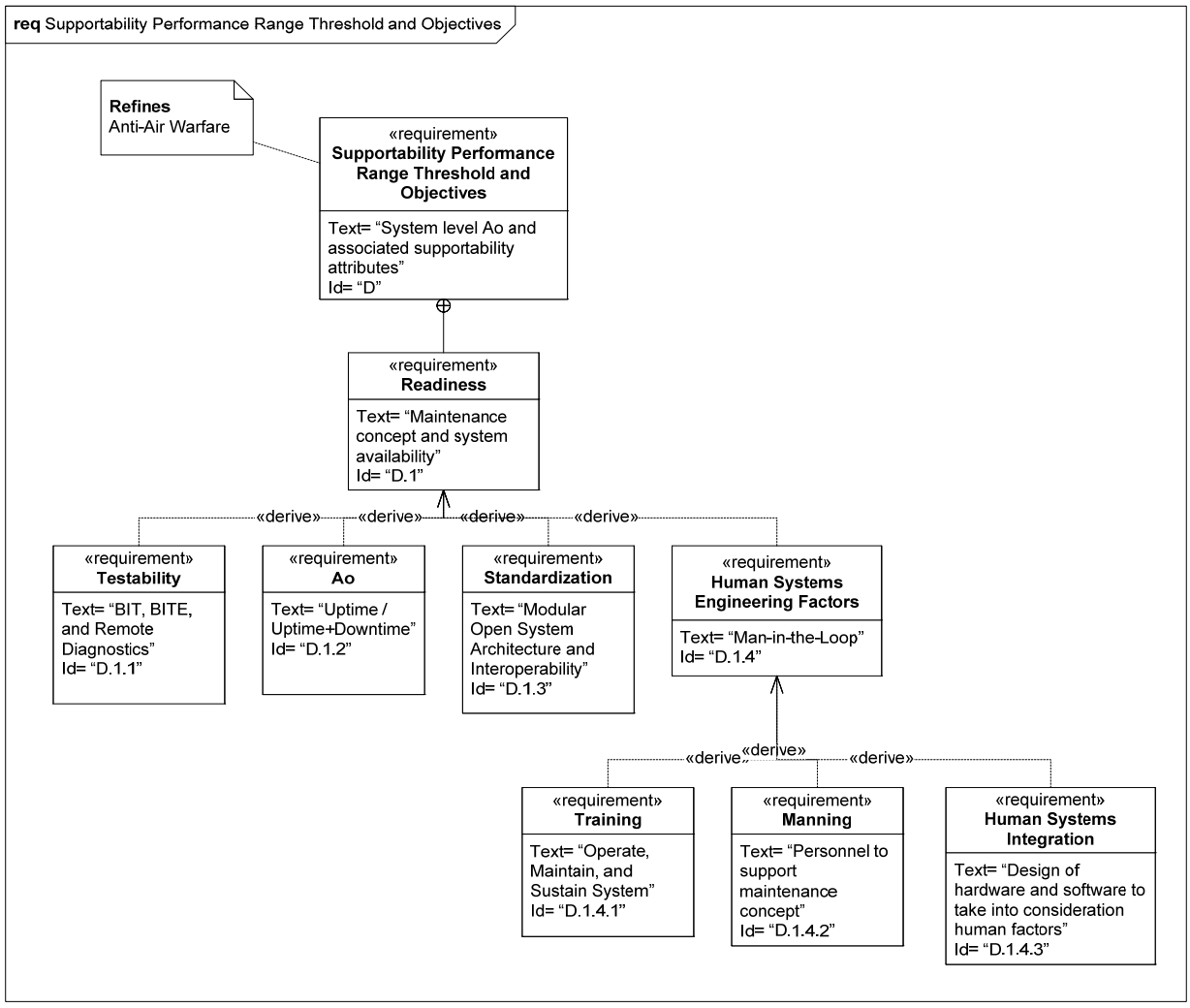
Figure 4-14 defines the Supportability requirements, which are refined from the high-level Supportability Range and Performance Objectives and Thresholds in Figure 4-8. Supportability requirements are Key Performance Parameters (KPPs) in our methodology. Supportability provides system and personnel readiness. Readiness was further refined to the ILS elements, which would be modeled to support the required A_O (system operational availability to meet mission requirements). As the requirements diagram for Supportability is decomposed, factors such as testability, standardization, Human Factors (anthropometrics, training, etc.), redundancy, reliability, test equipment, technical documentation, supply support (Failure Modes and Effects Analysis (FMEA), Level of Repair Analysis (LORA)), sustainability, maintainability and upgradeability need to be addressed by the systems architect and design engineers.

Decomposition of requirements was shown throughout the SysML requirements diagrams developed for the AAW mission. Stakeholder needs were expressed in mission requirements for Limited Area Defense, Self Defense and Surveillance. Mission requirements were met through system requirements (i.e., P_{RA} against specified threats) and accomplished through component requirements (i.e., search and detect, track, C2 and engage).

4.2.3.4 Model Validation

Validation analysis was conducted for each developed artifact to ensure the integrity and feasibility of AAW system requirements. Validation efforts consisted of maintaining requirements traceability throughout the systems engineering process. CORE and SysML models were used as primary tools for requirements validation. Both models established parent-child relationships for verifying requirements flow down and were utilized to validate requirements allocation to functions for hardware, mission and software. Validation analysis generated feedback which was used to update the model artifacts. The validation process continued as data from each artifact was input into the automated CORE tool (described in section 4.6). CORE products were used to ensure

that all follow-on models were developed using the same terminology and to guarantee model traceability back to the originating requirements.



Depicts the requirements traceability for the supportability, performance range, thresholds and objectives packages.

Figure 4-14: Supportability Requirements Diagram

4.3 AAW FUNCTIONAL ANALYSIS AND ALLOCATION

4.3.1 Systems Engineering Process for Architecture Development

The system requirements defined in section 4.2 were analyzed to define the required functional behaviors of the system. The requirements and functional behaviors were used to develop the AAW architecture system. A key requirements document was the ConOps document. "The operational concept is a vision of: what the system is, a statement of mission requirements, and a description of how the system will be used." (Buede 2000) The functional architecture of the AAW system was developed based on research and an understanding of legacy Navy AAW systems. The AAW architecture contains a hierarchical model of the functions performed by the system. The architecture illustrates the flow of information through the transformational processes of the system functions to the waiting external systems being serviced by the system.

4.3.2 AAW Artifacts Generated

AAW architecture development is documented in the following artifacts:

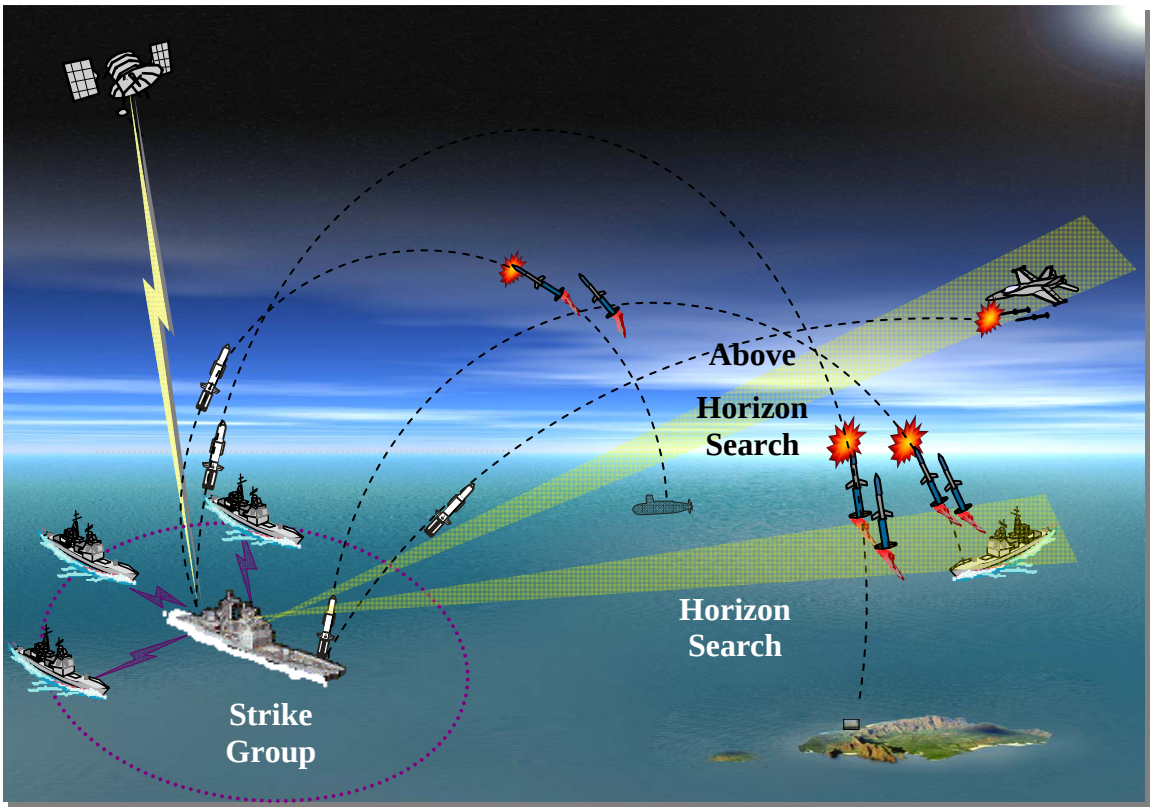
1. AAW Concept of Operations (ConOps) Document
2. AAW Functional Architecture
3. AAW Activity Diagram
4. AAW Enhanced Functional Flow Block Diagram (EFFBD)
5. AAW Sequence Diagram
6. AAW Block Definition Diagram (BDD)
7. AAW Integrated Block Diagram (IBD)

4.3.2.1 AAW Concept of Operations

Figure 4-15, the AAW Mission ConOps diagram (DoDAF OV-1), illustrates the mission requirements for Surveillance, Self Defense and Limited Area Defense of high value units in a marine environment. Each of the key scenarios (Surveillance, SD and LAD) was described in the AAW ConOps (Appendix G). The scenarios were used to

understand and uncover any underlying issues that were critical in meeting the objectives of the user. An initial statement of the objectives and an Excel analysis were used to help with the evaluation of objectives. When the problem was understood, the scenario development process began and the communication between the team and stakeholders proceeded. The scenarios helped define mission definitions, tactical objectives, marine environments, threat definition, and to some degree, limited tactics.

In Figure 4-15 (DoDAF OV-1), the AAW system is engaging eight (8) radially inbound ASMs. The ASMs can originate from either a single location or multiple locations, such as enemy ships, submarines, aircraft, or land based platforms. It is the primary task of the AAW system to provide a layered defense to detect, track, C2, engage and destroy the incoming ASM raid with a 0.99 P_{RA}. The OV-1 depicts system detection, using horizon and above horizon, short and long range search functions. Once the track is identified as hostile, the system will assign the weapon systems to engage and destroy the target, protecting ownship and other high value units in the area. Communication between ships and satellites is also important within the operational area. Therefore, a net-centric communication path is provided within the strike group to allow for long range distance support and C2 communications.



The AAW OV-1 illustrates the mission requirements for self defense and limited area defense of high value units in a marine environment.

Figure 4-15: AAW ConOps [OV-1]

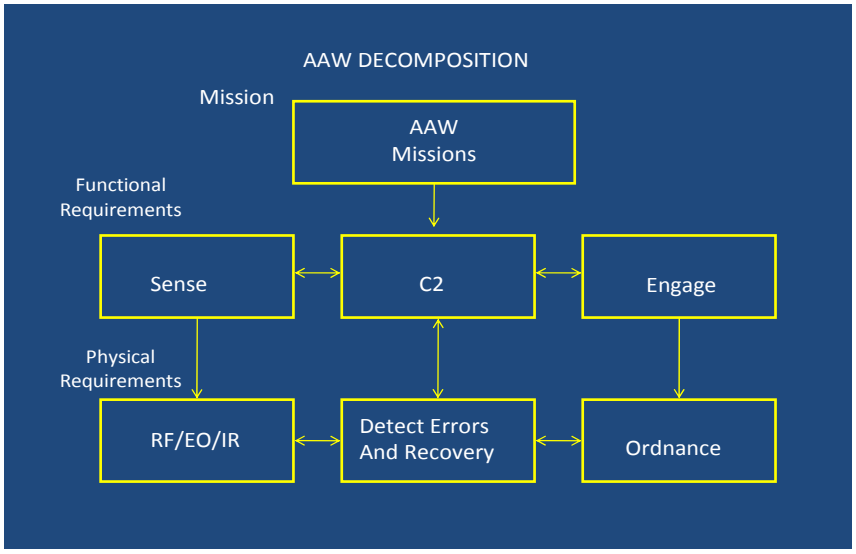
4.3.2.2 AAW Functional Architecture

Functional composition is recommended when a system is either unprecedented or is a radical departure from an existing system. Functional decomposition is sufficient when the system is an update or variation of an existing system per Buede. Using a combination of decomposition and composition sometimes referred to as middle-out, can produce the best results. Buede states the following:

> One can make use of simple functionalities associated with specific scenarios defined in the ConOps to establish a "sense" of the system. Then positioning a top-level decomposition that is likely to match the top-level segmentation of the physical architecture is common before proceeding to do decomposition that is reinforced by periodic reference to the functionalities to assure completeness. (Buede 2000)

A functional composition was used to allocate functionality into a layered architecture to optimize system performance based on low coupling and high cohesiveness. A decomposition approach is used to define and allocate AAW system functionality for legacy systems. The layered architecture functionality was allocated from a top-level system functionality required to execute AAW missions and then partitioning that function into several sub functions within domain components. The resultant "middle-out" approach was able to incorporate new functions in addition to integrating legacy architectures that kept system requirements in mind.

An exploration of the operational and support behavior of legacy AAW systems led to the decomposition depicted in Figure 4-16. There were several iterations of the AAW architecture developed during this study. Figure 4-16 provides a high level view of what AAW architecture is composed of: sense, C2, and engage functions required to further control sensors and weapons necessary to meet P_{RA} requirements. By dividing the system into smaller units, each function can be dissected and restructured to meet the stakeholder's needs. Decomposition is a form of reductionism or, simply stated, a simplification of a complex system (Melancon 2008). The top level functions correspond to the needs of the user while the bottom level represents the needs of the system.



This figure depicts the decomposition of the AAW architecture into various functional and physical requirements; top level depiction of AAW.

Figure 4-16: AAW Decomposition Diagram

From this level of functionality, it was clear that behavior and activities could not be readily understood or if requirements could be met. A review of Navy activities further led to a refinement of the architecture, as depicted in Figure 4-17. The activities/functions identified are in much greater detail than what was previously identified for detect, control, and engage functions.

1.0 Search and Detect
- 1.1 Detect Targets
  - 1.1.1 Use Radar
    - 1.1.1.1 Perform Transmit and Receive Functions
    - 1.1.1.2 Perform Signal Processing
    - 1.1.1.3 Determine Position
  - 1.1.2 Use EO/IR
    - 1.1.2.1 Perform Scanning
    - 1.1.2.2 Perform Signal Processing
    - 1.1.2.3 Determine Position
  - 1.1.3 Use EW

2.0 Tracking
- 2.1 Use Local Platform Tracking
  - 2.1.1 Use Single Sensor Tracking
    - 2.1.1.1 Determine Track Position
      - 2.1.1.1.1 Determine Track Vector
        - 2.1.1.1.1.1 Assign track Number
  - 2.1.2 Use Multiple Sensor Tracking
    - 2.1.2.1 Use Modern Data Association
      - 2.1.2.1.1 Perform Data Fusion
        - 2.1.2.1.1.1 Determine Track Vector
          - 2.1.2.1.1.1.1 Assign Track Number

3.0 Command and Control
- 3.1 Manage Tracks
  - 3.1.1 Assign New Tracks
  - 3.1.2 Monitor Air Tracks
  - 3.1.3 Correlate Air Tracks
  - 3.1.4 Update Air Tracks
  - 3.1.5 Drop Air Tracks
- 3.2 Manage Tactical Control
  - 3.2.1 Manage Information
  - 3.2.2 Provide Display Support for Tactical Operation
  - 3.2.3 Combatant Commanders Decision Aide
  - 3.2.4 Provide Alerts and Prompts
  - 3.2.5 Manage Sub-mode
  - 3.2.6 Perform Tactical Control
    - 3.2.6.1 Perform Threat Evaluation/Weapons Assignment (TEWA)
      - 3.2.6.1.1 Conduct Threat/Kill Assessment
        - 3.2.6.1.1.1 Assess Air Threats
          - 3.2.6.1.1.1.1 Review Air Track History
          - 3.2.7.1.1.1.2 Assess Air Track Intent
          - 3.3.7.1.1.1.3 Assess Air Threat Capabilities
          - 3.3.7.1.1.1.4 Assess Air Threat Risk
    - 3.2.6.2 Perform Readiness Control
    - 3.2.6.3 Perform Weapons Control
  - 3.2.7 Perform Initialization
  - 3.2.8 Reconfigure
  - 3.2.9 Perform Readiness Assessment
  - 3.2.10 Manage Ship Sensors
    - 3.3.10.1 Collect Sensor Data from Ship Sensors/Data Links
      - 3.3.10.1.1 Integrate and Manages Local and Remote Tracks
- 3.3 Plan
- 3.4 Direct

4.0 Engage
- 4.1 Manage Weapons (HK/SK)
  - 4.1.1 Designate Electronic Warfare
    - 4.1.1.1 Launcher
  - 4.1.2 Designate Fire-and-Forget Weapon
    - 4.1.2.1 Launcher
  - 4.1.3 Designate Semi-Active Homing Weapon
    - 4.1.3.1 Launcher
      - 4.1.3.1.1 Initialize/Control Launcher
        - 4.1.3.1.1.1 Assign Illuminator
          - 4.1.3.1.1.1.1 Illuminate

An Excel view of the AAW decomposition, showing the parsing of functions.

Figure 4-17: AAW Excel Based Problem Decomposition

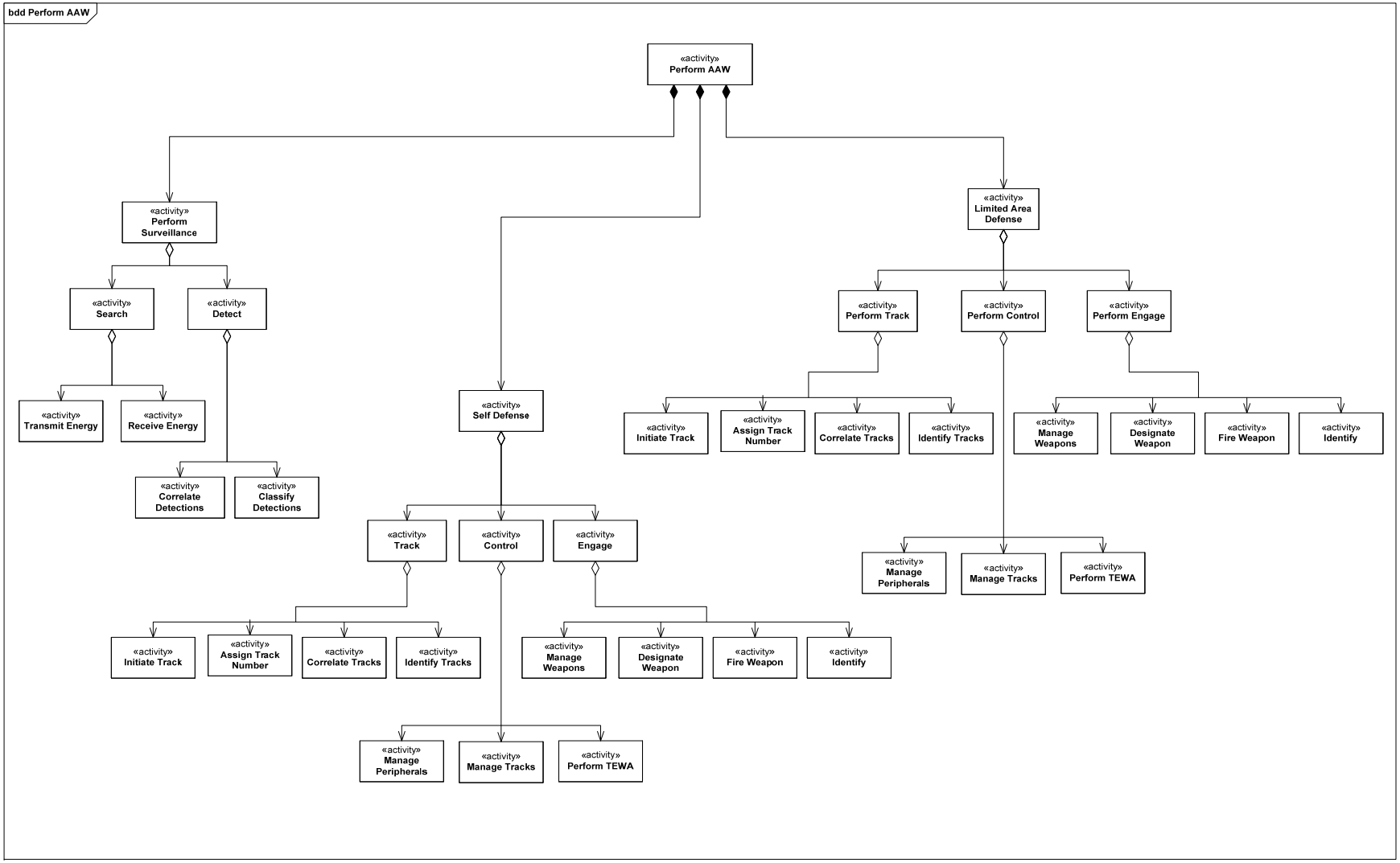
A key point to note here is that once the architecture becomes complex, Excel is not robust enough to maintain configuration. Changes are difficult to make and relationships must be maintained using manually entered notes. Robust tools, such as CORE and Dynamic Object Oriented Requirements System (DOORS), are more suitable to control and manipulate the architecture without the need to redraw functions.

The top level functions of the functional hierarchy are devoted to achieving AAW requirements. The four top-level functions in Figure 4-17 are Search & Detect, Track, Command & Control, and Engage. The top-level functionality was decomposed on the basis of certain stimulus or response threads that pass through the functions that were decomposed. For example, the track function identified in Figure 4-16 had to be further decomposed in order to gain further insight into behaviors needed to meet the overall mission requirement of $P_{RA.}$

When engineering complex systems, system requirements and specifications are often written "as the system shall do some described task." (Oliver et al. 1997) In applying this perspective to AAW requirements, one can say that destroying an incoming ASM is a primary task associated with AAW. The functionality that is decomposed from the high-level function of track must facilitate the requirement to destroy an incoming ASM. The question then becomes, what must the track function do in order to meet that requirement? It must determine the track position relative to ownship, determine the track vector (bearing) relative to ownship, and assign a track number necessary for accurate track updates. The track function has a detection report input and output. In order to meet the context of the problem, (the performance is defined by time in requirements) the function of tracking had to be broken down into additional detail to illustrate that correct processes and behaviors were attributed to the function. Furthermore, track time requirements (seconds) required automatic processes to account for the following: registration, time stamping of data, the building, validation, updating and management of tracks, and the display of track information. Detailed analysis would be required at this level and first order modeling would be used to identify timing

budgets/constraints. The above decomposition originated from the manage track function and is an example of what had to be accomplished to fully understand the function (other functions would require the same level of decomposition).

The final step in the architecture development process was to map the functional decomposition into a SysML standard. Figure 4-18, the SysML AAW Functional Architecture view, provides a more complete view of information when compared to the Excel view provided in Figure 4-17. Figure 4-18 represents the functions necessary to meet the requirements and ConOps described in this report.



This SysML diagram depicts the functions required to meet the AAW mission requirements and ConOps.

Figure 4-18: SysML AAW Functional Architecture

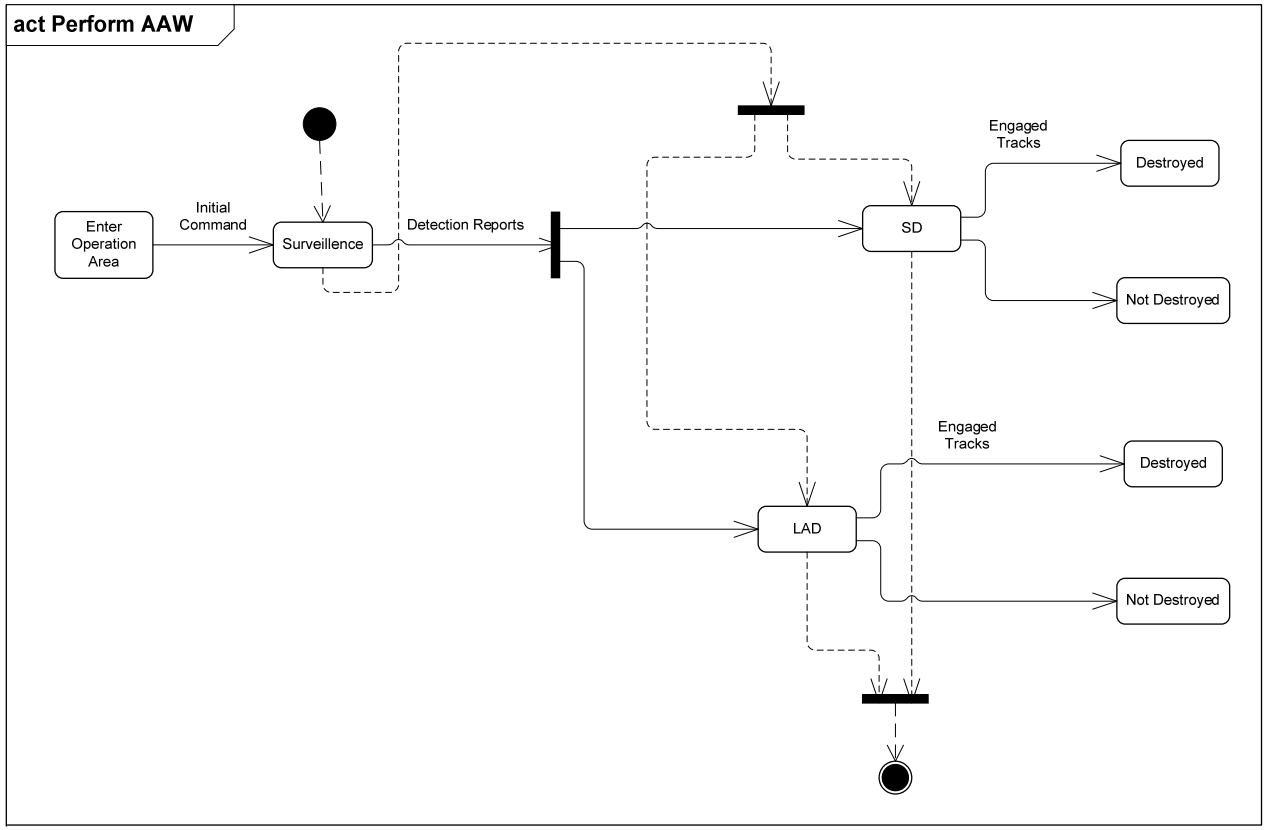
4.3.2.3 AAW Activity Diagram

Friedenthal provided the following description for activity in SysML:

> In SysML, an activity is a formalism for describing behavior that specifies the transformation of inputs to outputs through a controlled sequence of actions. The activity diagram is the primary representation for modeling flow-based behavior and is analogous to the functional flow diagram that has been widely used for modeling systems. (Friedenthal 2008)

The activity diagram represents the actions, the flow of inputs to outputs, and control (control is identified by the dashed lines in the diagram). An activity decomposes into a set of actions that describe how the activity executes and transforms its inputs to outputs. Each action can accept inputs and produce outputs. These outputs are called tokens and correspond to anything that flows (data, physical elements etc). (Friedenthal 2008) A particular type of action, referred to as a call action, can invoke other activities that can be further decomposed into additional actions.

The AAW Activity Diagrams were developed as a primary means of representing all activities of the AAW system. The AAW Activity Diagram is depicted in Figure 4-19. The remaining activities are provided in Appendix H (SysML Products). The activities are labeled in the box at the upper left corner of the frame. Each frame represents one activity of the AAW system. Figure 4-19 includes call actions that invoke other activities, such as the action Surveillance, which invokes the activities *Self Defense (SD) and Limited Area Defense (LAD).* Actions, such as *Initial Command*, have inputs and outputs that can accept tokens that represent units of information or matter. The AAW activity diagram is also a Functional Flow Block Diagram (FFBD).



This diagram depicts the call actions that make up the AAW action by invoking other activities, such as the action Detection Reports, which invokes the activities Self Defense (SD) and Limited Area Defense (LAD); data flow is left to right, control flow is top to bottom.

Figure 4-19: AAW Activity Diagram

4.3.2.4 AAW Enhanced Functional Flow Block Diagram (EFFBD)

In complex systems it is important to understand what the system does and how data will flow between functions. Models are developed to replicate the architecture design and assist understanding of complex system behavior. Models link requirements to functions and link functions to elements. The model described in this section is the behavior model (represented in SysML); it describes what the system will do and not how it will be done. The behavior model when executed provides time lines, time-dependent simulations and probabilistic calculations. (Oliver et al. 1997) The simulations can uncover conditions referred to as 'starvation', where inputs do not reach the various locations, and 'deadlock', where the system is queued waiting for the trigger. The simulation can expose design flaws which can be fixed in the next design iteration.

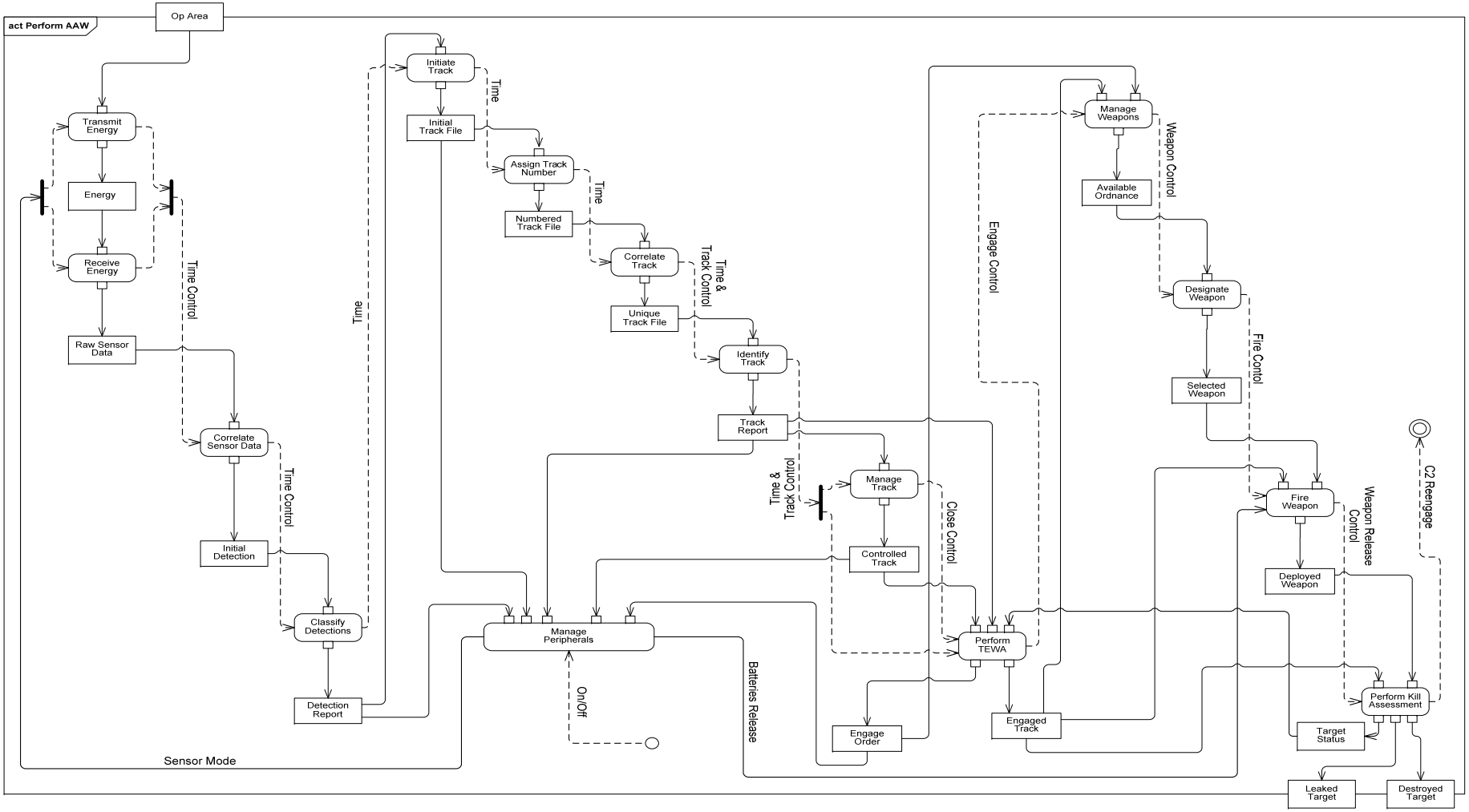
The EFFBD was developed using all of the activity views contained in Appendix H. These views have all the necessary information, control flows, data flows and processes. Figure 4-20, the EFFBD, is a cumulative view of the Detect, C2 and Engage processes. It is important to establish where and how items enter the EFFBD. Input and output are called items. Items can be external, entering from an outside source such as entering the Operation Area (OP Area). Items can be decomposed and referred to as time items; an item at the bottom of decomposition is called a discrete item. Discrete items are categorized as message, state, temporary or global items. (Buede 2000) For example entering the OP Area would be a time item and the trigger to enable the system. The discrete item would be the weapon function.

By using the EFFBD, simulations can be run that identify discreet time event controls, resources and characteristic links between systems. Behavior is distributed to the physical architecture where the links and interfaces pass between the functions. Lastly, the validation and verification simulation runs can be used to validate time sequences, identify lack of resources that cause time delay, and validate system and operational resources required to run the system. (Buede 2000)

When a message is sent to a function or control, it triggers the receiving function to execute as soon as the function is enabled by the control structure. (Buede 2000) It is here where details of functions and processes begin to be fully understood. Triggers are data items with control implications. A basic rule in EFFBD is that a function must be enabled and triggered prior to execution. Thus execution conditions occur by either control constructs or data triggers giving the designer detailed insight into the potential data or control flow issues. Modeling the EFFBD supports anomaly detection and provides a visual representation of system generated interrupts, alerts and notifications that are in the control system design. Most importantly, the EFFBD integrates the operation with the control system.

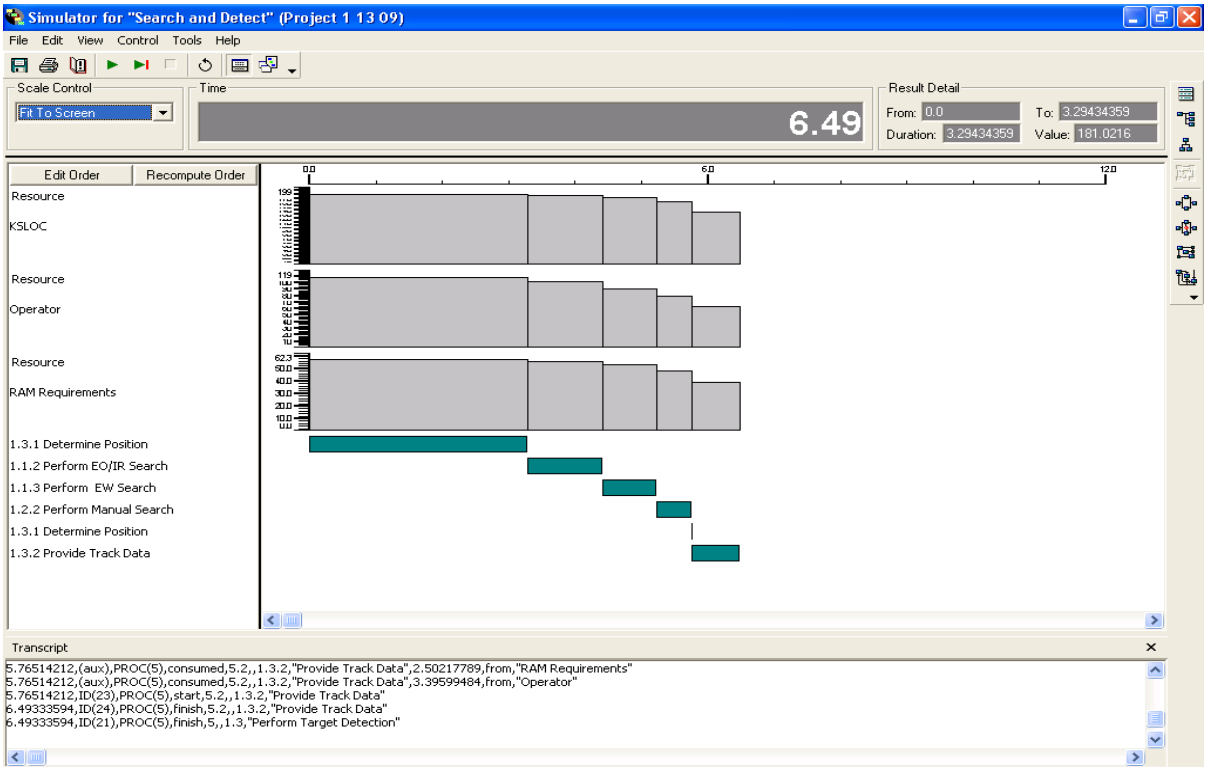
The EFFBD is in the same configuration as the FFBD but also includes looping, iteration, replication and control. The functions are the rectangular boxes that accept input and produce output. Control operators are the gatekeepers that determine the order of occurrence. There are primary functional flows that are used in the EFFBD: sequencing when there is a predecessor and successor, using "or"; concurrency when events occur simultaneously using 'and' operators; and iteration which repeats as a block sequence. Figure 4-20 conveys all the information identified in the behavior diagrams, but at a macro level. Additionally, the SysML standards use the "join" and "emerge nodes" to convey what the "and" and "or" operators did in previous models.

Once the EFFBD is complete it can be used in simulation. The AAW Architecture data was used to run a simulation in CORE to illustrate the interrelationship in the system. Figure 4-21 is a CORE simulation output that represents the Search and Detect function of the AAW model. The x-axis represents time and the y-axis represents availability of resources. In the event of a 'deadlock' or 'starvation' condition, the time sequence would stop, revealing an incomplete data flow. This type of modeling can be an important source of data to help decision makers and design engineers make timing/resource decisions.



This diagram allows for a visual representation of the system generated interrupts, alerts and notifications that are in the control system design and most importantly, it integrates the operation with the control system. For ease of viewing, the control lines (dashed lines) flow from top to bottom; they would normally flow left to right. The data lines are drawn as solid lines and convey data flow.

Figure 4-20: Enhanced Functional Flow Block Diagram

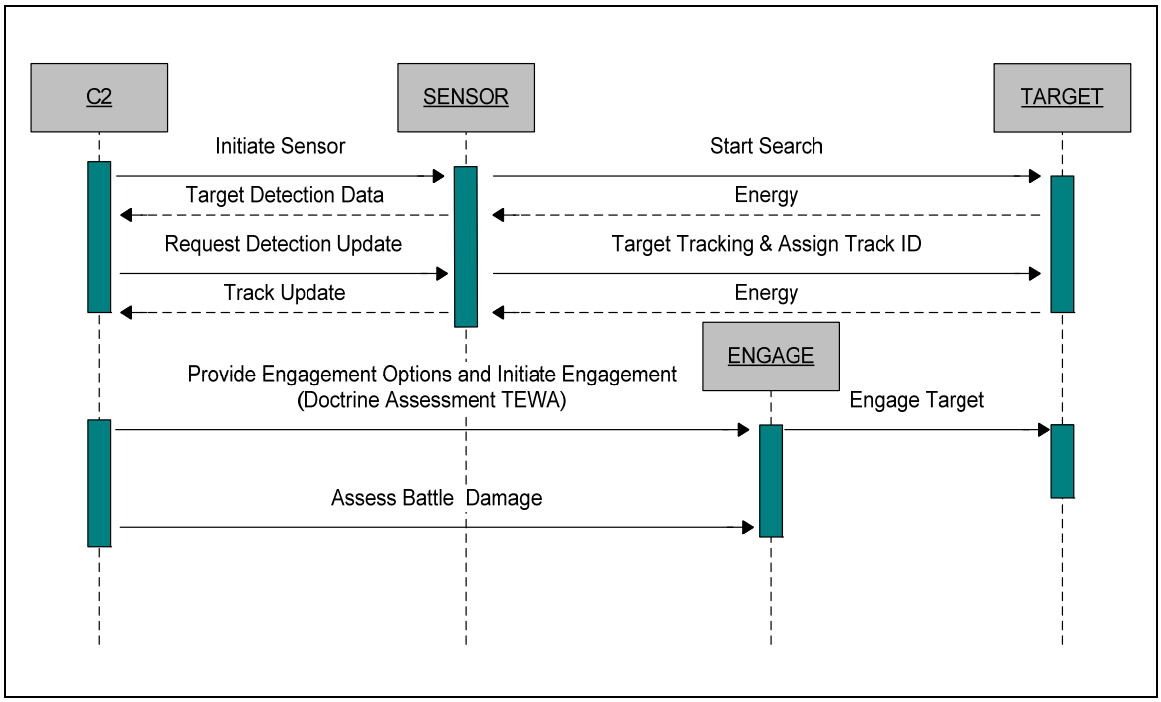


This CORE screenshot depicts the importance of resource utilization analysis. Had there been a 'deadlock' or 'starvation', the time sequence would have stopped, thus revealing an incomplete data flow. Note that Gray is the amount of resources available, while Green is the duration of the function.

Figure 4-21: EFFBD Simulation of Resource Utilization over Time

4.3.2.5 AAW Sequence Diagram

Figure 4-22, the AAW Sequence Diagram, was developed to illustrate the sequence of events and the interaction of components within the AAW System Architecture. The Figure 4-22 shows the higher level operational nodes, including Command and Control (C2), Sensor, Target, and Engage. Communication is represented as messages sent between the nodes. The AAW sequence diagram specifies interactions, which are illustrated by the send and receive messages between lifelines (vertical bars under nodes). The time sequence of the messages is indicated by the vertical placement of the messages on the diagram.

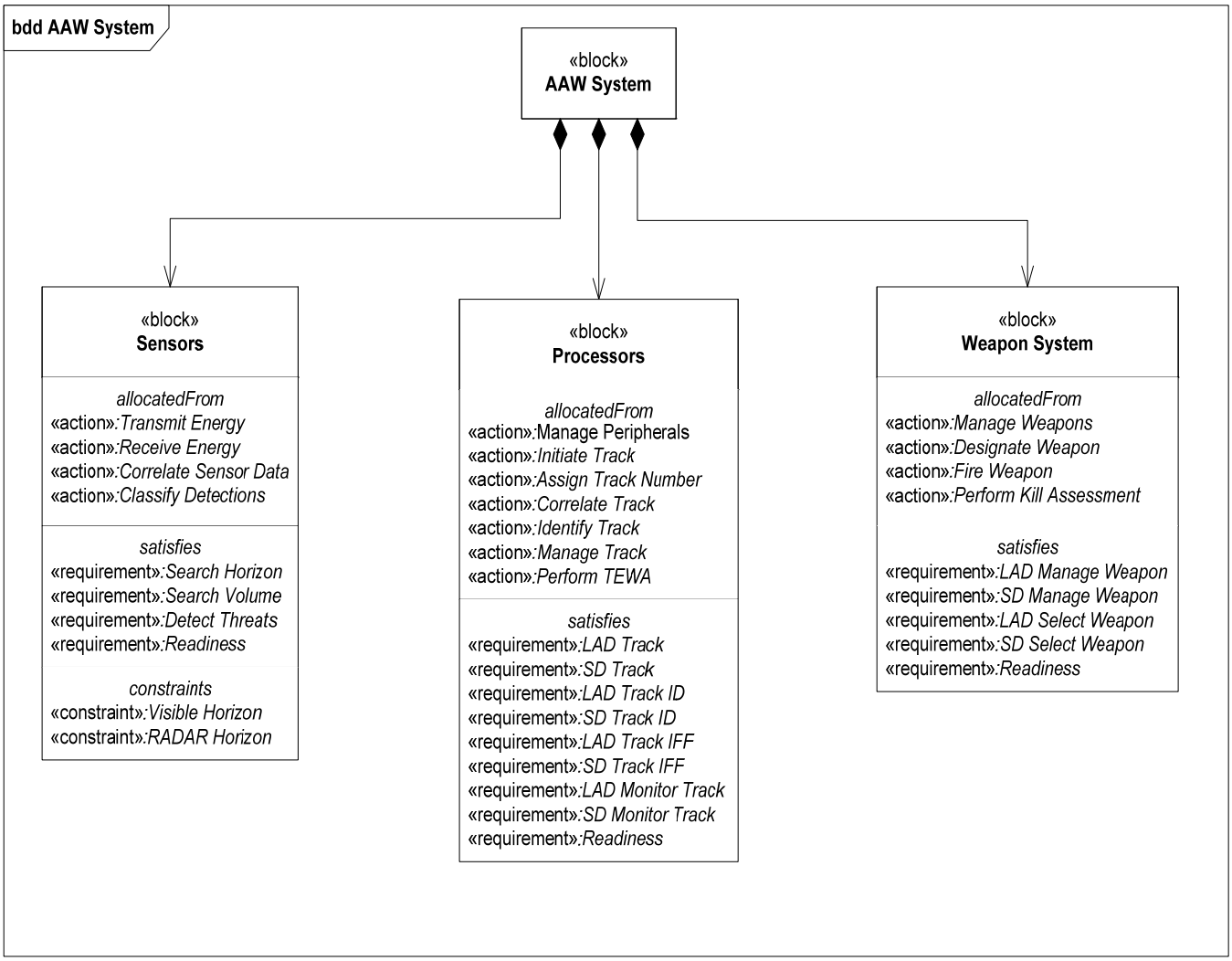


This diagram represents the behavior of the system in terms of a sequence of messages exchanged between elements to perform the AAW mission.

Figure 4-22: AAW Sequence Diagram

4.3.2.6 AAW Block Definition Diagram

Figure 4-23 is a basic structural element called a Block definition diagram which is used to build a disciplined-diagnostic view. The AAW system is represented by sensors, weapon systems and processors. The AAW SysML Block Definition Diagram (BDD) is the simplest way to describe the structure of the system. It is very similar to the class diagram in UML. A class diagram contains a name, identifies roles and responsibilities and provides constraints. A review of the Sensor part of the diagram reveals that it has all the requirements identified in section 4.2, addresses the activities in section 4.3 and would be suitable for verification using a timing model or the simulation in CORE.



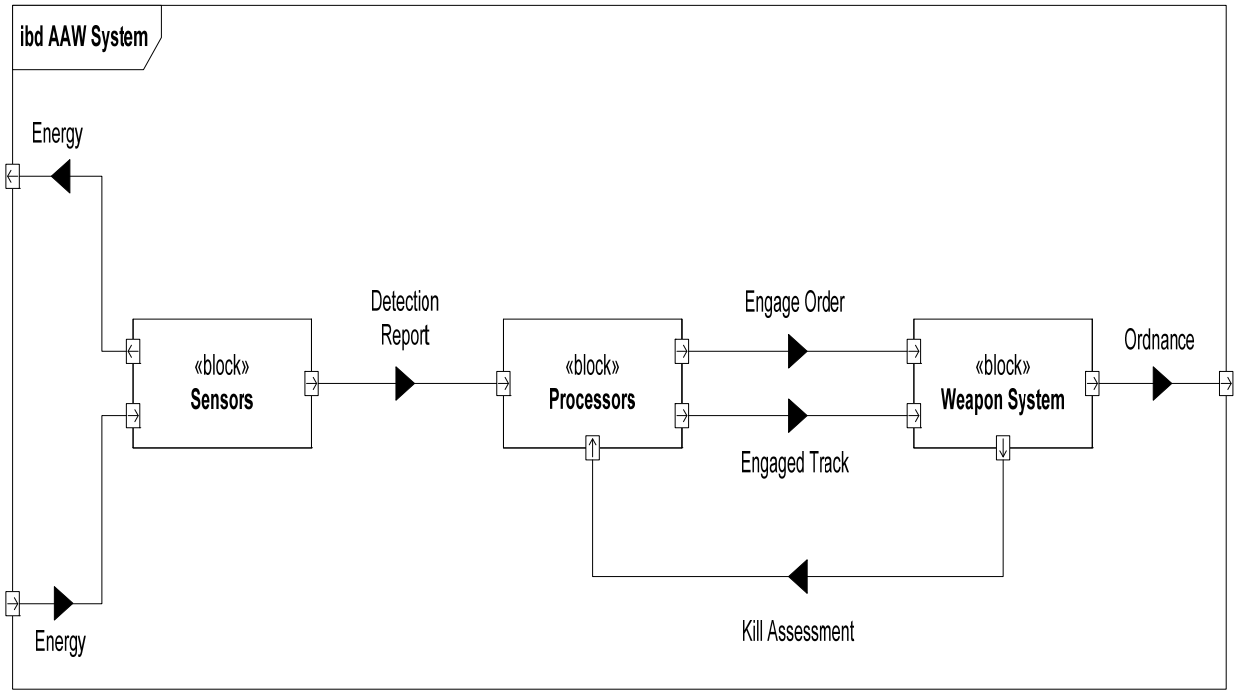
The AAW system is represented by sensors, weapon systems and processors; equivalent to a class diagram in UML.

Figure 4-23: AAW Block Diagram

4.3.2.7 AAW Internal Block Diagram

Figure 4-24, the SysML AAW Internal Block Diagram (IBD), refines the structural aspect of the model. The IBD is similar to the Composite Structure in UML. In the AAW IBD, properties are assembled to define how they collaborate to realize the behavior of the Block. The IBD allows the designer to refine the definition of the interaction between the usages of Blocks by defining ports. A part represents usage of another Block. Ports are parts available for connection from outside the owning Block. Ports are categorized according to type by the interfaces or Blocks that define what can be exchanged through them. Ports are connected using connectors that represent the use

of an association in the IBD. The Sensor block has inputs of energy and outputs of detection reports, maintaining the relationships throughout the model.



In the AAW IBD, properties are assembled to define how they collaborate to realize the behavior of the Block.

Figure 4-24: AAW Internal Block Diagram

4.3.2.8 SysML Artifacts

The SysML standard modeling language for systems engineering provides the design community a common language for documentation, communication and collaboration across development teams. The AAW views show that the behaviors, activities, data and control flows that were uncovered were appropriate for the level of functions required. The AAW views presented in this report include the:

- Context Diagram
- The Requirements Diagram
- Block Diagram
- Activity Diagram
- Sequence Diagram
- Block Definition Diagram (BDD)
- The Internal Block Diagram (IBD)

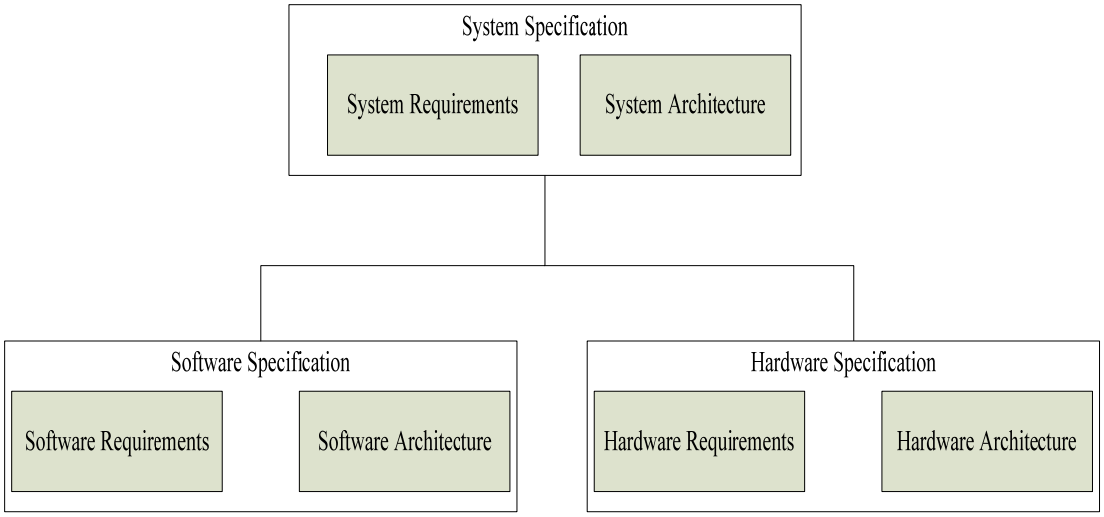
MBSE depends on information being captured and rendered in a meaningful way to the stakeholder/design community. The diagrams in this section document the structure and behavior for the AAW system architecture and meet the stakeholders requirements represented in the ConOps. The views and definitions provided support a detailed understanding of the products and how they satisfy customer needs.

4.4 AAW SOFTWARE ARCHITECTURE DEFINITION

This section defines the system as a whole, addresses partitioning of the system into Hardware/Software/People Ware (HW/SW/PW), and identifies how the system is structured. The results of addressing these areas of research are used to determine the right set of software architecture artifacts required to accomplish the AAW mission.

4.4.1 System Definition

The system defined in this report is an AAW system used to illustrate that MBSE can support an approach to meet the required Anti-Ship Cruise Missile P_{RA}. The ConOps (Appendix G) breaks the AAW problem into scenarios from which requirements have been captured in SysML artifacts. The program management plan raises questions about implementation of the systems engineering process for SPL development and the challenges it poses for the Navy. SPL designs are flexible, extensible, and feasible. The success of future software design relies on the ability of the software system to support reuse of source code. Oliver provides a collection of processes and information models. One of the approaches calls for views used to understand information for defining behaviors, functions, and data flow. These views are represented as both requirements and architecture. (Oliver et al. 1997) The Hatley-Pirbhai (HP) approach was chosen for the software architecture specification necessary to support the AAW architecture. Figure 4-25 represents the decomposition of the system specification; the HP method provides a process to determine relationships between requirements and architecture. (Hatley et al. 2000)



Decomposition of the system specification into software and hardware.

Figure 4-25: Software and Hardware System Specification Structure

The HP method was developed specifically for design, development, and decomposition of multi-disciplinary systems. The model allows for representations of physical interconnections, information, material, and energy movement between systems/components. Systems must operate in and interact with the real world, and they do so through different types of technology – electrical, mechanical, hydraulic, pneumatic, optical, radio, and many others. Moreover, these technologies interact with each other independently of software, in both intended and unintended ways. (Dorset House 2008)

The HP method is an extension of structured methods which include architecture models, information (data) models, and control flow models, and supports bottom-up as well as top-down design. A valid comparison for the HP method is a set of architectural prints for a house that includes and ties in all the appropriate views from which tradesman and craftsman could build. All models are tied together via a data dictionary. The models can be thought of as a tool kit to help designers understand a complex system so it can be designed and built. The models can also be used as a communications tool to help explain the system to others, including the customer and users. (Rickman 2000)

The Navy's traditional AAW architecture employs a Detect to Engage (DTE) approach for the control of sensors and the release of weapons. This functional formulation was used in the development of the Aegis and SSDS combat systems. The systems architecture was based on a shared memory concept. The limitations of this approach included dependency on legacy programming language (Aegis used CMS2 language – not many colleges teach this), limited scalability and extensibility (hardware choices were specific and unique) and high cost to modify (software and software retesting were projected to break the bank). The transformation to OA creates a foundation for future war fighting capabilities. Additionally, the open computing environment allows for modern programming languages (C++, JAVA etc), modular functional allocation, and libraries of common services (defined components).

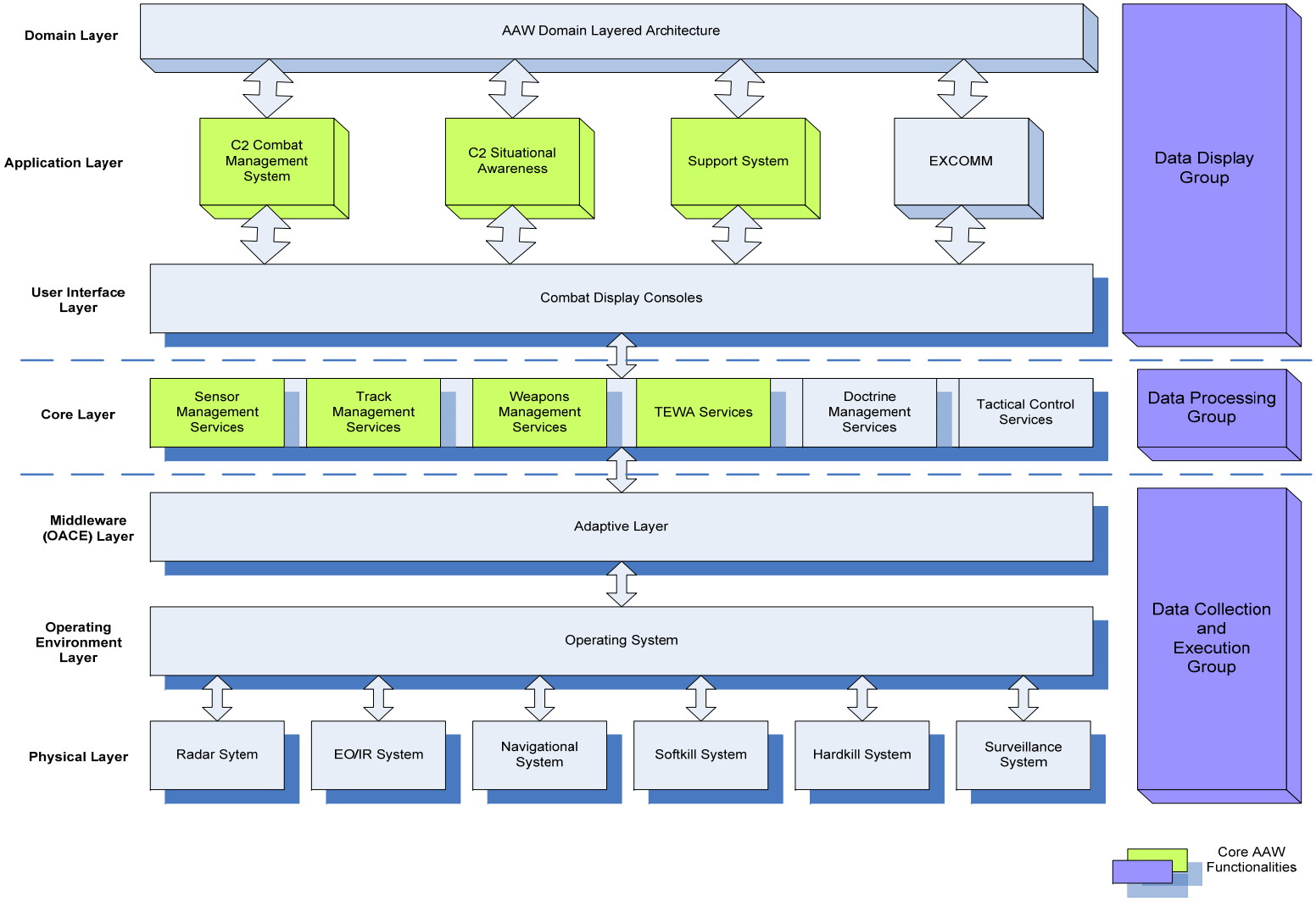
Both Aegis and SSDS have been re-architected to allow for the use of COTS products and the use of a layered communication approach. To minimize the cost of updating systems, a layered architecture is an appropriate choice. Layered architecture provides for modularized layers, which are relatively easy to update and are cheaper to adapt new technologies. An Analysis of Alternatives (AoA) (Appendix I) and a Software Integrated Product Team (IPT) Report (Appendix J) support a layered architecture approach for combat systems.

Layered software architecture is a prevalent approach used by client-server and web-based systems. It helps to structure applications that can be decomposed into *n* groups of subtasks in which each group is at a particular level of abstraction with well-defined interfaces. The *i*th layer could communicate with only the (*i*-1)th and the (*i*+1)th layer. Layered architecture is widely used in most web-based systems where performance is a critical factor. (Sharmal et al. 2005)

Rather than focus on point-to-point functional and physical interfaces, the layered architecture concept recognizes the need for continuous and concurrent combat system

data processing. What are important are the information being produced and the responsibility for its creation and dissemination. For example, the core layer consists of several components for the system's functions (Figure 4-26). The AAW core layer includes sensor management services, track management services, weapons management services, and TEWA services. Functions are assigned unique responsibilities; Sensor Management produces unique sensor system information and broadcasts this information as messages throughout the system. Each function has access to its appropriate layer from which to perform its responsibility.

In the AAW domain, information is generated upon change from a development view; the functional independence provided by a layered architecture removes the complications of sequential function coordination and communication. The communication hierarchy is developed from a commercial standard from which information can be shared. From this form of functional independence, it is easy to develop physical independence in the form of distributed processor architectures. A review of legacy systems, such as Aegis and SSDS, indicates that this approach allows for massive amounts of computing power to be focused where and when it is required. As systems grow, the addition of new system functions can be integrated into the system information (new processors that follow proper protocols can be added to the system to add capability). (Johns Hopkins APL 2001)



The layered architecture concept recognizes the need for continuous and concurrent nature of combat system data processing.

Figure 4-26: Proposed AAW Layered Architecture

Data entities and attributes are assigned to particular and unique functions within the layered architecture. This allows for clear separation of the contributions from each function. System information has a distinct source, directly related to a specific portion of software and traceable to specific message output to the data distribution architecture. (Johns Hopkins APL 2001) Benefits of using layered architecture with combat systems include low-latency, high-throughput, high-scalability, deterministic responses, and minimal consumption of network, processor, and memory resources.

Predictable distribution of data with minimal overhead is of primary concern to real-time applications (AAW and the requirements for ship self-defense apply). Since resources are finite, it is important to be able to determine available resources and implement policies that allow middleware to apply the resources to the most critical requirements. There are new standards on the market that address middleware and time data distribution requirements. (OMG 2008) To maintain accuracy in data calculations, data must be time stamped when it is either measured or created. This can be used to indicate valid or accurate time which can be used in later processing. Latency can be thought of as the delay from data measurement to the subsequent processing of the data. The ability to detect and form a track in rapid succession is where latency really becomes a concern. The AAW system described herein has about six seconds from which to gain firm track and engage with weapons. The AoA details quality attributes and uses measures and weights to evaluate architecture requirements.

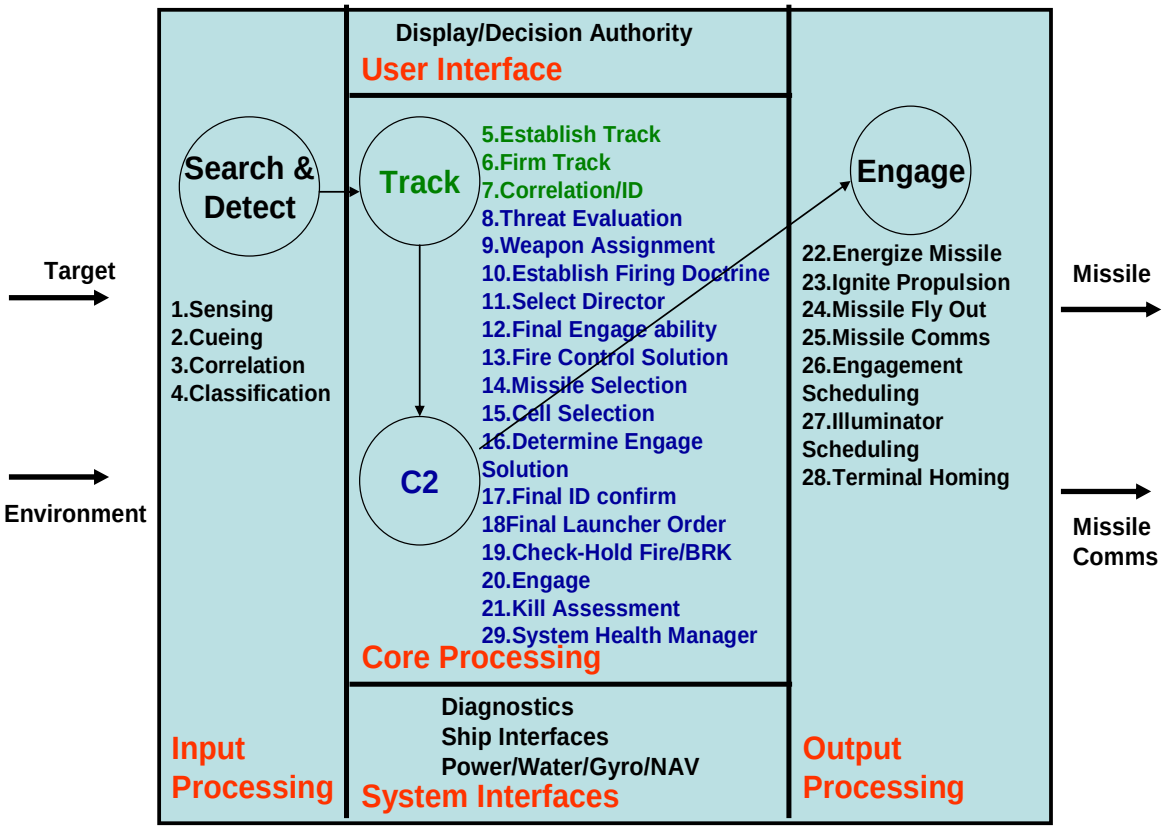
A layered architecture supports an OA environment that takes advantage of COTS protocols and time-to-market of new products. Developing software for every change can be expensive and time-consuming – a faster and cheaper software development approach is necessary. SPL refers to a software engineering method and techniques for creating a portfolio of similar software systems from a shared set of software assets using a common means of production. (Krueger 2008) When developing software for complex systems, identifying and/or creating commonality can be difficult. Layered architecture facilitates commonality with its well-defined interface requirements that make modularity

possible. Layered architecture places software applications in layers by their sub-level functions such as data handling. For software applications in a layer to communicate with others in different layers, all software applications in the layer need to meet the same interface requirement. That is, the same lines of code that satisfy interface requirements exist in each software application to communicate with other layers. For example, sensor management service and track management service in the core layer must have the same lines of code to communicate with the middleware layer and the user interface layer. The approach of assigning the role of meeting the interface requirements to the common lines of code is the key to designing the AAW software system for a SPL.

4.4.1.1 AAW Architecture Discussion

The Context Diagram in Figure 4-3 illustrates a shipboard AAW system. There are many relationships with other systems required in order to meet Navy AAW requirements. To constrain these relationships, the AAW system is viewed as a black box with a limited view of inputs/outputs and interfaces.

Figure 4-27 identifies AAW functions using the HP architecture template. For study purposes, the context of this system has been reduced and the AAW system can be viewed as stand-alone, i.e. no outside interface is required to reach a solution for P_{RA}. The boundaries and limitations illustrated in the ConOps are meant to limit the answer of the question of P_{RA} (and supportability requirements). The intent is not to decompose the problem down to specific hardware and software components (lines of code). The AAW problem is viewed as an operational challenge, and the solution will be an academic view of the logic required. Where appropriate, HW/SW/PW will be used to illustrate what could be accomplished if further decomposition was accomplished.



The context of this system has been reduced and the AAW system can be viewed as a stand alone, i.e., no outside interface is required to reach a solution for the probability of raid annihilation (P_{RA}).

Figure 4-27: AAW Architecture Model (Hatley-Pirbhai 2008)

A thorough understanding of AAW behaviors is required to support DTE, and the Navy has amassed a large body of research to determine activities/behaviors required for AAW systems. The Navy Tactical Data List (3.0) was developed to understand what behaviors are required for various mission requirements. A review of this document and an academic review of Detect-Control-Engage functions allowed for the development of the AAW HP Architectural Model Template. The functions are derived from the SE3123 Time Range Information. (Green 2008b)

4.4.2 Partitioning of HW/SW/PW

Systems engineering has changed significantly over the last decade. In today's system design there is a need to build a faster, cheaper, and more flexible system that can take advantage of COTS reuse potential. In other words, the system must be flexible enough to handle requirements changes within a given system and be adaptable for OA requirements. The first step toward satisfying an OA combat system is decoupling computing hardware from software and people ware. The requirements of HW/SW/PW partitioning need to reflect the layered architecture decision. If existing components, such as COTS, Government-Off-The-Shelf (GOTS) and reuse items, are going to be part of the system design, these should be considered as given components and included in the architecture. If a requirement for a modular part can be satisfied by an existing component, the function must be split in the requirements model so that the requirements not met by existing components are clearly identified. For requirements that cannot be allocated to existing components, those modules can follow directly from the requirements model and must be able to satisfy the minimization of coupling and maximization of functional cohesion.

Partitioning is the division of an element in a model into its subsidiary or child elements. (Hatley et al. 2000) As applied to this project, the AAW architecture model is the top-level element and the intent is for it to remain at a high level of abstraction. The model is made up of HW/SW/PW, which are the subsidiary or child elements, and will only be discussed to showcase what could be used to partition this model. It would take several iterations of this approach (HP views) to arrive at all the possible HW/SW/PW configurations that could be used to meet the AAW system requirements.

The AAW system requires a sensor to detect a threat in an assumed environment at a specific time interval, a reaction time, and an engage system. Major C2 tasks include weapon and sensor systems control, tactical picture compilation, situation interpretation and threat evaluation, weapon selection and engagement monitoring, and evaluation. There are also supportability requirements (A_O, MTBF, MTTR, etc) that must be met to

achieve the overall mission requirement (P_{RA}). The supportability requirements drive the need for system health monitoring, fault detection/isolation, system reconfiguration, and maintenance, all of which must be considered in the partitioning of the system architecture down to the HW/SW/PW elements.

The C2 process in the AAW system requires a highly dynamic flow of information and decision making. In a highly dynamic scenario, with a large number of contacts, handling large amounts of data can quickly overwhelm human capabilities. Similarly, the system supportability process also requires a highly dynamic flow of information and decision making to support system health monitoring, fault detection, and system reconfiguration decisions. These system functions must all be partitioned and allocated to the HW/SW/PW sub-elements of the AAW architecture.

Both autonomous and supportive behaviors are required in a Weapon Engagement Manager (WEM). In an AAW system, the WEM can be considered part of the algorithm or process required to meet the functionality of the system. It continually senses and reacts to occurrences, including events that take place in the combat environment and actions or activities involving WEM intervention, and responds to requests for support or commands from an operator. A variety of information is required; including knowledge about the behavior of entities in the combat environment, engagement doctrine (tactics), and conditions for the viability of specific actions as well as effects. The WEM continually adapts its reasoning to take account of an evolving tactical picture (situation).

A first-level analysis of the AAW problem identifies the need for a very quick response action, on the order of seconds from detect thru engage, based upon the speed of the target and the distance at which the sensor detects it. Hardware decomposition for the sensor would result from choosing an RF/EO/IR sensor available 24/7 and operating in a marine environment. Further first-order analysis leads to a very quick processing time and a human in the loop that could make the decision to fire (the expectation is that the

system can provide the operator an engagement order). Once the system recommends a release of energy, an operator can respond and fire a weapon. There can be lengthy discussion on where and when a human in the loop interacts with the system; the premise in the AAW system is that humans are there to control and for the release of energy. Hence, a large percentage of the system is logic/control based.

The system health management aspects of the system also require autonomous and supportive behaviors. System health management includes functions like system health monitoring, fault detection, and system reconfiguration decisions, all of which must continually sense and react to occurrences. In the context of supportability, the occurrences are typically faults or equipment performance degradation within the system, and reactions are the decisions to correct the occurrences. These reactions can require both autonomous and supportive behaviors. Autonomously, the system must constantly monitor the health of the system and react quickly to the detection of a fault or performance degradation. Further autonomous behaviors include making decisions to either allow continued operation with degraded performance, rapidly reconfiguring to redundant equipment, or requiring supportive behaviors from a human command.

System supportive behaviors include functions like further fault isolation by responding to operator commands, reconfiguration based on operator commands and system reboots, etc. Other supportive behaviors could include the operator allowing remote access to the system by an off-board technical support center that could further react to the occurrence. The bottom line is that the supportability aspects of the system must be integrated into the system architecture and, just as with the performance elements, must be partitioned down to the HW/SW/PW sub-elements of the system.

4.4.3 Hatley-Pirbhai Development Process

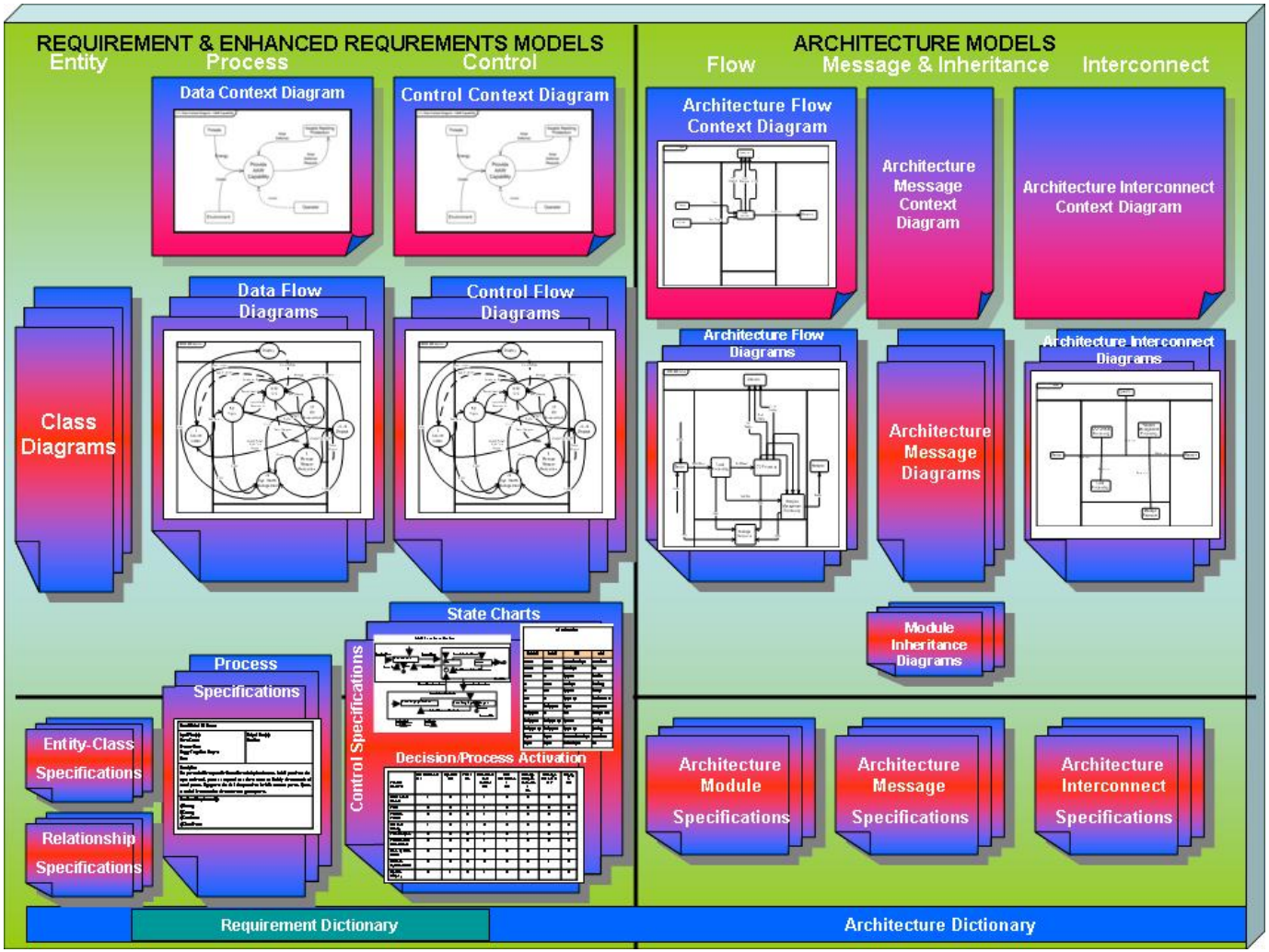
The HP methodology is a graphical method used to identify an AAW system's functions and its physical partitioning. The HP structure methods separate the AAW system specification into two models: requirements (data model & control model) and architecture. The requirements model describes what the AAW system should do, while the architecture model describes the entities in the AAW system and the allocation of requirements to hardware, software, and people. The HP methodology was chosen for the AAW system architecture due to its strong emphasis on software reuse as well as its bottom-up and top-design approach to system specification. The HP descriptive models (artifacts) can be compared to the DoDAF artifacts as outlined in Appendix K.

A set of Data Flow Diagrams (DFDs), and control and process specifications provide the basis of the AAW system requirement model. The model captures the core requirements of the AAW system, which are technology agnostic (independent of specific technology). The intent is to determine system behavior to arrive at a specification. The enhanced requirements model adds to the core model requirements for input, output, user interface, and maintenance/self test processing.

The architecture model was derived from Navy AAW activities and from basic Detect-Control-Engage functions. From this view and further decomposition, the Architecture Context Diagram (ACD), Architecture Flow Diagram (AFD), and Architecture Interconnection Diagram (AID) were developed. The ACD depicts the physical boundaries of the AAW system. The AFD shows the physical entities, called modules, in the AAW system. The AID illustrates the interconnections of modules in the AAW system.

The HP Model Development Process, depicted in Figure 4-28, consists of the following (Haggerty 2000):

1. Development of system core requirements, including context diagram, DFDs, EDFDs, control specs, and process specs
2. Enhancement of requirements
3. Determination of the architecture modules for the system
4. Allocated processes and control specs from the enhanced requirements to architecture modules using super bubbles
5. Drawing the core requirements for each module in the system
6. Enhancing the module core requirements as necessary for inter-module communication, new user interfaces, and maintenance/self-test
7. Drawing the flows and interconnections on the AFD and AID based on the boundaries of the enhanced module requirements.



The HP structure method separates the AAW system specification into two models: requirements (data model & control model) and architecture. The requirements model describes what the AAW system should do, while the architecture model describes the entities in the AAW system and the allocation of requirements to hardware, software and people.

Figure 4-28: Hatley-Pirbhai System Development Model

4.4.3.1 System Requirement Model

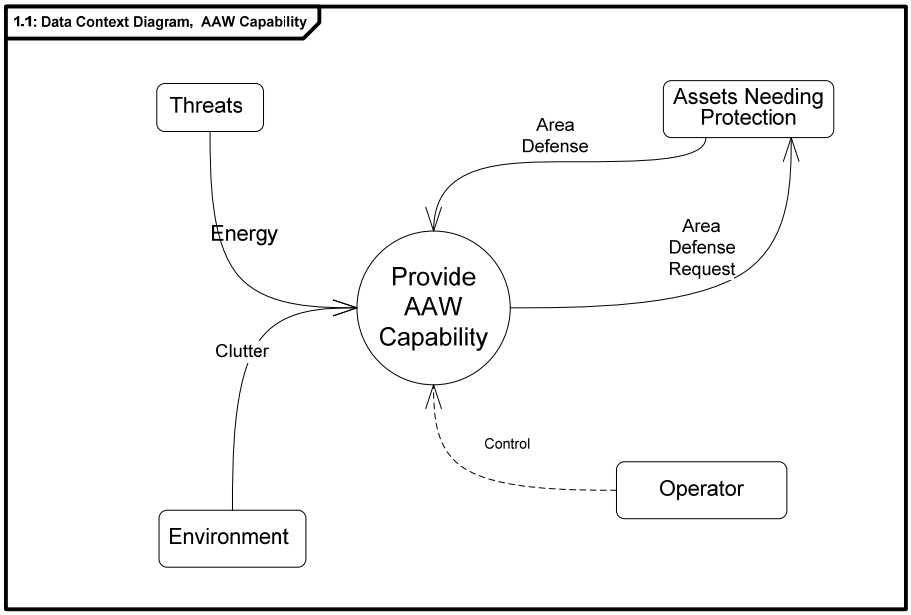
The Logical Architecture is illustrated as a single function to show the external inputs and outputs and is represented by the System Context Diagram. The system context diagram, Figure 4-29, was developed using the ConOps document. Once the boundaries were identified, the next step was the decomposition of the AAW system and the creation of the Data Flow Diagram (DFD) to demonstrate that the next level of process bubbles in the system could be developed. These bubbles were then further decomposed into lower levels. At the lowest level of this decomposition are the actual Process Specifications (PSPECs). The Logical Architecture was then mapped to a Physical Architecture that assigns the logical processes to physical subsystems and service packages. This mapping is documented in the Physical Architecture Data Dictionary.

The AAW Data Context Diagram (DCD) illustrates the high-level interactions of AAW as a stand-alone system with the various external and environmental inputs/outputs. The diagram is a derivation of the SysML requirement Context Diagram. The constraints and limitations of the AAW system are outlined in the ConOps document and further refined by the system requirements. The diagram represents all external entities that may interact with the AAW system.

Threats and atmospheric data are captured from the environment and fed to the AAW System. Atmospheric conditions can affect the performance of the AAW system and can be confused with threat data; clutter tracks can mask real tracks. Other assets within the limited area of influence require protection. The AAW system interacts with the operator via display consoles providing assessment and control of area defense.

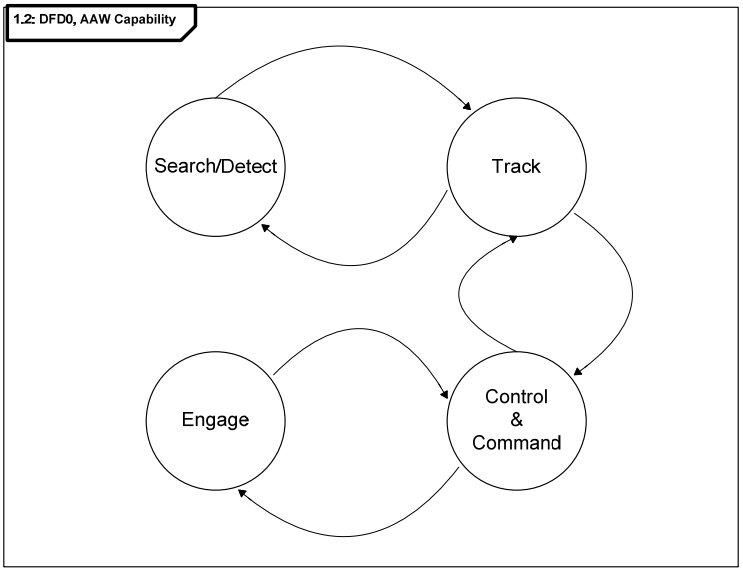
The Data Flow Diagram Level 0 (DFD0) (Figure 4-30) illustrates the functional architecture of the AAW system and shows the system as processes. The DFD is designed to show how a system is divided into smaller portions and to highlight the flow of data between those parts. This context-level DFD shows more detail of the system

being modeled. It provides a structural visualization of the main functionalities of the AAW system software architecture and how the AAW system is implemented. The diagram shows core process data flows within the functions of AAW, which include search/detect, track, C2, and engage.



Limited Area Context Based

Figure 4-29: AAW Context Diagram



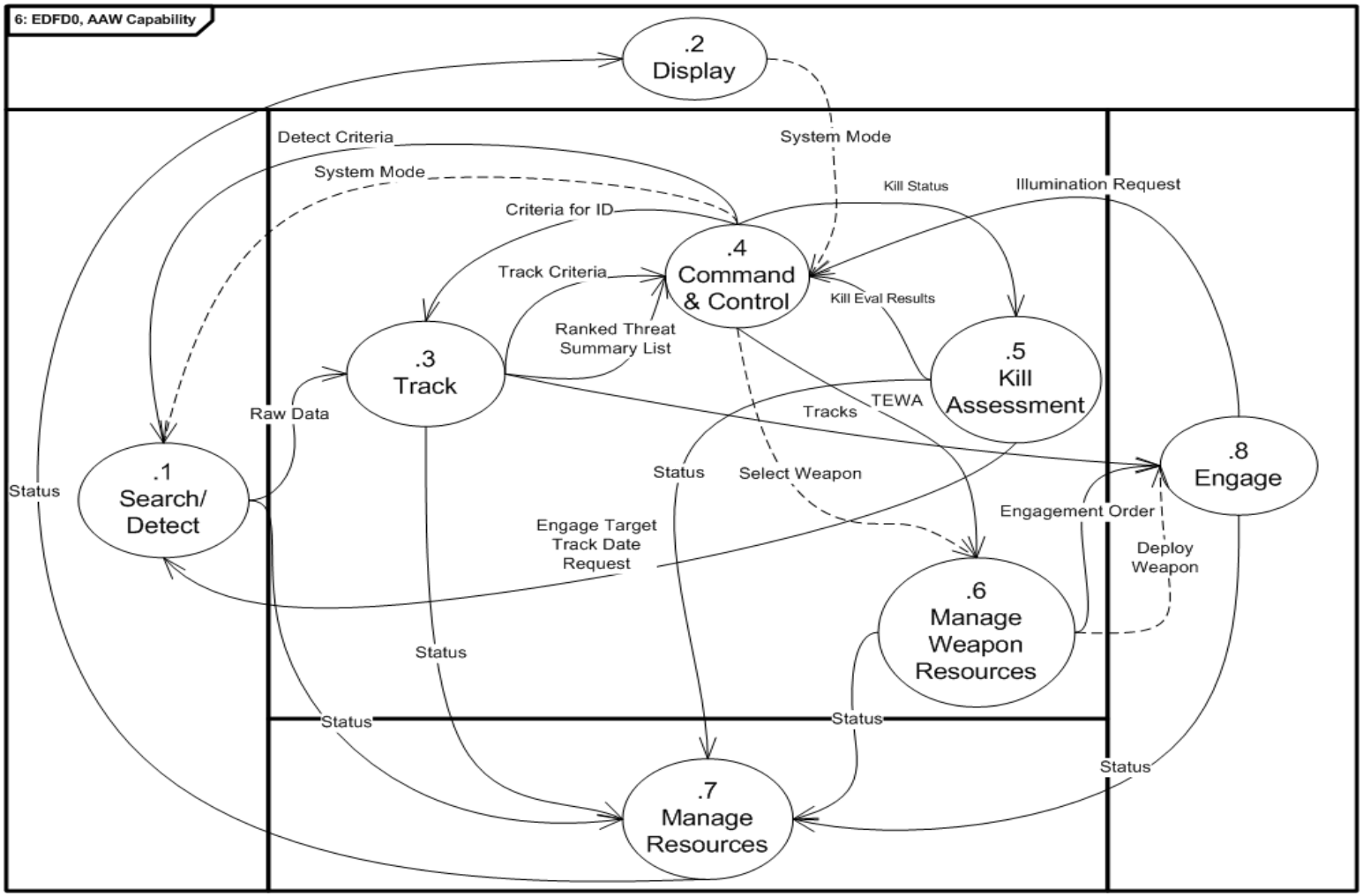
The DFD is designed to show how a system is divided into smaller portions and to highlight the flow of data between those parts; initial data flow diagram.

Figure 4-30: AAW Capability DFD0

The Enhanced Data Flow Diagram Level 0 (EDFD0) applies the architecture template to the AAW processing functions. The template adds input/output, human machine interface, and support process activities to the requirement functions presented within the DFD0 diagram for a complete AAW capability. The four enhanced regions of the template contain processes, behaviors, and items for the physical interface with the system environment. (Hatley and Pirbhai 2000) The diagram completes the system's enhanced requirement model of the HP model development process. The template is broken up into five separate and distinct regions of processing.

For example, search/detect is a logical input to Track processing and is found in the input processing section of the template. The intent is not to arrive at specific hardware at this level of decomposition, but rather to show that Search and Detect are main functions. From the name of the process, there is no way to understand how the AAW system will handle all the detections and how they will be distributed throughout the system. Search/Detect is comprised of many sub–functions (Figure 4-31). What, how, and when must be answered, and to meet the requirements described in the ConOps, the system must be able to distinguish if and when it detects a target and if it can be distinguished from other objects.

It is via this process that sub-functions are discovered; specifically, sensing, correlation, and classification are required before determining a valid detection has been made. An object could be anything in space that provides energy back to the system. The energy has to be correlated to other energy (contacts) and further classified to avoid confusion with other contacts. The same process of discovery would be used to explore all the functions in Figure 4-30. The inputs/outputs/control/data flows must be evaluated to describe performance and control. The descriptions can then be used to build the Search and Detect specification. The functions and behaviors of search and Detect require further breakdown and extensive study to allocate to a correct sensor for this function.



Process applied to HP template

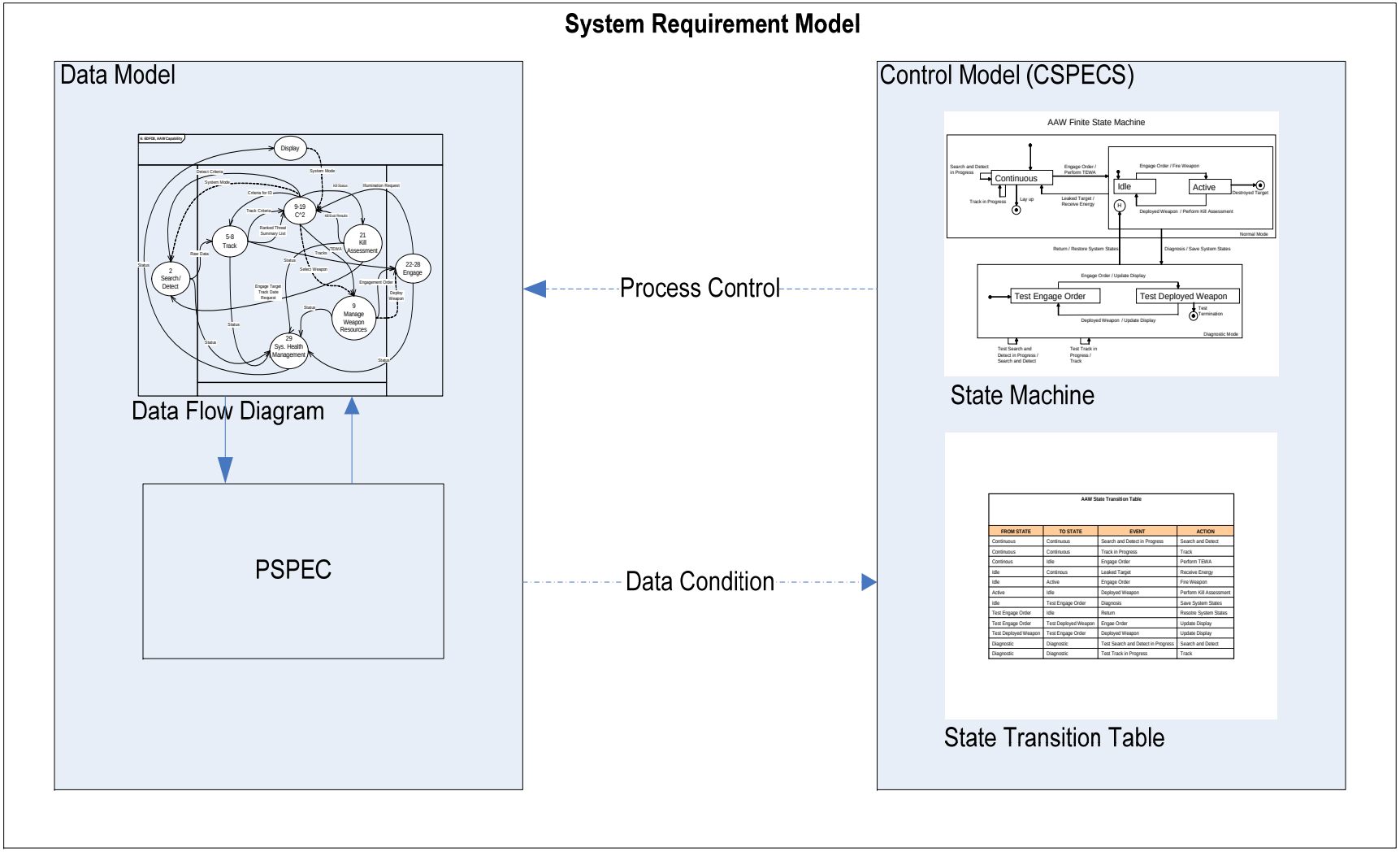
Figure 4-31: AAW Capability EDFD0

4.4.3.2 Control Specifications

The completion of the AAW System Requirement Model entails the development of the AAW System Control Model. The Control Model captures the major operating modes of the AAW System. The behavioral aspects of the AAW System are captured in the Finite State Machine (FSM). It interacts with the Data Model through control flows and by enabling/disabling processes from the DFD of the Data Model.

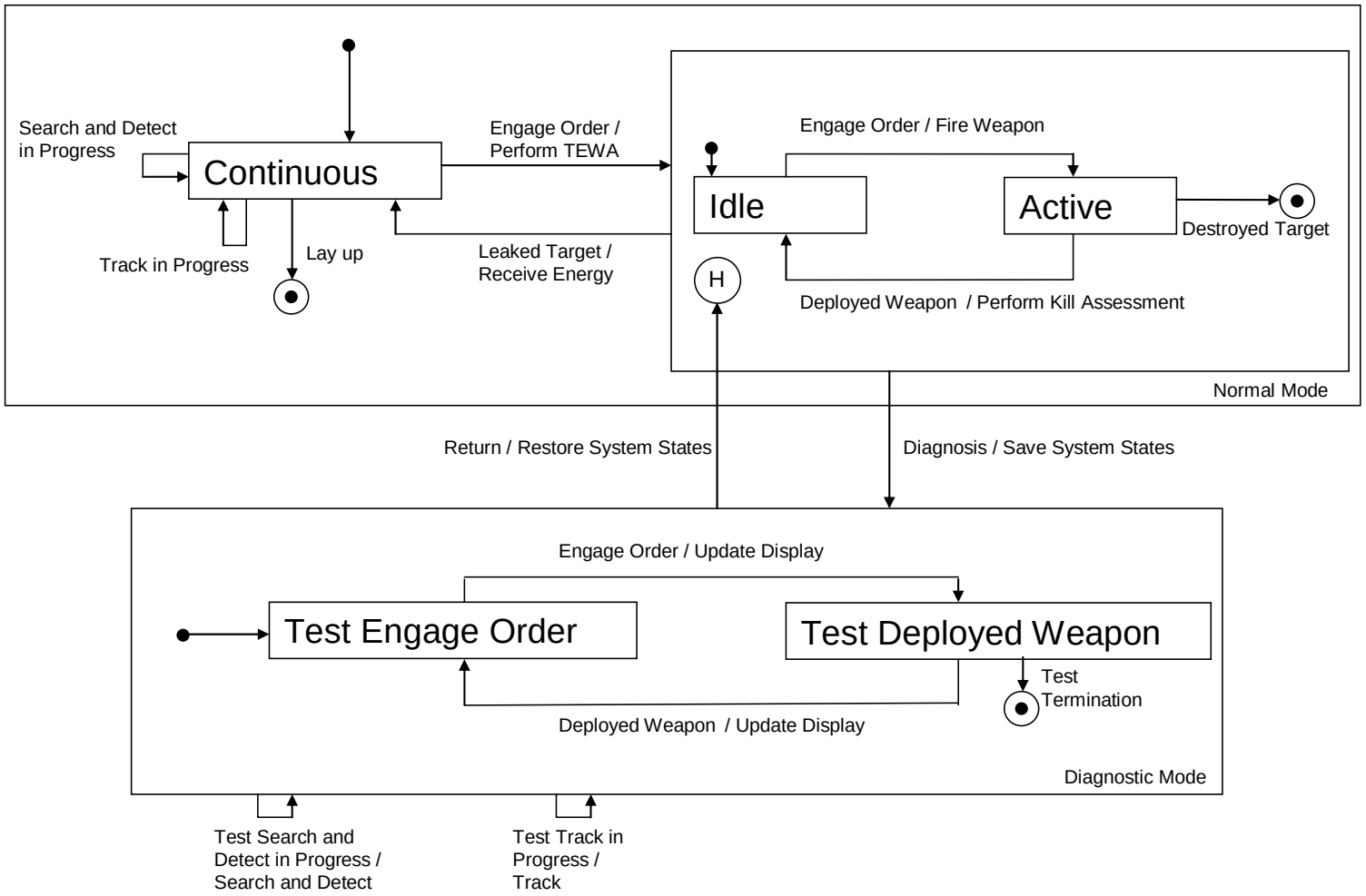
The Control Specifications (CSPECs) are used to indicate how the software behaves when an event or control signal is activated and which corresponding processes of the Data Model are invoked. It represents the behavior of the AAW System in two ways. The CSPECs contains a state transition table that is a sequential specification of AAW system behavior. It also contains a process activation table mapping the Data Model processes to the AAW System FSM control state.

Figure 4-33 provides a control view of the Enhanced Data Flow Diagram Level 0 and is a CSPECs artifact. It highlights the behaviors of the EDFD0, providing better fidelity of AAW architecture behavioral functionalities. The AAW System FSM consists of two modes: normal and diagnostic. In normal mode, there are three states: continuous, idle, and active. The event trigger causes the AAW system to transition from state to state. For example, in the normally idle state, an engagement request from command and control triggers the FSM to transition to an active state. In this state, AAW deploys a weapon, then transitions back to idle upon a trigger event of perform kill assessment. It transitions back to active only if kill assessment determines the need for a new engagement action. In the continuous state, in addition to search and detect, tracking functions are also performed with criteria set by command and control in a continuous manner. The trigger event to the idle state is an engagement order from command and control. The diagnostic mode is a subset FSM of the AAW system. It consists of test engage order and test deployed weapon. The diagnostic mode is a function of the shipboard training and system health assessment.



Data and Control Model Relationship

Figure 4-32: System Requirement Model



AAW Behavior Model

Figure 4-33: AAW Finite State Machine

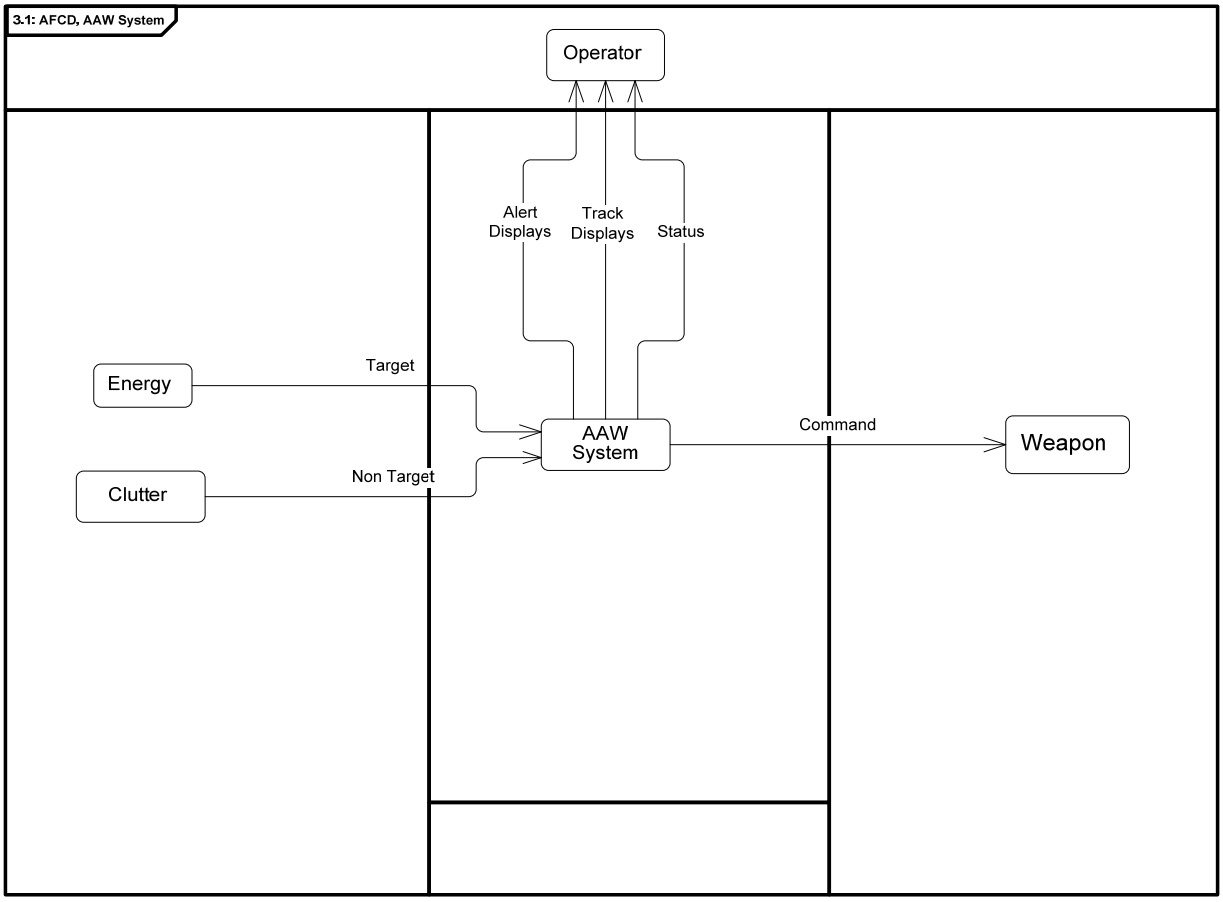
The State Transition Table (STT) (Table 4-2) depicts all the AAW system FSM changes of state. The change of state is triggered by an event derived from control flow. The control flow can originate from internal or external sources. It can be triggered by time, such as a periodic scan for threat in a defined perimeter. It can also initiate from internal processes such as command and control requesting an engagement order. The action column is the result of the trigger event causing the state machine to create a control signal to activate a process within the EDFD. The process activator enables the corresponding processes to execute the required functions.

Table 4-2: AAW State Transition Table

FSM state transition events and action (control)

AAW State Transition Table			
FROM STATE	**TO STATE**	**EVENT**	**ACTION**
Continuous	Continuous	Search and Detect in Progress	Search and Detect
Continuous	Continuous	Track in Progress	Track
Continous	Idle	Engage Order	Perform TEWA
Idle	Continous	Leaked Target	Receive Energy
Idle	Active	Engage Order	Fire Weapon
Active	Idle	Deployed Weapon	Perform Kill Assessment
Idle	Test Engage Order	Diagnosis	Save System States
Test Engage Order	Idle	Return	Resotre System States
Test Engage Order	Test Deployed Weapon	Engae Order	Update Display
Test Deployed Weapon	Test Engage Order	Deployed Weapon	Update Display
Diagnostic	Diagnostic	Test Search and Detect in Progress	Search and Detect
Diagnostic	Diagnostic	Test Track in Progress	Track

The Architecture Flow Context Diagram (AFCD) (Figure 4-34) shows the high-level flows and interconnects linking the various external and environmental inputs/outputs to the AAW system. The AFCD repackages the context diagram and keeps the input and output relationships. The diagram illustrates the data flow among the entities of the input processing (threat, atmospheric conditions), user interface (operator), output processing (weapon), and AAW system.



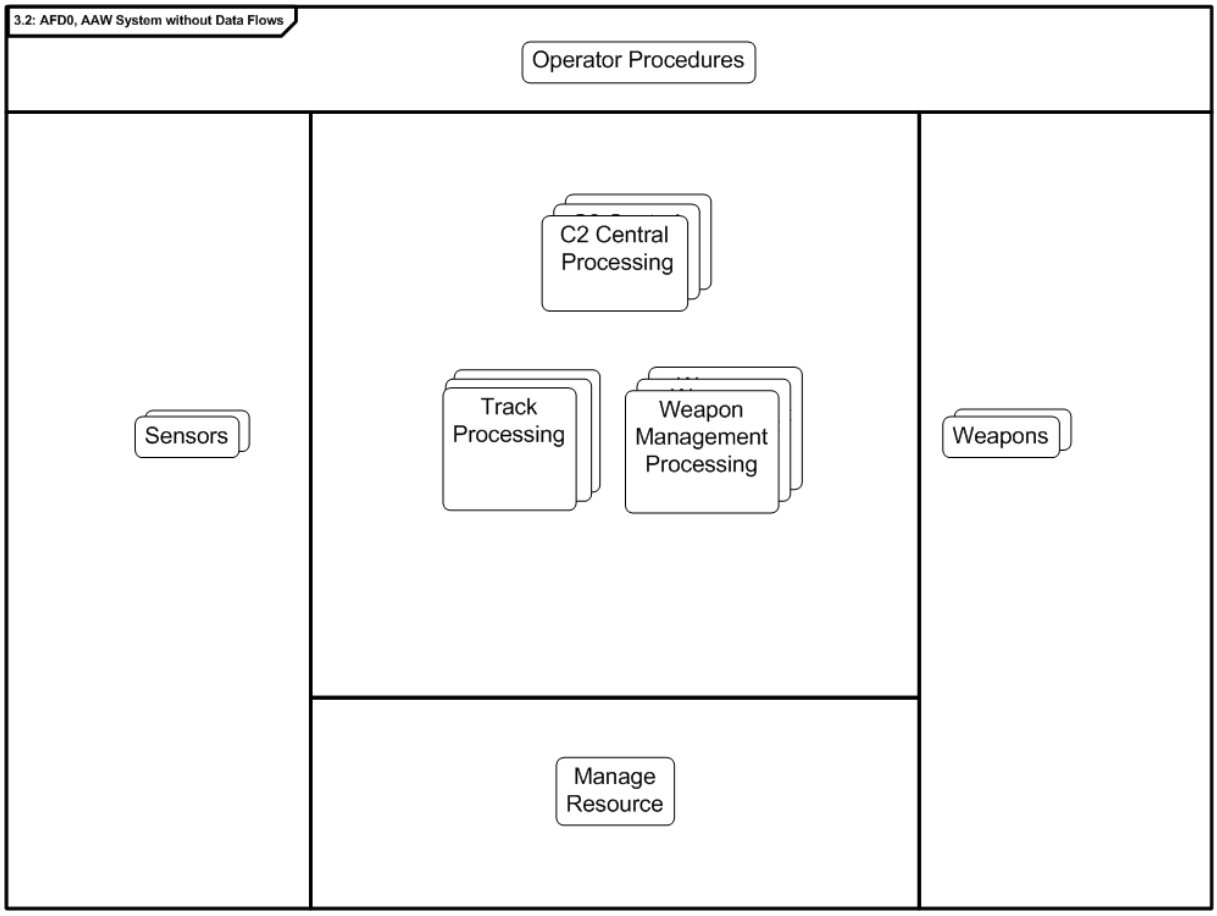
AAW High Level Architecture Context Diagram

Figure 4-34: AAW AFCD

The Architecture Flow (without data flows) Diagram Level 0 (AFD0) in Figure 4-35 establishes the derived software architecture modules from the system-enhanced requirements spec of the EDFD. This diagram focuses on core processing. In the context of the study problem track, C2 and weapons management require complex processing to maintain a tactical picture in the timelines specified given that the system will operate in

a marine environment. This results in a plethora of combinations of environmental conditions including friendly and enemy ships and aircraft and land masses. Given the magnitude of the problem, a separate study could be undertaken to determine if it makes sense to combine these processes. For this project, however, the intent was to maintain a relationship with the detect-control-engage paradigm.

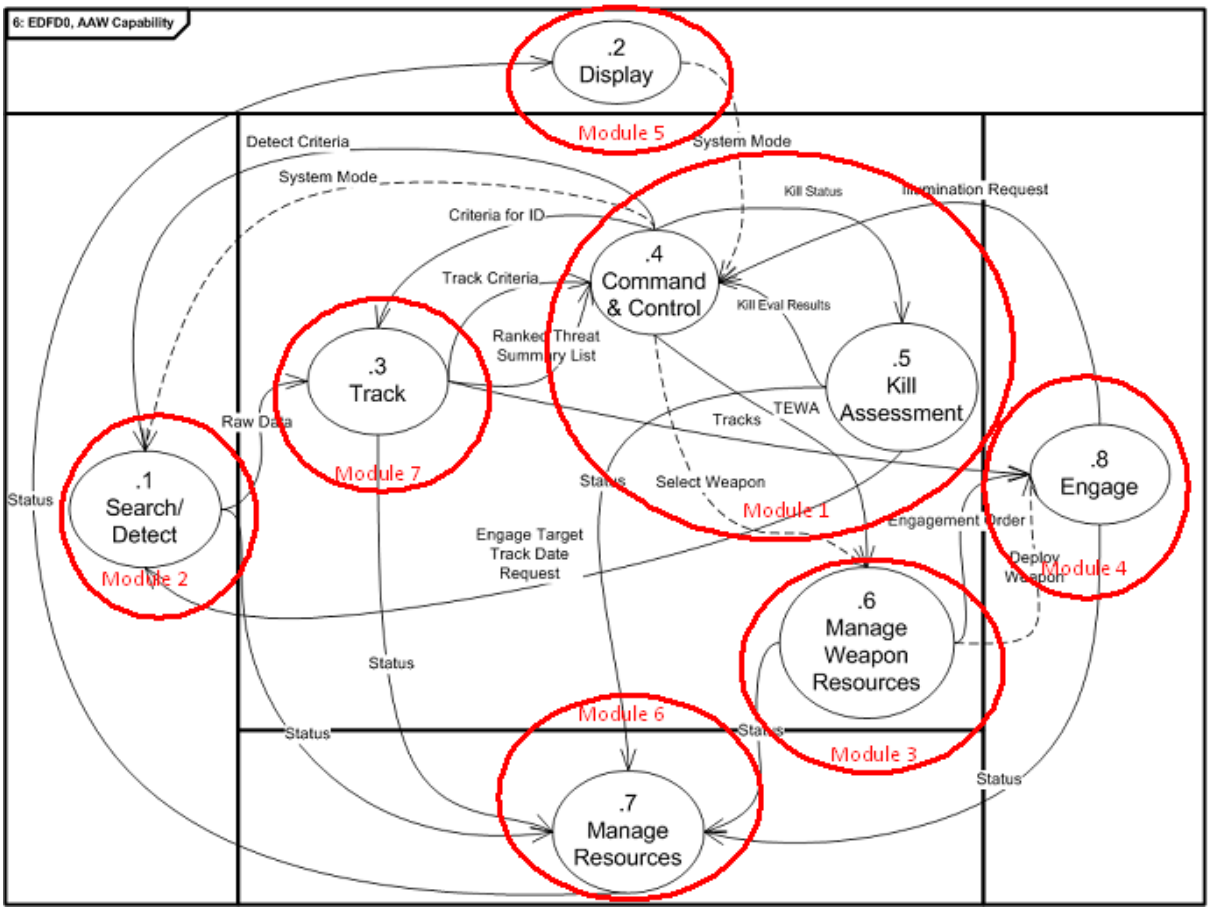
Software functions are grouped into modules illustrated in the diagram and allocated to their respective processors. At this stage, hardware choices are not required because a deeper understanding of the behaviors is still being uncovered to meet requirements. This initiates the next step of the development model to showcase SPL templates based on the modulation of the AAW system architecture.



Core Processing identified

Figure 4-35: AAW AFD0

The EDFD in Figure 4-36 illustrates the allocation of processes to super bubbles or to seven architecture modules. Super bubbles in the EDFD represent architecture modules that are created based on the potential for a shared resource. The super bubble (Module 1) groups both the C2 and Kill Assessment (KA) processes, and contains all individual processes included in C2 and KA.



AAW Module Grouping

Figure 4-36: AAW EDFD

The seven architecture modules are:

- **Module 1 (C2 Central Processing Module & KA):** This module contains all functions of C2 and KA. C2 resides in module 1. The KA process was also included in Module 1 because of the processing required to ascertain if the target that C2 engaged has been destroyed. The C2 and KA processing should have low coupling and high cohesion. With low coupling, a change in one module will not require a change in the implementation of another module. Low coupling is often a sign of a well-structured computer system, and when combined with high cohesion, supports the general goals of high readability and maintainability. The KA function is only required to be invoked when a target is engaged and a weapon is fired; the track is maintained by C2 and, at the appropriate time, KA reviews and analyzes it for relevant dynamic behavior. KA provides a confidence that the track is no longer viable or killed. If so, C2 does other work; if not, C2 must re-engage to attempt to kill the target again. C2 and KA processes must have a continuously stable interface to ensure requirements can be met.
- **Module 2 (Sensors Module):** Includes functions of sensors. Search and detect process is allocated.
- **Module 3 (Weapon Management Module):** Includes functions of weapon assignment and missile selection. Manage weapon resources process is allocated.
- **Module 4 (Weapons Module):** Includes functions of missile communication and energizing missiles. Engage process is allocated.
- **Module 5 (Operator Procedures Module):** Includes functions of command inputs and display. Operator process is allocated.
- **Module 6 (Manage Resource Module):** Includes functions of monitoring computers in the system. Manage resources process is allocated.
- **Module 7 (Track Module):** Includes functions of tracking. Track process is allocated.

4.4.4 Process Specifications

PSPECs are primitive specifications provided as structured English descriptions that use "shall" statements to describe the way in which they transform input information into desired outputs. PSPECs are functional requirements of the EDFD0. Each process bubble has corresponding PSPECs (Table 4-3). In this architecture development model, the PSPEC's input and output data flows represent all those to/from that process. It is essential to the goal of an open architecture that modules designed and produced by different manufacturers will interoperate and interface seamlessly. The PSPEC encapsulates each process's internal operational behavior and exposes only the external interfaces.

Table 4-3: PSPECs

2.1 Process Search/Detect	
Input Flow(s): Detect Criteria Engage Target Data Request	**Output Flow(s):** Raw Data Status
Description: This process shall be responsible for interface with shipboard sensors. It shall provide raw data report to the track process as requested via a detect criteria set forth by the command and control process. Engagement data shall also provide to the kill assessment process. Search and Detect status shall be provides to the manage process.	
Functional Requirement(s): a) Sensing b) Cueing c) Correlation d) Classification	
2.2 Process Operator	
Input Flow(s): Status	**Output Flow(s):** System Interface

Description:

This process shall be human and machine interface. It shall be the avenue of interaction of ship force with the AAW system via combat display consoles.

Functional Requirements:

a) Display

b) Decision Authority

2.3 Process Track

Input Flows:	**Output Flows:**
Raw Data	Track Criteria
Criteria for ID	Rank Threat Summary List
	Tracks
	Status

Description:

This process shall be responsible for creation of track database compiled from the raw data provided by the search/detect process. It shall provide a ranked threat summary list set forth by the criteria for ID from the command and control process. The process shall also provides status to the manage resources process.

Functional Requirements:

a) Establish Track

b) Firm Track

c) Correlation/ID

2.4 Process Command and Control

Input Flows:	**Output Flows:**
Track Criteria	Criteria for ID
Rank Threat Summary List	Detect Criteria
Illumination Request	TEWA
	Kill Status

Description:

This process shall be the set detection criteria for the search/detect process. It shall perform the TEWA functions and provide kill status to the kill assessment process. It shall plan and direct track database with criteria for ID and detection criteria to the search/detect process and the track process. It shall provide illumination function to the engage process.

Functional Requirements:

a) Thread Evaluation

b) Weapons Assignment

c) Establish Firing Doctrine

d) Select Director

e) Final Engage Ability

f) Fire Control Solution

g) Final ID Confirm

h) Final Launch Oder

i) Check-Hold Fire/BRK

j) System Health Manager

2.5 Process Kill Assessment

Input Flows:	**Output Flows:**
Kill Status	Kill Evaluation Results
Resource Status	Engage Target Track Data Request
	Status

Description:

This process shall perform kill assessment function from engagement track data provided by the search/detect process. It shall update the command and control process with a kill status report. System status shall be monitored via the resource management process.

Functional Requirements:

a) Kill Assessment

2.6 Process Manage Weapons Resources

Input Flows:	**Output Flows:**
TEWA	Engagement Order
	Status

Description:

This process shall manage onboard missile assets. It shall determine the proper missile and cell selection for engagement process. System status shall be monitored via the resource management process.

Functional Requirements:

a) Missile Selection

b) Cell Selection

c) Determine Engage Solution

d) Engage

2.7 Process Manage Resources

Input Flows:	**Output Flows:**
Status	Resource Status

Description:

This process shall provide shipboard interfaces and provide status the sensor/detect process, track process, command and control process, kill assessment process, and the engage process. It shall monitor the health of shipboard resources such as power, water, gyro, and NAV.

Functional Requirements:

a) Diagnostics

b) Ship Interfaces

b) Power/Water/Gyro/NAV

2.8 Process Engage

Input Flows:	**Output Flows:**
Engagement Order	Illumination Request
Tracks	Status

Description:

This process shall be responsible for routing the engagement order from Command and Control to the missile launching platform. It shall track and update missile status during the initial stage, homing stage, and the terminal stage. It shall upload illumination of threat to the missile. System status shall be monitored via the resource management process.

Functional Requirements:

a)Energize Missile

b) Ignite Propulsion

c) Missile Fly Out

d) Missile Comms

e) Engagement Scheduling

f) Illuminator Scheduling

g) Terminal homing

The Process Control Table (Table 4-4) maps the output action of the AAW system FSM to corresponding PSPECs. The control signal activates and deactivates all the matching processes of the PSPECs. For example, the control action receiver energy activates PSPECs 2.1 and 2.3. The process/control mapping binds the static data model to the behaviors of the control model. This methodology completes the system requirement model of the HP structure analysis.

Table 4-4: Process Control Table

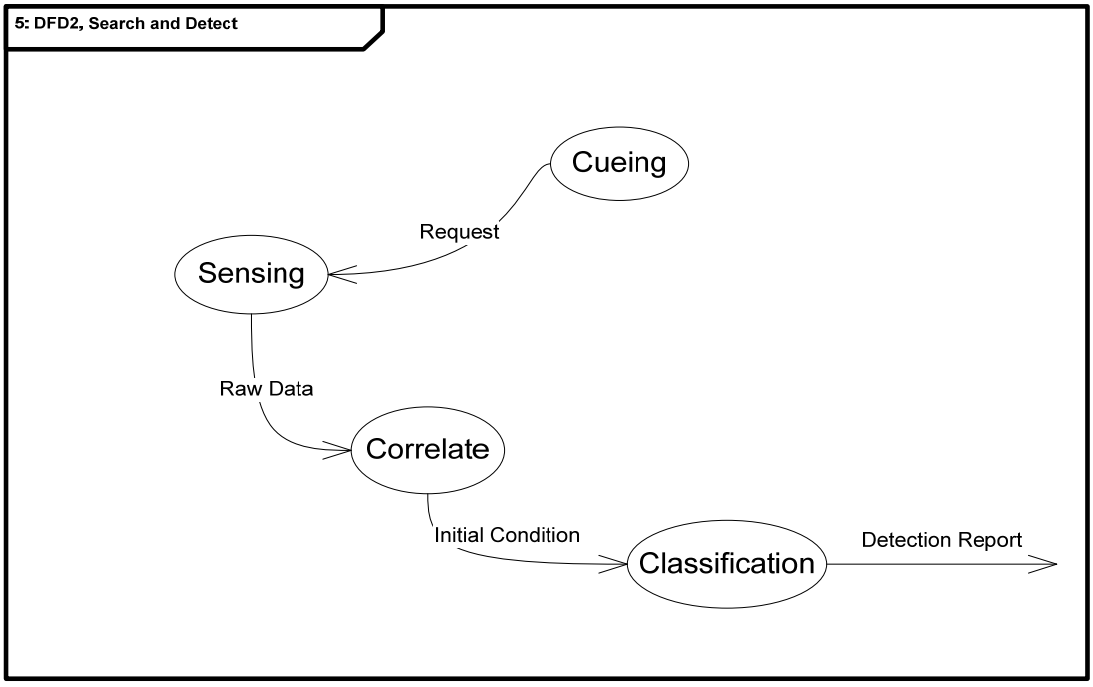
Mapping Control Action to Data Process

Process / **Control**	Search/Detect 2.1	Operator 2.2	Track 2.3	Command and Control 2.4	Kill Assessment 2.5	Manage Weapon Resources 2.6	Manage Resources 2.7	Engage 2.8
Search and Detect	1	0	1	1	0	0	0	0
Track	0	0	1	1	0	0	0	0
Perform TEWA	0	0	0	1	1	0	0	0
Receive Energy	1	0	1	0	0	0	0	0
Fire Weapon	0	0	0	1	0	1	0	1
Perform Kill Assessment	0	0	0	1	1	0	0	0
Save System States	0	0	0	0	0	0	1	0
Restore System States	0	0	0	0	0	0	1	0
Update Display	0	1	0	1	0	0	0	0

In a simple system, an architecture module might not need further breakdown because the process that is allocated to the architecture module might be enough to generate the desired outputs. However, a complex system like AAW cannot be described with one level of a process in the architecture module. Another example of how the HP process can help is provided in an example, illustrated in Figure 4-37. The sensor process is named Search & Detect. When a process message request from Cueing tells Sensing to sense for objects, the module needs to correlate and classify the sensed object data to prepare a detection report. There are four necessary processes to generate the output of the sensor module. DFD2 (Figure 4-37) shows the four sub-level processes and data flow.

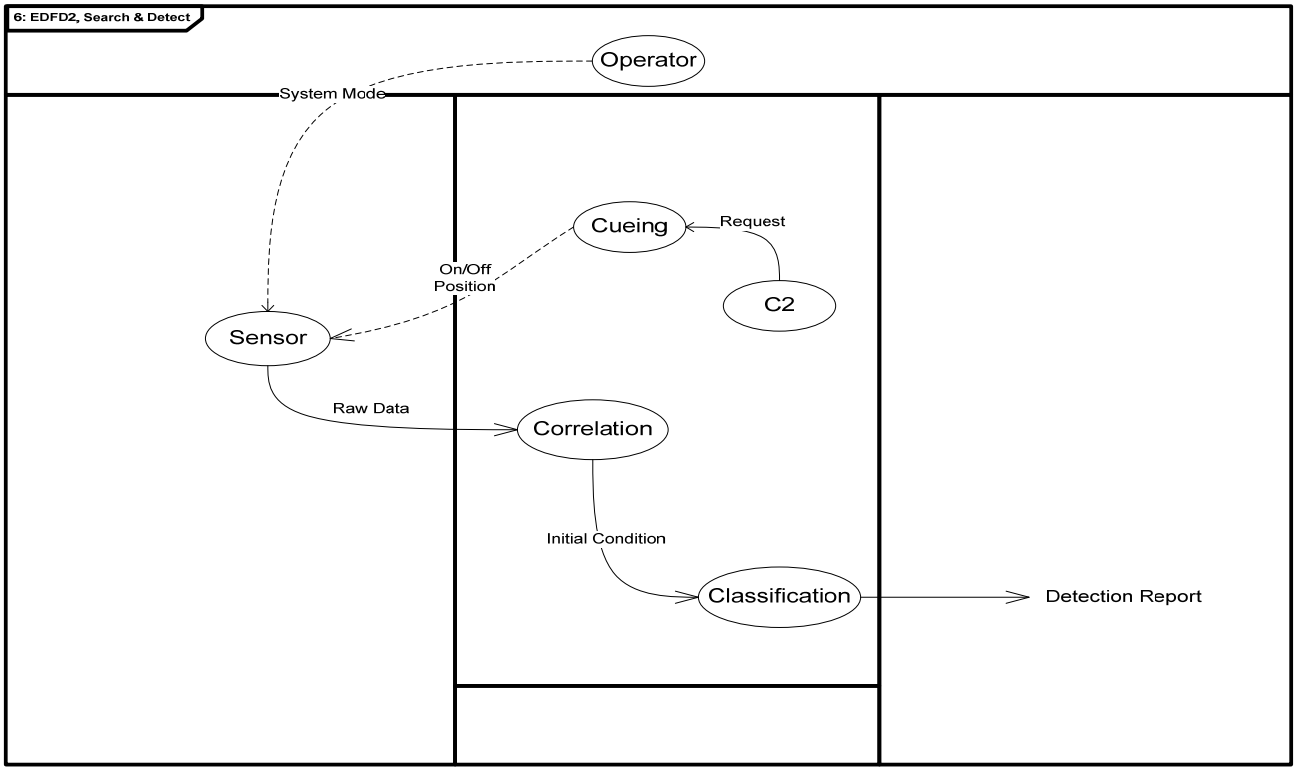
EDFD2 (Figure 4-38) shows external entities of the sensor module such as operator process and C2 process. These external entities give commands to sub-level processes of the sensor module. The operator process sends a system mode signal to turn on or off the sensor. The C2 process starts the cueing process by sending a request. An EDFD of the architecture module presents a complete picture of how data flows in the system. The views enhance the reader's understanding of control and data flow, conveying required information of system architecture.

The Architecture Flow Diagram (AFD) shows data and control flows among architecture modules defined in Figure 4-39. All flows have already been defined in EDFD (Figure 4-38). In Figure 4-39, all processes in EDFD were allocated to architecture modules, which were represented as super bubbles. All data and control flow lines shown in Figure 4-38 are lines that connected super bubbles in Figure 4-39.



First level of decomposition

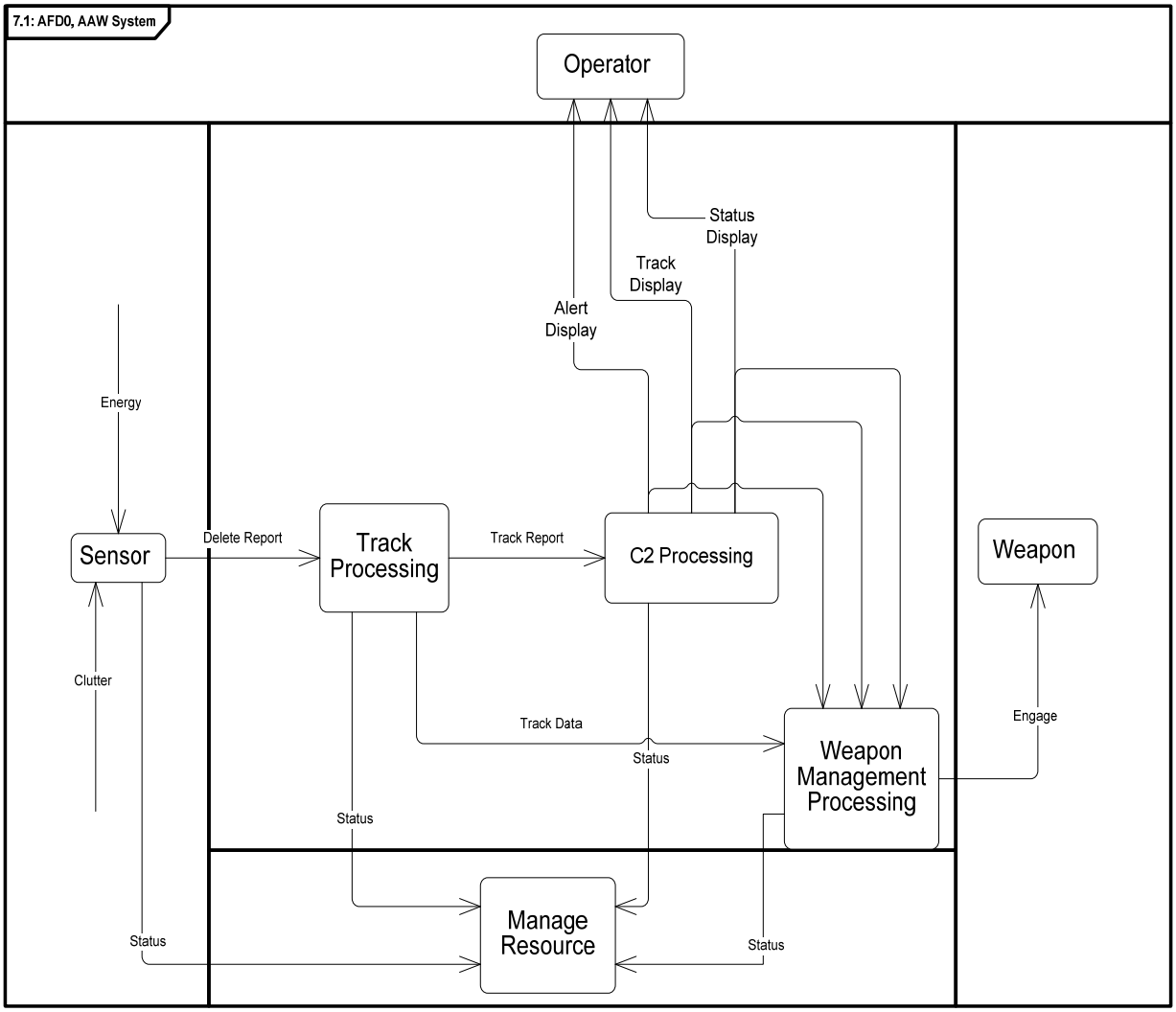
Figure 4-37: Search and Detect DFD2



Second Level of decomposition

Figure 4-38: Search and Detect EDFD2

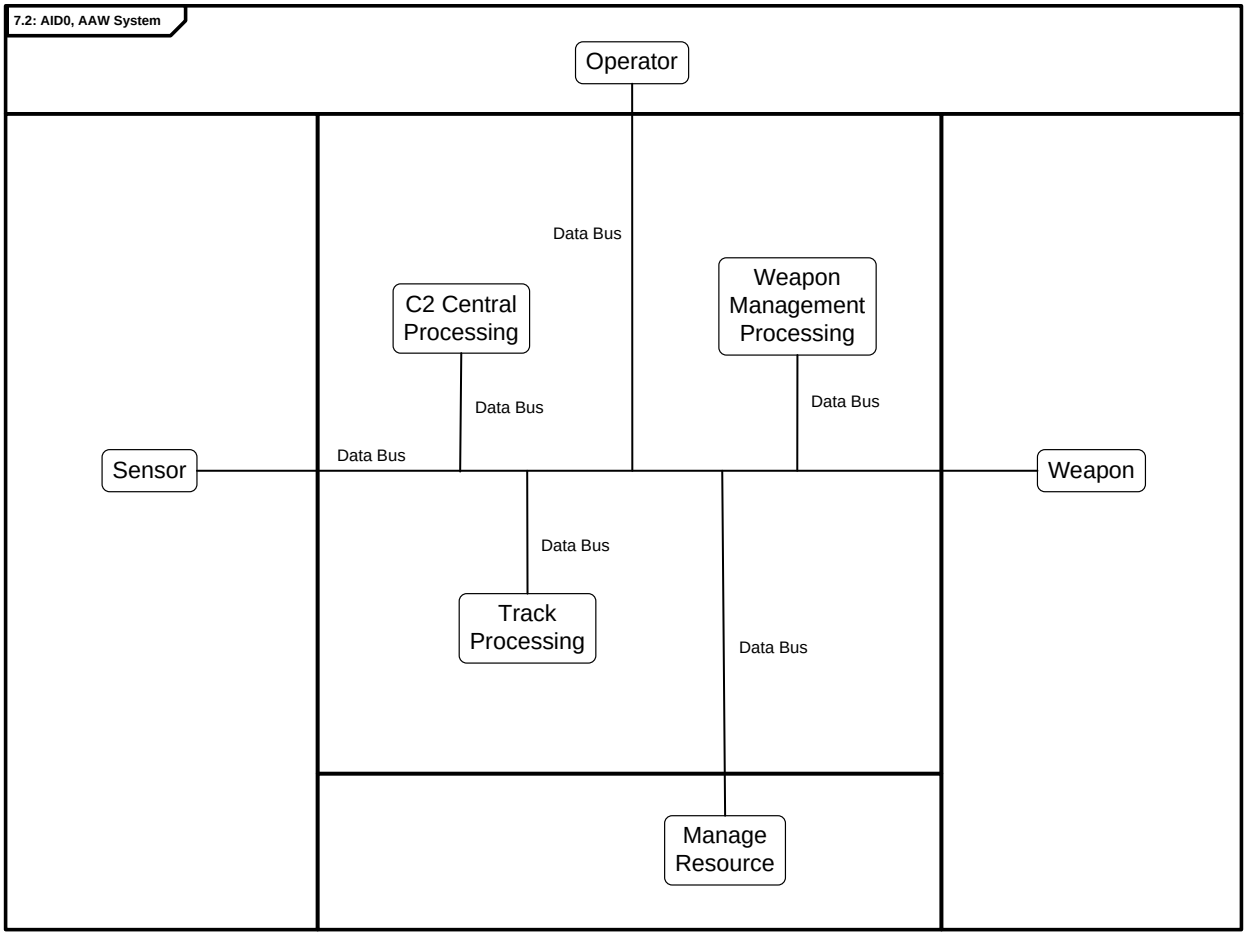
Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.



Architecture Module Data Flow

Figure 4-39: AAW AFD0

The Architecture Interconnect Diagram (AID) (Figure 4-40) illustrates the physical channels for exchanging information such as data and control. It depicts the allocation of data and control flows to channels. The proposed AAW system in this study was designed to use Transmission Control Protocol/Internet Protocol (TCP/IP), requiring the system to be smart enough to send information to where it is needed. This approach offers the advantage of reducing the amount of cable installations needed. AID shows one common bus that is connected to all architecture modules. To remove any ambiguity, all flow-to-channel allocations are recorded in the architecture dictionary (Table 4-5).



Architecture Module Data Bus Interconnection

Figure 4-40: AAW AID

Table 4-5: Architecture Dictionary

Name	Definition/Physical Description	Type	Source	Destination	Channel
Detect Report	Current location and speed of detected objects	Data	Sensor	Track Processing	Common Bus
Track Report	Calculated estimated heading and speed + current course + past course	Data	Track Processing	C2 Processing	Common Bus

4.4.5 Architecture Development Model, Layered Architecture and SPL

Eight processes were identified (through use of the HP method) to support an AAW software architecture. Each process requires software to perform the required functions. For software developers, the EDFD (and supporting information) helps to create a specification that outlines the control and data handling behaviors of the software system. A PSPEC, which is based on the functional requirement of the EDFD, is an important specification for software development. The architectural development model shown in section 4.4.3 (Figure 4-28) can serve as an example of defining processes and functions for a software system.

This report focuses on software architectures (See Appendix I for further detail), and determined that the modular structure of a layered architecture can meet the stringent and complex requirements of an AAW system while providing the added benefit of flexibility and commonality needed by OA and the SPL. Each process in the proposed AAW system has distinct data handling requirements. To determine layers, the architecture logically grouped the software in the system by its data handling, and each group was broken down into layers by the function of the layer as shown in Figure 4-26.

Lower level layers show flexibility of new systems by isolating changes of the system within them. Adaptive ware in the middleware layer (one of the lower level layers) converts diverse data formats into one standard data format that all higher-level layers can process. Because of this function of the middleware layer, software applications in higher-level layers do not need to be modified or newly developed when any change occurs in the system. Once this adaptive ware is developed for the middleware layer, the modularity of the layered architecture helps to reuse the adaptive ware in the same data translating applications.

The logical grouping of the software offers the benefit of commonality for SPLs. All software in a layer is required to meet the same interface requirements to communicate

with other software in other layers. The concept of using the layer interface requirement as a source of commonality has the advantage of moving a big portion of work for meeting interface requirements from the software to the common lines of code. This approach provides a path to create SPLs and offers the benefits that come with it, such as reducing software development time by reusing common lines of code.

SPLs are rapidly emerging as a viable and important software development paradigm allowing the Navy to realize an order-of-magnitude improvement in time to acquire new weapon systems. SPL engineering can also enable rapid market entry and flexible response, and provide a capability for mass customization. Appendix J provides further detail.

4.5 VERIFICATION AND VALIDATION

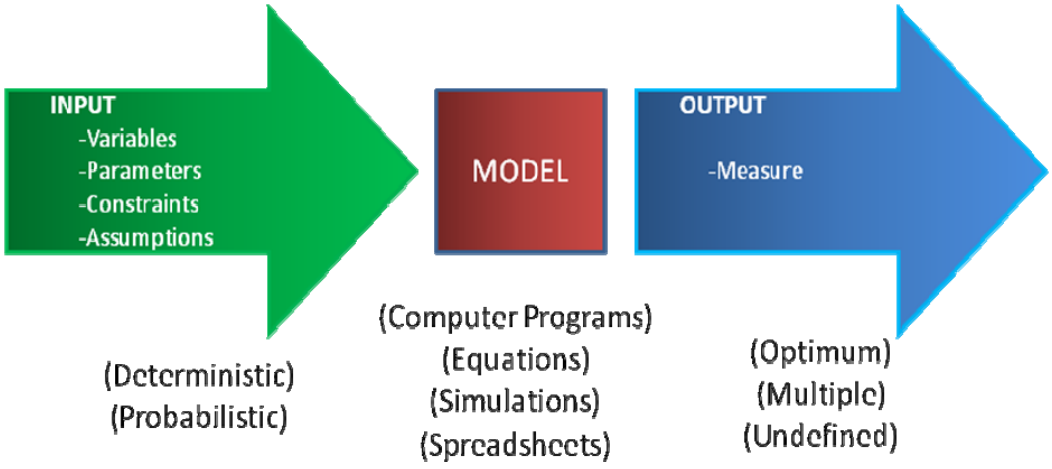
4.5.1 M&S Introduction

The purpose of M&S is to verify that the methodology and AAW architecture meet the functional requirements in the operational environment described in the ConOps. The primary objectives for M&S were to develop the necessary models and simulate defined scenarios. These models and simulations serve to verify the AAW architecture and ConOps requirements. To accomplish these objectives, research findings were applied to develop a viable and effective methodology for M&S application. These resources included academic lectures (e.g., NPS), textbooks (e.g., Johns Hopkins APL Technical Digest 2001), software (e.g., Extend), and research articles (e.g., Roedler and Jones 2005) The goal is to utilize this M&S methodology as an element of the V&V process to validate the methodology for AAW architecture development.

4.5.2 M&S Methodology

This section describes the strategy and approach used to create the M&S methodology. The proposed methodology can be applied once requirements have been generated and implemented throughout the development of the AAW architecture. Requirements must be defined to enable initialization of the M&S process as early as practical. Defining and refining the requirements was critical for enabling the use of an incremental approach toward meeting the stated objectives. The proposed M&S methodology conforms to the MBSE approach by establishing requirements traceability and linking the V&V of the model's architecture and Extend Model. This was accomplished by employing a model-driven approach for developing, designing, and verifying the system. Extracting the necessary documentation and artifacts from the central design repository in CORE was critical for ensuring the traceability of requirements, functions, and behaviors to the actual model.

This methodology also conforms with applying supportability requirements early in the design phase (prior to the model development) and throughout the V&V phase. MOEs are established to allow for V&V to meet mission requirements. (Johns Hopkins APL 2001) Models are developed to represent elements in an architecture and put into mathematical simulations to produce an output. Figure 4-41 represents the Model Formulation, which consists of input variables, parameters, constraints, and assumptions. These inputs are entered into a mathematical model comprised of equations, simulations, and/or spreadsheets.



This is the process of using given inputs to create a model that dispenses a measurable output.

Figure 4-41: Model Formulation (Posadas 2007)

Modeling begins with identifying and defining a problem. The next step is to represent the process and collect data inputs and requirements from the stakeholders. The model is then developed, tested, and analyzed to interpret and verify results against the originating requirements for the mission.

When top-level requirements are defined by the customer, they are decomposed into concepts for the model. From the concepts, MOEs are generated to verify the validity of the functions in the model. In the example case, the top-level requirement was to achieve a P_{RA} of 0.99. To achieve the requirement, concepts of how the model should flow were broken down into combat system functions (Figure 4-17). These functions include: (1) Search and Detect, (2) Track, (3) Command and Control, and (4) Engagement. To validate the functions, MOEs were compared with the results from the functions to assess the model's performance.

The proposed M&S methodology was formulated into a high-level process as shown in Chapter 3. This process highlights M&S guidelines throughout the course of the AAW Architecture development.

1. **Define the M&S Objectives:** The first step is establishing specific objectives for M&S within the context and scope of the system architecture. This step requires a detailed review of the ConOps, system requirements, functional architecture, behavioral analysis, supportability functions, and/or any other applicable documents and artifacts that may assist in defining the main objectives.

   Development of this project's AAW architecture began by using the system requirements and ConOps to create main objectives for M&S to satisfy specific scenarios for combat system engagements. As the AAW architecture development progressed and the functional architecture became better defined, the primary objectives remained unchanged; however, the M&S objectives were further refined to specify the necessity for modeling the AAW Self Defense and the Limited Area Defense aspects of the architecture.

2. **Characterize the Model:** The second step is to define the critical measures of the system and identify the inputs for the models. Tables 4-6 and 4-7 organize the model's inputs for specifications and requirements.

Table 4-6: Given Requirements

Given Requirements and Spec				
Threat Profile				
Na (number of attack)	8	each		
Tar (target arrival rate)	4	sec/target		
V_target (mach 0.9)	306.261	m/s	0.165	nm/s
Ht (target max height)	9	m	0.0090	km
RCS	1	m^2		
Combat System Specs				
Tr (Reaction Time)	6	sec		
Tft (Firm track time)	1.5	sec		
Te (kill evaluation time)	2	sec		
Radar Height	50	ft	15.24	m
Probability of detection	0.995			
EO/IR Sensor Height	60	ft	18.288	m
Two X-band illuminator				
Tup (tie-up time)	6	sec		
Defense Requirement				
Pra	0.99			
Rout (keep out range)	2	km	1.08	nm
Weapon Choice and Specs				
Two 8-Cell Medium Launcher				
L (launch rate)	1	sec/missile		
Two Short RangeLauncher				
TL (initial launch)	5	sec after detection		
L (launch rate)	1	sec/missile		
Nc (number of cell)				
Missile Choice and Specs				
Short Range (fire and forget)				
Vm (velocity) - mach 2	0.368	nm/s	680	m/s
Rmin (min range)	0.54	nm	1	km
Rmax (max range)	5.4	nm	10	km
P(k)	0.85			
Medium Range HK				
Vm (velocity) - mach 3	0.551	nm/s	1020.9	m/s
Rmin (min range)	0.54	nm	1	km
Rmax (max range)	16.2	nm	30	km
P(k)	0.8	nm		
Long Range HK	0.8			
Vm (velocity) - mach 2.5	0.459	nm/s	850.73	m/s
Rmin (min range)	1.62	nm	3	km
Rmax (max range)	43.2	nm	80	km
P(k)	0.7	nm		
Soft Kill				
P(k)	0.6			
Execute Time	10	s		
Activation Range	7	nm		

This table is an example of the parameter to be used in a model.

Table 4-7: Derive Requirements and Assumption

The table identifies the engineering assumptions represented in the model.

Derive Requirements			
Radar Horizon (to calculate maximum radar detection range)			
Rd(km) = sqrt(17Ht) + sqrt(17Hr) =		28.46528	km
Assumption			
1. Assume initial detection position is at Rd at time 0			
2. Assume re-scan time, Ts =	1	sec	

The critical measures of the architecture include combat system timing, time latency, target characteristics, and environmental aspects that relate to values in the model. Examples of measures for the AAW Architecture include: radar detection ranges, threat velocities, and weapon reaction times between missile engagements.

Establishing MOEs is the next step in the process. The MOEs are used to evaluate the primary objectives for M&S with regard to the critical measures. The MOE for the AAW architecture model was P_{RA}. According to Ortiz,

> P_{RA} is defined as the ability of a particular stand-alone ship, as an integrated system, to detect, control, engage, and defeat a specified raid of anti-ship cruise missile (ASCM) threats with a specified level of probability in the operational environment. The P_{RA} MOE is a system-of-systems measure which is levied on the ship defense suite as a whole to properly detect, control, and engage (annihilate) a raid of incoming threat ASCMs. Thus, it doesn't measure the performance of any particular ship defense element; rather it measures the system performance of all the ship defense elements across the complete battle timeline. (Ortiz 2006)

MOPs and KPPs are also defined to further characterize the model. Roedler et al provided the following clarification for MOPs and KPPs:

> MOPs measure attributes considered as important to ensure that the system has the capability to achieve operational objectives.

> MOPs are used to assess whether the system meets design or performance requirements that are necessary to satisfy the MOEs. MOPs should be derived from or provide insight for MOEs or other user needs. MOPs often map to KPPs, or requirements in the system specification. Each KPP has a threshold and Objective value. KPPs are the minimum number of parameters needed to characterize the major drivers of operational performance, supportability and interoperability. (Roedler and Jones 2005)

After the critical and technical measures (MOEs, MOPs and KPPs) are defined, the inputs for the model must be identified. The inputs and parameters identified in Tables 4-6 and 4-7 are variable in nature and are important to outline the dynamic characteristics of the model. Randomization of the parameter's behavior can be based on statistical analysis and functional scenario review for more realistic simulation results. Characterizing the example model was an important step for the application of M&S methods and analysis research towards V&V of the AAW Architecture. The following data were included in the example model:

- Threat profile
  - Number of attacks
  - Target attack frequency
  - Target velocity
  - Target height
- Combat Systems
  - Radar characteristics
  - Kill evaluation time
  - Salvo size
- Mission requirements
  - Probability of raid annihilation (P_{RA})
  - Keep out range
- Weapons
  - Type
  - Velocity
  - Range
  - Probability of kill (P_K)

In the case of developing a LAD or surveillance scenario model, additional inputs from the customer would involve the parameters of the defended asset and weapons doctrine setup. If range is a factor, other combatants and warfare centers are required to help communicate and relay information.

3. **Identify Tools:** The goal for this step is to research and define the tools to be utilized for modeling and simulation. It is important to ensure that appropriate tools are used in given situations. For example, if the model is simple, computer spreadsheets will suffice for minor to moderate calculations. If the model is more intricate, a higher level modeling tool such as Extend or Arena is recommended.

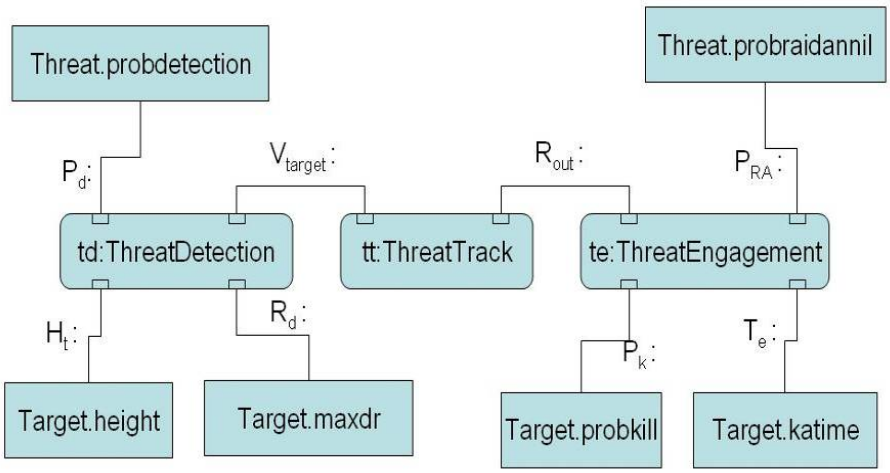
   For the AAW architecture example, spreadsheets were developed for the initial mathematical calculations (Table 4-8). Extend was used to conduct the more detailed and intricate modeling aspects of the architecture.

4. **Build a Parametric Diagram:** After characterizing the model and identifying all the necessary tools, the next step is to review the critical measures, input variables, and required calculations for the system to facilitate the development of a parametric diagram (Figure 4-42). The parametric diagram is used to structure the characteristics of the model into a high-level mathematical diagram that will aid in depicting the flow of calculations for the simulation.

Table 4-8: Spreadsheet Modeling Samples

Spreadsheets were developed for the initial mathematical calculations and reasoning for the AAW example.

Med Range HK and Soft Kill Rounds							
TGT #	TGT Detection Range	TGT Detection Time	Med Range HK #	Director #	Med Range HK Launch Time	TGT Range at Launch	Intercept Range
1	16.10	0.00	1	1	4.50	14.87	11.44
			2	1	5.50	14.59	11.22
			3	1	6.50	14.32	11.01
2	16.10	4.00	4	2	4.50	14.87	11.44
			5	2	5.50	14.59	11.22
			6	2	6.50	14.32	11.01
3	16.10	8.00	7	1	20.54	10.46	8.05
			8	1	21.54	10.19	7.84
			9	1	22.54	9.92	7.63
4	16.10	12.00	10	2	20.54	10.46	8.05
			11	2	21.54	10.19	7.84
			12	2	22.54	9.92	7.63
5	16.10	16.00	13	1	32.89	7.08	5.45
			14	1	33.89	6.80	5.23
			15	1	34.89	6.53	5.02
6	16.10	20.00	16	2	35.89	6.26	4.81
			17	2	36.89	5.98	4.60
			18	2	37.89	5.71	4.39
7	16.10	24.00	19	1	42.38	4.47	3.44
			20	1	43.38	4.20	3.23
			21	1	44.38	3.93	3.02
8	16.10	28.00	22	2	45.38	3.65	2.81
			23	2	46.38	3.38	2.60
			24	2	47.38	3.10	2.39

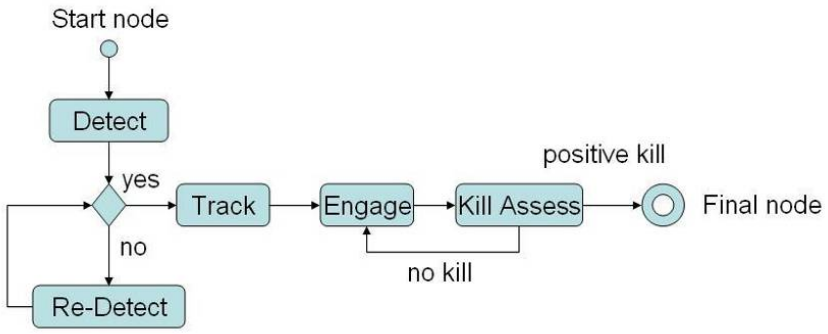


The parametric diagram is used to structure the characteristics of the model into a high-level mathematical diagram that will aid in depicting the flow of calculations for the simulation.

Figure 4-42: Parametric Diagram

5. **Build a High Level Model:** When designing the high level model, start from the known requirements and functional architecture. These outline the main functions and flow of the model.

   For example, the three main functions (Detect, Control, and Engage) of the AAW architecture represent a baseline model. Next, a simple activity diagram was created to show the directional and logical flow of the model (Figure 4-43). It was imperative that the main functions and activities were traceable to the requirements and functional architecture as well as to any major scenario activity in the ConOps.



The model is a simple activity diagram to show the directional and logical flow of the model.

Figure 4-43: High Level Model

6. **Construct Main Functions:** After the High Level Model is generated, the process begins for constructing the main functions of the model. It may be helpful to take one main function of the model and separate it from its interfacing counterparts to help focus on the particular logical aspects of the function. For example, take the High Level Model that was used for the AAW example in Figure 4-43 and focus solely on the Re-Detect function. There are several sub-functions and events that make up Re-Detect. These sub-functions include re-scan timing delays, target distance calculations, and target priority scheduling. Generating the specific input variables, constants, parameters, and dynamic characteristics of the Re-Detect sub-function is also required.

7. **Integrating Supportability:** Modeling is the creation of abstractions or representations of the system to predict and analyze performance, cost, schedules, and risks, and to provide guidelines for systems research, development, design, manufacture, and management. (Maier and Rechtin 2002) As such, it is imperative that the model represent supportability factors and incorporate those elements into the model from the beginning. Supportability factors are critical in predicting availability, reliability, readiness, and life cycle costs of the system being modeled, and should be integrated throughout the requirements hierarchies and functional architecture design. The engineering and supportability teams should examine the main functions of the model and employ changes at the high level before starting the detailed system modeling.

   In the AAW architecture, serial and parallel redundancies in the model allow decision-makers to meet availability and reliability requirements. The AAW architecture is defined in section 4.6.5.

8. **Build, Execute, and Analyze the Model:** The last step in the methodology may either be the easiest or most difficult to implement depending on how well the upfront planning and design for the model proceeded. The key is to utilize all the previously identified techniques, processes, and tools to help construct the model. Extend was chosen to model and simulate data for analysis, interpretation, and decision making of the AAW architecture. Based on these results the model can be modified as required to meet the M&S objectives. For example, the short range missile was not selected to be a weapon in the model because it did not provide a large enough intercept range to support the customer's request of having a 2 km keep-out zone. Even though it did have a higher P_K than other missiles and soft kill (as stated Table 4-6), its limited range precludes it from target interception at the required distance.

4.5.3 Model Description

The Self Defense model was built using Extend software language. It was built to model a process, represents a realistic threat scenario, and was simulated to prove out the combat systems architecture. The model was built based on assumptions, constraints, and specifications from requirements and ConOps stated previously in the report. Since the model is a part of a validation process, arbitrary values were used as necessary to depict performance examples of weapons and combat system elements. The model was broken down into the following functions or processes:

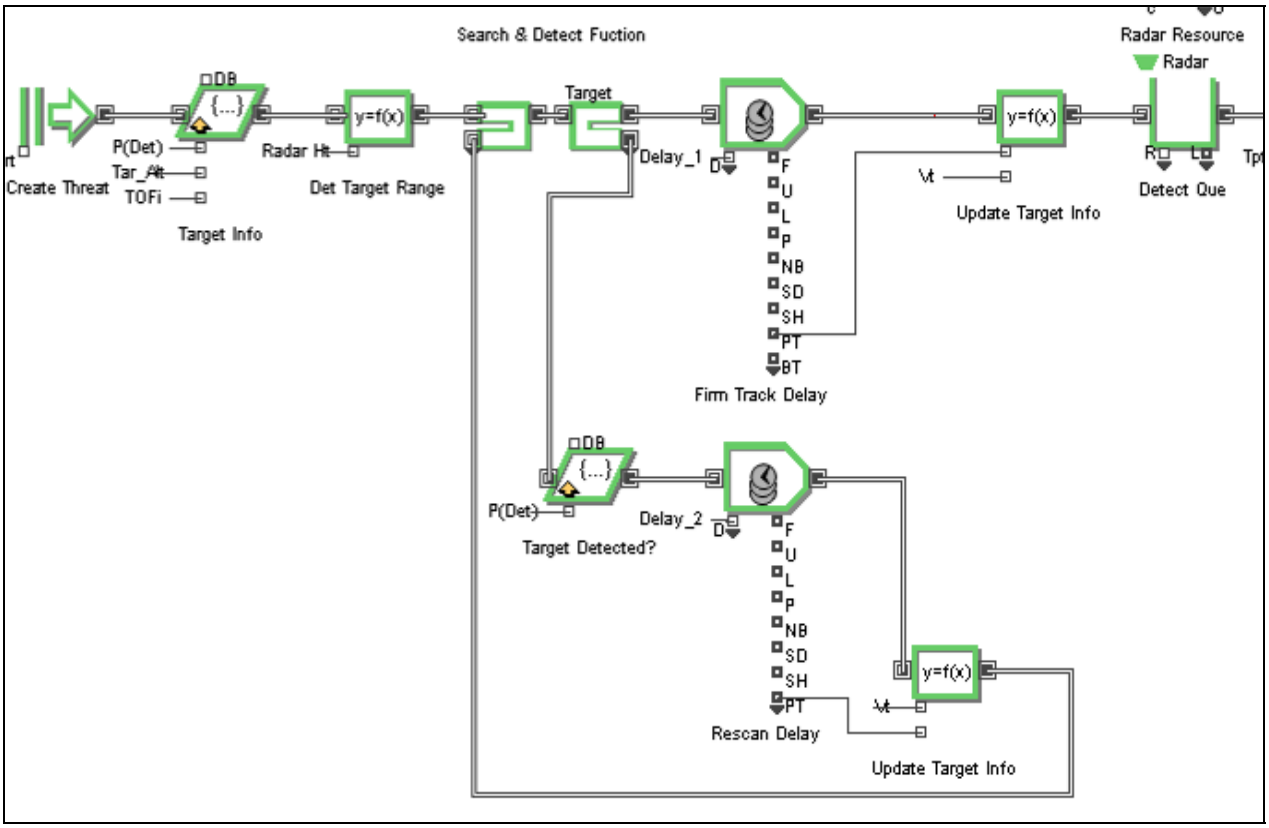
- Search and Detect
- Tracking
- Engagement

4.5.3.1 Search and Detect Function

The first stage of the model is target creation. AAW targets were generated and deployed every four seconds. Each target was assigned probability of detection, target altitude, time of flight, and target range position. The probability of detection was set at 0.995. Target altitude was a constant value of nine meters. The target's initial time of flight of flight was set at zero seconds. The next step was to calculate the detection range of the target. The radar horizon equation was used to calculate the maximum detection range.

$$T\arg etRange = \sqrt{(17 \bullet RadarHeight)} + \sqrt{(17 \bullet T\arg etAltitude)}$$

If the target is detected by the radar, it is assigned a binary value of one and continues to the next function, track. If the target is not detected by the radar, it is assigned a binary value of zero and is rescanned by the radar for re-detection. During rescan, the target's velocity and range is updated and fed back into the search and detect function with the same probability of detection of 0.995. This loop will continue until the target is detected successfully. Upon detection, the target enters the track function. Figure 4-44 depicts the Extend *Search and Detect Function*.



These functions were built in Extend to capture the detection of the simulated threats.

Figure 4-44: Search and Detect Function

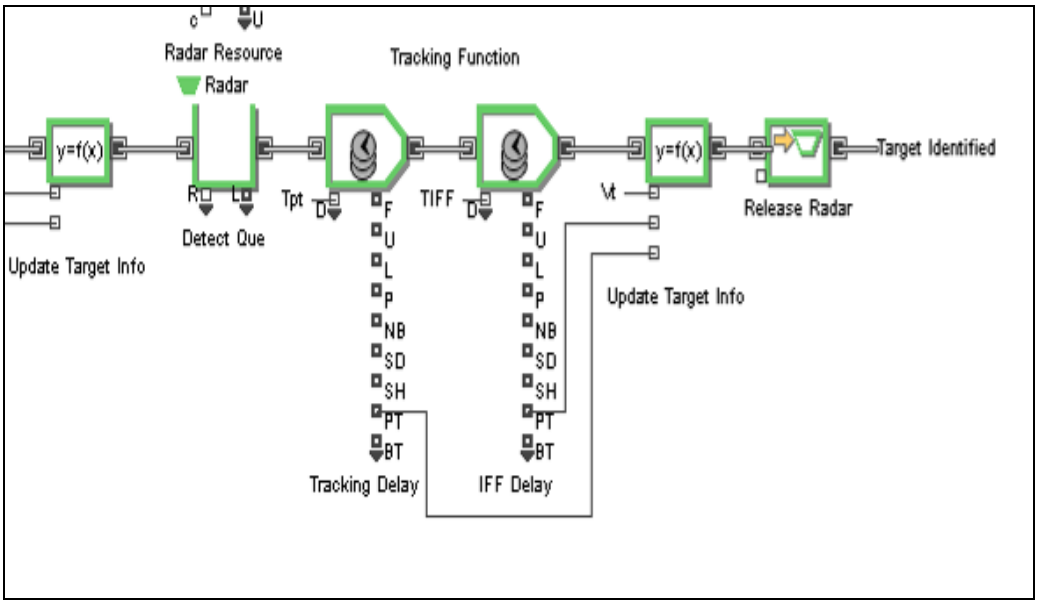
4.5.3.2 Track Function

The *Track* function follows the *Search and Detect* function. The model will calculate the target's time of flight (TOF) and range (equations seen below). Figure 4-45 depicts the Extend *Track* function.

$$T\arg etTOF = T\arg etTOF + Delay\_1$$

$$T\arg etRange = T\arg etRange - T\arg etVelocity \bullet T\arg etTOF$$

By knowing the target's characteristics, the system will make the determination of friend or foe, which is called the Identification Friend/Foe (IFF) sub-function. In the AAW architecture model, it was assumed that every threat was a foe. Regardless of friend or foe, the IFF sub-function adds a two-second delay into the model. Since the target was assumed to be a foe, it proceeds to the *Engage* function.



The tracking function may be called the Identification Friend/Foe (IFF) function.

Figure 4-45: Track Function

4.5.3.3 Engage Function

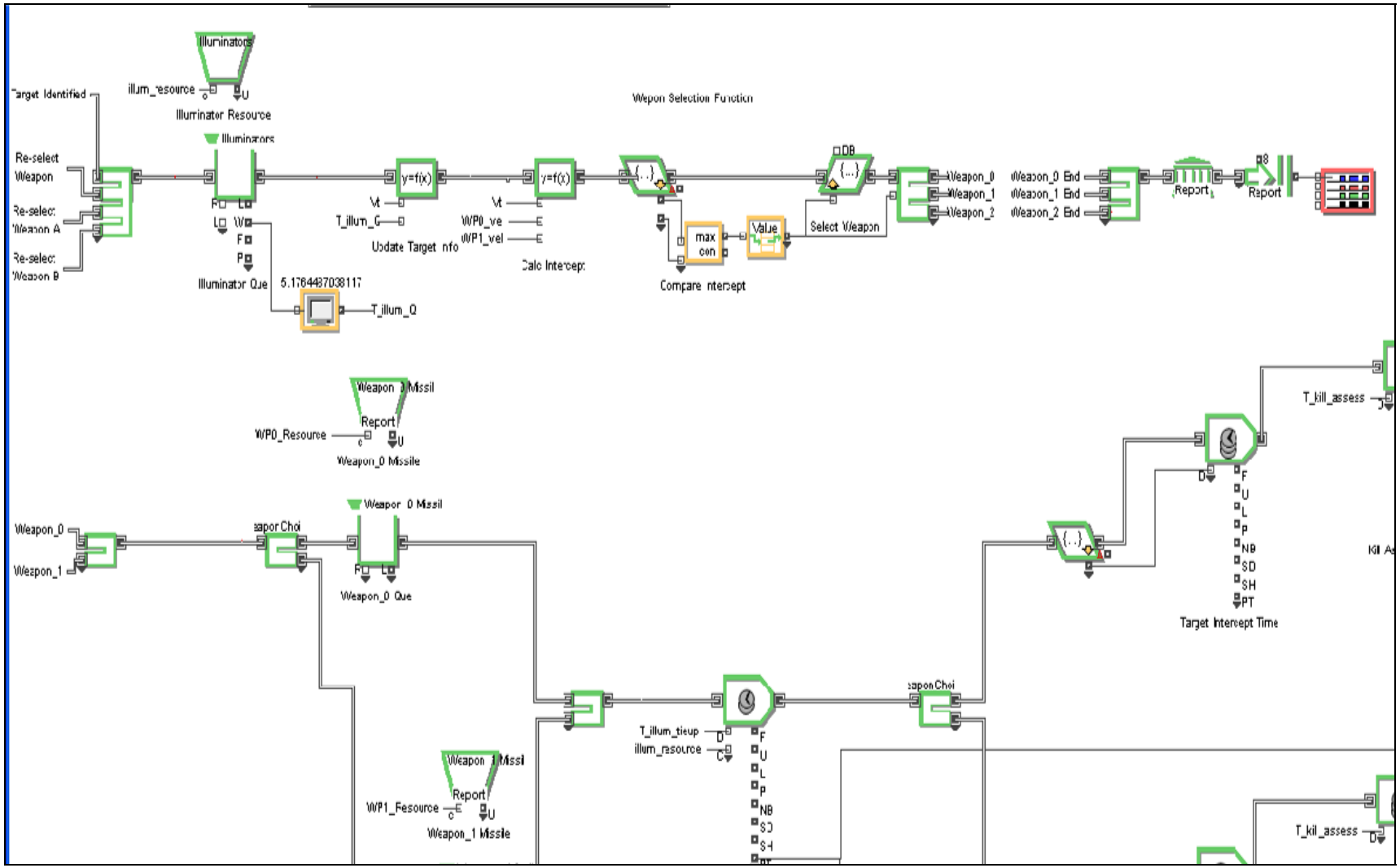
Upon system engagement of the threat, a weapon selection sub-function will determine the best available weapon to defend ownship. Based on the target's range, the appropriate weapon will be selected and fired. Illuminators will be used during the terminal phase of the weapon's flight to provide guidance to the calculated intercept range. After the intercept point is determined by the weapon system, the illuminator will become available for the next target. Kill assessment is calculated and incorporated as a two-second delay, which is based on the following probability of kill for the provided weapons.

P_K = Weapon_0 (3-80 km range) = 0.7

P_K = Weapon_1 (1-30 km range) = 0.8

P_K = Weapon_2 (less than 5 km range) = 0.6

If the target is not destroyed (determined from the kill assessment sub-function), the system will re-engage by selecting another weapon to fire. The weapon selection will be based on the updated threat profile (target range) and another illuminator will be assigned to provide terminal phase guidance. Figure 4-46 depicts the *Engage* function.



This function provides system engagement of the threat; a weapon selection sub function will determine the best available weapon to defend ownship.

Figure 4-46: Engage Function

4.5.4 M&S Analysis

The Extend model of the AAW architecture was modeled as a sequence of processes. An ideal scenario is one in which the probability of detection and probability of kill of the AAW architecture is set to 1.0. That is, each incoming threat is detected, tracked, and destroyed within the first engagement. The process however, simulates threats that may be detected, tracked, and destroyed at the first engagement or in repeated engagements, if required.

The model captured data of each threat's time at intercept, range at intercept, time of flight, the final weapon selected to destroy the threat, and records of ship's weapon resources (ammunitions). The model was run at a rate of one thousand times per iteration and was repeated for ten iterations. The model counts how many of the eight threats were destroyed per run and records the results one thousand times per iteration. A sample size equivalent to ten thousand runs was generated using Extend. Portions of the sample data are enclosed in Appendix L. The data was exported to Microsoft Excel for statistical analysis.

To verify and validate whether the AAW architecture could self-defend against a raid attack consisting of eight threats with a P_{RA} of 0.99, Excel was utilized to calculate the average P_{RA} and the average number of threats destroyed per iteration for one thousand runs. Ten P_{RA} averages along with the average number of threats destroyed were generated and utilized to calculate a 98% confidence interval of the P_{RA} for the AAW architecture and a 98% confidence interval of the number of threats destroyed. Additionally, the average of the threat's time at intercept, range at intercept, time of flight, and the average number of weapon type used for the raid attack were calculated.

Table 4-9 summarizes the AAW architecture's MOP pertaining to the average $P_{RA,}$ average number of threats destroyed and average consumption of weapon resources. Its purpose was to compare the effectiveness of the AAW architecture against alternative

AAW architectures. The table shows that the original AAW architecture (Configuration A) with radar height at 15.24 meters, two illuminators, and the given weapon capability onboard will not satisfy the 0.99 P_{RA} requirement. The current AAW architecture can only deliver an average P_{RA} of 0.674. The average number of threats the AAW architecture can destroy is approximately 7.4 threats. Furthermore, the probability of surviving a raid attack of eight threats is virtually impossible if the current AAW architecture loses one illuminator during self-defense operation.

The model was further utilized to determine three alternatives (Configuration B, C, and D) possible of achieving the required $P_{RA.}$ The alternatives either required an increase in the number of illuminators, the probability of kill for weapon_1, or a combination of both, while maintaining the radar height of 15.24 meters in the combat system. Theoretically, the three alternatives are able to provide 0.99 P_{RA}, but consideration should be given to the feasibility of each alternative. Configuration B limits the illuminators to two but increased weapon_1's probability of kill to 0.98. The increased accuracy may not be currently achievable. Configuration C adds an additional illuminator and requires weapon_1 to be 0.90 effective. Configuration D increased weapon_1's probability of kill to only 0.85 but requires two additional illuminators which increases the overall cost.

Table 4-8 provides information regarding the utilization of the weapon resource. There were no instances in which the weapon resources were depleted. In fact, the model also determined that weapon_0 is not vital for a successful self defense mission in each of the combat system configurations. Weapon_0 was never consumed because the radar height limited the threat detection range to the range capability of weapon_1. Weapon_1 was selected over weapon_0 because the probability of kill was higher. This information suggests that the AAW architecture is overstocked with weapon_0. The AAW architecture can improve the availability of weapon_1 by reducing the number of weapon_0 carried onboard and replacing them with more weapon_1 missiles which were most frequently used.

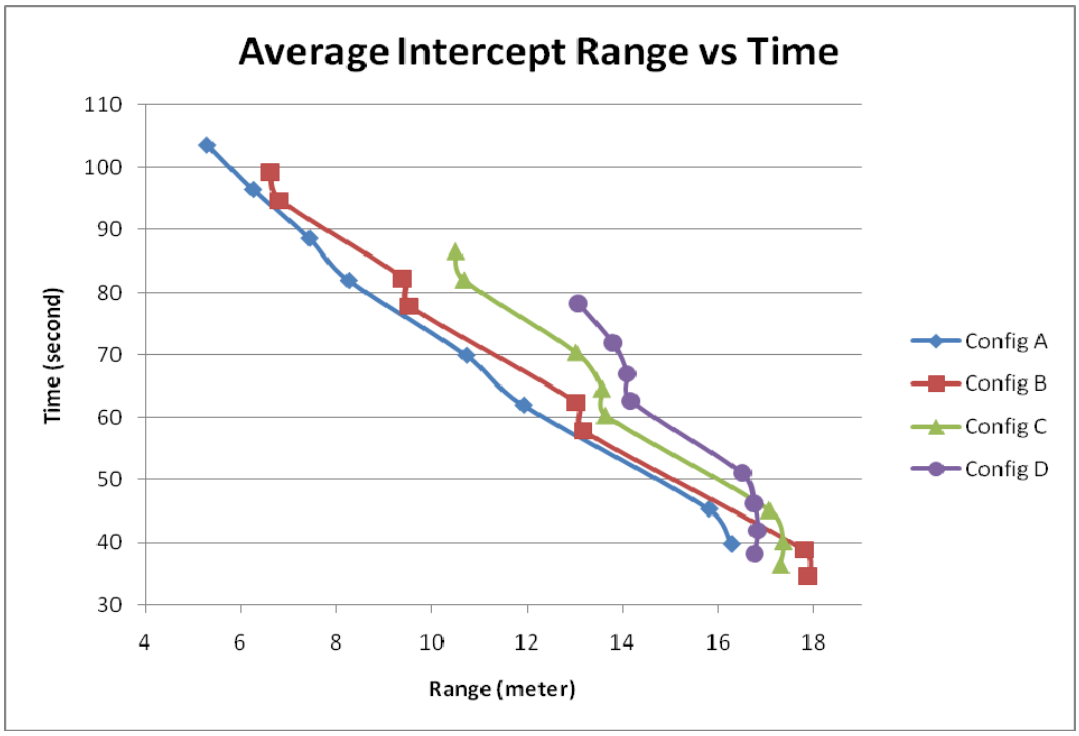
Table 4-9: 98% Confidence Interval for P_{RA} Average Threat Killed & Weapon Consumed

Summary metric of the AAW architecture's MOP pertaining to the average $P_{RA,}$ average number of threats destroyed and weapon consumed.

Combat System Configuration	98% Confidence Interval						Type of Resource	Total Equip	Avg Used	Std Dev Used	98% CI Used	
	Min P_{RA}	AVG P_{RA}	MAX P_{RA}	Min Kill	AVG Kill	MAX Kill					Min	Max
A (Original Requirement)	65.8%	67.4%	68.9%	7.369	7.397	7.425	Weapon_0 Missil	12	0	0	0	0
							Weapon_1 Missil	16	8.40	0.02	8.38	8.41
							Weapon_2 SK	20	1.10	0.03	1.08	1.12
B (Alternative 1)	98.7%	98.9%	99.1%	7.985	7.988	7.990	Weapon_0 Missil	12	0	0	0	0
							Weapon_1 Missil	16	8.09	0.01	8.09	8.10
							Weapon_2 SK	20	0.08	0.01	0.08	0.09
C (Alternative 2)	99.2%	99.4%	99.7%	7.989	7.993	7.997	Weapon_0 Missil	12	0	0	0	0
							Weapon_1 Missil	16	8.79	0.02	8.77	8.81
							Weapon_2 SK	20	0.10	0.01	0.09	0.11
D (Alternative 3)	98.7%	98.9%	99.0%	7.986	7.988	7.989	Weapon_0 Missil	12	0	0	0	0
							Weapon_1 Missil	16	9.34	0.03	9.32	9.35
							Weapon_2 SK	20	0.06	0.01	0.05	0.07

Combat System Configuration	Radar Height (m)	Illuminators	P(kill) weapon_0	P(kill) weapon_1	P(kill) Weapon_2
A	15	2	0.7	0.8	0.6
B	15	2	0.7	0.98	0.6
C	15	3	0.7	0.9	0.6
D	15	4	0.7	0.85	0.6

Figure 4-47 plots the threats' range at intercept against time of intercept for the four configurations of the AAW architecture. The corresponding data for each configuration are displayed below the plot in Table 4-10. On average, the eight threats from the raid attack are intercepted in consecutive order. That is, the first threat is intercepted earlier and further away than the second threat. The pattern remains the same when comparing the second threat with the third, the third with the fourth, and so forth. A comparison of the four configurations shows that threats were intercepted earlier and further away when there were more illuminators onboard. The illuminator tie-up time becomes less of a factor with more illuminators onboard. Configuration D has four illuminators so the threats are intercepted in two groups of four. Similarly, configuration C has three illuminators and the threats are intercepted in two groups of three plus one group of two. Configuration A and B both have two illuminators so the threats are intercepted in four groups of two. These patterns are shown in Figure 4-47. Threats were usually unable to penetrate into the two-kilometer keep-out zone.



Plots the threats range at intercept against time of intercept for an AAW architecture with various configurations.

Figure 4-47: Average Interecept Range vs. Intercept Time

Table 4-10: Intercept Time, Range, TOF, and Weapon Resource Used (1 Illuminator)

Provides information regarding the utilization and availability of the weapon resource.

Combat System Configuration	98% Confidence Interval											
	Threat	Time of Intercept (sec)			Range at Intercept (km)			Threat TOF (sec)			Weapon Choice	
		Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
A Original Requirement	1	39.422	39.764	40.106	16.184	16.289	16.394	39.422	39.764	40.106	1.031	1.035	1.040
	2	44.924	45.316	45.708	15.694	15.814	15.934	40.924	41.316	41.708	1.028	1.034	1.039
	3	61.735	62.010	62.285	11.843	11.927	12.011	53.735	54.010	54.285	1.026	1.031	1.037
	4	69.655	69.926	70.196	10.645	10.728	10.811	57.655	57.926	58.196	1.027	1.031	1.035
	5	81.782	81.989	82.197	8.195	8.259	8.322	65.782	65.989	66.197	1.056	1.060	1.065
	6	88.498	88.678	88.859	7.380	7.435	7.491	68.498	68.678	68.859	1.162	1.172	1.182
	7	96.486	96.543	96.600	6.234	6.252	6.269	72.486	72.543	72.600	1.227	1.237	1.246
	8	103.592	103.718	103.845	5.241	5.280	5.318	75.592	75.718	75.845	1.164	1.176	1.189
B Alternative 1		Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
	1	34.425	34.534	34.644	17.857	17.890	17.924	34.425	34.534	34.644	1.000	1.000	1.001
	2	38.683	38.816	38.949	17.764	17.804	17.845	34.683	34.816	34.949	1.000	1.000	1.001
	3	57.859	57.911	57.963	13.166	13.182	13.198	49.859	49.911	49.963	1.000	1.000	1.001
	4	62.329	62.412	62.495	13.003	13.029	13.054	50.329	50.412	50.495	1.000	1.000	1.001
	5	77.763	77.813	77.863	9.522	9.537	9.553	61.763	61.813	61.863	1.000	1.001	1.001
	6	82.241	82.315	82.389	9.361	9.384	9.407	62.241	62.315	62.389	1.016	1.018	1.021
	7	94.695	94.707	94.719	6.810	6.814	6.818	70.695	70.707	70.719	1.011	1.015	1.018
	8	99.295	99.360	99.425	6.594	6.614	6.634	71.295	71.360	71.425	1.013	1.016	1.019
C Alternative 2		Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
	1	35.819	36.445	37.071	17.114	17.305	17.497	35.819	36.445	37.071	1.000	1.001	1.002
	2	39.698	40.273	40.849	17.182	17.358	17.534	35.698	36.273	36.849	1.005	1.008	1.010
	3	44.369	45.223	46.078	16.805	17.067	17.329	36.369	37.223	38.078	1.005	1.009	1.012
	4	59.888	60.425	60.962	13.473	13.637	13.802	47.888	48.425	48.962	1.005	1.008	1.010
	5	64.061	64.642	65.223	13.393	13.571	13.749	48.061	48.642	49.223	1.003	1.006	1.009
	6	69.385	70.422	71.458	12.708	13.026	13.343	49.385	50.422	51.458	1.004	1.005	1.007
	7	81.522	82.085	82.648	10.507	10.679	10.851	57.522	58.085	58.648	1.006	1.010	1.014
	8	85.939	86.662	87.384	10.281	10.502	10.724	57.939	58.662	59.385	1.007	1.010	1.013
D Alternative 3		Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
	1	37.983	38.205	38.428	16.698	16.766	16.834	37.983	38.205	38.428	1.002	1.004	1.005
	2	41.696	41.974	42.253	16.752	16.837	16.922	37.696	37.974	38.253	1.001	1.002	1.004
	3	46.052	46.218	46.384	16.712	16.762	16.813	38.052	38.218	38.384	1.002	1.003	1.005
	4	50.768	51.055	51.342	16.418	16.506	16.594	38.768	39.055	39.342	1.002	1.003	1.004
	5	62.470	62.679	62.888	14.108	14.172	14.236	46.470	46.679	46.888	1.004	1.005	1.007
	6	66.703	66.927	67.151	14.027	14.096	14.164	46.703	46.927	47.151	1.002	1.003	1.004
	7	71.736	71.908	72.081	13.743	13.795	13.848	47.736	47.908	48.081	1.002	1.003	1.004
	8	78.108	78.297	78.485	13.006	13.064	13.122	50.108	50.297	50.485	1.009	1.011	1.013

Combat System Configuration	Radar Height (m)	Illuminators	P(kill) weapon_0	P(kill) weapon_1	P(kill) Weapon_2
A	15	2	0.7	0.8	0.6
B	15	2	0.7	0.98	0.6
C	15	3	0.7	0.9	0.6
D	15	4	0.7	0.85	0.6

4.5.5 Supportability in M&S

4.5.5.1 Objective

A key objective of this project is to incorporate supportability elements as an essential requirement in the design and development of systems to perform as required in their operational environment. The model can support program objectives by demonstrating the cause and effect of graceful degradation, redundancy, reliability, and other supportability factors as trade-offs and decisions regarding cost, performance, and availability are made. By modeling the design alternatives and performing trade-off analyses, the model will provide decision makers with the data required to select the optimum system architecture.

4.5.5.2 Assumptions

Prior to model development, supportability requirements were addressed as part of the system requirements and identified as KPPs. The model represents an AAW architecture with the minimum amount of hardware components required to meet the 0.99 P_{RA}. The hardware (radar, illuminators, and weapons) is assumed to have built-in redundancy and reliability which will provide for an Ao sufficient to meet the 0.99 P_{RA}. Due to limitations within the model, certain assumptions were made. For example, there is only one radar in the model. However, it is recommended to plan for either a second radar of the same type or a backup radar that could compensate for the lost functionality should the primary fail. The designer could alternatively build enough redundancy and health monitoring into a single radar to meet required reliability and A_O.

4.5.5.3 Limitations

There is limited hardware, software, data, and personnel (manpower/training) associated with the model, and therefore only a portion of the supportability aspects of the systems architecture are represented in it. MTBF, MTTR, MLDT, mean corrective maintenance time, mean preventive maintenance time, and other factors that relate to A_O and readiness

were not available for the AAW architecture model. Since this was an academic exercise, the scope of the model does not consider many of the supportability factors that should be modeled early in the system architecture design and development process. However, it is recommended that all applicable supportability factors be modeled when applying this methodology to an actual program.

4.5.5.4 Observations of the Model

The radar represented in the model is a single-point failure, yet it is assumed that there is built-in redundancy and inherent reliability to ensure probability of success. Modeling could be utilized to demonstrate the effect of a second radar and/or the amount of redundancy or graceful degradation required within the single radar to achieve the required A_O. Parallel and/or serial redundancies in system architectures are design decisions that need to be made based on sound reliability data and modeling.

Redundancy not only drives costs in the initial system procurement, it also increases the cost of system support, due to more spare components to supply and support. Therefore, if the primary system has a reliability that meets the 0.99 P_{RA}, it wouldn't be fiscally prudent or technically required to build in any additional redundancy. The weapon selection sub-function within the model is sufficiently reliable and redundant to deploy either an Evolved Sea Sparrow Missile (ESSM), Standard Missile (SM), or chaff to kill or divert the threat. The weapon selection sub-function enters a loop so that if there is a misfire, equipment failure, or failure to destroy the intended target, another weapon will be selected.

The two illuminators in the system represent the minimum system hardware required to meet the threats presented in the given scenarios. With two illuminators, the system will be available to engage two targets simultaneously. Should one of the illuminators fail, the P_{RA} would not meet 0.99. This presents a dilemma similar to that of the radar. The model could be utilized to establish scenarios in which various levels of redundancy,

graceful degradation, and/or reliability are modeled, and trade-off analyses are provided to the decision makers as they establish the optimum system architecture.

System health monitoring and diagnostics will be present within the system to detect and isolate failures and switch to the alternate/redundant system. Additionally, system health monitoring would be used to detect failures within systems and equipment and accomplish such functions as allowing for a specified amount of performance degradation, using redundant equipment, and/or rebooting the system. It is assumed that adequate spares are available onboard to correct failures, the sailors will be properly trained to identify and make repairs, and the technical documentation will be sufficient to troubleshoot the faults presented.

The missiles are wooden-round and high reliability is anticipated. There are multiple numbers of missile types as well as multiple types of countermeasure systems onboard ship. The launching system is comprised of two 8-Cell VLS Launchers. A high reliability of the launchers will be achieved. Analysis of the reliability requirements would identify the level of redundancy required to meet the operational availability. There is built-in redundancy so that if one cell does not work (cell hatch, misfire, etc.), a secondary cell with the same type of missile will be selected.

4.5.6 M&S Summary

During the initial development phase, Extend access, research, and familiarization occurred concurrent with defining M&S objectives (validating a developed methodology to support AAW combat systems). The primary objective filtered into customer inputs (top-level requirements). It was decided that a 0.99 P_{RA} would be used as the determining factor to V&V the architecture. Functional concepts for the combat systems architecture derived from the requirements. MOEs were established following conceptual functions development. The MOEs measure the models components that make up the 0.99 P_{RA}.

Through the Extend model example of the Self Defense scenario, it was observed that Configuration A (as described in Section 4.5.4) failed to meet 0.99 $P_{RA.}$ Common engineering practices were implemented to pinpoint the reason(s) of failure. Through data analysis and investigation, the reasons for failure included having a limited number of illuminators in the architecture and having weapons with low probabilities of kill. In Configurations B, C, and D, the Extend model example showcased an increase in the number of illuminators and also a variance of the probability of kill for weapon_1. Through many runs and iterations, all three configurations surpassed the requirement of 0.99 P_{RA} (results in Table 4-9).

Tradeoffs were required to achieve high probabilities of raid annihilation. These tradeoffs included increasing the quality of kill from a weapon (increased probability of kill), increasing number of illuminators (providing more weapon_1 engagements), increasing the radar height (increases threat target detection), modification of original secondary objective (2 km keep-out zone), and modifying the weapons salvo policy (employ multiple layers of defenses). Slight changes in decision making will impact the architecture's modeling and simulation results, and in turn provide the customer (warfighter) with the best plan of attack against a given self-defense scenario.

4.6 SUB PROCESS METHODOLOGY USING CORE

4.6.1 CORE Introduction

The originating set of documents available for developing the AAW architecture were the AAW ConOps document, the SysML Requirements diagram, and the AAW Architecture Functional Behavior diagram. The SysML Requirements diagram was developed in Visio and the Functional Behavior diagram was developed using Excel and Word. These documents and diagrams were used as the basis for populating the CORE database. Once the CORE database was populated, the tool was used to generate an executable AAW architecture with traceability back to the SysML Requirements diagram and Functional Behavior diagram. Additionally, CORE was used to generate a set of DoDAF Views,

diagrams, and systems engineering reports, and verify the AAW architecture was executable using COREsim.

The following paragraphs discuss how the MBSE approach to Target System Specification and Target System Architecture Generation was used in CORE to develop the AAW architecture and DoDAF artifacts.

4.6.2 MBSE Process Using CORE

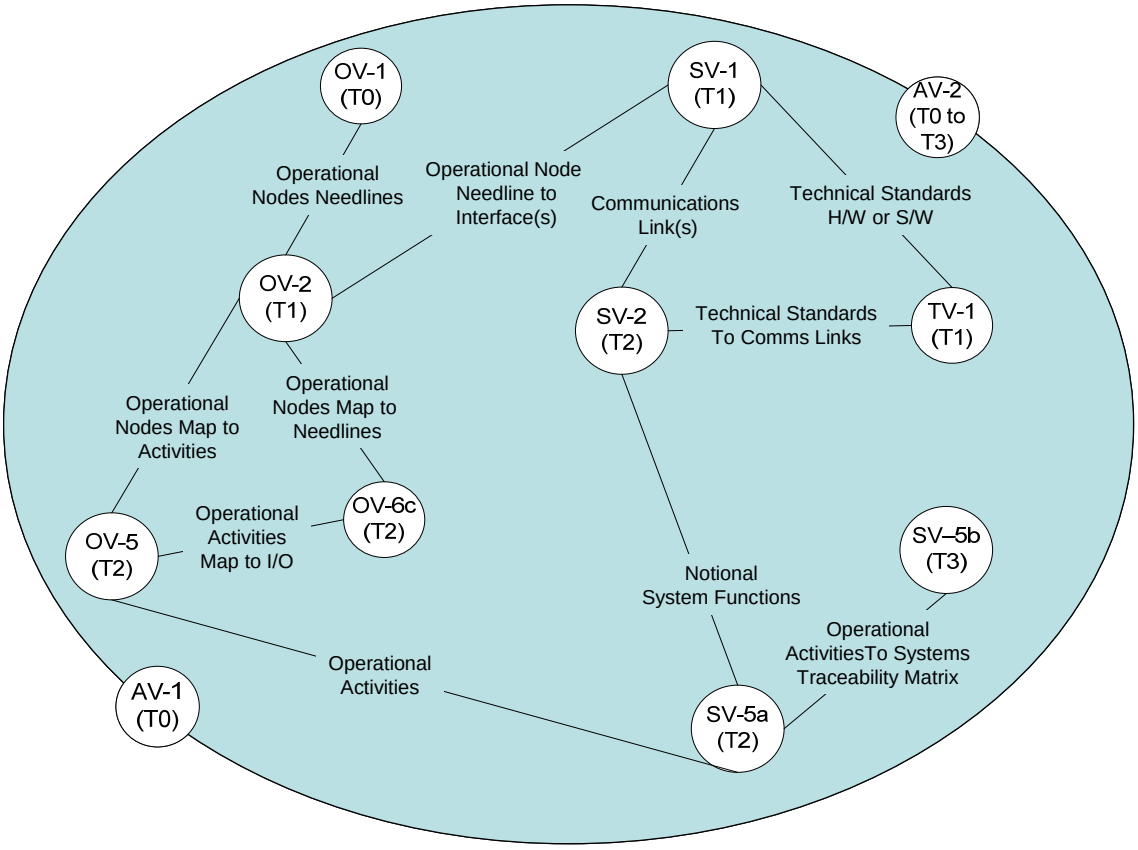
The scope of this effort was to develop the AAW architecture and DoDAF views using CORE. The following sections describe the process used to develop the AAW architecture and DoDAF v1.5 views using the CORE Workstation 5.1.5 tool, which is a single-user version of CORE.

4.6.3 Requirements Generation and Analysis (Process 1)

To use the CORE tool to develop the AAW architecture, it was necessary to enter the problem statement into the CORE database. This was done by using the Element Extractor function of CORE to import the AAW ConOps. The ConOps document file was captured using the External File Path attribute in the ExternalFile Class. This established a hyperlink between the original document and the CORE database. The size of the CORE database can be kept small through use of hyperlinks. (Dam 2006) By using this process, CORE can serve as a repository for the SysML Use Cases and Sequence Diagrams, SPL artifacts, etc.

The AAW requirements used in CORE were derived in part from the SysML Requirements diagrams. The process of developing the AAW architecture began by taking the AAW requirements defined in the SysML diagrams and entering them into the CORE database. The CORE program contains a DoDAF schema with predefined

Classes (e.g., Architecture, Component, Function, etc.) and Attributes (i.e., critical properties of elements) that can be used to capture and later execute (with COREsim) the architecture. Requirements capture continued throughout the development process. The requirements analysis process in CORE allows for the capture of issues and components. Issues include information on problems that need to be resolved as part of the analysis process. Although the AAW architecture is not an existing system, it was necessary to define physical components in the Component Class of CORE to enable the tool to produce some of the DoDAF views such as SV-1 (Systems/Services Interface Diagram). Figure 4-48 illustrate the DoDAF views that were to be developed for the AAW architecture. The views were to be developed in tiers, with T0 being the first tier and then T1 and so on.



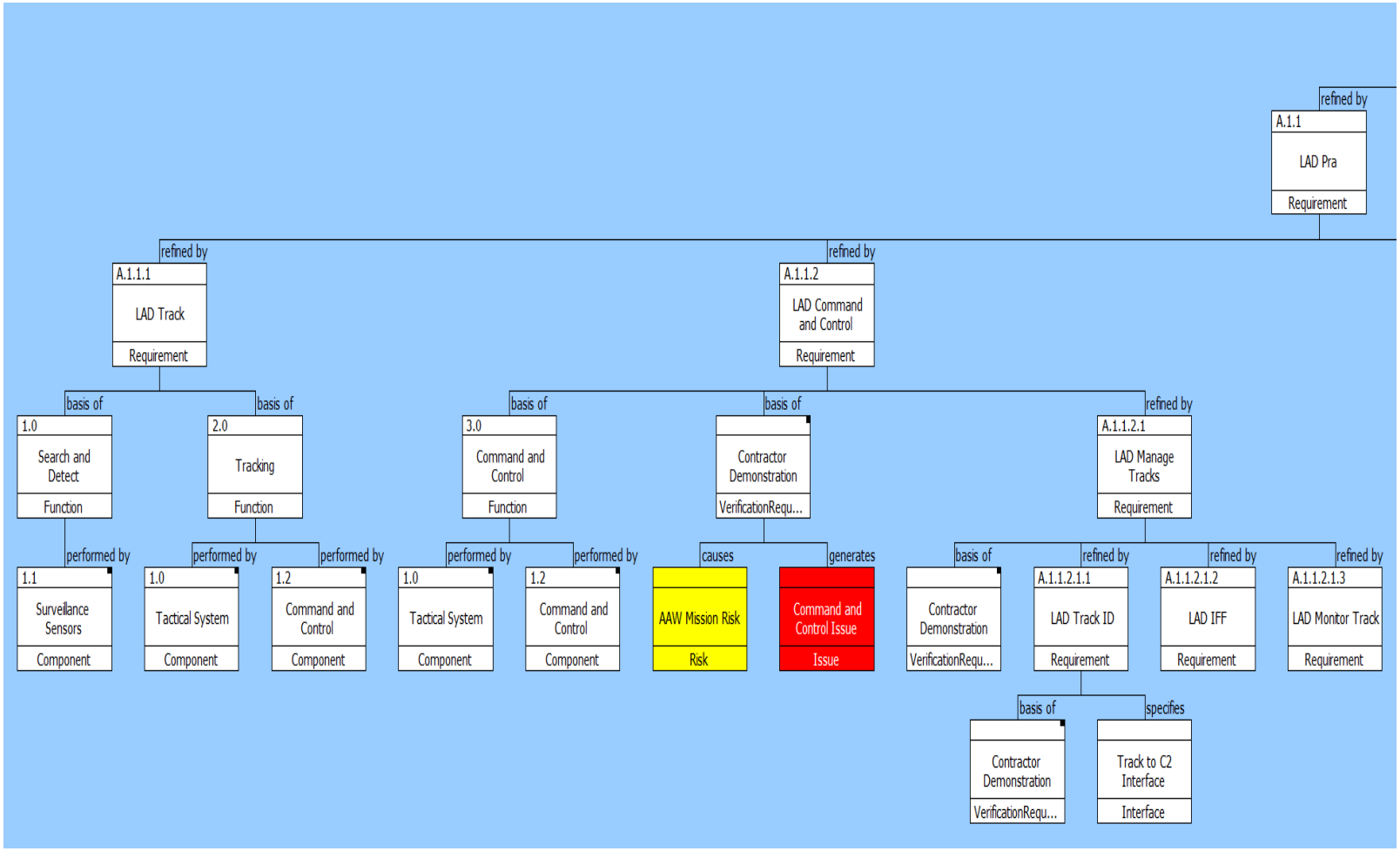
This depicts the DoDAF products produced in support of the Capstone objectives.

Figure 4-48 DoDAF Products Relationships

4.6.3.1 Artifacts Produced

CORE dynamically generates diagrams directly from the database repository ensuring that they are consistent with the design details. A change made to the database is automatically reflected in the views. (Vitech Corp. 2007c) The Requirements Hierarchy Diagram was developed in CORE from the SysML Requirements diagram. The elements (i.e., objects or entities) used to populate a class in CORE were taken directly from the SysML diagram. Since the SysML Requirements diagram did not provide all the attributes that CORE requires, many of the attributes had to be created as part of the CORE development process. As such, there is no direct traceability of the attributes from CORE to SysML. Several of the SysML requirement names were identical and had to be changed to enter them into the database because CORE software requires that all elements have unique names. The SysML requirements did not have all the relationships defined, so it was necessary to define them in the CORE database (a relationship defines a link between two elements; in this case, between two requirements). The team had to ensure that the correct relationships and attributes were entered in the CORE database to generate the DoDAF views (artifacts).

DoDAF OV-1 (Operational Concept Graphic) was produced by establishing a hyperlink from the CORE database to the file containing the OV-1 graphic. The DoDAF OV-1 view is illustrated in Figure 4-15. The AAW Requirements Traceability Diagram was generated in CORE. A partial view (the complete diagram cannot be shown on this size of paper) of the diagram is shown in Figure 4-49. It can be seen from the diagram that elements such as components, functions, verification requirements, interfaces, issues, and risks can all be traced back to the originating requirements. Diagrams produced by CORE automatically show issues in red and risks in yellow. OV-2 (Operational Node Connectivity) was developed by defining the elements and associated needlines and operational nodes. The information required to build this view in CORE was not provided in the SysML Requirements Diagram. To produce SV-1 (Systems/Services Interface Diagram), the links between the components (physical) had to be defined and entered into the Component Class in the CORE database.



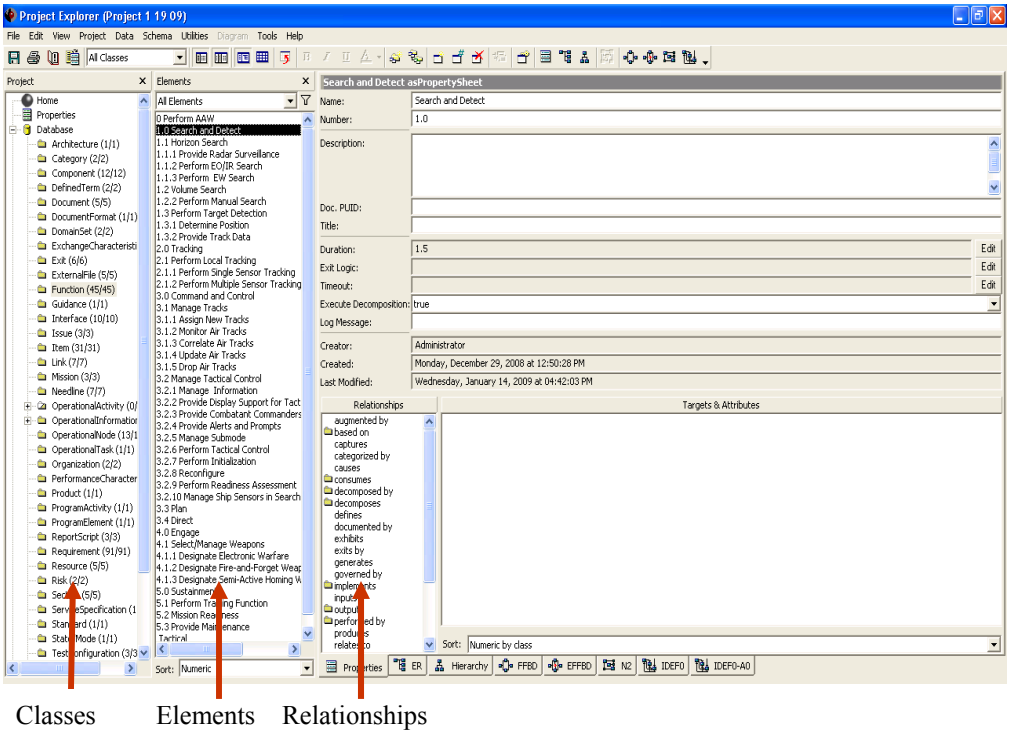
This figure shows a partial view of the AAW Requirements Traceability Diagram generated by CORE. It can be seen from the diagram that elements such as components, functions, verification requirements, interfaces, issues and risks can be traced back to the originating requirements. Note: Risks are shown in yellow and issues are shown in red.

Figure 4-49: AAW Requirements Traceability Diagram

4.6.4 Functional Analysis and Allocation (Process 2)

The AAW Functional Behavioral model was built in CORE in part from the AAW Functional Architecture diagram. The AAW Functional Architecture Model functions and hierarchy were entered into the CORE database in the Function Class. All functions with similar names were changed to make them unique so they could be entered into the CORE database. To produce the DoDAF views and to verify the architecture with CORE within the given time constraints of this project, no functional requirements at a level of indenture deeper than three were entered into CORE. The Functional Behavior diagram also lacked the proper relationships and attributes required by CORE. Figure 4-50 shows an example of the Classes, Elements, and Relationships for the Perform AAW Function. The missing relationships and attributes were created in the CORE database. Behavior analysis for the AAW architecture was performed by building the Functional Flow Block Diagrams (FFBDs) in CORE. FFBDs show functional flow including control logic. (Vitech Corp. 2007c) N-squared (N2) diagrams and Enhanced FFBDs (EFFBDs) were also generated. An N2 diagram shows the data flow between functions. EFFBDs show functional flow, control logic, inputs, outputs, and triggers. Used in conjunction with an FFBD, the N2 diagram helps to capture and analyze the functional behavior of the system. (Vitech Corp. 2007c) Function inputs, outputs and triggers were not provided in the AAW Functional Architecture diagram and had to be defined and added to the EFFBDs. Additionally, functions were allocated to components as part of the CORE development process.

In addition to the Requirements and Function Classes, another 39 classes were created in the CORE database. These classes included Component, Interface, Item, Link, Needline, Operational Activities and Nodes, and Resources, to name a few. A total of 291 different elements and 38 different relationships were developed.



CORE screen capture showing Classes, Elements and Relationships.

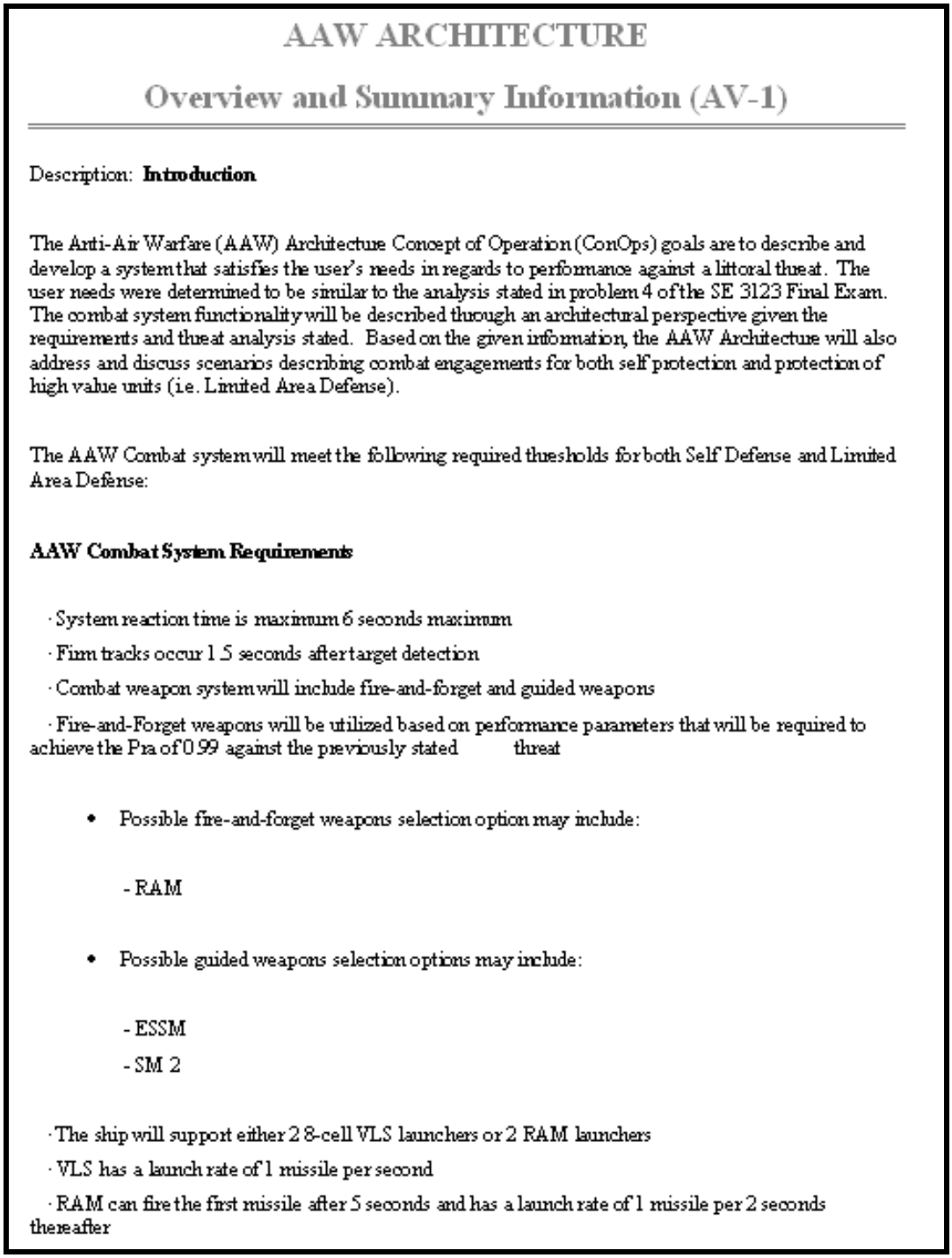
Figure 4-50: CORE Screen Capture

4.6.4.1 Artifacts Produced

The CORE project file containing the database (not included in this report) was produced in eXtensible Markup Language (XML) format. CORE offers the user a choice of different file output formats for the DoDAF artifacts, diagrams, and reports. Rich Text Format (RTF) was selected for the DoDAF views and reports and Windows Metafile (WMF) was selected for the diagrams. CORE will also produce documents in Hypertext Markup Language (HTML) format. The following are the DoDAF v1.5 views and reports generated in CORE and provided in Appendix M of this report (the nomenclature used is consistent with the one used by the CORE tool):

- **Overview and Summary Information [AAW Architecture] (AV-1):** AV-1 was developed by using the Element Extractor feature of CORE to capture the relevant sections from the AAW ConOps. It contains the high-level description, operational concept, operational scenarios, scope, time frame, and mission of the AAW architecture. Figure 4-51 is a screen capture of the first page of the DoDAF

AV-1 view (AV-1 contains a total of five pages). The header and footer information is automatically generated by the CORE Run Script feature.



AAW ARCHITECTURE

Overview and Summary Information (AV-1)

Description: **Introduction**

The Anti-Air Warfare (AAW) Architecture Concept of Operation (ConOps) goals are to describe and develop a system that satisfies the user's needs in regards to performance against a littoral threat. The user needs were determined to be similar to the analysis stated in problem 4 of the SE 3123 Final Exam. The combat system functionality will be described through an architectural perspective given the requirements and threat analysis stated. Based on the given information, the AAW Architecture will also address and discuss scenarios describing combat engagements for both self protection and protection of high value units (i.e. Limited Area Defense).

The AAW Combat system will meet the following required thresholds for both Self Defense and Limited Area Defense:

AAW Combat System Requirements

· System reaction time is maximum 6 seconds maximum

· Firm tracks occur 1.5 seconds after target detection

· Combat weapon system will include fire-and-forget and guided weapons

· Fire-and-Forget weapons will be utilized based on performance parameters that will be required to achieve the Pra of 0.99 against the previously stated threat

- Possible fire-and-forget weapons selection option may include:

  - RAM

- Possible guided weapons selection options may include:

  - ESSM
  - SM 2

· The ship will support either 2 8-cell VLS launchers or 2 RAM launchers

· VLS has a launch rate of 1 missile per second

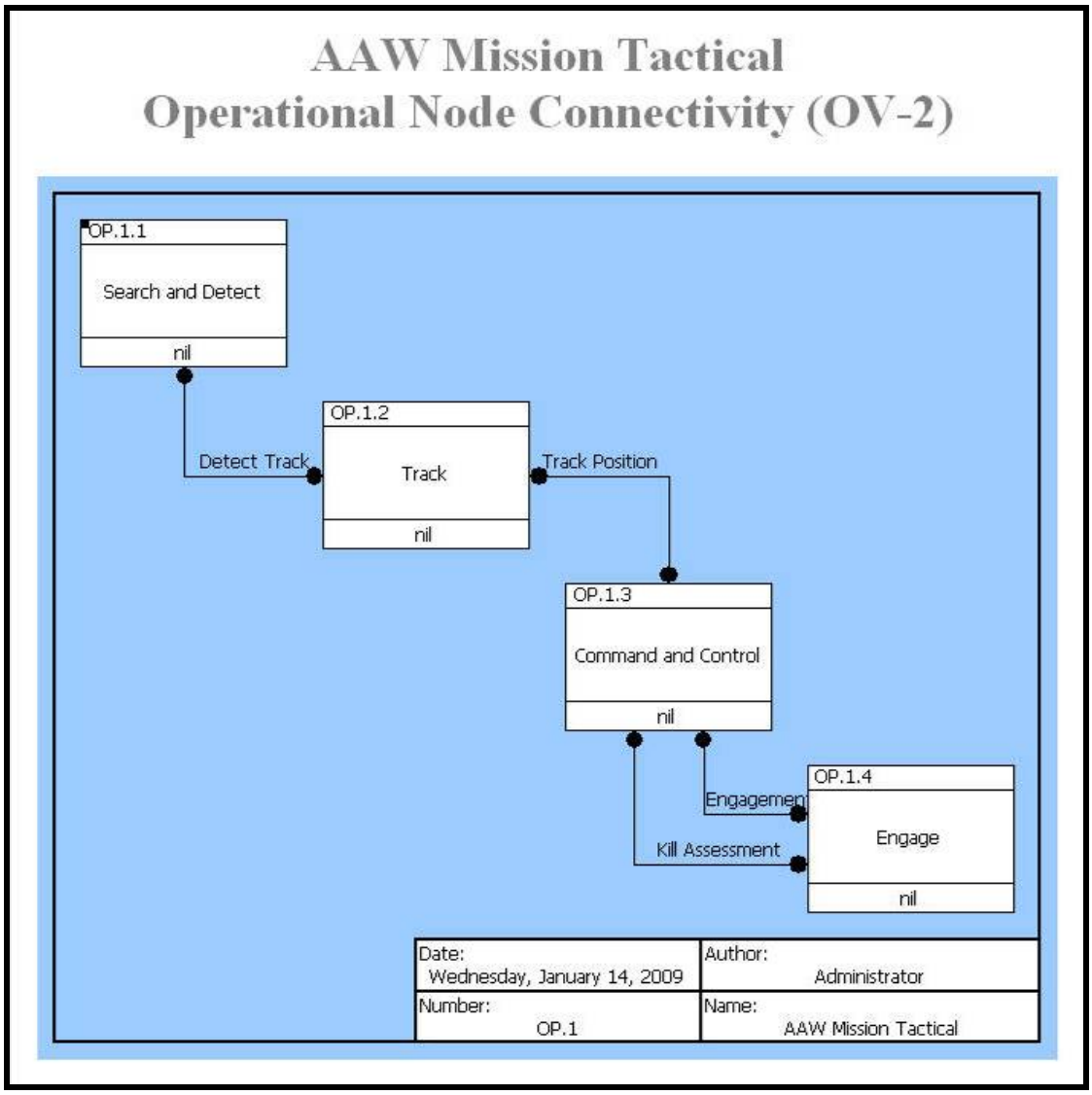
· RAM can fire the first missile after 5 seconds and has a launch rate of 1 missile per 2 seconds thereafter

CORE screen capture of the first page of the AV-1 DoDAF View.

Figure 4-51: AV-1 DoDAF View Screen Capture

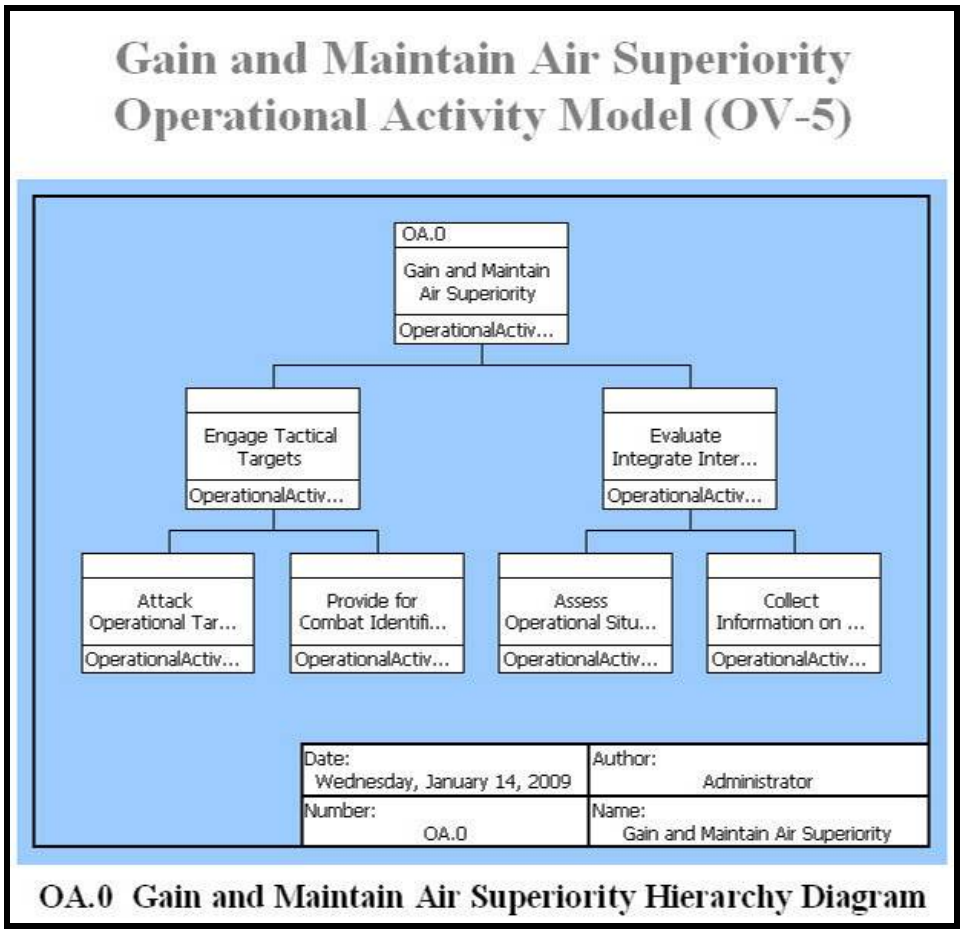
- **Architecture Description Document for AAW Architecture (AV-2):** AV-2 was generated automatically by CORE using the Run Script feature. This document includes an architecture description, guidance section, operational overview, relevant operational nodes, operational connectivity needs, activity model, systems overview, systems/components section, interfaces, functional model, item dictionary, functional model resources, issues and decisions, risks, acronyms, and glossary.
- **Operational Concept (OV-1):** OV-1 was captured in CORE from a graphics file using the External File Path attribute. This feature creates a hyperlink to the file it uses to bring in the graphic when a DoDAF view is generated. Figure 4-15 shows the DoDAF OV-1.
- **Operational Node Connectivity Description (OV-2):** OV-2 provides a diagram that shows the connectivity between the Search and Detect, Track, Command and Control, and Engage operational nodes. The lines between the operational nodes are labeled with the names of the data that flows between the nodes. OV-2 also provides a table that lists each element and the definition of each needline associated with the element and operational node. Figure 4-52 shows a Physical Block Diagram for the AAW Mission Tactical Operations Node that illustrates the operational nodes, needlines, and data flow exchange.
- **Operational Activity Model (OV-5):** OV-5 was developed by defining Operational Activities and Operational Nodes. The view provides hierarchy, Integration Definition for Function Modeling (IDEF0), EFFBD, and N2 diagrams. It also provides an Associated Element Definition table that lists the elements and their associated operational activity definitions. Figure 4-53 shows the Gain and Maintain Air Superiority Operational Activity Hierarchy diagram for the OV-5 DoDAF View.
- **Operational Activity Sequence Model (OV-6):** OV-6 produced FFBDs, EFFBDs, and N2 diagrams for the Gain and Maintain Air Superiority operational activity. Figure 4-54 shows the Gain and Maintain Air Superiority Operational Activity FFBD for OV-6. The reference nodes, shown as grey blocks in Figure 4-54, indicate the source and sink of the control flow (i.e., the function before and

the function after the function flow shown by the two nodes in the middle). Since there is no source or sink for the Gain and Maintain Air Superiority context, they are labeled as "Ref".



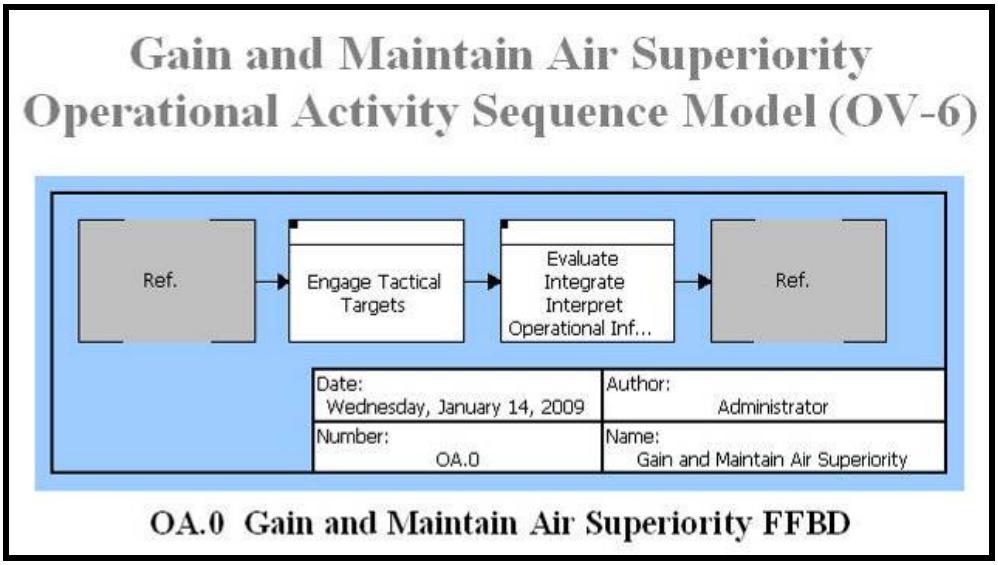
CORE screen capture of the title page of the AV-2 DoDAF View.

Figure 4-52: OV-2 DoDAF View Physical Block Diagram Screen Capture



CORE screen capture showing the Gain and Maintain Air Superiority Operational Hierarchy diagram (OV-5).

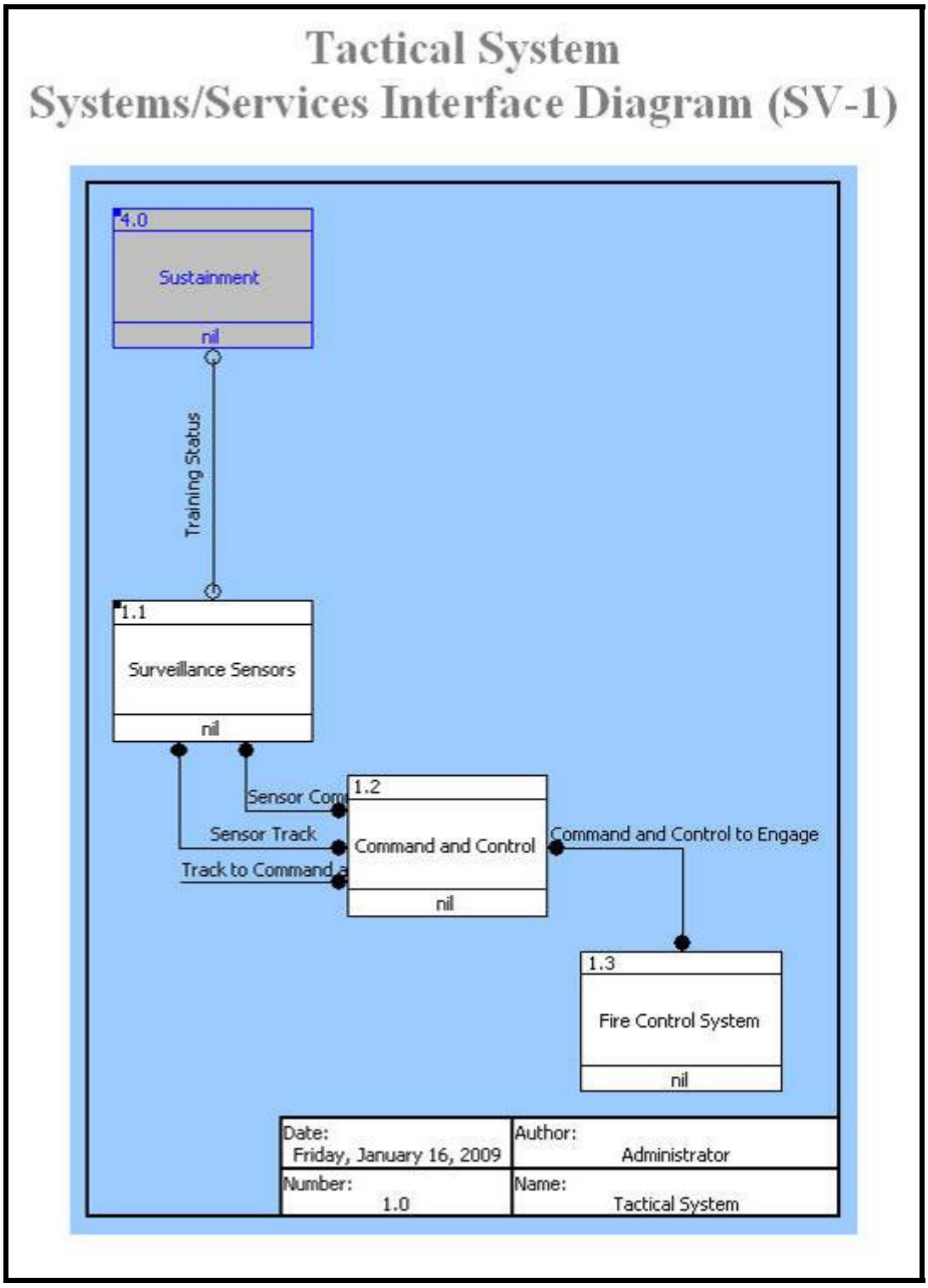
Figure 4-53: OV-5 DoDAF View Hierarchy Diagram Screen Capture



CORE screen capture showing the Gain and Maintain Air Superiority Operational FFBD (OV-6).

Figure 4-54: OV-6 DoDAF View EFFBD Screen Capture

- **Systems/Services Interface Diagram (SV-1):** SV-1 produced diagrams that show the interface and communications links and data flow between the Sustainment, Surveillance Sensors, Command and Control, and Fire Control System components. Figure 4-55 shows the Physical Block Diagram for the Tactical System that illustrates the needlines and data flow exchange. The data exchanged is labeled with the name of the data.
- **Systems/Services Communication Diagram (SV-2**): SV-2 also produced diagrams that show the interfaces, communication links, and data flow between the Sustainment, Surveillance Sensors, Command and Control, and Fire Control System components. The SV-2 physical block diagram for the Tactical System is identical to the one shown in Figure 4-55.
- **Operational Activity to Systems Function Traceability Matrix (SV-5a):** SV-5a produces a traceability matrix. Figure 4-56 shows the traceability between the Command and Control, Fire Control System, and Surveillance Sensors, and the Engage Tactical Targets and Evaluate Integrate Intercept Interpret Operational Information operational activities.
- **System Description Document for Tactical System:** This was generated automatically by CORE using the Run Script feature. This document includes a component overview, originating requirements, design constraints, issues and decisions, risks, functional behavior model, item dictionary, resources, components, interfaces, requirements traceability matrix, and acronyms sections.



CORE screen capture showing Tactical System Physical Block Diagram for the SV-1 DoDAF View.

Figure 4-55: SV-1 DoDAF View Physical Block Diagram Screen Capture

Gain and Maintain Air Superiority to Tactical System Operational Activity to Systems Function Traceability Matrix (SV-5)

Component	Function	Operational Activity	
		Engage Tactical Targets	Evaluate Integrate Interpret Operational Information
Command and Control [WS 21340]	Command and Control	X	X
	Engage	X	
	Tracking		X
Fire Control System [WS-31333]	Engage	X	
Surveillance Sensors [WS 21200]	Horizon Search		X
	Search and Detect		X

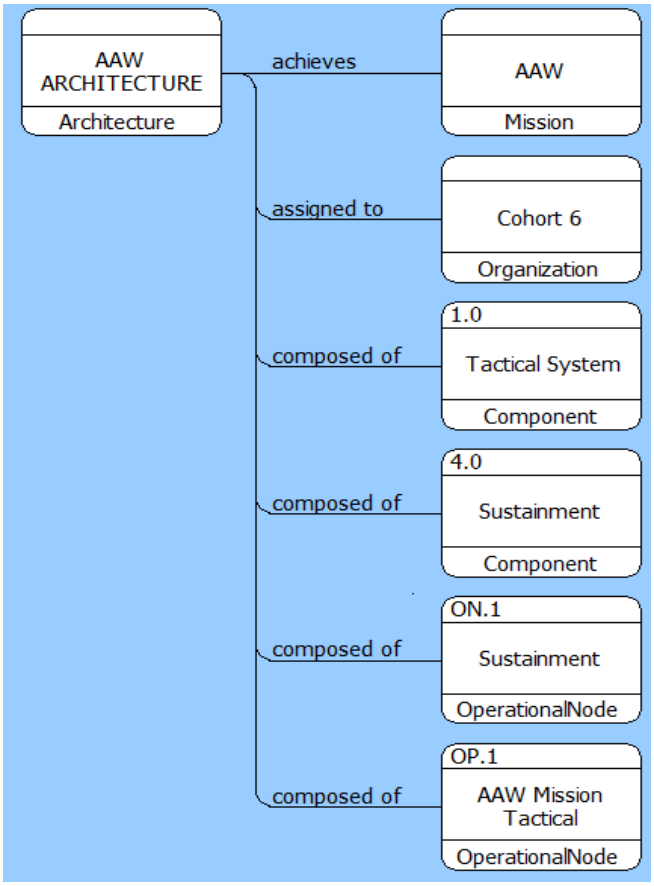
CORE screen capture showing Gain and Maintain Air Superiority to Tactical System Operational Activity to Systems Function Traceability Matrix for the SV-5a DoDAF View.

Figure 4-56: SV-5a DoDAF View Traceability Matrix Screen Capture

In addition to the DoDAF artifacts discussed, the following were also produced but are not included in this report or in Appendix M:

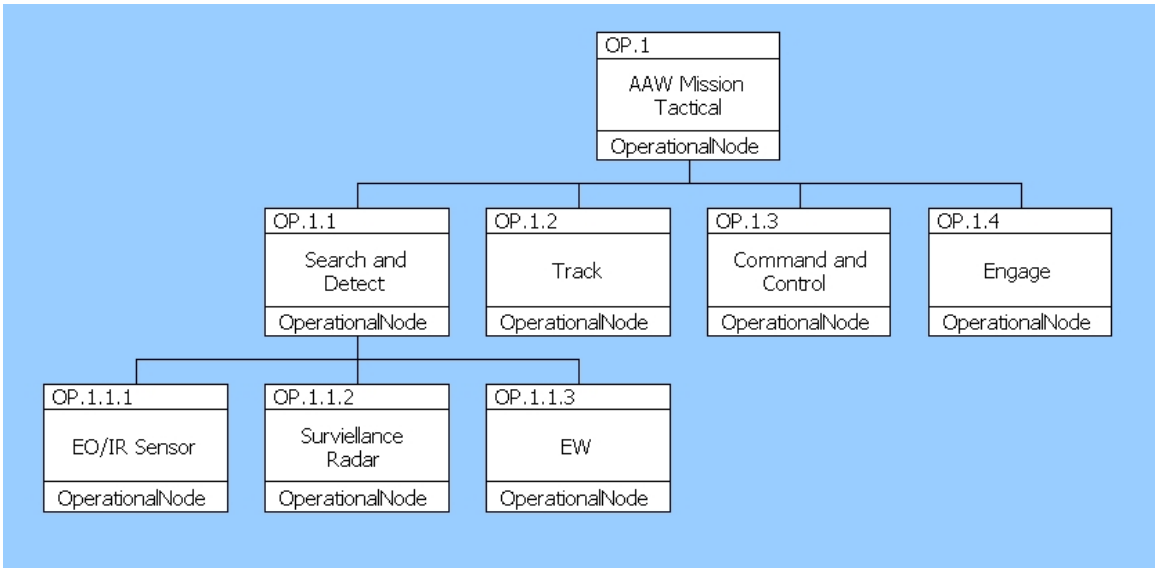
- Summary Operational Information Exchange Matrix (OV-3)
- Summary Systems Data Exchange Matrix (SV-6)
- Systems/Services Functionality Sequence Model (SV-10)
- IDEF0 Node Index Reports
- Schema Definition Report for All Classes
- Test and Evaluation Plan (TEP) for the Surveillance Sensors (draft)
- Database Statistics Report (All Classes Facility Statistics)

Over 150 different diagrams (not all of which are included in this report) were developed in CORE. A few examples of the Element Relationship (ER), Hierarchy, FFBDs, EFFBD, and N2 diagrams produced are illustrated in Figures 4-57 through 4-61.



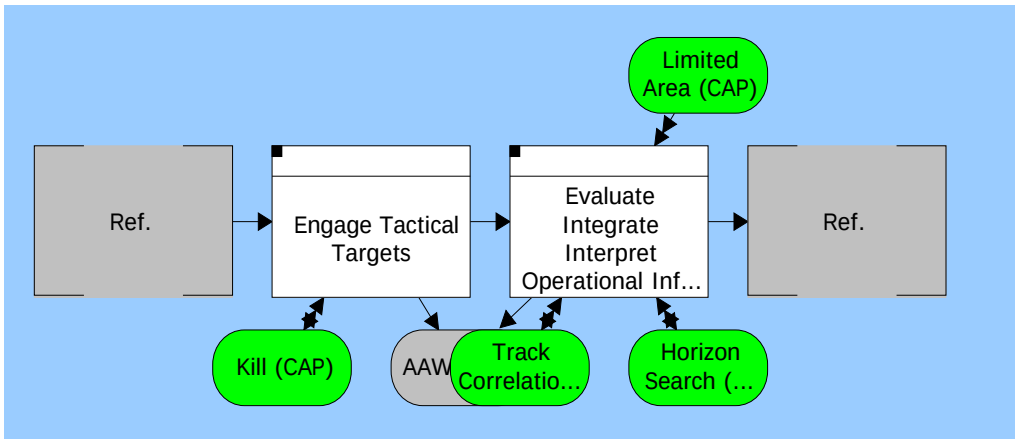
CORE screen capture showing the ER Diagram for the AAW Architecture Element.

Figure 4-57: ER Diagram



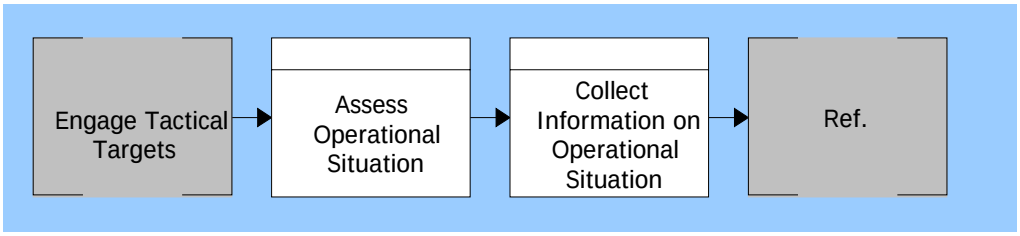
CORE screen capture showing the Hierarchy Diagram for the AAW Mission Tactical Operational Node.

Figure 4-58: Hierarchy Diagram



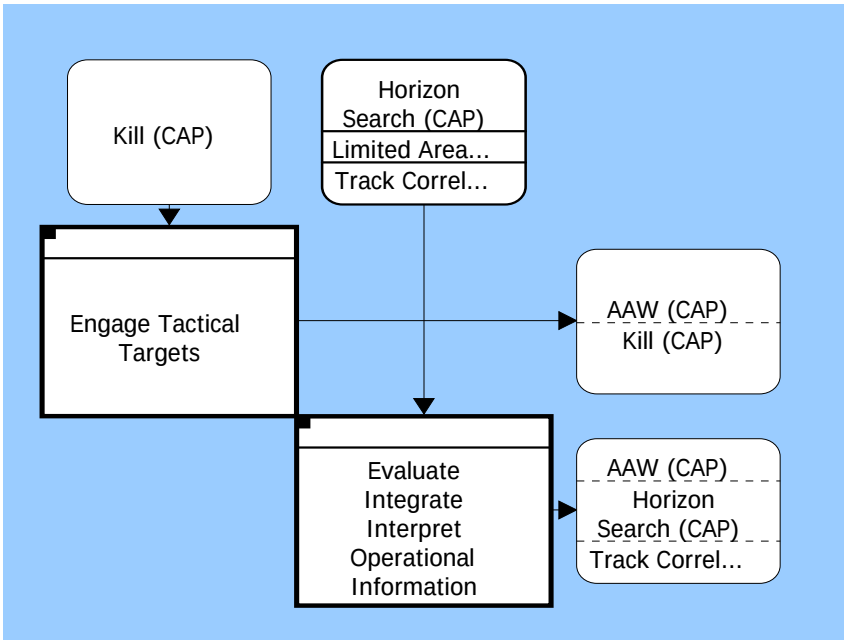
Illustrates the CORE output for the Enhanced FFBD for the Gain and Maintain Air Superiority Function. Triggering events are indicated in green.

Figure 4-59: Gain and Maintain Air Superiority Enhanced FFBD



Illustrates the CORE output for the FFBD for the Evaluate Integrate Interpret Operational Information Function.

Figure 4-60: Evaluate Integrate Interpret Operational Information FFBD



Illustrates the CORE output for the N2 Diagram for the Gain and Maintain Air Superiority Function.

Figure 4-61: Gain and Maintain Air Superiority N2 Diagram

4.6.5 Architecture Definition (Process 3)

The development of the AAW architecture using CORE was hampered by the team's unfamiliarity with how to use CORE and its limited knowledge of how to define a system architecture. As stated earlier, the ConOps, requirements, and functions were entered into the CORE database without following any structured method or process. It soon became obvious that much more information was required to define the AAW architecture than what was available in the documentation. As a result, the AAW architecture definition was incomplete and none of the DoDAF views could be generated initially. Once the missing information was entered, the AAW architecture was defined sufficiently enough to allow CORE to generate some of the required DoDAF views. After additional data was entered, it became possible to run a simulation of some of the functions of the AAW architecture. The process used to define and develop the AAW architecture in CORE was not optimal and could be improved. The following paragraphs provide a structured approach to populating the CORE database using the DoDAF schema to develop an architecture that is executable and can be used to produce DoDAF views. (Vitech Corp. 2007a) This approach is provided in the CORE Architecture Definition Guide published by Vitech.

In CORE, an architecture is defined as consisting of an operational architecture domain and a systems architecture domain. "The operational domain is used to capture originating concepts, capabilities, and the supporting operational analysis to expose the requirements leading to, and implemented in, the system architecture domain." (Vitech Corp. 2007a) The system architecture is composed of operational nodes and components. Operational nodes represent elements of the operational architecture that produce, consume, or process information. A component represents a physical or logical element that performs a function. (Vitech Corp. 2007a)

Operational Architecture

- **Step 1 – Operational Concept Capture:** The first step in developing the architecture should be to define and capture the architecture's operational concept. The ConOps, requirements, and any guidance documents should be used for this purpose. The Operational Architecture provides classes, attributes, and relationships required to capture the originating requirements and guidance for a system. At this step, the CORE database should be populated with the elements (along with the appropriate relationships and attributes) for each of the following classes as is deemed appropriate: (Vitech Corp. 2007a)
    - Document (an entry for each available source document)
    - Guidance (any available guidance documents or statements)
    - Mission (an entry for each stated mission)
    - OperationalTask (operational tasks)
    - Requirement (requirements from the SysML diagrams for example)
    - ExternalFile (any applicable documents or graphics files)
    - DefinedTerm (any acronyms and abbreviations)

  Capture the organizations in the architecture in the Organization class using the appropriate elements and relationships. Enter the operational boundary into the CORE database using the OperationalNode class. Information that this exchanged between the nodes should be defined by elements in the Needline class. (Vitech Corp. 2007a)
- **Step 2 – Conduct Operational Activity Analysis:** The second step is to derive the operational behavior for the operational architecture required to fulfill the stated mission. Operational behavior is defined in the operational scenarios using the OperationalActivities class populated with the appropriate elements and relationships. The definition of an operational activity is an action or process that is needed to accomplish a mission. (Vitech Corp. 2007a)
- **Step 3 – Conduct Architecture Synthesis:** The third step is to decompose operational activities and operational nodes to refine the operational architecture.

Additionally, the external and internal Needline definitions are defined or refined further. (Vitech Corp. 2007a)

- **Step 4 – Validate Operational Model:** The fourth step is to validate the operational model using COREsim. "COREsim dynamically interprets the behavior model in conjuction with the needline model and identifies and displays timing, resource utilization, operational information flow, and model inconsistencies". (Vitech Corp. 2007a)

System Architecture

- **Step 1 – Requirements Capture:** The first step is to identify the system and its mission. Capture the physical elements in the Component class along with the appropriate relationships. Capture the originating requirements in the Document, Requirement, ExternalFile, and DefinedTerm classes. Define the system boundary by identifying all interfaces between the system and each external by creating elements in the Interface class. Finally, identify applicable documents in the Document class. (Vitech Corp. 2007b)
- **Step 2 – Perform Requirements Analysis:** The second step is to identify requirement issues and risks and capture them in the Issue and Risk classes along with applicable relationships and attributes. Part of requirements analysis is to characterize the requirements as functional, performance, constraint or verification. These attributes are available for selection in the requirement's Type Attribute. (Vitech Corp. 2007b)
- **Step 3 – Perform Functional Analysis:** The third step is to organize functions by states or modes if required. This can be captured in the State/Mode class. The system functional hierarchy is developed and performance requirements are allocated to functions. Any functional performance risks and issues should be captured in the database using the Risk and Issue classes. A function's inputs, outputs, and triggers are captured using the proper relationships. (Vitech Corp. 2007b)

- **Step 4 – Perform Architecture Synthesis:** The fourth step is to allocate functions to components. External and internal interfaces are defined and captured in the Interface class. The Item and Link classes are used to further refine the interfaces. Finally, define any constraints for components and capture the physical architecture issues and risks. (Vitech Corp. 2007b)
- **Step 5 – Verification/Validation:** The fifth step is to establish verification requirements. These can be captured in the VerificationRequirement class along with the appropriate relationships and attributes. Verification events and test procedures can be captured in the VerificationEvent and TestProcedure classes. Finally, COREsim is used as explained in Step 4 of the Operational Architecture procedure steps. (Vitech Corp. 2007b)

4.6.6 Discrete Event and System Timing Models

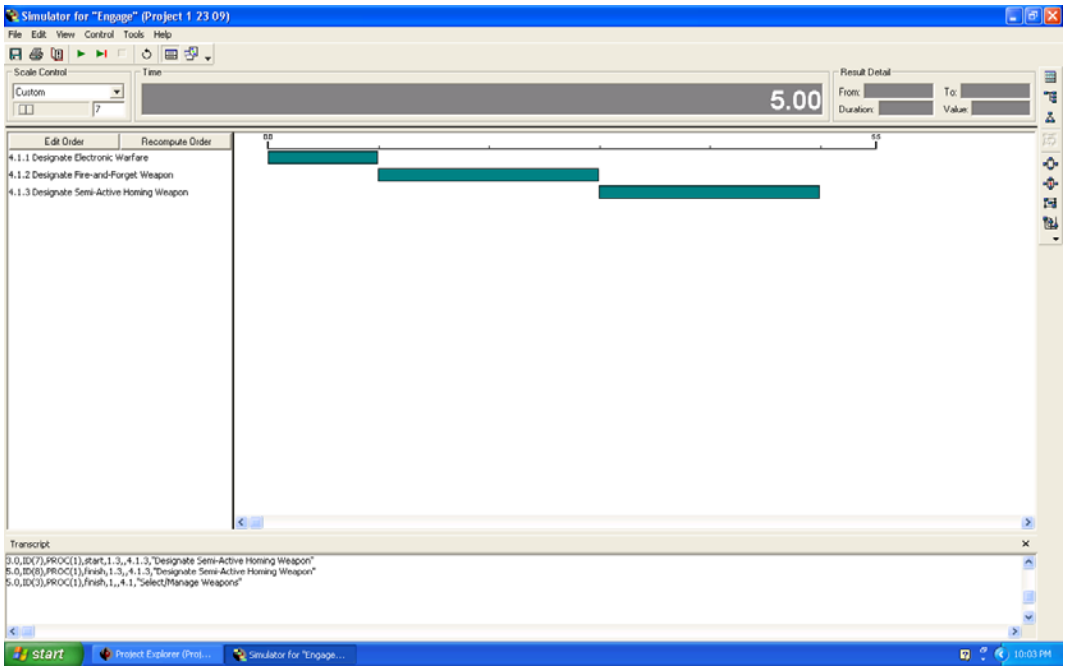
The COREsim feature was used to perform simulations of the behavioral model functions for the AAW architecture.

> COREsim is an integrated discrete event simulator that dynamically interprets the behavior model to support timeline analysis, resource analysis, and consistency analysis of the integrated architecture. The simulator can be constrained to execute a single layer of a behavior model or it can navigate the entire logical decomposition to execute the entire system model. Use of COREsim's robust analysis capability provides a means to ensure that the system definition is complete and executable. (Vitech Corp. 2007c)

Best engineering judgment was used to select the values for the duration (timing) of the individual functions. Since the team did not have information that defined the resource consumption rates or minimum and maximum resource values, "place holder" values were used. Although the values for the resources were not representative of realistic parameters, the fact that the simulation executed correctly did demonstrate that CORE could be used to verify that the architecture was executable.

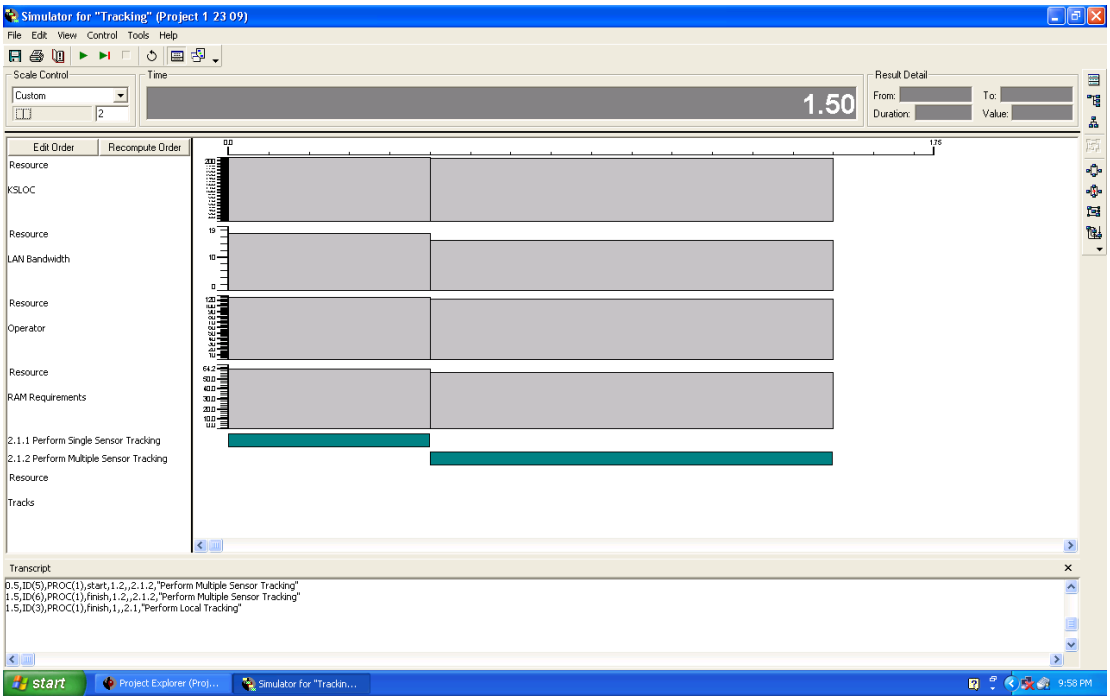
The screen capture shown in Figure 4-62 illustrates a successful simulation of Engage functions, 4.1.1 Designate Electronic Warfare, 4.1.2 Designate Fire-and-Forget Weapon, and 4.1.3 Designate Semi-Active Homing Weapon. Had this part of the architecture not been executable, the simulation would have not run to completion. The window in the top center block labeled "Time" shows that the simulation ran for a total of 5.00 seconds. This time corresponds to the total time it took all three functions to execute. Additional information is provided in the "Transcript" window at the bottom. This CORE simulation output could provide the software product team with information required for performing an assessment of the architecture. The output could also be used to provide M&S parameters for processing time of major functions for P_{RA} simulation.

The CORE simulation output in Figure 4-63 shows the KSLOC (Source Lines of Code – in thousands), LAN (Local Area Network) Bandwidth, Operator and RAM (Random Access Memory) resources consumed by the Tracking Function. From the output it can be seen that function 2.1.1 Perform Single Sensor Tracking took 0.5 seconds to execute and function 2.1.2 Perform Multiple Sensor Tracking took 1.0 seconds to execute. The total simulation runtime was 1.50 seconds. The simulation also shows the reduction of the resources over time as they are consumed by the functions.



CORE screen capture showing the simulation results (timing only) for the Engage Function. The green colored bars indicate the time it took to execute the function. The time divisions on the timeline (x-axis) are in 1 sec increments.

Figure 4-62: Simulation Results for Engage Function



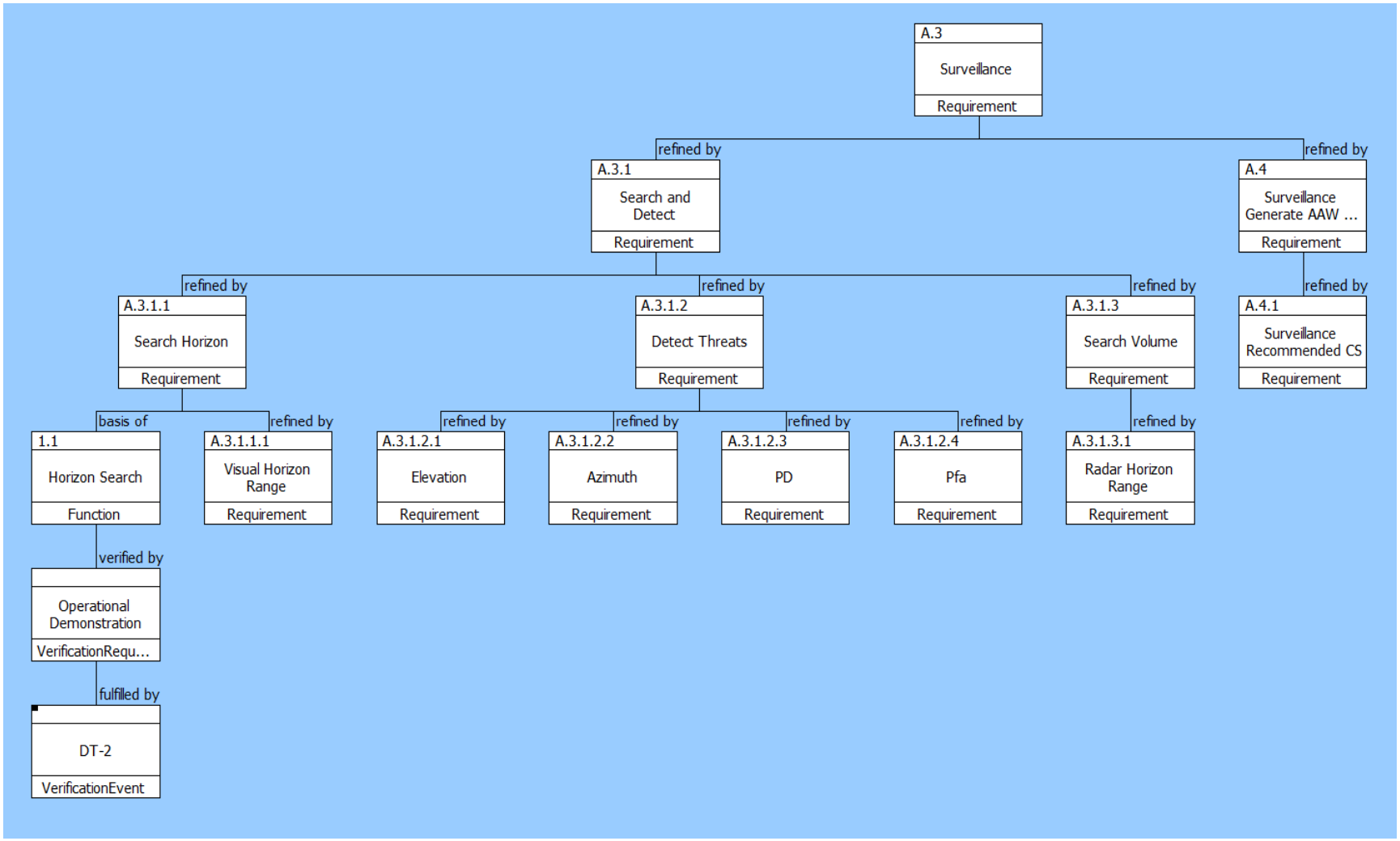
CORE screen capture showing the simulation results (resource usage and timing) for the Tracking Function. The green bars indicate the amount of time it took the given function to execute. The time divisions on the timeline (x-axis) are in 0.1 sec increments. The gray bars indicate the amount (y-axis) of resource available at the start and conclusion of a function's execution.

Figure 4-63: Simulation Results for Tracking Function

4.6.7 CORE Issues in Developing DoDAF Artifacts

Difficulties were encountered when using CORE to capture the requirements and functionality of the AAW architecture to develop DoDAF artifacts. CORE is missing some supportability classes, and there were issues translating functional behavior and SysML requirements into CORE. Additionally, there were no individual user software licenses, limited documentation, or user training for the CORE tool. The following paragraphs provide some of the details on the difficulties encountered in using CORE:

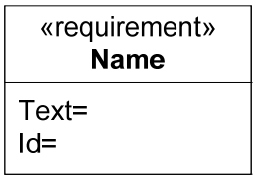
1. Inability to directly translate the SysML AAW requirements into CORE. The main issue was that some of the relationships and elements in SysML don't have a corresponding relationship or element in CORE. Although this did not prevent the entering of the SysML requirements into the CORE database, there is no assurance the SysML requirements were captured correctly, and therefore no guarantee of complete traceability of requirements from CORE back to SysML.
    a. There does not appear to be a way to show packages (packaging of requirements) in CORE; thus, everything is displayed in a hierarchical view (i.e., not packaged). CORE deals with classes and elements but not packages. Figure 4-9 shows the requirements traceability for the SysML surveillance package and Figure 4-64 shows how the SysML package diagram translates into CORE.



This figure is a CORE output (traceability view) based on the SysML Surveillance Requirements Package.

Figure 4-64: CORE Surveillance Requirements (Traceability View)

b. The requirements elements in SysML provide only the name, requirement ID number, and text attributes. Figure 4-65 is an example of a SysML Requirements Element symbol. When entering requirements elements in CORE, it is required at a minimum to know the type and origin of those requirements, neither of which is provided in the SysML requirement element block.



A screen capture from the MicroSoft Visio (2003) SysML 1.0 Plug-in for a SysML Requirements Element symbol.

Figure 4-65: SysML Requirement Element Symbol

i. Choices available for the attribute "Type" in CORE are:
   1. Composite
   2. Constraint (limitation on the design or construction of the system)
   3. Functional (what the system must do)
   4. Performance (how well the system or function must perform)

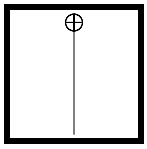
ii. Choices available for the attribute "Origin" in CORE are:
   1. Derived
   2. Design decision
   3. Operational
   4. Originating

c. There is no way to show "trace" or "copy-to" elements in CORE. Figure 4-11 illustrates the use of the "trace" relationship in SysML while Figure 4-12 illustrates the use of the "copy-to" relationship. The "refined by" relationship was used for all requirements in the CORE database.

i. SysML defines "trace" as, "a general purpose relationship between a requirement and any other model element." (Soares and Vrancken 2007) CORE contains the "traced from" and "traces to" relationships. The "traced from" relationship "identifies a higher-level document to which the requirements in the subject *document* should be traced". The "traces to" relationship "identifies a lower-level document to which the requirements in the subject *document* should be traced".

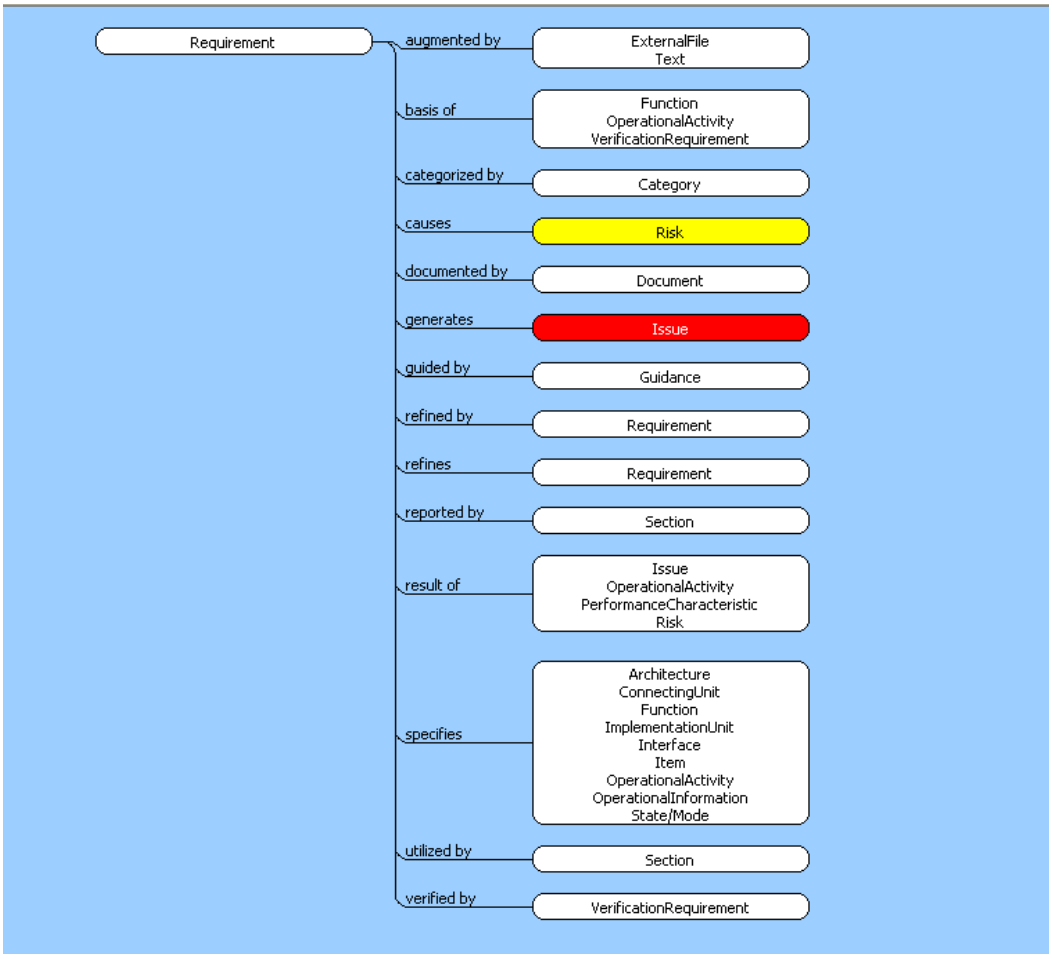
ii. SysML defines "copy to" as a master/slave relationship. The slave is a requirement whose text property is a read-only copy of the text property of a master requirement. (Soares and Vrancken 2007) There is no "copy to" relationship defined in CORE.

d. In SysML, the use of the "containment" relationship between elements does not directly translate into CORE. Figure 4-66 shows an example of the SysML Containment relationship symbol. The closest relationships in CORE are "contains" or "contained by". "Contains" identifies the sections that comprise a *document* generated from the contents of the database. "Contained by" identifies the *documents* that include the section. The Requirements IPT interpreted this relationship as "categorized by". However, in CORE the "categorized by" relationship identifies a grouping that includes this element. Further, the Requirements class for this relationship has a target class of "category." Figure 4-67 shows an example of the requirement class and its relationships to its target classes. In CORE, a Category groups related elements. A category is used for formal documentation to group non-functional requirements by subject matter. Figure 4-68 shows a screen capture of a category class and the possible target classes.



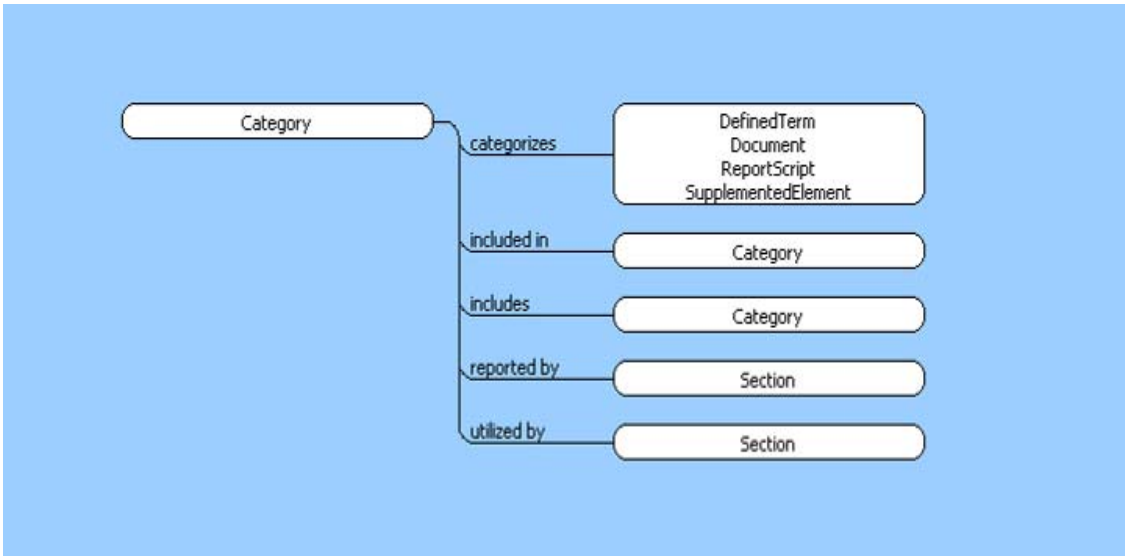
A screen capture from the MicroSoft Visio (2003) SysML 1.0 Plug-in for a SysML Containment Relationship symbol.

Figure 4-66: SysML Containment Relationship Symbol



CORE screen capture of an example of a Requirement Class and possible target classes. CORE automatically highlights the Risk class in yellow and Issue class in red.

Figure 4-67: CORE Requirement Class with Target Classes

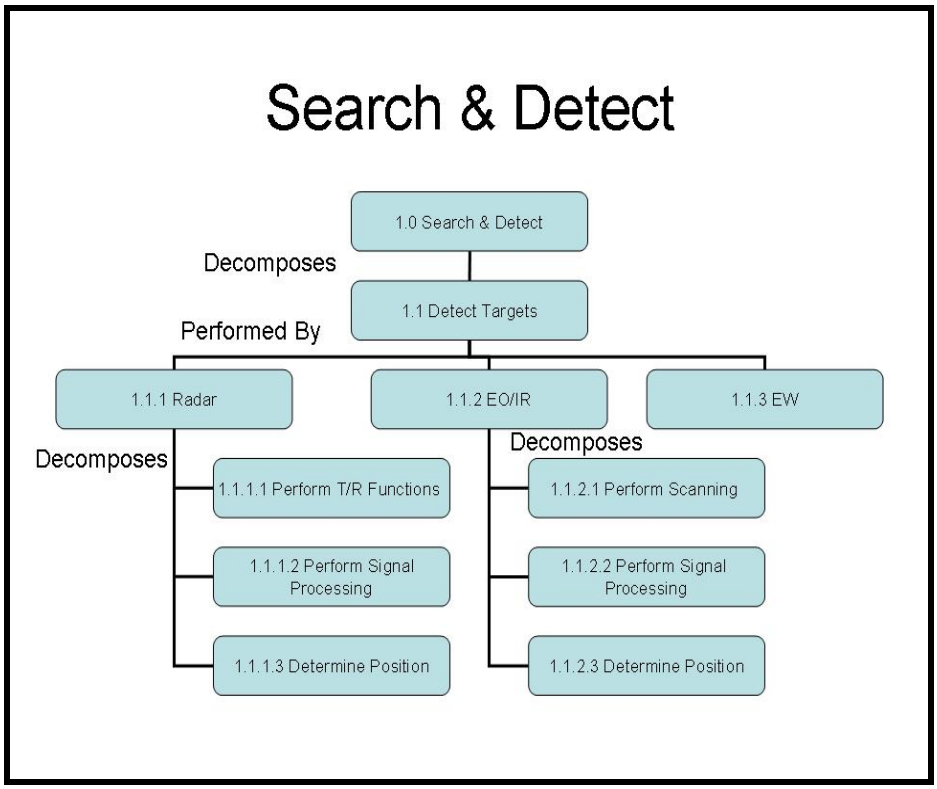


CORE screen capture of an example of the CORE Category Class and the possible target classes.

Figure 4-68: CORE Category Class with Target Classes

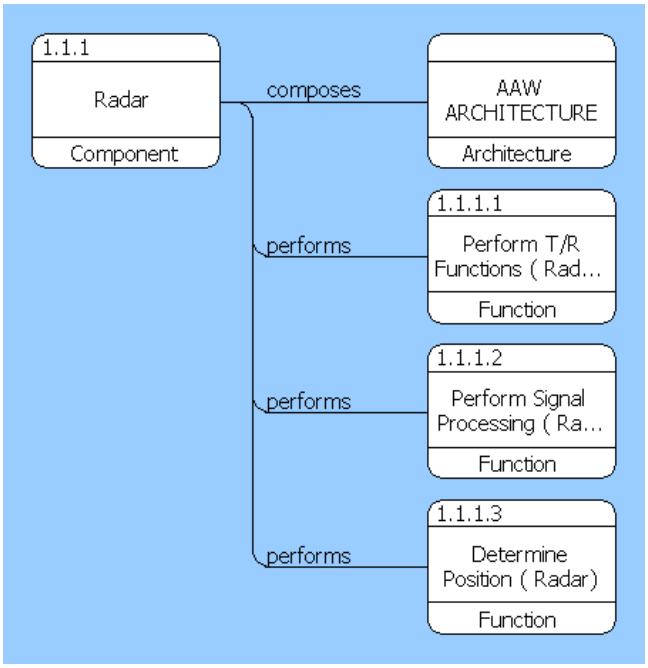
e. CORE requires that all elements be uniquely named. The SysML requirements diagram shows the same names for requirements elements in the Limited Area Defense as well as the Self Defense packages. Examples are Track, Command and Control, Engage, Manage Tracks, Select Weapon, and Manage Weapon. For CORE to accept the inputs, the names had to be changed. Assigning different element numbers to the requirements did not make a difference; CORE still needs unique names

2. The same issue was encountered when translating the Functional Diagram into CORE. Figure 4-69 shows the Functional Diagram for the Search and Detect Function. All functions that were not uniquely named such as Perform Signal Processing and Determine Position had to be modified in the CORE database as shown in Figures 4-70 and 4-71.



Illustrates a section of the Search and Detect Functional Diagram used to develop the AAW architecture in CORE.

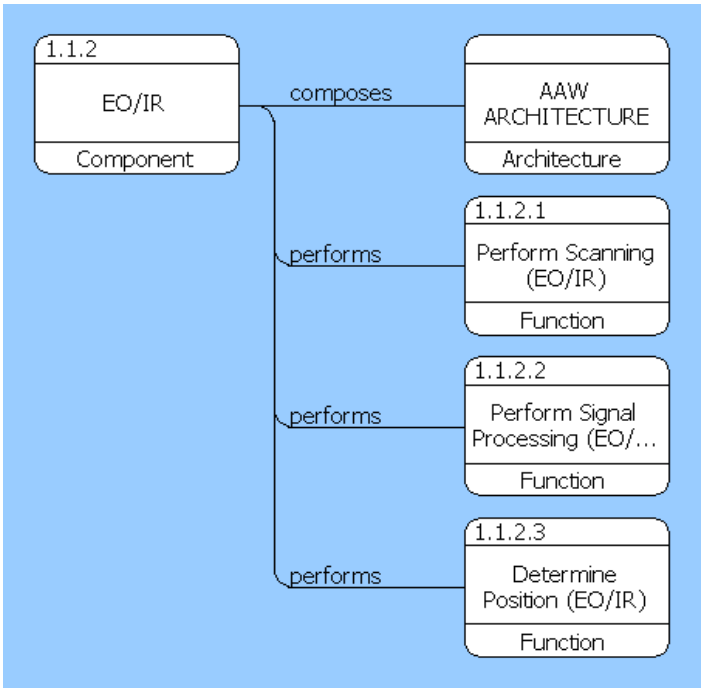
Figure 4-69: Functional Diagram



Illustrates how the names of functions with similar names in SysML had to be modified (note the addition of "Radar" to the name) in order for them to be accepted for entry into the CORE database.

Figure 4-70: CORE ER Diagram for Radar

Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.



Illustrates how the names of functions with similar names in SysML had to be modified (note the addition of "EO/IR" to the name) in order for them to be accepted for entry into the CORE database.

Figure 4-71: CORE ER Diagram for EO/IR

3. CORE does not appear to have the classes needed to model supportability requirements. These classes are Fault Detection/Fault Isolation (BIT), embedded Computer-based Training (CBT), and embedded Technical Manuals TMs)/Interactive Electronic Technical Manuals (IETMs). Elements for a connectivity function and adequate bandwidth to allow for off-board monitoring and distance support are required for supportability. These are the elements required to implement the requirements driven by the system support concept which is part of the ConOps. The CORE schema needs to be "extended" to include the classes discussed in this paragraph. This requires research into the CORE features to determine how this can be done.

4. CORE 5.1.5 Software Issues: The lack of CORE documentation, training (in CORE as well as lack of familiarity with how to produce DoDAF views), and the inability to install CORE on personal or work computers adversely affected the

progress of populating the project information in CORE. This in turn affected the development of the DoDAF products that needed to be developed using CORE.

a. Limited documentation (this includes documentation available on the Vitech web site, the CORE program, and the Internet).
b. Documentation available via the HELP feature in CORE does not cover all issues a user will encounter. There is no comprehensive user's manual that can be referenced to help troubleshoot issues that a typical user will encounter when using CORE. For example, issues with the External File Path attribute, which outputs graphics as a broken link in the DoDAF OV-1 view. Error messages are cryptic and provide little insight as to what is causing a problem. Figure 4-72 below shows a typical CORE script error output.

```
<!-- Target element specified by ID {9DFC57EA-
4422-4E60-A597-077F46DFB311} is not defined
in project Capstone Project 12 04 08. -->
  <relationship-reference source-id="{B4F1C763-
        1187-4A53-B336-2550C474264C}"
        definition="decomposed by" target-
        id="{9DFC57EA-4422-4E60-A597-
              077F46DFB311}"/>
```

Illustrates one of the CORE error messages received during the development of the AAW Architecture.

Figure 4-72: CORE Error Script

c. There is little to no help available via the Internet (i.e., no users' group forums) to help resolve CORE issues.
d. CORE software is only available via a remote connection to computers at the NPS campus. Special VPN software must be installed on personally-owned computers to access the remote NPS computers.
e. No access to the remote NPS computers from PHD NSWC is available because the VPN software cannot be installed on Navy-owned computers

since the VPN software is not approved for installation by Navy Marine Corps Intranet (NMCI).

f. CORE software is approved for installation on NMCI computers, but since there is no PHD NSWC sponsor for this software program, it cannot be installed on Navy-owned computers. This limitation forced the personnel working on CORE to work independently using personal computers. Additionally, a high-speed Internet connection is required when working with CORE via a remote connection to the NPS computers.

g. There was no opportunity to have all the IPTs collaborate on a real-time basis on CORE inputs involving the different areas of responsibility. This limitation affected the progress in populating the project data in CORE. It was not until the first week of the third quarter of the project that a LAN connection at PHD NSWC became available on a not-to-interfere basis (in a secure room requiring special access privileges). While some progress was made, the CORE license at NPS expired approximately three weeks later. Access was restored after two weeks, but by then critical development time had been lost and the team was forced to freeze the database to begin writing the CORE section of this report.

h. CORE was not part of the NPS MSSE curriculum, which caused a steep learning curve. This learning curve affected the progress in populating the project data in CORE.

4.7 SUMMARY

Requirements were developed to support the problem statement given by the stakeholder, and SysML was chosen to represent requirements in a graphical and tabular form. The SysML requirements diagram can facilitate the transformation of stakeholder requirements into system requirements and improve requirements traceability throughout the design life cycle. (Soares and Vrancken 2007)

The ConOps (Appendix G) was developed in parallel with the requirements and helped in describing the characteristics of the proposed AAW system from the viewpoint of an end user. The ConOps was used to bound the problem, in that an AAW system was set in a clear marine environment (i.e., no clutter and no jamming). The system was further bound to include a supportability profile that allowed for a requirement to be defined and documented in SysML. Additionally, technical/stakeholder discussions helped in developing the OV-1. The questions posed led to an AAW system with Detect-Control-Engage functions to meet the requirements and support the ConOps.

Both SysML and HP were used in the development of the AAW architecture. The systems engineering process (outlined in Chapter 3) was used to develop and refine artifacts. Additionally, the process was function-driven and based on the identification and elaboration of operational contracts, a message-based interface communication concept. (Hoffmann 2007) The HP Models served two purposes:

1. To help the designers understand a complex system so that it can be designed, produced and fielded.
2. To be used as a communications tool to help explain the system to others including the customer and users.

The M&S task was broken down into three main phases:

1. Process development
2. Model development
3. Modeling analysis

The accomplishment of each phase is essential in an acquisition development program. The development of a working level model follows the completion of a high-level process development. Following the completion of the model development and successful execution of the simulation is the analysis of requirements traceability.

During the initial development phase, the Extend (software) model was agreed upon as the tool of choice for defining M&S objectives (validate a developed methodology to support AAW Mission Requirements). The overall objective of the model is to validate that the proposed system meets the Top-Level Requirements (TLRs). The TLR for the system was stated as a 0.99 P_{RA} for a given number of targets in a given amount of time.

Modeling analysis consisted of taking the raw data generated in the Extend simulations and inserting into Microsoft Excel for analysis. Excel was used as an analysis tool because of its capability to display statistical data. During the analysis of the data, it was concluded that the main constraint to the Self Defense model was the limited number of illuminators. Additional manipulation of the model and continuing simulations proved that this was a key factor in achieving a 0.99 P_{RA}. Additional sensitivity analysis was conducted and revealed that sensor height, P_K of the missile, and reaction time would be other areas to look into for both supportability and performance tradeoffs. It is important to remember that M&S is an element of the entire framework of validating a methodology. The objective was to prove that given inputs and parameters can lead to a successful P_{RA} using Extend as a tool.

CORE is a powerful tool for developing architectures and DoDAF artifacts. CORE was selected as a tool to aid in the use of capturing the AAW requirements and its ability to illustrate the relationships between components in the architecture model.

Once the AAW model was entered into CORE, the tool was used to display graphical representations that were used to determine if the architecture was built correctly. The tool asked the modeler numerous questions to develop the relationships necessary to construct the AAW system. The answering of these questions led to a descriptive interpretation of the system. The relationships that were developed from the backbone of the model allowed for behaviors, control and data flows to be understood. CORE developed DoDAF views and reports, which are provided in section 4.6 and Appendix M.

THIS PAGE INTENTIONALLY LEFT BLANK

5 CAPSTONE OBJECTIVE SUMMARY

This project was undertaken as a result of the Navy's history of deploying combat systems designed using several distinct architectures. Developing a common domain-based set of requirements necessary to produce what the warfighter needs is critical prior to selecting a particular combat system architecture. Future requirements are derived from shortcomings of existing systems or are identified capability gaps in systems or ships fielded today.

Stakeholders and systems acquisition professionals require detailed analyses of behaviors and processes (documented via requirements) necessary to meet tomorrow's mission needs. System designers should take advantage of software or code that has already been developed for mission areas. The advantage of software reuse is that the Navy pays for writing code once and using it many times. When developing a capability for reuse, one must first consider the need for a library that can store the appropriate artifacts that describe associated requirements. The Navy lacks a central storage area for mission requirements of current systems.

Since Operation and Support (O&S) costs comprise up to 70% of Total Ownership Cost (TOC), supportability factors should be given the same degree of consideration as performance factors. (Miles 2008) The integration of supportability as a top-level requirement provides focus and resources necessary to achieve reductions in out year costs. The result of this thinking is the latest DoDI 5000.02 that prescribes early integration of supportability requirements.

M&S, although used in performance assessment areas, is not used to the same degree for supportability, which results in a lack of quantitative data to support gap analysis and decision-making. Use of M&S tools to validate requirements is largely undocumented and its ability for reuse is unknown. This creates significant challenges in establishing

traceability between artifacts to reduce development and T&E efforts. More effective use of M&S during requirements generation could be achieved by increased tool interoperability and easier tool verification and validation. Performing early and iterative analysis using M&S to establish boundaries for SPL domains is underutilized.

The methodology described herein was developed to create open and supportable system architectures. This methodology uses a common language, domain analysis to support SPL reuse, maintains traceability of requirements and architecture functionality, and integrates supportability, sustainment, and life cycle cost considerations. The methodology was evaluated by applying it to the Anti-Air Warfare (AAW) mission thread, in particular Self Defense (SD). The AAW implementation included the development of a system architecture and design artifacts, including DoDAF views. The project demonstrated the benefits of a MBSE approach tailored to developing architectures that support OA, SPL, and integrating supportability early in the system development process. Follow-on research topics are identified to expand the use of the MBSE approach for developing system architectures.

5.1 TECHNICAL CONCLUSION

5.1.1 Systems Engineering Process - Experience

Numerous methods and tools are available to design and develop system artifacts; however, the quality of those artifacts has not fully evolved. System artifacts were developed using SysML, HP and CORE, and their quality varied significantly due to the relative experience levels of the Subject Matter Experts (SMEs) who supported artifact development. Although languages such as SysML provide a common interpretation of graphical requirements depiction, while HP provides an approach for conducting the systems engineering work in sequential steps and CORE provides an automated capability to verify and correlate data, none of these processes provide the technical detail necessary to develop a valid architecture. Additionally, there are limitations on

integrating supportability into early systems engineering and design efforts due to conflicts between mission and supportability requirements. The collaboration of logisticians and engineers early in design will benefit the Navy since decisions made during system design have long-term effects on life cycle considerations (cost, performance, maintenance concept, manning, reliability, training, open systems architecture reuse).

5.1.2 Supportability Integration in the Systems Engineering Process

The Navy advocates the integration of supportability early in the concept development and design phases, but very little training or guidance is provided on how to effectively do this. Many logisticians are equipped with neither the knowledge nor experience to adequately support initial system concept and architecture development. Similarly, many design engineers lack the training and experience of considering supportability during concept exploration, design, and development. On this project, engineers and logisticians collaborated to meet an expressed objective of integrating supportability into design as depicted in the resulting artifacts. Supportability was considered during requirements generation, functional analysis, and architecture composition, and is depicted in context diagrams and other artifacts. The M&S process illustrates how supportability and RAM requirements can be integrated using operational artifacts to depict A_O values given the allocation of reliability factors. Integrating supportability early in design provided the maintenance concept and maintenance planning with a solid foundation on which they can be built, and a basis for conducting tradeoff decisions between operational enhancements and life cycle sustainment considerations.

5.1.3 Tools for Verification and Validation of Engineering Artifacts

Use of a tool to conduct traceability is critical on large complex systems due to the sheer volume of technical data and the likelihood of human error when trying to conduct V&V manually using engineering artifacts. Requirements generation and traceability were achieved using SysML as the modeling language and CORE as the architecture tool.

Requirements traceability was established both from a parent-child relationship and from requirements to the engineering, architecture, and M&S artifacts. SysML contains methods based on the allocation relationship depicted in the artifacts to verify traceability, so manual efforts that would significantly increase the likelihood of human error are obviated. CORE provides numerous automated functions for verifying traceability, including assessment reports for unallocated requirements, functions, signals, test requirements, and other reports that correlate information and system design attributes. It further provided limited simulation capability to verify proposed operation using functions and input/output data and criteria. SysML was used to manually verify traceability between artifacts while CORE was utilized to verify overall traceability. Sample test criteria and events were used to successfully verify CORE could be used to assess demonstration of requirements.

To support an enterprise approach, T&E requirements and test status could be associated to requirements linked to SPLs or domains that will be reused in future configurations. Architecture tools have the ability to improve traceability analysis, increase verification capability, and reduce manual efforts for production of systems engineering artifacts. CORE was successfully used to develop a proposed architecture and produce engineering, architecture, program management, and T&E artifacts including the DoDAF views. CORE was populated using the engineering artifacts as a baseline with additional information being entered as required to produce examples of products. CORE provided for complete traceability of requirements and verification that the artifact data contained in it were linked correctly. It was also able to execute limited simulation and produce data to support M&S through allocation of resources to software and hardware components.

5.1.4 M&S Limited to Operational Scenarios

M&S can provide significant value in conducting tradeoffs during design. However, the majority of M&S is focused on verifying operational parameters within scenarios vice

optimizing system design. M&S was applied using a top-down approach to verify system operational behavior and validate initial operational requirements. The software tool Extend was used to perform the simulation of a raid scenario. Through multiple variations of models and simulations, it was found that there can be anomalies or elements that need adjustment in the architecture. The abnormal (or unlikely) results from the raw data led to more extensive research of the initial inputs, which led to additional simulation runs. Defining objectives, processes, and model development were all key milestones in building the Extend model. Recognizing that ship systems operate in dynamic environments drives the need to address supportability requirements during the initial discussions of requirements generation. By understanding how to build the Self Defense model, the same techniques can be reused to develop other models (i.e., Limited Area Defense and Surveillance). The only differences would be the incorporation of additional range parameters and probability of kill. With the exception of those inputs, the same fundamental approach could be used for developing a model. M&S was not applied to validate system supportability or to optimize system design by assessing the performance of alternate architecture styles. A quality assessment was made during architecture development that led to a layered approach.

5.2 CONTRIBUTION TO KNOWLEDGE

5.2.1 Integration of Concepts

The methodology, based on tenets from multiple authors, resulted in a more robust process that successfully met the Capstone objectives. Key aspects were included in requirements generation, architecture development, and M&S.

The requirements generation process was based on Martin's approach (Martin 1997) and the application of SysML. Top-level requirements were derived from stakeholder needs and captured in the ConOps, while lower-level requirements were derived through

decomposition. The requirements were captured in SysML, demonstrating an MBSE approach that encapsulated traceable behavior, structure, and parametric requirements.

Architecture generation, based on Martin's approach and augmented by HP and Bosch methods, produced a structure for software reuse that enables support for life cycle objectives. The HP method was used to define and arrange key mission functions into a logical sequence. These functions were grouped into a component line based on loose coupling and high cohesion. Consequently, these functions were capable of being utilized in a variety of styles, all of which can be assessed using Bosch's methods. Following assessment of the selected style, the architecture can then be captured in a SysML diagram to establish a common format and language that improves communication among designers, developers and stakeholders.

M&S tenets were derived from previous coursework, the Navy Modeling and Simulation Office (NMSO), and assessment methods identified by Bosch. While the NMSO provides critical information on overall development of processes and systematic functions, Bosch provides methods for assessing the architecture against stakeholder goals. The application of architecture assessments complemented traditional M&S activities focused on verification of mission capability, while the use of architecture assessments, described by Bosch, provided a method for improving life cycle objectives. The use of a MBSE-based language (SysML) in conjunction with a MBSE tool (CORE) allowed multiple methods to achieve requirements traceability and improve verification capability.

5.2.2 Technical Lessons Learned

The project was ultimately successful. However, there were many difficulties encountered in performing the work. M&S preplanning, tool selection, learning curve, accessibility, and time constraints presented challenges that led to lessons learned.

Initial planning should account for use of engineering and architecture artifacts for development of M&S. HP analyses of performance and sustainment factors for SPL domains were used to identify architecture risks based on sensitivity analysis. The methodology demonstrated that the engineering artifacts could be used as a basis for M&S. The use of SysML and parametric requirements provided a basis for the modeling assumptions and variations. Artifacts developed during architecture allocations, such as the activity diagrams and EFFBD, provided a basis for the construct of the model. Functional allocation provides a depiction of the major functions that need to be modeled. Use of an automated tool can provide the interface and configuration data for validating the model. Depending on tool selection, inherent capabilities may be available to model system performance factors given resource constraints. These constraints, such as bandwidth or queue size limitations, can be inputted to the architecture tool. M&S can be used to understand the variations, which should make the modeled system representative under various environmental and operational conditions.

The tools used to execute the project (CORE, Extend, SysML) were selected based on availability and perceived capability. No one on the project team had experience using these tools, so there was a substantial learning curve that had to be overcome. None of these tools are supported on NMCI computers, so work had to be conducted on personal, non-NMCI computers.

The learning curve for CORE, SysML, and the HP method could not be fully surmounted during the time allocated for this project. CORE was found to be a very powerful tool with automated features that support an iterative development methodology; however, input requirements to use CORE are substantial and were not well understood. User guides and other sources of information on using CORE were limited; as a result, CORE learning was mainly via trial and error and on-the-job training. SysML was found to be a viable language to document artifacts, but initial products required considerable time due to unfamiliarity, which caused schedule conflicts. The systems engineering value of using the HP method was not fully realized until late in the project. Development time

for the HP method took longer than anticipated, but when completed, it provided the level of detail necessary to produce the engineering artifacts needed to populate CORE. Many of the challenges can been eliminated or reduced in severity by implementing the following recommendations:

- Tool and language training should be conducted prior to beginning the project.
- Careful consideration should be given to the sequence of formal courses.
- Understanding the project's scope and intent prior to taking courses would improve application of knowledge gained.
    - Having the Capstone requirements defined early in the program would facilitate the application of the Key System Attributes (KSAs) obtained through coursework.
- Selecting tools with adequate operator and user manuals will significantly reduce learning curve issues.

Use of CORE via remote NPS computers proved to be too limiting for use in a real world project. A full version with local accessibility should be available for future project efforts. The project was significantly impacted by the lack of sponsorship for CORE to be installed on local machines. Having the ability to access CORE from NMCI workstations will go a long way toward facilitating real time collaboration on future Capstone projects that use it.

Consistency between the tool and the language is critical. The lack of consistency created challenges. Using SysML as a language and CORE as a tool led to difficulties transferring data from the SysML artifacts when populating CORE, as described in Section 4.6. This resulted in the CORE development team interpreting and developing unique data, which resulted in traceability issues. Future Capstone projects that use CORE to develop system architectures and DoDAF views should use CORE from the beginning. It provides the capability to fully develop architectures and DoDAF artifacts without the use of SysML. Time that should have been invested in developing the CORE artifacts was spent developing SysML artifacts that do not translate directly to CORE.

Many of the relationships and attributes that CORE requires are not provided by SysML. Much of the architecture had to be created in CORE to produce the DoDAF views. SysML does not appear to provide a clear way to develop the DoDAF views; therefore, time could have been better invested by concentrating on CORE. Several tools are used in industry to implement SysML in system design. Among the industry leaders are MagicDraw, Enterprise Architect, SysML Toolkit, and Rhapsody. All of these products are structured for usability, drawing, executability, and standards compliance.

A solid requirements foundation is essential to system success. Requirements generation is the first in a *series* of steps in this iterative systems engineering process. However, due to time constraints and the size of the project, IPTs were forced to work in *parallel*. The requirements generation process did not drive follow-on work in system design, and architecture development did not drive M&S, forcing IPTs to be reactive rather than proactive. To follow the methodology as developed, compromises needed to be made. Some of what was compromised, including project objectives, should be more tightly controlled from the start or, if possible, more time should be allocated to complete the project.

5.2.3 Organizational and Programmatic Lessons Learned

Approaches and process are developed without consideration to schedule and resource constraints, creating risk since most programs are faced with fiscal, schedule, and technical limitations. Numerous methods are available to develop a viable SPL, each with their own benefits and limitations. However, none of the approaches address processes or methods for execution when a program is in a constrained environment due to schedule, resource availability, or technical challenges. This project was under schedule constraints and resource limitations that resulted in the level of detail and artifact quality being sacrificed. Programs should consider in-process reviews at periodic intervals during initial planning to ensure the systems engineering process is not compromised when operating in a constrained environment.

5.2.3.1 Organizational Lessons Learned

Team size, composition, multitasking, and lack of understanding of future IPT requirements created challenges in execution. Having a team with 28 members resulted in a lack of common direction and underutilization of available resources. Personnel were assigned to teams in a somewhat arbitrary manner, instead of being assigned according to their KSAs. Multitasking was both beneficial and harmful to project objectives. Some personnel involved in multiple IPTs gained valuable experience in developing the different artifacts, although this also led to resource constraints due to heavy workloads. A lack of understanding of when the various IPTs would be required to execute resulted in IPTs being formed without adequate understanding or planning. The team had received adequate information on IPT planning from previous courses, but the sense of urgency to execute overrode applying the IPT best practices.

5.2.3.2 Virtual and Physical Facility Planning

Virtual facility planning focused on ensuring a data repository was available to maintain historical and current data. Physical facility planning focused on meeting spaces to hold working group meetings and conduct training. Best practices applied in data management were used successfully to maintain configuration management and communication, despite geographical dispersion and continuous travel requirements. A central online database (NPS Blackboard) was used to manage data, schedule meetings, and conduct reviews. This was considered critical in keeping personnel informed of current developments despite significant differences in work requirements. Facilities initially were not dedicated resulting in challenges for personnel to conduct meetings and perform physical reviews of material. Although the use of a virtual war room improves communication, it does not eliminate the need for a physical location to conduct working group meetings. A dedicated room with the capability to accommodate an event similar to those conducted for quality improvement is needed.

5.2.3.3 Language Selection for Models

Although SysML shares many characteristics with UML, which was taught to the MSSE team members, the language barrier proved to be significant given the limited amount of time, resources and training available. It may take months to master SysML through formal coursework. For this project, however, it was up to the team members to determine the differences between the two languages and how to syntactically communicate in SysML in a very short period of time. Since this is a systems engineering project and SysML is quickly becoming the industry standard for systems modeling, it would make sense for NPS to offer a course in SysML for this program in lieu of UML. Unfortunately, none of the industry-leading tools were available for this project. Instead, an OMG SysML 1.0 template for Microsoft Visio was used to represent system design concepts in the modeling language. The team determined that there would not be a steep learning curve in becoming familiar with the tool, since team members gained sufficient experience using Visio in previous projects. Since the application and template were available from NMCI, NPS and Pavel Hruby (www.softwarestencils.com), the wait time for software licenses and installs would be minimal. Unlike MagicDraw, Enterprise Architect, SysML Toolkit and Rhapsody, Visio proved to be very limiting in SysML implementation as it is only a drawing tool. It was up to the designer to ensure that all requirements were adequately captured, that follow-on diagrams were traceable back to requirements, and each was consistent with the modeling language standard.

5.3 RECOMMENDATIONS

The following recommendations include observations made throughout the project and suggestions for future projects to be able to perform functions that were not able to be executed in this one due to time and resource constraints.

5.3.1 Logistician Training to Support Architecture Development

It is recommended that the Navy place a greater emphasis on training logisticians in the systems engineering discipline. Many of the issues uncovered in this project point to the fact that the Navy advocates involving logistics early in the concept development and design processes, but very little training or guidance is provided on how to do this effectively. Engineers and logisticians enter the project with similar objectives; however, their respective disciplines lead them to focus on the requirements that most affect their responsibilities.

5.3.2 Logistics Strategy and Products

Use of the artifacts could allow development of a logistics strategy and development of core supportability focus areas, such as maintenance, support and training, to meet milestone reporting requirements for mandatory life cycle sustainment metrics, including material availability KPP and associated KSAs.

5.3.3 Quality Attributes for Domain-Specific Components

Developing quality attributes specific to domain requirements would improve the architecture performance based on stakeholder goals. The methodology applied quality attributes to the total architecture when comparing event-based versus layered architecture; however, when assessing the subcomponents of the SPL, different quality attributes may be selected based on the subcomponent. For example, SPL components that are mission-critical and sensitive to time delays may be optimized by use of performance quality attributes that highlight the need for high fidelity in the information exchange. Components such as readiness assessments, which are not instantaneous, can accept higher signal latency and can be architected using a layered approach.

5.3.4 SPL Library Criteria and Characteristics

Develop specific criteria and attributes necessary to capture and characterize software components for reuse. These can be contained in a software tool used to identify suitable software for reuse. The criteria would provide input to a method to characterize the level of modification required for the software component to be reused or identify critical information that would limit reuse such as performance parameters. The criteria would be based on common attributes across software domains, and serve as a Dewey Decimal system for the library. The effort would establish a common model, which based on the ConOps or life cycle scenario, would allow entry of criteria and provide quantitative data for selection of the optimal component based on the cost of non-conforming criteria that would require modification.

5.3.5 System Decomposition Based on Methodology

Generate a SPL and allocate functionality and software to hardware components. Numerous lessons learned were captured when the methodology was used at a high level. It is felt that as the system is further decomposed, additional lessons learned would be identified and captured due to the level of detail, which was limited based on schedule.

5.3.6 M&S Model Life Cycle Based on Methodology

Develop a process for using M&S to estimate life cycle costs and optimize support concept tradeoffs early in design. M&S provides significant value in validating requirements, providing data to decision-makers, and demonstrating the objectives and constraints of system performance. Supportability requirements are applicable to M&S, since this is where decisions regarding tradeoffs, reliability equations, and other KPPs that affect the supportability concept and O&S costs are derived. The use of a ConOps in addition to the application of reliability data to the DoDAF views and architecture could be combined with historical cost data to generate life cycle supportability cost estimates. Maintenance concepts and technical initiatives such as distance support could be prioritized based on mission-centric activities and functions captured in DoDAF views.

This could be used to characterize and optimize architecture composition to meet modernization and maintenance requirements during life cycle, as well as to determine optimal sustainment concepts through tradeoff and business case analysis.

APPENDIX A: STATEMENT OF WORK

RESEARCH PROPOSAL – PHD Cohort 6

Date: June 27, 2008

Tentative Project Title: A Methodology for Software Product Line Development in Navy Weapons System Acquisition: An Approach to Open Architecture

Advisors: John M. Green, GSEAS

Rachel Goshorn, Ph.D., GSEAS

Period of Performance: 7 July 2008 – 23 March 2009

Project Synopsis:

Background

The Navy has long recognized the value of an open architecture for combat systems but was limited in the past by two key factors: 1) computer technology and 2) the acquisition organization. The combination of these two factors resulted in the procurement of diverse combat systems such as AEGIS and SSDS MK-2. In the last 10-12 years the Navy has undertaken several efforts to resolve this diversity through programs such as Sea Athena and Common Command and Decision, which morphed into the current Open Architecture program under the auspices of PEO IWS 7.

The focus of the current program is on procuring open source software that is reusable and interoperable. What this really means is what is referred to as "best of breed" software with design disclosure for evolutionary improvement combined with integration of common services such as time reference and common applications such as track management.[1]

The following diagram (Figure 1) represents the domain model in use by PEO IWS[2]. While it is described as a functional architecture model, it is, in the DODAF vernacular, a SV-1.

---

[1] https://acc.dau.mil/CommunityBrowser.aspx?id=31396

[2] This model was used by CAPT Tom Strei of PEO IWS 7 and by CAPT Ric Ruston of N-76 in numerous briefs

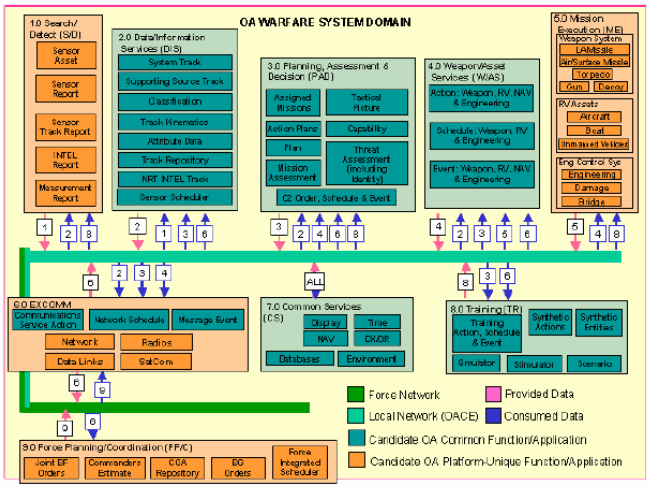


FIGURE 5.2 Open architecture functional architecture. NOTES: INTEL, intelligence; C2, command and control; NRT, near real time; RV, remotely controlled vehicle; NAV, navigation; EXCOMM, Executive Committee; SatCom, satellite communications; DX/DR, direct exchange/dead reckoning; BF, battle force; COA, common operating area; BG, battle group; OA, open architecture.

Figure 1: Domain model in use by PEO IWS

Figure 2 is from a memo dated 3 June 2008 from PEO IWS that establishes the charter for the CG(X) Combat System integrated product team.

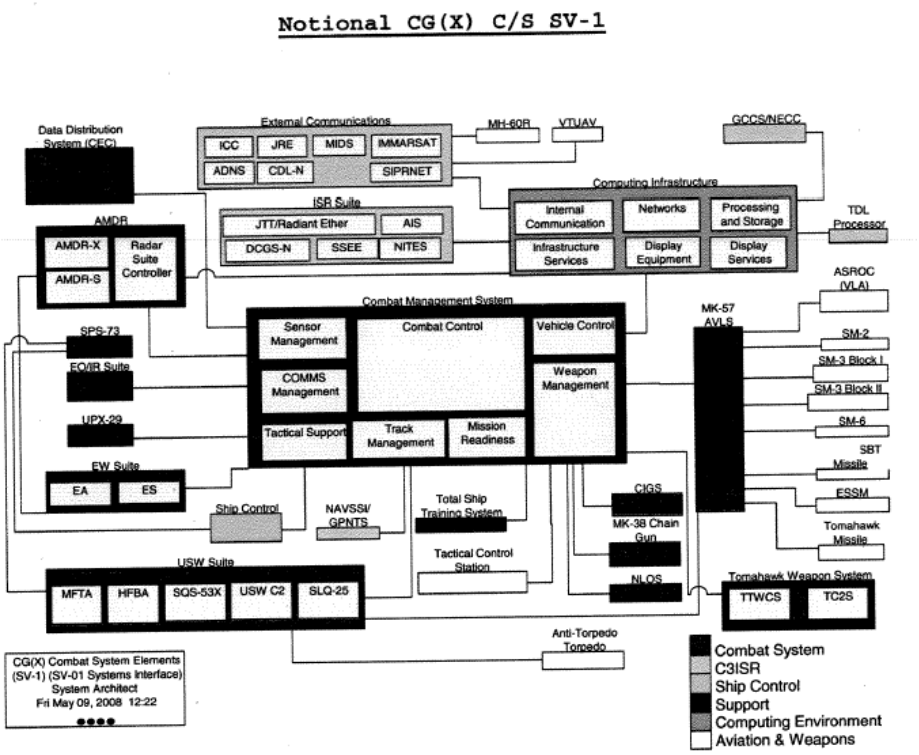


Figure 2: Charter for the CG(X) Combat System integrated product team

A comparison of the two diagrams shows that while the CG(X) diagram is similar to the domain model it relies on existing systems for much of its definition which suggests business as usual. This diagram does not reflect a logical progression from the operational needs to system requirements. This link is crucial: the operational architecture defines the rules for interoperability and defines the context for common services and applications. Most importantly it captures them in terms of required system performance.

It is clear that a shift in the current open architecture acquisition paradigm is required. This project would investigate the use of mission domain specific (e.g., AAW or ASW) software product lines as an approach because of its focus on the operational needs. It is also an approach that has proven its worth to European firms such as Terma and Saabtech.

Rationale

The basic software product line approach has demonstrated its merit. The Navy needs a defined approach or paradigm by which it can acquire the next generation of combat systems such that much of the software can be reused. The current strategy is one of opportunistic reuse where PEO IWS hopes that software being developed for LCS and DDG-1000 can be reused in other applications. The literature is replete with warnings against this approach.[3] The key is predictive reuse obtained through the software product line approach. Figure 3 illustrates the concept of predictive reuse through the use of a domain life cycle model.

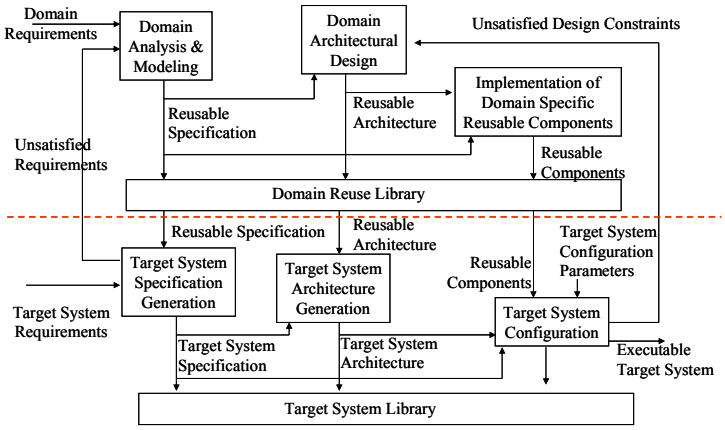


Figure 3: Evolutionary Domain Life-Cycle Model (Gomaa and Farrukh)

Tasking/Objectives

The overall goal is to develop a methodology for creating complex system architectures that are inclusive of concepts such as open architecture (OA) and services-oriented architecture (SOA) in a net-centric environment. This overall goal is captured in the following three objectives:

[3] Jan Bosch has written extensively on the topic: http://www.janbosch.com/index_files/Page392.htm

Show the viability of software product lines in developing warfare domain specific systems that are built from components that allow software reuse across multiple, diverse platforms (e.g., submarines and destroyers).

Develop a methodology by which to develop and model product lines consistent with DoD practice to include simulation and its ability to support development of operational, system, and logistics/support architectures as an integrated process. Methodology is defined to consist of three elements: technique or theory, process, and tools. The methodology should incorporate the DODAF Architecture Framework into a system specification format.

Develop a management/organizational approach to support 1 and 2 above.

Research questions

The central research question is how do the following key concepts interrelate?

1. OA
2. SOA
3. DODAF
4. Domain analysis
5. Software product lines
6. Model-based systems engineering
7. Process for Systems Architecture and Requirements Engineering (Hatley et al)
8. The Modular Command Evaluation Structure (MORS)
9. The Systems Engineering "VEE" model
10. Context sensitive modeling (Clymer)
11. Reliability theory
12. Net-centric architectures

The answer should define the appropriate role of simulation in model-based systems engineering and identify if there is a process that can be put in place to develop combat system software product lines that uses the DODAF framework as a specification and support software reuse.

Benefit

If proven viable, the concepts that are the focus of this research will provide the Navy with a methodology that supports the goals of the Open Architecture effort. Specifically it will allow the Navy to direct how industry will comply with the Open Architecture mandates. Second, it will comply with the requirements to use a simulation-based approach. Third, the Navy will have a repository of combat system software built on

predictive reuse thus, reaping the cost benefits that they seek. Finally, the navy will have a path by which legacy systems can be incorporated with the network-centric systems of the future.

Project Deliverables:

First Quarter: NPS MSSE Capstone Project Plan, Chapters 1 and 2 of the Capstone Report., end-of-quarter progress review

Second Quarter: Chapters 3 and 2 of the Capstone Report, end-of-quarter progress review

Third Quarter: NPS MSSE Capstone Technical Report submitted at the completion of the project. Final briefing.

References:

1. https://acc.dau.mil/CommunityBrowser.aspx?id=18016&lang=en-US. The official site for all thing related to Navy Open Architecture.
2. https://acc.dau.mil/CommunityBrowser.aspx?id=17608&lang=en-US . The Systems Engineering Community of Practice (SECoP) site.
3. http://www.acq.osd.mil/sse/docs/SE-Guide-for-SoS.pdf
4. https://acc.dau.mil/CommunityBrowser.aspx?id=200754&lang=en-US
5. http://www.sei.cmu.edu/mbse/index.html. Info on model-based systems engineering.
6. http://www.sei.cmu.edu/productlines/. The source of all good things related to software product lines.
7. http://www.janbosch.com/index.htm. Another good site for reference materials on software product lines.
8. Bass, Len, Paul Clements, and Rick Kazman. *Software Architecture in Practice, 2nd Edition*, Addison-Wesley. Boston, 2002.
9. Clements, Paul, et al. *Documenting Software Architectures: Views and Beyond*, Addison-Wesley. Boston, 2002.
10. Clements, Paul and Linda Northrop. *Software Product Lines: Practices and Patterns*, Addison-Wesley. Boston, 2002.
11. Clymer, John R. *Simulation-Based Engineering of Complex Systems*, John R. Clymer and Associates, Placentia CA, 2001.
12. Davis, Alan M., Eward H. Bersoff, and Edward R. Comer. "A Strategy for Comparing Alternative Software Development Life Cycle Models," *Software Requirements Engineering, 2nd Edition*, pp. 44-72, Richard Thayer and Merlin Dorfman, Editors, IEEE Computer Society Press, Los Alamitos, CA, 1997.

13. Fairley, Richard E., and Richard H. Thayer, "The Concept of Operations: The Bridge from Operational Requirements to Technical Specification," *Software Requirements Engineering, 2nd Edition*, pp.73-83, Richard Thayer and Merlin Dorfman, Editors, IEEE Computer Society Press, Los Alamitos, CA, 1997.
14. Garlan, David. "Software Architecture: a Roadmap," *The Proceedings of the conference on The Future of Software Engineering*, p.91-101, June 04-11, 2000.
15. Gomaa, H. and G.A. Farrukh. "Methods and Tools for the Automated Configuration of Distributed Applications from Reuasable Software Architectures and Components," *IEE Proceedings-Software*, Vol.146, No. 6 pp.277-285, December 1999.
16. Hatley et al, Derek. *Process for System Architecture and Requirements Engineering*. New York, NY: Dorset House Publishing, 2000
17. Hatley, Derek J. and Imtiaz A. Pirbhai, *Strategies for Real-Time System Specification.* New York, NY: Dorset House Publishing, 1987.
18. http://en.wikipedia.org/wiki/Hatley-Pirbhai_modeling
19. http://www.dtic.mil/ndia/systems/Rickman1.pdf
20. Kazman, Rick, Paul Clements and Mark Klein. *Evaluating Software Architectures: Methods and Case Studies*, Addison-Wesley. Boston, 2002.
21. Lieberman, Benjamin A., *The Art of Software Modeling*, Auerbach Publications, Boca Raton, 2007

APPENDIX B: PROJECT MANAGEMENT PLAN

1 Introduction

1.1 Project Description

The members of the Naval Surface Warfare Center Port Hueneme Division (NSWC PHD) Cohort 6 Capstone Project Team will develop a Systems Engineering (SE) methodology that can be used to create complex system architectures that will provide the Navy a process to achieve its goal of Open Architecture (OA), satisfy its requirements for a simulation based approach, and build a repository of combat system software for predictive reuse. The team will explore the applicability of the methodology by examining an Anti-Air Warfare (AAW) mission thread. A detailed description of the problem and a vision of what the end state will be are provided in the following sections.

1.1.1 Problem Statement

New combat system architectures are too often defined by existing systems instead of the requirements of the stakeholder. The Navy currently develops software for a specific purpose (e.g., the DDG-1000 combat system) with the hope that it can be reused on other platforms. Alternatively, DoD development should ensure products are created with known potential reuse. Experience has demonstrated that the management of large complex systems and technologies in pursuit of meeting DoD requirements demand a structured, repeatable method for evaluating investments and investment alternatives. The systems engineering process should construct architectures that inherently produce the compulsory Department of Defense Architecture Framework (DoDAF) artifacts. The DoDAF is intended to ensure that architecture descriptions can be compared and related across programs, mission areas, and, ultimately, the enterprise; thus, establishing the foundation for analyses that supports decision-making processes throughout the DoD.[4] This research project will produce a methodology that will fully integrate the generation

[4] Department of Defense. 2007. *DoD Architecture Framework Version 1.5.* http://www.defenselink.mil/cio-nii/docs/DoDAF_Volume_I.pdf.

of DoDAF artifacts with the acquisition requirements development process. This project will also investigate the use of mission domain specific Software Product Lines (SPLs) as an approach because of its focus on the operational needs, as well as develop an assessment of the current usage of Modeling and Simulation (M&S) in system requirements generation. The goal is to understand the implications that this information has on new initiatives, like Simulation Based Acquisition, where M&S will be more fully integrated into the military acquisition process.

1.1.2 Vision

The overall objective is to document a methodology that employs a Model-Based Systems Engineering (MBSE) approach, facilitates OA and Service-Oriented Architecture (SOA), clearly defines the role of M&S, integrates logistics/support architectures, and inherently produces the required artifacts for progression in the current acquisition life cycle process.

1.1.3 End State

If proven viable, the concepts covered by this capstone project will provide the Navy with a methodology that supports the goals of OA with regards to SPL development. It will allow the Navy to direct how industry should comply with OA mandates. The emerging methodology will be simulation based with emphasis on predictive reuse of combat system software. Furthermore, the Navy will have a path by which legacy systems can be incorporated with the network-centric systems of the future.

1.2 Background

Software product lines refer to engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production. SPLs are rapidly emerging as an important software development paradigm, allowing companies to realize vast improvements in time to market, cost, productivity, quality and other business drivers.[5] SPL engineering enables rapid market entry and flexible response, and provides a capability for mass customization within the Naval Open Architecture (NOA).

The NOA is the convergence of business and technical practices yielding modular, interoperable systems that adhere to open standards with published interfaces. This approach significantly increases opportunities for innovation and competition, enables reuse of components, facilitates rapid technology insertion, and reduces maintenance constraints. The NOA aims to deliver increased warfighting capabilities in a shorter time at reduced cost.

The Navy and Marine Corps have adopted OA as a way of reducing the rising cost of naval warfare systems and platforms, as well as to increase its capabilities. The NOA allows the incorporation of more commercial-off-the-shelf (COTS) technology in warfare systems, and enables reuse of software and related assets. In addition, NOA is an enabler of FORCEnet, the operational construct and architectural framework for naval warfare in the information age. More importantly, OA contributes to greater competition and collaboration among system developers through the use of open standards and standard published interfaces. OPNAV has mandated that individual domains (Air, Submarines, Surface, C4I, Space and Marine Corps) and Program Executive Offices (PEOs) pursue common architectures across their platforms or capabilities.[6]

---

[5] Software Engineering Institute. Software Product Lines. Carnegie Mellon University. http://www.sei.cmu.edu/productlines/.

[6] PEO-IWS 7. 2007. *Naval Open Architecture Contract Guidebook, Version 1.1.* https://acc.dau.mil/CommunityBrowser.aspx?id=183088&lang=en-US.

The Navy has long recognized the value of SPL and OA for combat systems, but was limited in the past by two key factors: 1) computer technology and 2) the acquisition organization. The combination of these two factors resulted in the procurement of diverse combat systems such as Aegis and Ship Self Defense System (SSDS) MK-2. In the last 10-12 years, the Navy has made several efforts to resolve diversity through programs such as Sea Athena and Common Command and Decision, which morphed into the current OA program under the auspices of PEO IWS 7. The focus of current acquisition strategy is to procure open source software that is reusable and interoperable for future systems.

1.3 Program Acquisition Discussion

Implementing an SE process to SPL development is complex and has long been a challenge for the Navy. Software design needs the rigors of an SE process that will ensure the success of the program and its ability to meet DoDAF requirements. The SPL design should be flexible, extensible and feasible through an OA approach; therefore, the success of software design relies upon the ability to design in fragments and the reuse of design ideas and source codes. The program acquisition objective is to develop and validate commonality and consistency among the architectures.

A review of the literature was conducted on graphic languages to understand if there is a unifying view available for MBSE. The findings indicate there are a number of engineering views available with shortcomings in most of them. Further study is required to understand if there are opportunities to use graphical representation, data models in aggregate or if Enhanced Functional Flow Block Diagrams (EFFBDs) are the best alternative for programs and program managers.

1.4 Organization Structure

Our Capstone Team consists of 28 team members located in three different geographical locations. We will be using an Integrated Product and Process Development (IPPD) management technique, which utilizes multi-discipline Integrated Product Teams (IPTs) to optimize our processes. Our IPT organization chart is displayed in Figure 1. We reserve the right to reassign personnel between and within the different IPTs to balance the workload as the project progresses. The following sub-sections will describe the responsibilities of each IPT and potential sub-IPTs:

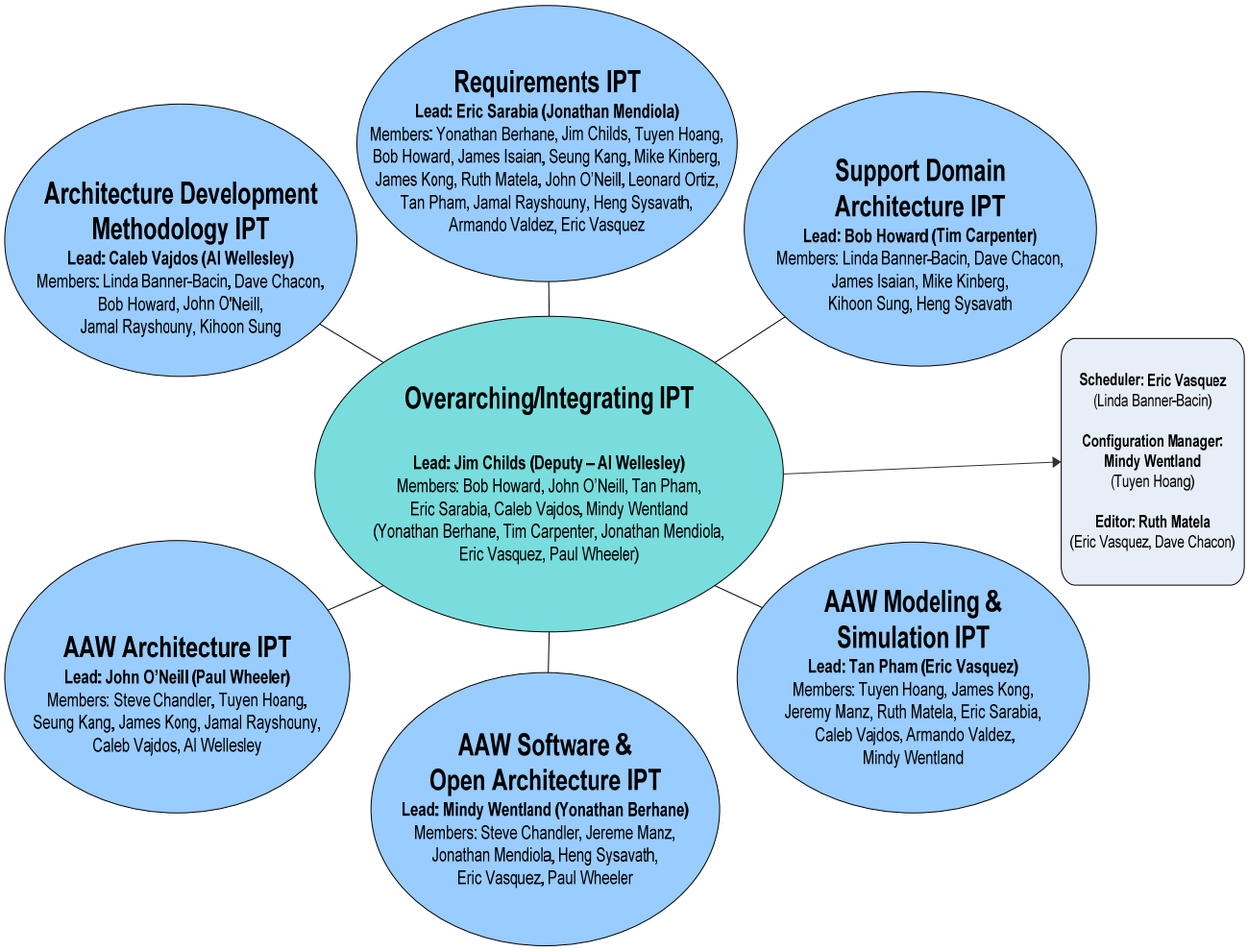


Figure 1: Organization Chart

1.4.1 Overarching / Integrating IPT

a. Integrate schedules and products, and ensure program milestones are met. Products include the Product Management Plan, In Process Status Reports, Summary Presentation, Program Schedules and Risk Assessments.
b. Oversee development of systems engineering products, as required.
c. This IPT shall assign team members to perform the following functions:

- **Team Leader / Deputy Team Leader**

  The Team Leader's primary responsibility will be to facilitate the overall coordination of the project. This includes being the chair of total team meetings, preparing the agenda, reviewing the schedule, getting collaboration on issues, reaching decisions, assigning action items with due dates, and managing the project risks. In the absence of the Team Leader, the Deputy Team Leader will fill these responsibilities. The Team Leader and Deputy Team Leader will be the final sign off of the Project Management Plan. Their signatures represent the concurrence of the whole class.

- **Scheduler**

  The Scheduler will be responsible for developing project schedules and tracking group progress versus planned due dates. The Scheduler will also be responsible for posting the current schedule on Blackboard upon Team Leader approval. Finally, the Scheduler will be responsible for providing status of group performance in meeting timelines during the scheduled In Process Reviews (IPRs).

- **Configuration Manager**

  The Configuration Manager will be responsible for keeping a complete audit trail of decisions, design modifications and documented changes. A Blackboard group file exchange will be created and utilized to store and exchange versioned files of all project deliverables. The Configuration

Manager will maintain backup copies of all files posted on the Blackboard group file exchange. Documentation of any systems engineering project must be maintained throughout the entire lifecycle, ensuring integrity of the information and engineering process. It also enables quality management of products as the methodology is developed for the final proposal.

- **Editor**

The Editor will be responsible for the editorial aspects of project reports and other documentations prepared by the Capstone Team. This includes reviewing, rewriting and editing the work of teammates. The Editor will be responsible for formatting, spelling, grammar, resolving conflicts, and making the report a cohesive document. During the editorial process, it is important that the Editor communicate with the author and the rest of the team. The Editor will collect, merge and render a final editorial decision on each submission. The Editor will maintain version control of all project documentations, and will provide periodic revisions for Blackboard posting. The Editor will also be responsible for verifying the correct format of all references; however, it is the task of the author to provide the necessary references.

- **Meeting Minutes**

Since our Capstone Team consists of 28 people from three different locations, effective meetings are required. An important element of these meetings will be documenting the decisions reached and the actions taken. Therefore, an individual will be assigned at each meeting to take minutes and ensure it gets posted on Blackboard to keep everyone informed of project progress. Furthermore, this individual is responsible for keeping track of the status of all action items to ensure success of the project.

1.4.2 Architecture Development Methodology IPT

a. Develop a methodology for creating complex system architectures (key concepts identified in the project Statement of Work (SOW) will be addressed as required). Methodology is defined to consist of three elements:
   - Techniques/theory
   - Process
   - Tools

b. Develop process to trace requirements from performance specifications through test and evaluation.

c. Collaborate on a regular basis as a cohesive IPT. Report weekly status to Overarching IPT.

1.4.3 Requirements IPT

a. Develop AAW system requirements for analysis and verification of the defined methodology through the development of a combat system architecture example.

b. Develop DoDAF artifacts, use cases, and/or node activity diagrams to support requirements definition in the defined methodology.

c. Support requirement traceability concepts.

1.4.4 AAW Architecture IPT

a. Develop architecture products required to meet AAW performance requirements. Includes identification and development of associated DoDAF views, use cases, node activity diagrams, and preliminary architecture functionality.

b. Process will include method for identifying and leveraging existing requirements.

1.4.5 AAW Software and Open Architecture IPT

a. Develop a conceptual model of predictive reuse through the SPL approach.
b. Develop guidelines for industry to follow NOA mandates.
c. Develop requirements for a Navy Combat System repository.
d. The software development tasks include: software configuration management (CM), configuration identification, configuration control, and configuration audit.
e. All products will conform to OA, SOA and net-centricity concepts.

1.4.6 AAW Modeling & Simulation (M&S) IPT

a. Ensure methodology complies with Navy requirements for simulation-based approach.
b. Collaborate on a regular basis as a cohesive IPT. Report weekly status to Overarching IPT.
c. Provide inputs to the Verification Plan and Report.
d. Provide M&S schedule depicting milestones and deliverables.
e. Develop M&S strategy and other M&S documents required to support milestone decision.
f. Identify and utilize appropriate tools and processes for optimizing the use of M&S to reduce risk and cost.
g. Methods will depict ability to optimize reuse with minimal testing requirements and will address incremental approach along with use of various types of models, such as physical, functional and mathematical as applied to SE process.
h. Simulation will be addressed for determining standard process to develop and execute simulation based on return value.
i. Define the appropriate role of simulation in MBSE.
j. Modelers will be responsible for developing M&S that is certifiable in order to provide a complete predictive analysis of a model system.

1.4.7 Support Domain Architecture IPT

a. Develop common architecture necessary to sustain capability, including training (Watch team and individual), logistics and material management architecture, and mission readiness reporting and personnel support.
   - Excludes ship level functions, such as administrative and medical, but does depict Combat System interface requirements.
   - Identification and development of associated DoDAF views, use cases, node activity diagrams, and preliminary architecture functionality.

b. Develop process to trace architecture to performance specifications developed by Requirements IPT.

c. Process will also include method for identifying and leveraging existing architecture.

d. Plan and develop lifecycle logistics in support of system architecture.

e. Establish and develop maintenance products required for milestone approval, such as Life Cycle Cost Estimate (LCCE), Life Cycle Support Plan (LCSP), Programmatic Environmental, Safety, and Health Evaluation (PESHE), and other deliverables as defined by 5000.2R.

f. Develop method for developing logistics in incremental and spiral development process, and identify process for supporting system KPPs.

g. Establish method for minimizing LCCs through implementation of organic support capabilities and Performance Based Logistics (PBLs), and establish a phased approach to developing logistics based on system and architecture maturity.

h. Develop class maintenance planning strategies for managing COTS and Military Standard (MIL-STD), including determination of optimal refresh cycles, logistics strategy based on refresh cycles, and maintenance updates.

1. 5 Stakeholders

The following organizations and people have been identified as Stakeholders for this Capstone Project. Stakeholders include resource sponsors, acquisition program offices, technical organizations and product users with a vested interest in the successful development and implementation of improved SE processes and methodologies for combat system architecture design. Stakeholders for this project include the following:

a. PEO IWS: Program Executive Office, Integrated Warfare Systems. PEO IWS is responsible for determining the combat system performance requirements and architecture design. POC: TBD
b. NAVSEA 05: Naval Sea Systems Command. NAVSEA 05 is the Navy's Technical Authority for ships and submarines, including all installed systems. NAVSEA 05 is responsible for establishing technical standards and processes to be followed in the development and life cycle support of naval systems. POC: TBD
c. Naval Postgraduate School: NPS is responsible for maintaining academic rigor in all of its degree programs, and ensuring that research projects and theses meet academic requirements and provide products of value to the Navy's technical and operational community. POCs: Mike Green, Rachel Goshorn, Wally Owens
d. NSWC PHD: NSWC PHD has an interest in ensuring PHD employees in the MSSE/MSSEM program receive a quality education in SE that will improve the productivity of the engineers and logisticians that graduate from the program. NSWC PHD is also interested in ensuring capstone projects produce technical products of value to NAVSEA and the Navy. POCs: Tim Troske, Karen Brower
e. PEO C4I/SPAWAR Systems Center: Office of Chief Engineer. POC: Dr. Nichil Dave, Bryan Larish
f. Jeff Grady, International Council on Systems Engineering (INCOSE) Fellow (JOG Systems Engineering, San Diego, CA)
g. Dr. Steve Dam, SPEC (Marshall, VA)
h. Paul Clements, SEI

1.6 Risk Management

The Overarching IPT will develop and implement a risk management process. The risk management process will include risk planning, risk identification, risk assessment, risk handling, and risk monitoring activities. A risk management plan will be developed to identify and track risk drivers, risk mitigation plans, and to perform continuous risk assessment.

The Project IPT Team Leaders will identify schedule and project performance objectives risks in their respective IPTs and provide them to the Overarching IPT during scheduled team meetings. The Overarching IPT will analyze and prioritize all risks. A risk-rating matrix containing the likelihood of an event happening and the consequences of that event happening will be developed and updated on a continuous basis. An overall risk rating will be assigned to each identified risk. The Overarching IPT will then determine if the risk can be mitigated, managed or transferred. The Overarching IPT will maintain a record of all risk mitigation actions taken, track their resolution status, and report results at scheduled team meetings.

2 Systems Engineering Approach

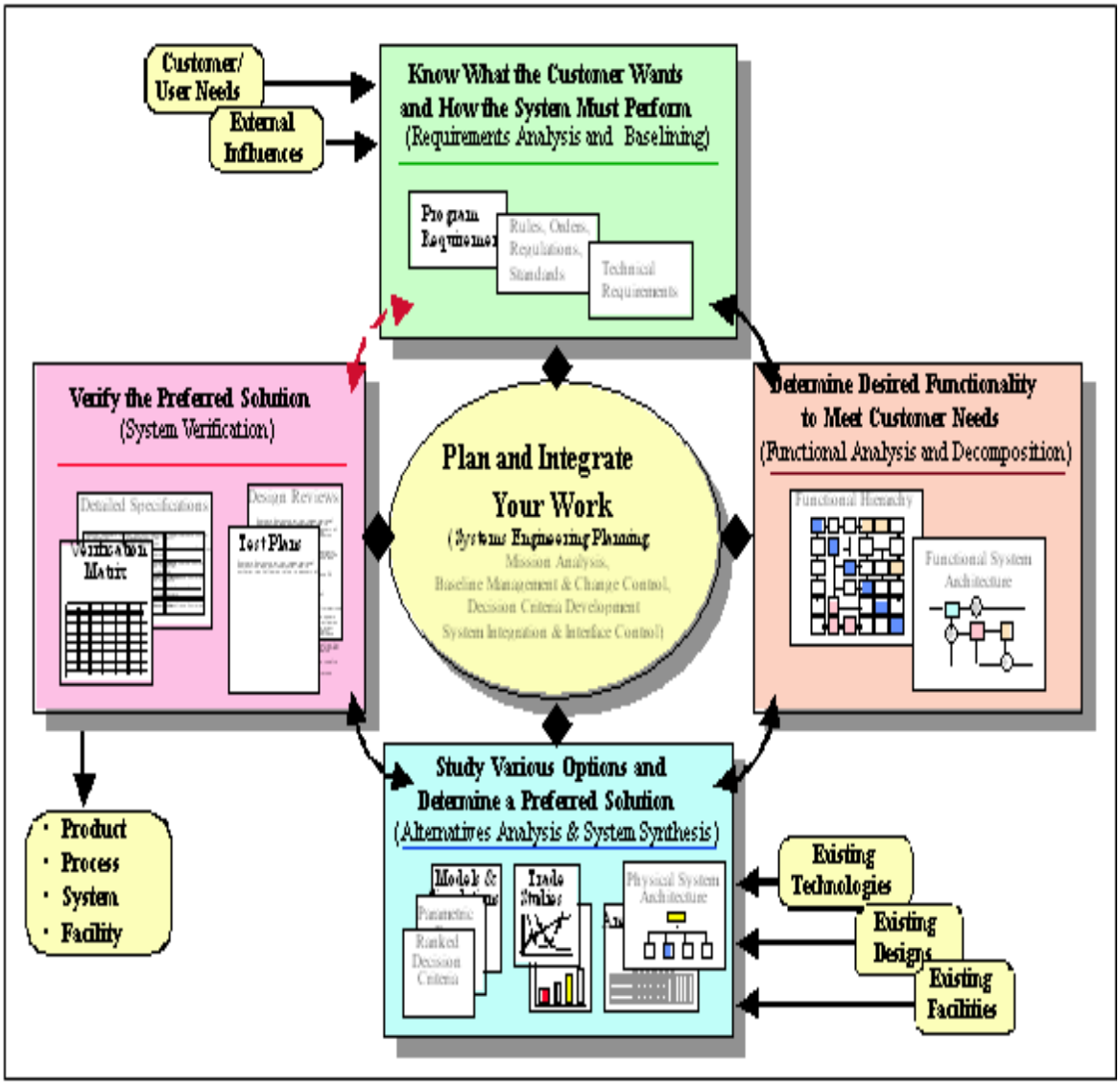
2.1 Overview

The SE approach to achieving our objectives will focus on developing a methodology that will integrate the following key concepts, as well as the necessary tools (e.g., Extend, CORE):

1. Open Architecture
2. Service-Oriented Architecture
3. DoDAF
4. Domain analysis
5. Software product lines
6. Model-based systems engineering
7. Process for Systems Architecture and Requirements Engineering (Hatley et al)
8. The Modular Command Evaluation Structure (MORS)
9. The Systems Engineering "VEE" model
10. Context sensitive modeling (Clymer)
11. Reliability theory
12. Net-centricity

The methodology will consist of three elements: technique/theory, process, and tools. The emerging methodology will determine the appropriate role of simulation in MBSE, and a process for developing combat system SPLs that uses the DoDAF framework as a specification and supports software reuse. Several SE processes were examined in choosing an approach most suitable to the project, including the *Vee* model, the Spiral model and the Waterfall model. Although all the examined SE processes were found to be suitable for the task, the Plowman's SE Process Model illustrated in Figure 2 was selected because of its inclusion of the following main activities vital to the development of our Capstone SE methodology: Requirements Analysis, Functional Analysis, Alternatives Analysis, and System Verification. Proper requirements analysis that is clearly defined and traceable to what the stakeholders require needs to be ensured. Tasks will then be applied to the refined requirements through rigorous SE to develop a set of

feasible alternatives, resulting in a final recommendation to the stakeholders. The following sections will provide further details regarding each SE component.



Plowman's SE Process Model is a cyclic and recursive model that is applied at different levels of rigor depending on the type of program or project involved. If you "unroll" the SE Process, the central "Plan and Integrate Your Work" element breaks into five distinct activities that bridge the efforts from the four main blocks. While many cycles of the SE process typically occur during the course of a project, each cycle covers the same basic steps only with different levels of emphasis and rigor.

Figure 2: Plowman's Systems Engineering Process Model[7]

[7] INCOSE. 2003. *Guide to the Systems Engineering Body of Knowledge (G2SEBoK) Versin 3.50.* http://g2sebok.incose.org/

2.2 Requirements Analysis

The initial process in the SE Approach is the Requirements Analysis. Stakeholder-generated requirements are inputs to the Requirements Analysis, and become outputs as effective requirements through the use of relevant tools. The relevant tools, or Requirements Analysis Toolkit, may consist of a combination of the following:

1. Stakeholder Analysis
2. Threat Analysis
3. Functional Analysis
4. Futures Analysis
5. Use Cases

The application of each tool will be determined as the project progresses. A stakeholder analysis, threat analysis and functional analysis are typically performed in the development of an effective requirements statement. The effective requirements statement is then examined in collaboration with the stakeholders.

2.3 Functional Analysis

Functional Analysis consists of the Functional Area Analysis (FAA), Functional Needs Analysis (FNA), Functional Solutions Analysis (FSA) and the Post-Independent Analysis (PIA), as specified in the Capabilities Based Assessment (CBA) functional area of the Joint Capabilities Integration Development System (JCIDS) analysis process.[8] Functional Analysis must be accomplished to deal with the complexity of today's systems and to provide a complete design with minimum errors.

Functional Analysis is a methodology for analyzing the mission and performance requirements of a system and decomposing them into discrete tasks or activities. It is a top-level conceptual definition approach, which allows the determination of what the system must do at varying levels of abstraction. It involves the hierarchical

[8] Defense Acquisition University. 2005. *JCIDS Manual (CJCSM 3170.01B).* https://acc.dau.mil/CommunityBrowser.aspx?id=19936.

decomposition of the primary system functions into sub-functions at increasing levels of detail, promoting structured design with the designer proceeding from general to specific levels of abstraction in an orderly fashion. Functional Analysis provides the link between what the system must do and the resulting architecture. It helps the architect to develop a complete design solution by providing a systematic means of identifying all of the functional elements that need to be incorporated into the system. Furthermore, it provides an integrated view of the design to help the architect understand and manage the system under development as an integrated whole.[9]

2.4 Supportability Analysis

The maintenance and supportability concept will be developed during the conceptual design phase within the SE process. This concept will evolve from the system operational requirements. The maintenance plan will define the follow-on requirements for system support based on the known design configuration and the results of the Supportability Analysis. The maintenance and support domain architecture will be built to support the defined maintenance and supportability concept. Corrective and preventive maintenance will be considered when designing the architecture. The architecture will also consider reliability, maintainability, usability, supportability, serviceability, producibility, disposability and affordability. The software architecture will take reconfiguration into account. Furthermore, an evaluation of logistics support factors will be considered, including depot support, repair policy, spares and inventory cost, reliability vs. maintainability, built-in test (BIT) capability and test methods. The identification and prioritization of technical performance measures associated with the prime system is essential when designing the maintenance and support domain architecture.

---

[9] Cole Jr., L. Elbert. 1998. *Functional Analysis: A System Conceptual Design Tool.* http://ieeexplore.ieee.org/iel4/7/14739/00670319.pdf?isnumber=14739&prod=JNL&arnumber=670319&arSt=354&ared=365&arAuthor=Cole%2C+E.L.%2C+Jr.

2.5 Alternatives Analysis

Alternatives Analysis is a tool used to compare various phases of the design and development process, which may raise new issues and questions traditional analysis methods may not yield. MOPs and MOEs can be developed from the requirements and functional analyses results. Using methods such as Zwicky's morphological box and trade studies, alternatives are generated based on the developed MOPs and MOEs. Using various methods, such as M&S, to analyze and compare the alternatives, feasible alternatives are segregated from those that are infeasible. Examples of software applications that may be used for M&S are CORE, Extend, Arena, Crystal Ball and @Risk. The DoDAF, which utilizes the Integrated Definition for Function Modeling (IDEF0) diagram and Unified Modeling Language (UML), and the Functional Flow Block Diagram (FFBD) will also be used extensively for modeling. The best alternatives will be chosen based on numerical results as well as subjective judgments. The stakeholders will then review the remaining alternatives and determine the preferred alternative. If the stakeholders reject all the submitted alternatives, the process will be repeated until an acceptable alternative is approved.

2.6 System Verification

System Verification is when the solution is compared to the requirements for each application of the SE process. Requirements at each level of development must be verifiable. For this capstone project, the examination of the AAW mission will serve as a means of System Verification by proving the SE methodology developed meets the specified requirements.

3 Milestones & Deliverables

Milestone	*Description*	*Deliverable*	*Date*
1	Project Management Plan Approval	Project Management Plan	**22 Aug 2008**
2	In Process Review #1 ***(End Summer Qtr 08)***	Requirements Analysis Research Status	**22 Sept 2008**
3	Analysis Review	Functional Analysis Supportability Analysis	**11 Nov 2008**
4	In Process Review #2 ***(End Fall Qtr 08)***	Methodology Outline Verification Status	**15 Dec 2008**
5	Verification Review	Methodology Verification (Modeling and Simulation)	**21 Jan 2009**
6	Final Report Review	Final Report	**9 Feb 2009**
7	**In Process Review #3** ***(End Winter Qtr 09)***	**Project Presentation and Final Report**	**23 Mar 2009**

Table 1: Milestones and Deliverables

3.1 Schedule

ID	Task Name	Duration	Start	Finish
1	**Project Management Plan**	**35 days**	**Mon 7/7/08**	**Fri 8/22/08**
2	Develop Plan	21 days	Mon 7/7/08	Mon 8/4/08
3	Submit Plan	14 days	Tue 8/5/08	Fri 8/22/08
4	**Conceptual Design (1.1)**	**110 days?**	**Thu 8/21/08**	**Wed 1/21/09**
5	**Requirements Analysis (2.0)**	**23 days?**	**Thu 8/21/08**	**Mon 9/22/08**
6	Needs Analysis	10 days?	Thu 8/21/08	Wed 9/3/08
7	System Oper Req	17 days?	Thu 8/21/08	Fri 9/12/08
8	Func Requirements	17 days?	Thu 8/21/08	Fri 9/12/08
9	System Maint Conc	23 days?	Thu 8/21/08	Mon 9/22/08
10	Status Report (1st Quarter Report)	11 days?	Mon 9/8/08	Mon 9/22/08
11	**Functional Analysis**	**36 days?**	**Tue 9/23/08**	**Tue 11/11/08**
12	Funct Analysis	31 days?	Tue 9/23/08	Tue 11/4/08
13	System Oper Funct	31 days?	Tue 9/23/08	Tue 11/4/08
14	System Maint Functions	31 days?	Tue 9/23/08	Tue 11/4/08
15	System Analysis	36 days?	Tue 9/23/08	Tue 11/11/08
16	**Requirements Allocation**	**24 days?**	**Wed 11/12/08**	**Mon 12/15/08**
17	Alloc Perf , Design Fact/Effec Req,	24 days?	Wed 11/12/08	Mon 12/15/08
18	Alloc of Sys Supprt Req	24 days?	Wed 11/12/08	Mon 12/15/08
19	Sys Analysis	24 days?	Wed 11/12/08	Mon 12/15/08
20	**Trade-off Studies**	**36 days?**	**Mon 12/1/08**	**Mon 1/19/09**
21	Sys and Subs Tradeoffs Eval of Alter	36 days?	Mon 12/1/08	Mon 1/19/09
22	Sys and Subsystem Analysis	36 days?	Mon 12/1/08	Mon 1/19/09
23	**Synthesis and Evaluation**	**51 days?**	**Wed 11/12/08**	**Wed 1/21/09**
24	Prelim Design	51 days?	Wed 11/12/08	Wed 1/21/09
25	Configuration and Arrangement of Syst	51 days?	Wed 11/12/08	Wed 1/21/09
26	Detail Specs	51 days?	Wed 11/12/08	Wed 1/21/09
27	Status Report (2nd Quarter Report)	11 days?	Mon 12/1/08	Mon 12/15/08
28	**Final Report**	**146 days?**	**Mon 7/21/08**	**Mon 2/9/09**
29	Final Report Development	129 days?	Mon 7/21/08	Thu 1/15/09
30	Final Report Review	11 days?	Fri 1/16/09	Fri 1/30/09
31	Final Report Submssion	1 day?	Mon 2/9/09	Mon 2/9/09
32	**GRADUATION Presentation**	1 day?	Mon 3/23/09	Mon 3/23/09

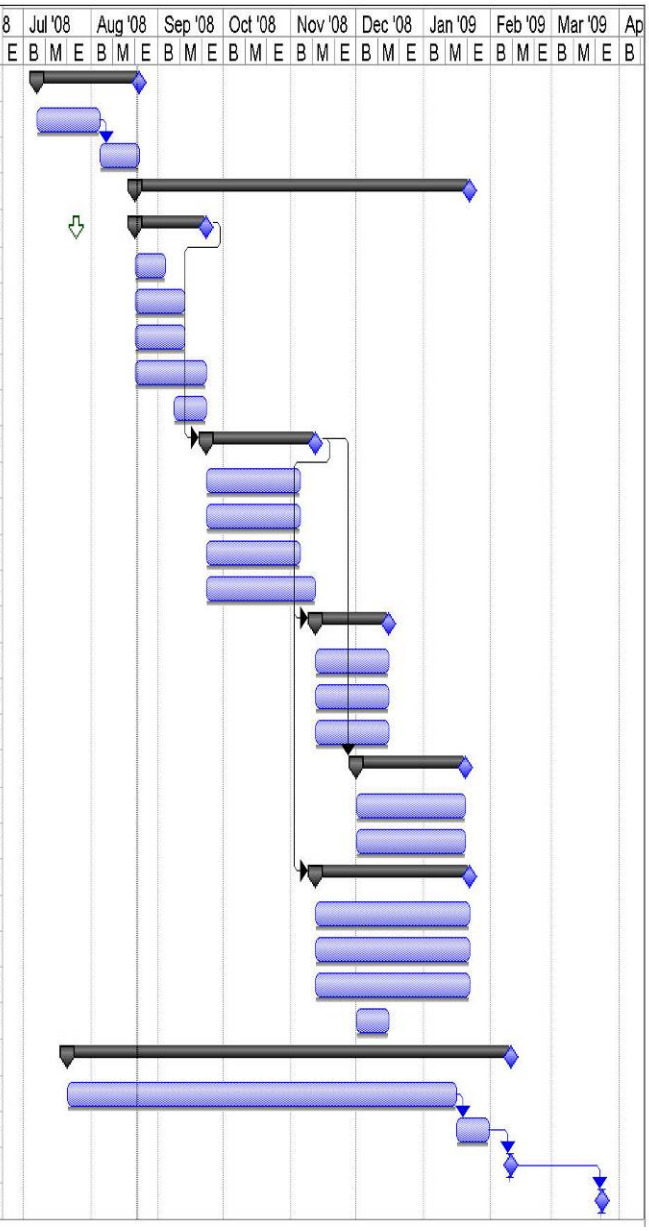


Figure 3: Project Schedule

3.2 Critical Path Method

Milestones are developed to identify major events in the program development process. A milestone usually indicates the completion of a set of events. The milestones for the project have been identified and will trigger the occurrence of certain events. Milestones represent high-level events and are considered to be communications points; they should not be too large or too small in number. The milestones for this project indicate the completion of a design phase and requirement for review and feedback, at which time team agreement must be reached before proceeding to the next phase.

The project will use a Gantt chart to develop a schedule with start and end dates for each activity, as illustrated in Figure 3. The Gantt chart also helps identify what predecessors are required prior to moving ahead. With this type of project it is imperative that we maintain the schedule and allow ample time for development, review, buy in, simulation and analysis. The end result will be the most effective SE methodology with regards to SPL development.

The critical path is an essential method in determining the overall model of activities that are required to occur in outlining the project's completion routine. This method includes the duration and inter-dependencies of all critical activities. The critical path for our project is scheduled as follows: Project Management Plan, Requirements Analysis, Functional Analysis, Requirements Allocation, Trade-off Studies concurrent with Synthesis and Evaluation, Final Report completion, and the Capstone Project Presentation. The duration and predecessors for these critical activities are detailed in the Gantt chart.

The project schedule outlines a one-month duration for the Project Management Plan. After this period, the management functions are expected to proceed in creating IPTs with required charter development and individual roles and responsibilities. Following this activity, the major Conceptual Design period can be expected to occur, with duration

of approximately five months. Conceptual Design includes all the major processes and activities that are required to produce the design goals for this project. It is important to note that within the Conceptual Design period, the Requirements Allocation, Trade-off Studies, and Synthesis and Evaluation activities will all occur in parallel processes, and share Functional Analysis as a predecessor. Schedule analysis has outlined this parallel activity period as being the most efficient way of dividing work among the capstone team members and completing all the tasks within the allotted time periods. Final Report generation is scheduled concurrently with the Conceptual Design activity, which is expected to last approximately six weeks before the Capstone Presentation to all stakeholders, academic faculty and interested parties.

Acronyms

Acronym	Term
AAW	Anti-Air Warfare
AoA	Analysis of Alternatives
BIT	Built-In-Test
C4I	Command, Control, Communications, Computers, & Intelligence
CBA	Capabilities Based Document
CM	Configuration Management
COTS	Commercial-Off-The-Shelf
DDG (1000)	Destroyer, Guided Missile, Next Generation
DoD	Department of Defense
DoDAF	Department of Defense Architecture Framework
EFFBD	**Enhanced Functional Flow Block Diagram**
FAA	**Functional Area Analysis**
FFBD	**Functional Flow Block Diagram**
FNA	**Functional Needs Analysis**
FSA	**Functional Solutions Analysis**
GPR	Government Purpose Rights
ICD	Initial Capabilities Document
IDEF0	Integrated Definition for Function Modeling
INCOSE	International Council on Systems Engineering
IPPD	Integrated Product and Process Development
IPR	In Process Review
IPT	Integrated Product Team
IWS	Integrated Warfare Systems
JCIDS	Joint Capabilities Integration Development System
KPP	Key Performance Parameter
LCC	Life Cycle Cost
LCCE	Life Cycle Cost Estimate
LCSP	Life Cycle Support Plan
M&S	Modeling and Simulation

MBSE	Model-Based Systems Engineering
MIL-STD	Military Standard
MOE	Measure Of Effectiveness
MOP	Measure Of Performance
MSSE	Masters of Science in Systems Engineering
MSSEM	Masters of Science in Systems Engineering Management
NAVSEA	Naval Sea Systems Command
NOA	Naval Open Architecture
NPS	Naval Postgraduate School
NSWC	Naval Surface Warfare Center
OA	Open Architecture
OPNAV	Office of the Chief of Naval Operations
PBL	Performance Based Logistics
PEO	Program Executive Office; Program Executive Officer
PESHE	Programmatic Environmental, Safety, and Health Evaluation
PHD	Port Hueneme Division
PIA	Post Independent Analysis
PMP	Project Management Plan
POC	Point of Contact
SE	Systems Engineering
SEI	Software Engineering Institute
SOA	Service-Oriented Architecture
SPAWAR	Space and Naval Warfare
SPL	Software Product Line
SSDS	Ship Self Defense System
UML	Unified Modeling Language

References

Blanchard, S. Benjamin, and Wolter J. Fabrycky. 2006. *Systems Engineering and Analysis.* 4th Edition. New Jersey: Prentice Hall International.

Cole Jr., L. Elbert. 1998. *Functional Analysis: A System Conceptual Design Tool.* http://ieeexplore.ieee.org/iel4/7/14739/00670319.pdf?isnumber=14739&prod=JNL&arnumber=670319&arSt=354&ared=365&arAuthor=Cole%2C+E.L.%2C+Jr.

Defense Acquisition University. 2006. *Defense Acquisition Guidebook.* https://akss.dau.mil/dag.

Defense Acquisition University. 2005. *JCIDS Manual (CJCSM 3170.01B).* https://acc.dau.mil/CommunityBrowser.aspx?id=19936.

Department of Defense. 2007. *DoD Architecture Framework Version 1.5.* http://www.defenselink.mil/cio-nii/docs/DoDAF_Volume_I.pdf.

INCOSE. 2003. *Guide to the Systems Engineering Body of Knowledge (G2SEBoK) Version 3.50.* http://g2sebok.incose.org/

Long, James. 2002. *Relationships between Common Graphical Representations in System Engineering.* http://www.vitechcorp.com/whitepapers/files/200701031634430.CommonGraphicalRepresentations_2002.pdf.

PEO-IWS 7. 2007. *Naval Open Architecture Contract Guidebook, Version 1.1.* https://acc.dau.mil/CommunityBrowser.aspx?id=183088&lang=en-US.

Software Engineering Institute. Software Product Lines. Carnegie Mellon University. http://www.sei.cmu.edu/productlines/.

APPENDIX C: INTEGRATED DICTIONARY

Term	Acronym	Definition
Architecture Interconnection Diagram	AID	Depicts the system architecture from a logical point of view (3SL, Cradle)
AP233	AP233	AP233 is an information model designed as a neutral data exchange capability for data created by Systems Engineering computer applications. This document also specifies how to use the AP233 XML Schema to represent SE concept for exchange between tools. The purpose of this document is to be the target for links from documentation mapping other information models, standards and notations into AP233. (Open Systems Joint Task Force)
Acoustic-Rapid Commercial Off the Shelf Insertion	A-RCI	Ability to rapidly install a marked technological refresh in equipment at a lower cost. (Lockheed Martin)
Acquisition Category	ACAT	Categories established to facilitate decentralized decision making and execution and compliance with statutorily imposed requirements. The categories determine the level of review, decision authority, and applicable procedures. (DAU)
Acquisition of Complex C4I Systems		The DoD has been working for decades to develop and field systems that are interoperable with other systems within each of the U.S. military services, among the services, and with allied forces. Improved technologies and increased capability requirements have made the fielding of interoperable systems of systems more challenging. System designers and technical agents have had to create complex architectures to describe and define these systems of systems. With no standards in place for depicting these architectures, inconsistencies arose in the way architectures were developed and documented. These inconsistencies with requirements and design documentation exacerbated the incompatibility problems of the underlying systems. (DAU)
Acquisition Personnel		The Defense Acquisition Workforce Improvement Act (DAWIA) of 1990 specified acquisition career fields and educational, training and experience requirements to fill positions at various levels in those fields. (DAU)
Acquisition Reform		This initiative focused on eliminating unnecessary military technical standards and adopting common commercial standards and processes to the maximum extent possible. (DAU)

Acquisition Strategy		A business and technical management approach designed to achieve program objectives within the resource constraints imposed. It is the framework for planning, direction, contracting for and managing a program. It provides a master schedule for research, development, test, production, fielding, modification, postproduction management and other activities essential for program success. The AS is the basis for formulating functional plans and strategies, Test and Evaluation Master Plans, Acquisition Plan, competition, systems engineering. (DAU)
Acquisition, Technology, & Logistics	AT&L	Identifies key processes that must work in concert to deliver capabilities required by the warfighter. The requirement processes are: Joint Capabilities Integration & Development System (JCIDS), Defense Acquisition System, Planning, Programming, Budgeting and Execution (PPBE). DAU
Activity Diagram		Activity diagrams represent the business and operational workflows of a system. An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state. (Developer.com)
Affordability		A determination that the Life Cycle Cost (LLC) of an acquisition program is in consonance with the long-range investment and force structure plans of the DoD. (DAU)
Agile Methods		Agile methodologies generally promote a project management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices that allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals. (Wikipedia)
Air Contacts	AC	Identify activity above surface ground. (lb)
All View	AV	An architecture view that provides a summary and overview information. It describes the scope, purpose, intended users, environment depicted, and analytical findings associated with the architecture. (CJCSM 3170.01C)
Alternative System Review	ASR	A multi-disciplined technical review to ensure that the resulting set of requirements agrees with the customers' needs and expectations and that the system under review can proceed into the Technology Development phase. (DoD 5000.1)
Analysis of Alternatives	AoA	The evaluation of the performance, operational effectiveness, operational suitability, and estimated costs of alternative systems to meet a mission capability. The

		analysis assesses the advantages and disadvantages of alternatives being considered to satisfy capabilities, including the sensitivity of each alternative to possible changes in key assumptions or variables. The AoA is normally conducted during the Concept Refinement phase of the Defense Acquisition Framework to refine the system concept contained in the Initial Capabilities Document (ICD) approved at the Concept Decision. (DoDI 5000.2 and CJCSI 3170.01F)
Anthropometrics		Typical anthropometric measurements include standing stature, weight, distance between eyes, and circumference around waist. However, sensory abilities may also be measured, such as hearing ability, sight, and the ability to sense touch. (BSU)
Anti-Air Warfare	AAW	A US Navy/US Marine Corps term used to indicate that action required destroying or reducing to an acceptable level the enemy air and missile threat. It includes such measures as the use of interceptors, bombers, anti-air craft guns, surface-to-air and air-to-air missiles, electronic attack and destruction of the air or missile threat both before and after it is launched.
Anti-Ship Cruise Missile	ASCM	A cruise missile is a guided missile that carries an explosive payload and uses a lifting wing and a propulsion system, usually a jet engine, to allow sustained flight; it is essentially a flying bomb. Cruise missiles are generally designed to carry a large conventional or nuclear warhead many hundreds of miles with high accuracy. Modern cruise missiles can travel at supersonic or high subsonic speeds, are self-navigating, and fly on a non-ballistic very low altitude trajectory in order to avoid radar detection. (Wikipedia)
Anti-Ship Missile	ASM	Guided missiles designed for use against ships. Most anti-ship missiles are of the sea-skimming type and use a combination of inertial guidance and radar homing. These missiles can be launched from a variety of platforms including ships, aircraft (including helicopters), land vehicles and submarines. (Wikipedia)
Anti-Ship Missile Defense	ASMD	Anti-ship missile defense systems are a defense system that would be the only defense against the incoming missiles. U.S. Aegis destroyers equipped with missile defense system. The SSDS system correlates sensor information, assesses own-ship defense readiness and recommends optimal tactical defense responses. With enhanced target tracking, SSDS expedites the assignment of weapons for manual threat engagements and provides operators with a 'recommend engage' solution. When operating in the automatic mode, the SSDS will coordinate both soft and hard kill employment.

Anti-Submarine Warfare	ASW	Littoral ASW operations protect naval forces, commercial and logistics shipping from enemy submarines, and thereby enable naval forces to project power ashore, conduct strategic sealift operations, and control or interdict sea lines of communications (SLOCs) that affect littoral objectives. (Naval Doctrine Command)
AP233		Application Protocol for Systems Engineers (233) INCOSE
Applied Physics Laboratory	APL	The Applied Physics Laboratory (APL) is a not-for-profit center for engineering, research and development. APL is a division of one of the world's premier research universities, Johns Hopkins. Our 399-acre campus, 20 miles north of Washington, DC, is home to 4,300 men and women. We recruit and hire the best and the brightest from top colleges, 68% of those hires are engineers and scientists. We work on more than 400 programs that protect our homeland and advance the nation's vision in research and space science, at an annual funding level of about $680M.
Architecture Block Diagram	ABD	Block diagram is a diagram of a system, in which the principal parts or functions are represented by blocks connected by lines, that show the relationships of the blocks. They are heavily used in the engineering world in hardware design, software design, and process flow diagrams. The Architecture Block Diagram is a high level view that shows the lay out of the system.
Architecture Context Diagram	ACD	Context Diagram is a data flow diagram showing data flows between a generalized application within the domain and the other entities and abstractions with which it communicates. Carnegie Mellon Software Engineering Institute
Architecture Dictionary		Glossary of terms used for design
Architecture Flow Diagram	AFD	A diagram comprising architecture modules and architecture data flows, or in an AFCD, one module, terminators and architecture flows. (Process for System Architecture, Hatley, Pirbai and Hruschka)
Architecture Flow Diagram level 0	AFD0	A diagram comprising architecture modules and architecture data flows, or in an AFCD, one module, terminators and architecture flows. 0 is the original diagram that the various levels are built from. (Process for System Architecture, Hatley, Pirbai and Hruschka)
Architecture Interconnection Diagram	AID	Interconnection diagrams show the cabling between electronic units and how the units are interconnected. All terminal boards are assigned reference designations according to the unit numbering method described

		previously. Individual terminals on the terminal boards are assigned letters and/or numbers according to *Standard Terminal Designations for Electronic Equipment*, NAVSHIPS 0967-146-0010. (Integrated Publishing, Draftsman)
Assistance Secretary of the Navy	ASN	Assistant Secretary of the Navy (ASN) is the title given to certain senior officials in the United States Department of the Navy. As of 2007, there are four Assistant Secretaries of the Navy: Assistant Secretary of the Navy for Research, Development and Acquisition, Assistant Secretary of the Navy for Manpower and Reserve Affairs, Assistant Secretary of the Navy for Financial Management and Comptroller, Assistant Secretary of the Navy for Installations and Environment. (Wikipedia)
Block Definition Diagram	BDD	A block definition diagram describes the system hierarchy and system/component classifications. The engineering block diagram is the representation of a system or part of a system in forms such as rectangles for system elements and lines that represent interfaces. (Hatley, Pirbhai, Hruschka)
Built-In-Test	BIT	An integral capability of the mission system or system which provides an automated test capability to detect, diagnoses, or isolate failures. (DAU)
Capabilities Based Document	CBA	Document that is based on capabilities of the system.
Capabilities Development Document	CDD	A Capability Development Document (CDD) provides operational performance attributes, including supportability, for those responsible for the acquisition of military equipment in the military of the United States. It includes "key performance parameters" (KPPs) and other parameters that guide the development, demonstration, and testing of the current increment. It also outlines the overall strategy for developing full capability. The format for the CDD is spelled out in the Chairman of the Joint Chiefs of Staff Manual (CJCSM) 3170.01
Carrier Vessel Nuclear; also nuclear powered aircraft carrier	CVN	The Nimitz Class aircraft carriers are the largest warships ever built. With over 6,000 personnel (crew and aircrew), the carrier has a displacement of 102,000 tons, and a flight deck length of 332.9 meters. Newport News Shipbuilding (now Northrop Grumman Ship Systems), based in Virginia, has built all nine nuclear-powered Nimitz Class carriers in.
Chief Of Naval Operations	CNO	The Chief of Naval Operations (CNO) is the highest-ranking officer in the United States Navy and is a member of the Joint Chiefs of Staff. The CNO reports directly to the Secretary of the Navy for the command, utilization of resources and operating efficiency of the operating forces

		of the Navy and of the Navy shore activities assigned by the Secretary. Under the authority of the Secretary of the Navy, the CNO also designates naval personnel and naval resources to the commanders of Unified Combatant Commands. The CNO also performs all other functions prescribed under 10 U.S.C. § 5033 and those assigned by the secretary or delegates those duties and responsibilities to other officers in his administration under his name. Like the other joint chiefs, the CNO is an administrative position and has no operational command authority over United States naval forces. Current CNO is Admiral Roughhead. (Wikipedia)
Closest Point of Approach	CPA	An estimated point in which the distance between two objects, of which at least one is in motion, will reach its minimum value; abbreviated CPA. The estimate is used to evaluate the risk of a collision of e.g. two ships. (Wictionary)
Combat Air Patrol	CAP	A combat air patrol is an aircraft patrol provided over an objective area, over the force protected, over the critical area of a combat zone, or over an air defense area, for the purpose of intercepting and destroying hostile aircraft before they reach their target. Combat air patrols apply to both overland and overwater operations, protecting aircraft, fixed and mobile sites on land, and ships at sea. (Wikipedia)
Combat Systems		A complete system, integrating state-of-the-art radar and missile systems, highly integrated and capable of simultaneous warfare on several fronts -- air, surface, subsurface, and strike. (Global Security.org)
Command and Control	C2	Command and control can be defined as the exercise of authority and direction by a properly designated commanding officer over assigned and attached forces in the accomplishment of the mission.
Command, Control, Communications, Computers, & Intelligence	C4I	C4I goal is to establish and maintain information superiority in support of the National Security Strategy of the United States. To fulfill this goal, the Department must: Provide the secure information capabilities needed by war fighters and other command authorities to effectively and successfully prosecute any mission. Enable the commanders of military forces and the managers of support activities to achieve the highest effectiveness, agility, and efficiency in their operations through the effective use of information. Assure a global capability to share and exchange information, and to provide required information in sufficient depth, security, clarity, and timeliness for decision makers to arrive at informed decisions. Ensure that quality, timely intelligence and counterintelligence support the operational needs of DoD

		and national-level decision makers. Continuously re-evaluate security practices and costs and applies appropriate risk management wherever possible. Forge a partnership with industry, allies, and coalition partners to define, nurture, promote, and exploit C4I concepts and technologies to meet defense requirements. (US DOD Official Website, www.defenselink.mil)
Command, Control, Communications, Computers, & Intelligence, Surveillance	C4ISR	C4ISR programs integrate data and information sources to increase situational awareness and provide command and control and decision support at all levels of command. (Lockheed Martin)
Command, Control, Communications, Computers, Combat Systems, Intelligence, Surveillance and Reconnaissance	C5ISR	C5ISR includes Combat Systems, see definitions above.
Commercial-Off-The-Shelf	COTS	Commercial items that require no unique government modifications or maintenance over the life cycle of the product to meet the needs of the procuring agency. (DAU)
Compiler Monitor Systems - 2	CMS2	A multi-level programming, which allows the system designer to define the levels of language, constructs which are appropriate for the various types of program modules in a large self-contained software system. The approach taken is to design a language and compiler-monitor system (CMS-2RS), which will facilitate the multi-level programming concept and the top-down programming method of software engineering in a production library environment. (NPS, Vincent Sacades, David Rummler)
Computer Based Training	CBT	Computer-based training, a type of education in which the student learns by executing special training programs on a computer. CBT is especially effective for training people to use computer applications because the CBT program can be integrated with the applications so that students can practice using the application as they learn. (Webopedia, Computer Dictionary)
Computer-Aided Systems Engineering	CASE	A category of software that provides a development environment for programming teams. CASE systems offer tools to automate, manage, and simplify the development process.
Concept of Operations	ConOps	A verbal or graphic statement, in broad outline, of a commander s assumptions or intent in regard to an operation or series of operations. It is designed to give an overall picture of the operation. It is also called the Commanders Concept. (CJCSI 3170.01F)

Configuration Management	CM	Configuration management (CM) is a field of management that focuses on establishing and maintaining consistency of a product's performance and its functional and physical attributes with its requirements, design, and operational information throughout its life. For information assurance, CM can be defined as the management of security features and assurances through control of changes made to hardware, software, firmware, documentation, test, test fixtures, and test documentation throughout the life cycle of an information system. (Wikipedia)
Constrained Environment		(DOD) In the context of joint operation planning, a requirement placed on the command by a higher command that dictates an action, thus restricting freedom of action. Operational Limitation; restraint. (DOD) An action required or prohibited by higher authority, such as a constraint or a restraint, and other restrictions that limit the commander's freedom of action, such as diplomatic agreements, rules of engagement, political and economic conditions in affected countries, and host nation issues.
Context Diagram		Software engineering and systems engineering are diagrams that represent all external entities that may interact with a system. This diagram is the highest-level view of a system, similar to Block Diagram, showing a, possibly software-based, system as a whole and its inputs and outputs from/to external factors. (Wikipedia)
Context Sensitive Modeling		Refers to a program feature that changes depending on what you are doing in the program.
Control Flow Diagram	CFD	A control flow diagram can consist of a subdivision to show sequential steps, with if-then-else conditions, repetition, and/or case conditions. Suitably annotated geometrical figures are used to represent operations, data, or equipment, and arrows are used to indicate the sequential flow from one to another. (Wikipedia)
Control Specifications	CSPECS	CSPECS are used to indicate how the software behaves when an event or control signal is activated and which corresponding processes of the Data Model are invoked. (Capstone Project)
CORE		Vitech's automated tools allow systems engineers to capture, analyze, and verify design integrity before problems embed themselves in the system. Vitech's model-based systems engineering methodology and CORE software, provides: Comprehensive traceability, Extensive behavioral modeling notations representing control flow, function flow, data flow, resource utilization, and interface-link capacity, System simulations automatically synchronized with behavioral models,

		System documentation generated directly from the database as a by-product of the engineering effort. (Vitech Website)
Data Context Diagram	DCD	The highest-level diagram of a layered set of data flow diagrams, showing the system functions as a single process, the terminators with which the system must interact, and the data flows between the system function and the terminator. (Process for System Architecture, Hatley, Pirbai, Hruschka)
Data Flow Diagram	DFD	A diagram consisting of processes, stores and data flows, or in a DCD, one process and terminators, connected by data flows between them. (Process for System Architecture, Hatley, Pirbhai, Hruschka)
Data Flow Diagram Level 0	DFD0	A diagram consisting of processes, stores and data flows, or in a DCD, one process and terminators, connected by data flows between them. Lowest level of decomposition. (Process for System Architecture, Hatley, Pirbhai, Hruschka)
Data Flow Diagram Level 2	DFD2	A diagram consisting of processes, stores and data flows, or in a DCD, one process and terminators, connected by data flows between them. Higher level decomposition. (Process for System Architecture, Hatley, Pirbhai, Hruschka)
Deadlock		A situation wherein two or more competing actions are waiting for the other to finish, and thus neither ever does. It is often seen in a paradox like 'the chicken or the egg'. Wikipedia
Decomposition		Refers to the process by which a complex problem or system is broken down into parts that are easier to conceive, understand, program, and maintain. (Wikipedia)
Defense Acquisition University	DAU	Authorized by Title 10, United States Code 1746, and chartered by Department of Defense (DoD) Directive 5000.57, the Defense Acquisition University provides practitioner training, career management, and services to enable the DoD Acquisition, Technology and Logistics community to make smart business decisions and deliver timely and affordable capabilities to the warfighter. DAU provides a full range of basic, intermediate, and advanced curriculum training, as well as assignment-specific and continuous learning courses to support the career goals and professional development of the DoD.
Defense Acquisition Workforce Improvement Act	DAWIA	Mandated that PEO positions be designated as critical acquisition positions (CAP) DAU.
Defense Science board	DSB	Over its 40 plus years, the Board has ably served the nation in numerous ways by providing innovative

		solutions to myriad technological, operational and managerial problems. In the transformation processes needed to ensure success in the military and national security endeavors, such wise counsel will be of even greater value to the Department. (DAU)
Define Mission		Missions as homeland defense and civilian support; deterrence operations; major combat operations; irregular warfare; military support to stabilization, security, transition and reconstruction operations; and military contribution to cooperative security. (Defenselink.com)
Department of Defense	DoD	DoD is the federal department charged with coordinating and supervising all agencies and functions of the government relating directly to national security and the military. The organization and functions of the DOD are set forth in Title 10 of the United States Code.
Department of Defense Architecture Framework	DoDAF	The Department of Defense Architecture Framework (DoDAF) defines how to organize the specification of enterprise architectures for U.S. Department of Defense (DoD) applications. All major DoD weapons and information technology system procurements are required to document their enterprise architectures using the view products prescribed by the DoDAF. DoDAF is well suited to large systems and systems-of-systems (SoSs) with complex integration and interoperability issues.
Department of Defense Directive	DODD	The principal DoD directive on acquisition, it states policies applicable to all DoD acquisition programs. These policies fall into five major categories: 1) Flexibility, 2) Responsiveness, 3) Innovation, 4) Discipline, and 5) Streamlined and Effective Management
Department Of Defense Instruction	DODI	Establishes a simplified and flexible management framework for translating mission needs and technology opportunities, based on approved mission needs and requirements, into stable, affordable, and well managed acquisition programs. Specifically authorizes the Program Manager (PM) and the Milestone Decision Authority (MDA) to use discretion and business judgment to structure a tailored, responsive and innovative program.
Department of Navy	DoN	One of several branches of the Depart of Defense.
Destroyer, Guided Missile	DDG	Guided missile destroyers are fast warships that help safeguard larger ships by operating in support of carrier battle groups, surface action groups, amphibious groups and replenishment groups. Guided missile destroyers are multi-mission surface combatants, which are also able to provide naval gunfire support. (Navy DDG Website)
Destroyer, Guided Missile, Next	DDG (1000)	Developed under the DD (X) destroyer program, DDG-1000 Zumwalt is the lead ship in a class of next-

Generation		generation, multi-mission surface combatants tailored for land attack and littoral dominance, with capabilities designed to defeat current and projected threats as well as improve battle force defense.
Detect Control Engage	DCE	*Detect, Control, and Engage*. A precision engagement starts with gathering and processing intelligence, surveillance, reconnaissance, and targeting information *(Detect)*. The results, along with the commander's intent, then pass to decision makers who determine how and with what weapons an engagement will be conducted *(Control).* These decisions are then passed to the detailed weapons planners and finally to the executers — those who operate the ships, submarines, or aircraft that launch the weapons *(Engage)*. Weapons systems may provide kinetic effects, such as the Tomahawk conventional weapon or non-kinetic effects such as the support of enemy surface-to-air missiles by the EA-6B aircraft. (JPL)
Detect to Engage	DTE	The process of target detection, resolution or localization, classification, tracking, weapon selection, and ultimately neutralization. (Integrated Publishing)
Developmental Test	DT	DT is the verification and validation of the systems engineering process and must provide confidence that the system design solution is on track to satisfy the desired capabilities. (DAU)
Developmental Test and Evaluation	DT&E	A well planned and executed DT&E program supports the acquisition strategy and the systems engineering process, providing the information necessary for informed decision making throughout the development process and at each acquisition milestone. DT is the verification and validation of the systems engineering process and must provide confidence that the system design solution is on track to satisfy the desired capabilities. Rigorous component and sub-system developmental test and evaluation (DT&E) ensures that performance capability and reliability are designed into the system early. DT&E then should increase to robust, system-level and system-of-systems level testing and evaluation, to ensure that the system has matured to a point where it can meet IOT&E and operational employment requirements. (DAU)
Digital Data Storage	DDS	Digital Data Storage (DDS) is a format for storing and backing up computer data on tape that evolved from the Digital Audio Tape (DAT) technology. (Whatis.com)
Director of Operations Test and Evaluation	DOT&E	Executive Level IV - Presidential Appointment with Senate Confirmation. (Prunes online)
Discrete Event and System Timing Model		In discrete-event simulation, the operation of a system is represented as a chronological sequence of events. Each event occurs at an instant in time and marks a change of

		state in the system. (Wikipedia)
Domain analysis	DA	Domain analysis is "the process of identifying, collecting, organizing, and representing the relevant information in a domain, based upon the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory, and emerging technology within a domain. Def from CMU/SEI-90-TR-21
Dynamic Object Oriented Requirements System	DOORS	DOORS is a software tool for managing complex projects. It is used to store multiple Documents and Tables containing project requirements and other information. You can readily import Word and Excel documents and Access Tables into DOORS as it is both document-centered as well as spreadsheet-like.
Electro Optic	EO	The electro-optic effect is a change in the index of refraction for certain crystals as a function of applied voltage. The index change is dependent on the direction and polarization of the incident beam. (John Simcik, Electro-option course module)
Electronic Warfare	EW	
Element Relationship	ER	You can view the traceability relationships of model elements in all open models. You can assess the impact of change by viewing how model elements relate to implementations and specifications. (IBM Websphere)
Enhanced Data Flow Diagram	EDFD	A significant modeling technique for analyzing and constructing information processes.
Enhanced Data Flow Diagram Level 0	EDFD0	A significant modeling technique for analyzing and constructing information processes. DFD literally means an illustration that explains the course or movement of information in a process. DFD illustrates this flow of information in a process based on the inputs and outputs. A DFD can be referred to as a Process Model. (Edraw Soft)
Enhanced Functional Flow Block Diagram	EFFBD	Functional Flow Block Diagrams provide a hierarchical decomposition of the system's functions and show a control structure that dictates the order in which the functions can be executed at each level of decomposition. The enhanced FFBD enables you to see how the inputs and output affect the functional sequencing. The Engineering Design of Systems, Dennis Buede, DoDAF, Steven Dam
Enterprise Data Bus	EDB	Integrate old and new, service-oriented architecture (SOA) to an infrastructure that can connect any IT resource, whatever its technology or wherever it is deployed. To be flexible and meet the needs of an infrastructure that can easily combine and re-assemble services to meet changing

		requirements without disruption and to be dependable. (Progress Sonic)
Event-Based		Event-based systems are systems in which producers deliver events, and in which messaging middleware delivers events to consumers based upon their previously specified interest. (DEBs)
Evolved SeaSparrow Missile	ESSM	The Evolved Sea Sparrow Missile (ESSM) is a short-range missile intended to provide self-protection for surface ships. It will provide each ship with the capability to engage a variety of anti-ship cruise missiles (ASCMs) and aircraft to support self-defense. (Global Security.com)
External Communications	EXCOMM	External communication covers how a provider interacts with those outside their own organization. This may be with the public, employers, community organizations, local authorities, job centers, careers offices, funding bodies, specialist agencies and other training providers. (QIA.org)
Extended Memory Interconnect	XMI	The name given to memory in an Intel PC above 1MB. Starting with the Intel 286, it was used directly by Windows and OS/2 as well as DOS applications that ran with DOS extenders. It was also used under DOS for RAM disks and disk caches. Contrast with expanded memory (EMS), which was specialized memory above 1MB. Today, the term is rarely heard, because the 1MB barrier was broken long ago.
eXtensible Markup Language	XML	The XML Metadata Interchange (XMI) is an OMG standard for exchanging metadata information via Extensible Markup Language (XML). It can be used for any metadata whose metamodel can be expressed in Meta-Object Facility (MOF). The most common use of XMI is as an interchange format for UML models, although it can also be used for serialization of models of other languages (metamodels).
Failure Modes and Effects, Analysis	FMEA	Procedure by which each potential failure mode is analyzed to determine its effects on the system and then classified according to its severity. (DAU)
Failure Modes and Effects, Critical Analysis	FMECA	An extension of Failure Mode and Effects Analysis (FMEA). In addition to the basic FMEA, it includes a criticality analysis, which is used to chart the probability of failure modes against the severity of their consequences. The result highlights failure modes with relatively high probability and severity of consequences, allowing remedial effort to be directed where it will produce the greatest value. (Wikipedia)
Family-of-Systems	FoS	A set or arrangement of independent systems that can be arranged or interconnected in various ways to provide

		different capabilities. The mix of systems can be tailored to provide desired capabilities, dependent on the situation
Federal Emergency Management Agency	FEMA	On March 1, 2003, the Federal Emergency Management Agency (FEMA) became part of the U.S. Department of Homeland Security (DHS). The primary mission of the Federal Emergency Management Agency is to reduce the loss of life and property and protect the Nation from all hazards, including natural disasters, acts of terrorism, and other man-made disasters, by leading and supporting the Nation in a risk-based, comprehensive emergency management system of preparedness, protection, response, recovery, and mitigation. (FEMA website)
Finite State Machine	FSM	A model of computation consisting of a set of states, a start state, an input alphabet, and a transition function that maps input symbols and current states to a next state. Computation begins in the start state with an input string. It changes to new states depending on the transition function. There are many variants, for instance, machines having actions (outputs) associated with transitions (Mealy machine) or states (Moore machine), multiple start states, transitions conditioned on no input symbol (a null) or more than one transition for a given symbol and state (nondeterministic finite state machine), one or more states designated as accepting states (recognizer), etc. (National Institute of Standards and Technology NIST)
Firm track time	Tft	
Formalism		Holds that mathematical statements may be thought of as statements about the consequences of certain string manipulation rules. (Wikipedia)
Functional Analysis		A part of the design process that addresses the activities that a system, software, or organization must perform to achieve its desired outputs, that is, the transformations necessary to turn available inputs into the desired outputs.
Functional Architecture		(a) Logical architecture that defines what the system must do, a decomposition of the systems' top-level function. This very limited definition of the functional architecture is the most common and is represented as a directed tree. (b) Logical model that captures the transformation of inputs into outputs using control information. This definition adds the flow of inputs and outputs throughout the functional decomposition. (c) Logical model of a functional decomposition plus the flow of inputs and outputs, to which input/output requirements have been traced to specific functions and items (inputs, outputs and controls). Buede, pg.434
Functional Area Analysis	FAA	Identifies the mission area or mission problem to be assessed, the concepts to be examined, the timeframe in

		which the problem is being assessed, and the scope of the assessment, and describes the relevant objectives and concept of operations (ConOps) or concepts and the relevant effects to be generated. (CJCSI 3170.01F)
Functional Flow Block Diagram	FFBD	FFBD is a multi-tier, time-sequenced, step-by-step flow diagram of a system's functional flow.
Functional Needs Analysis	FNA	Assesses the ability of the current and programmed warfighting systems to deliver the capabilities the Functional Area Analysis (FAA) identified under the full range of operating conditions and to the designated measures of effectiveness. The FNA produces a list of capability gaps that require solutions and indicates the time frame in which those solutions are needed. It may also identify redundancies in capabilities that reflect inefficiencies. (CJCSI 3170.01F)
Functional Solutions Analysis	FSA	Operationally based assessment of all potential Doctrine, Organization, Training, Materiel, Leadership, Personnel, and Facilities (DOTMLPF) approaches to solving (or mitigating) one or more of the capability gaps (needs) previously identified.
Government Accountability Office	GAO	Formerly the General Accounting Office. An agency of the Legislative Branch, responsible solely to the Congress, which functions to audit all negotiated government office contracts and investigate all matters relating to the receipt, disbursement, and application of public funds. Determines whether public funds are expended in accordance with appropriations
Government Furnished Equipment	GFE	Property in the possession of or acquired directly by the government, and subsequently delivered to or otherwise made available to the contractor. (DAU)
Government Furnished Information	GFI	Limitations and provisions for GFI technical data that are marked with restrictive legends. (DAU)
Government Purpose Rights	GPR	In Defense Department acquisitions, the resulting contract can permit delivery of technical data and computer software using a "middle way," known as Government Purpose Rights, which is an Intellectual Property licensing system that is available to DOD acquisitions. Government Purpose Rights ("GPR") lie somewhere between the broad Unlimited Rights license rights allowing unrestricted Government release of information and the more restrictive Limited or Restricted Rights licensing rights that forbid most releases outside the Government.
Government-Off-The-Shelf	GOTS	Government off-the-shelf (GOTS) is a term for software and hardware products that are typically developed by the technical staff of the government agency for which it is created. (Wikipedia)

Graphical User Interface	GUI	A GUI offers graphical icons, and visual indicators, as opposed to text-based interfaces, typed command labels or text navigation to fully represent the information and actions available to a user. (Wikipedia)
Guided Missile Cruiser	CG	Guided missile cruisers are large combat vessels with multiple target response capability. They perform primarily in a battle force role and are multi-mission surface combatants capable of supporting carrier battle groups, amphibious forces, or of operating independently and as flagships of surface action groups. Due to their extensive combat capability, these ships have been designated as Battle Force Capable (BFC) units. (Navysite.com)
Hard-kill		Defense by use of missiles
Hardware	HW	A general term that refers to the physical artifacts of a technology. It may also mean the physical components of a computer system, in the form of computer hardware. (Wikipedia0
Hardware in the Loop	HWIL	Hardware-In-the-Loop is a form of real-time simulation. Hardware-In-the-Loop differs from pure real-time simulation by the addition of a "real" component in the loop. This component may be an electronic control unit (ECU for automotive, FADEC for Aerospace) or a real engine.
Hatley-Hruschka-Pirbhai (H-H-P) Methods:		This methodology uses Data Flow Diagrams, Control Flow Diagrams, Object-Relationship Diagrams and Information Architecture diagrams to define software (technical) requirements, all of which are rigorously traceable to functional requirements. (Bill Meacham, PMP)
High Value Units	HVU	Equipment of considerable cost, a ship.
Human Systems Integration	HSI	Focus attention on the human part of the system and by integrating and inserting manpower, personnel, training, human factors, safety, occupational health, habitability, and personnel survivability considerations into the Defense acquisition process.
Hyper Text Markup Language	HTML	HTML is a computer language devised to allow website creation. These websites can then be viewed by anyone else connected to the Internet. It is relatively easy to learn, with the basics being accessible to most people in one sitting; and quite powerful in what it allows you to create. It is constantly undergoing revision and evolution to meet the demands and requirements of the growing Internet audience under the direction of the W3C, the organization charged with designing and maintaining the language. (HTMLsource)

Identification Friend or Foe	IFF	Radar and identification friend or foe (IFF) equipment constitutes the forward elements of complex systems that have appeared throughout the world. Examples include the semiautomatic ground environment (SAGE), augmented by a mobile backup intercept control system called BUIC in the United States, NATO air defense ground environment (NADGE)...(Encyclopedia Britannica)
In Process Review	IPR	Interim Program or Progress Review
In Service Engineering Agent	ISEA	The activity, delegated functions by, and in support of system manager for the overall engineering, test, maintenance, technical analysis and logistics requirement incident to a specific operational equipment. A government organization providing field changes (not ECPs) after procurement and initial system fielding have occurred. Traditionally, ISEAs connote organic support, funded by a lead service in joint programs (However, this type of support may be contracted out as well.)
Information Technology	IT	Any equipment or interconnected system or subsystem of equipment, that is used in the automatic acquisition, storage, manipulation, management, movement, control, display, switching, interchange, transmission, or reception of data or information by the executive agency. IT includes computers, ancillary equipment, software, firmware and similar procedures, services (including support services), and related resources, including National Security Systems (NSSs). It does not include any equipment that is acquired by a federal contractor incidental to a federal contract. (CJCSI 6212.01C)
Infrared	IR	Infrared (IR) radiation is electromagnetic radiation whose wavelength is longer than that of visible light (400-700 nm), but shorter than that of terahertz radiation (3-300 μm) and microwaves (~30,000 μm). Infrared radiation spans roughly three orders of magnitude (750 nm and 1000 μm). (Wikipedia)
Initial Capabilities Document	ICD	Documents the need for a materiel approach, or an approach that is a combination of materiel and non-materiel, to satisfy specific capability gap(s). The ICD defines the gap in terms of the functional area; the relevant range of military operations; desired effects; time and Doctrine, Organization, Training, Materiel, Leadership and education, Personnel, and Facilities (DOTMLPF); and policy implications and constraints. The outcome of an ICD could be one or more DOTMLPF Change Recommendations (DCRs) or Capability Development Documents. (CJCSI 3170.01F)
Initial Capabilities Document	ICD	Defines the gap in terms of the functional area; the relevant range of military operations; desired effects; time

		and Doctrine, Organization, Training, Materiel, Leadership and education, Personnel, and Facilities (DOTMLPF); and policy implications and constraints. The outcome of an ICD could be one or more DOTMLPF Change Recommendations (DCRs) or Capability Development Documents. (CJCSI 3170.01F)
Initial Launch	TL	First launch of the weapon system
Initial Operational Test and Evaluation	IOTE	Dedicated Operational Test and Evaluation (OT&E) conducted on production, or production representative articles, to determine whether systems are operationally effective and suitable, and which supports the decision to proceed Beyond Low Rate Initial Production (BLRIP).
Integrated Definition for Function Modeling	IDFM	IDEFO models the decisions, actions and activities of a system in order to communicate the functional perspective of the system.
Integrated Logistics Support	ILS	The ILS process begins during mission analysis and continues throughout the lifecycle of a product or service. It progresses from analysis and planning during mission and investment analysis to acquisition during solution implementation to steady-state operations during in-service management. The process then iterates during the in-service management as ILS planning is adjusted to ensure services continue to be supported in a cost-effective manner. (Federal Aviation Administration)
Integrated Logistics Support Plan	ILSP	Master logistics planning document that describes necessary logistics, assigns responsibility for those activities and establishes schedule for completion.
Integrated Product and Process Development	IPPD	IPPD is the DoD management technique that simultaneously integrates all essential acquisition activities through the use of multidisciplinary teams to optimize design, manufacturing, and supportability processes. One of the key IPPD tenets is multidisciplinary teamwork through Integrated Product Teams.
Integrated Product Team	IPT	IPTs are an integral part of the Defense acquisition oversight and review process. For Acquisition Category ID and IAM programs, there are generally two levels of IPT: the Overarching Integrated Product Team and the Working-level Integrated Product Team(s). Each program should have an OIPT and at least one WIPT. WIPTs should focus on a particular topic such as cost/performance, test, or contracting. An Integrating Integrated Product Team (IIPT), which is itself a WIPT, should coordinate WIPT efforts and cover all topics not otherwise assigned to another IPT. IPT participation is the primary way for any organization to participate in the acquisition program. DAU

Integrated Warfare Systems	IWS	Systems that deliver Enterprise solutions for Naval warfare systems that operate seamlessly and effectively within the Fleet and Joint Force.
Integration Definition for Function Modeling	IDEF	IDEF methods are used to create graphical representations of various systems, analyze the model, create a model of a desired version of the system, and to aid in the transition from one to the other. IDEF is sometimes used along with gap analysis. (Tech Target)
Interactive Electronic Technical Manuals	IETM	IETM is a technical manual that is prepared (authored) in digital form on a suitable medium, by means of an automated authoring system. (DAU)
Internal Block Diagram		The internal block diagram describes the internal structure of a system in terms of its parts, ports, and connectors. The package diagram is used to organize the model
International Council on Systems Engineering	INCOSE	INCOSE is a not-for-profit membership organization founded in 1990. The mission is to advance the state of the art and practice of systems engineering in industry, academia, and government by promoting interdisciplinary, scalable approaches to produce technologically appropriate solutions that meet societal needs.
International Organization for Standardization	ISO	ISO is a non-governmental organization that forms a bridge between the public and private sectors. On the one hand, many of its member institutes are part of the governmental structure of their countries, or are mandated by their government. On the other hand, other members have their roots uniquely in the private sector, having been set up by national partnerships of industry associations. (ISO website)
Interoperability		The ability of systems, units, or forces to provide data, information, materiel, and services to and accept the same from other systems, units, or forces and to use the data, information, materiel, and services so exchanged to enable them to operate effectively together. National Security System (NSS) and Information Technology System (ITS) interoperability includes both the technical exchange of information and the end-to-end operational effectiveness of that exchanged information as required for mission accomplishment. (CJCSI 3170.01E).
Interoperable Joint Warfighting		Ensuring that components of all services can operate together on the same mission.
Interoperable War Fighting Capabilities		Ensuring that all systems of all services can operate together on the same mission.
Joint Capabilities Integration Development System	JCIDS	The Joint Capabilities Integration Development System, or JCIDS, is the formal United States Department of Defense (DoD) procedure, which defines acquisition

		requirements and evaluation criteria for future defense programs. JCIDS was created to replace the previous service-specific requirements generation system, which allegedly created redundancies in capabilities and failed to meet the combined needs of all US military services. In order to correct these problems, JCIDS is intended to guide the development of requirements for future acquisition systems to reflect the needs of all four services (Army, Navy, Marines, and Air Force) by focusing the requirements generation process on needed *capabilities* as requested or defined by one of the US combatant commanders. In the JCIDS process, regional and functional combatant commanders give feedback early in the development process to ensure that their requirements are met. (Wikipedia)
Keep out Range	Rout	
Key Performance Parameter	KPP	Those attributes or characteristics of a system that are considered critical or essential to the development of an effective military capability and those attributes that make a significant contribution to the key characteristics as defined in the Joint Operations Concept. KPPs are validated by the Joint Requirements Oversight Council (JROC) for JROC Interest documents, and by the DoD Component for Joint Integration or Independent documents. The Capability Development Document (CDD) and the Capability Production Document (CPD) KPPs are included verbatim in the Acquisition Program Baseline (APB). (CJCSI 3170.01E)
Kill Assessment	KA	Kill assessment is the real-time, remote determination of missile engagement lethality.
Kill evaluation time	Te	Time relevant to missile engagement lethality. (lb)
Landing Platform Dock	LPD	An amphibious warfare ship, a warship that embarks, transports, and lands elements of a landing force for expeditionary warfare missions. (Wikipedia)
Launch Rate	L	Rate between missile launch. (lb)
Layered architecture		An architecture in which data moves from one defined level of processing to another. (PC Magazine)
Layered Systems		Layered Systems use layers to separate different units of functionality. Each layer only communicates with the layer above and the layer below. Each layer *uses* the layer below to perform its function. Communication happens through predefined, fixed interfaces. (Garfixia Software Architects)
Level Of Repair Analysis	LORA	LORA is a rigorous empirical method for determining if and at what level of maintenance an item will be repaired. A trade study conducted by a contractor as part of the

		system/equipment engineering analysis process. A basis on which to evolve an optimum approach to repair recommendations concurrent with the design and development process. Also referred to as Repair Level Analysis or Level of Repair Analysis (LOR/A).
Life Cycle Cost	LCC	The total cost to the government of acquisition and ownership of that system over its useful life. It includes the cost of development, acquisition, operations, and support (to include manpower), and where applicable, disposal. For defense systems, LCC is also called Total Ownership Cost (TOC).
Life Cycle Cost Estimate	LCC	Estimate of the cost of a program from cradle to grave
Life Cycle Support Plan	LCCE	The total phases through which an item passes from the time it is initially developed until the time it is either consumed in use or disposed of as being excess to all known materiel requirements. The plan covers the entire period.
Life-cycle Affordability		Lifecycle cost consists of research and development costs, investment costs, operating and support costs, and disposal costs over the entire Lifecycle. These costs include not only the direct costs of the acquisition program, but also include indirect costs that would be logically attributed to the program.
Limited Area Defense	LAD	Range limitation of weapons systems.
Littoral Combat Ship	LCS	Littoral Combat Ships are the first examples of the U.S. Navy's next-generation surface combatants: the Freedom Class and the Independence Class. Intended as a relatively small surface vessel for operations in the littoral zone (close to shore), the LCS designs are slightly smaller than the Navy's guided missile frigates, and have been compared to the corvette of international usage. (Wikipedia)
Live Fire Test & Evaluation	LFT&E	A test process to evaluate the vulnerability and /or lethality aspects of a conventional weapon or conventional weapon system. LFT&E is a statutory requirement (Title 10 U.S.C. § 2366) for covered systems, major munitions programs, missile programs, or product improvements to a covered systems, major munitions programs, or missile programs before they can proceed Beyond Low Rate Initial Production (BLRIP). By law, a covered system is any vehicle, weapon platform, or conventional weapon system that includes features designed to provide some degree of protection to users in combat and that is an Acquisition Category (ACAT) I or ACAT II program. (Note: The term covered system can also be taken to mean any system or program covered by Title 10 U.S.C. § 2366,

		including major munitions and missile programs.) (DAU)
Local Area Network	LAN	A local area network (LAN) is a group of computers and associated devices that share a common communications line or wireless link. Typically, connected devices share the resources of a single processor or server within a small geographic area (for example, within an office building). Usually, the server has applications and data storage that are shared in common by multiple computer users. A local area network may serve as few as two or three users (for example, in a home network) or as many as thousands of users (for example, in an FDDI network). (Search.network.com)
Local Area Network	LAN	A local area network (LAN) is a computer network covering a small physical area, like a home, office, or small group of buildings, such as a school, or an airport. The defining characteristics of LANs, in contrast to wide-area networks (WANs), include their usually higher data-transfer rates, smaller geographic range, and lack of a need for leased telecommunication lines.
Manpower Requirements		The total supply of persons available and fitted for service. Indexed by requirements including jobs lists, slots, or billets characterized by descriptions of the required people to fill them. (DAU)
Masters of Science in Systems Engineering	MSSE	A rigorous study and an Advanced degree for Systems Engineering offered by the Naval Post Graduate School
Masters of Science in Systems Engineering Management	MSSEM	A rigorous study and an Advanced degree for Systems Engineering Management that incorporates the Systems Engineering and Supportability aspects of a System offered by the Naval Post Graduate School
Mathematic Modeling Language	MathML	A MML model is a set of nested components. There must be one top-level component, in this case of type "math". This document will consider only top-level components of type math. Content of the math component is delimited by curly braces.
Mathematical Modeling		A mathematical representation of a process, device, or concept by means of a number of variables which are defined to represent the inputs, outputs, and internal states of the device or process, and a set of equations and inequalities describing the interaction of these variables. A mathematical theory or system together with its axioms
Maximum Range	r_{MAX} or Rmax	Trajectory is the path of a moving object that it follows through space. The object might be a projectile or a satellite, for example. It thus includes the meaning of orbit - the path of a planet, an asteroid or a comet as it travels around a central mass. A trajectory can be described mathematically either by the geometry of the path, or as

		the position of the object over time
Mean Logistics Delay Time	MLDT	Indicator of the average time a system is awaiting maintenance and generally includes time for 1) Locating parts and tools, 2) Locating, setting up or calibrating test equipment, 3) Dispatching personnel 4) Reviewing technical manuals, 5) Complying with supply procedures, and 6) Awaiting transportation. The MLDT is largely dependent upon the logistics support structure and environment. (DAU)
Mean Time Between Failure	MTBF	For a particular interval, the total functional life of a population of an item divided by the total number of failures (requiring corrective maintenance actions) within the population. The definition holds for time, rounds, miles, events, or other measures of life unit. A basic technical measure of reliability recommended for use in the research and development contractual specification environment, where "time" and "failure" must be carefully defined for contractual compliance purposes. (DAU)
Mean Time To Repair	MTTR	The total elapsed time (clock hours) for corrective maintenance divided by the total number of corrective maintenance actions during a given period of time. A basic technical measure of maintainability recommended for use in the research and development contractual specification environment, where "time" and "repair" must be carefully defined for contractual compliance purposes. (DAU)
Measure Of Effectiveness	MOE	Measure designed to correspond to accomplishment of mission objectives and achievement of desired results. (CJCSI 3170.01E) MOEs may be further decomposed into Measures of Performance and Measures of Suitability. See operational effectiveness, Measure of Performance, operational suitability, and Measure of Suitability.
Measure Of Performance	MOP	Measure of a system□s performance expressed as speed, payload, range, time on station, frequency, or other distinctly quantifiable performance features. Several MOPs and/or Measures of Suitability may be related to the achievement of a particular Measure of Effectiveness (MOE). See Measure of Suitability, operational suitability, and Measure of Effectiveness.
Measure of Raid Annihilation	M_{RA}	Probability of Raid Annihilation, P_{RA}, is the Navy's Measure of a single ship with its combat systems to detect, control, engage and defeat a specified raid of threats within a specified level of probability in an operational environment. Threat performance and combat system performance both can vary significantly with natural environment conditions so the P_{RA} federation incorporates these effects.

Measures of Effectiveness	MOE	Measure designed to correspond to accomplishment of mission objectives and achievement of desired results. (CJCSI 3170.01E) MOEs may be further decomposed into Measures of Performance and Measures of Suitability. See operational effectiveness, Measure of Performance, operational suitability, and Measure of Suitability.
Measures of Performance	MOP	The DoD Performance Assessment Guide contains 3 stand-alone modules that work together to help you: (1) benchmark your organization's quality climate and quality management strengths and weaknesses (The Quality and Productivity Self-Assessment Guide); (2) track your organization's performance over time (The Guide for Developing Performance Measures); and, (3) track what your organization's customers think about the service they receive (The Guide For Measuring Customer Satisfaction. (DoD Performance Assessment Guide)
Meta-Data		Is "data about other data", of any sort in any media. An item of metadata may describe an individual datum, or content item, or a collection of data including multiple content items and hierarchical levels, for example a database schema. In data processing, metadata is definitional data that provides information about or documentation of other data managed within an application or environment. The term should be used with caution as all data is about something, and is therefore metadata. (Wikipedia)
Metadata Repository		A metadata repository is a database of data about data (metadata). The purpose of the metadata repository is to provide a consistent and reliable means of access to data. The repository itself may be stored in a physical location or may be a virtual database, in which metadata is drawn from separate sources. Metadata may include information about how to access specific data, or more detail about it, among a myriad of possibilities.
Microsoft Visio		Microsoft Visio is diagramming software for Microsoft Windows. It uses vector graphics to create diverse diagrams. It is currently available in two editions, Standard and Professional.
Middle-out		The objective is to avoid the problems of top-down and bottom-up approaches, by designing a very high-level language specific to the application domain. Domain knowledge is captured in the design of this language, which retains a strong formal basis. (IEEE Explore)
Military Standard	MIL-STD	A United States Defense Standard, often called a military standard, "MIL-STD", "MIL-SPEC", or (informally) "MilSpecs", is used to help achieve standardization objectives by the U.S. Department of Defense.

		Standardization is beneficial in achieving interoperability, ensuring products meet certain requirements, commonality, reliability, total cost of ownership, compatibility with logistics systems, and similar defense-related objectives.
Minimum Range	Rmin	
Model Based Systems Engineering (MBSE):		The discipline of systems engineering in a "model-based" or "model-driven" context. (Estefan year)
Model-based systems engineering	MBSE	"Model-based systems engineering is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing through out development and later life cycle phases". INCOSE, Systems Engineering Vision 2020, Version 2.03, TP-2004-004-02 Sept 2007
Modeling and Simulation	M&S	Modeling and Simulation is a discipline for developing a level of understanding of the interaction of the parts of a system, and of the system as a whole.
Modeling Languages		Artificial language that can be used to express information or knowledge or systems in a structure that is defined by a consistent set of rules. The rules are used for interpretation of the meaning of components in the structure. (Wikipedia)
Modular Command Evaluation Structure	MCES	Modular Command and Control Evaluation Structure (MCES) provides the framework for management of the process
Modular design		Characterized by the following: Functionally partitioned into discrete scalable, reusable modules consisting of isolated, self-contained functional elements, Rigorous use of disciplined definition of modular interfaces, to include object oriented descriptions of module functionality, Designed for ease of change to achieve technology transparency and, to the extent possible, makes use of commonly used industry standards for key interfaces. (OS Guide Appendix C)
Modular Open Systems Architecture	MOSA	Department of Defense implementation of "open systems". DAU
Nautical Mile	nm or NM or nmi	The nautical mile was based on the circumference of the earth at the equator. Since the earth is 360 degrees of longitude around, and degrees are broken into 60 so-called "minutes", that means there are 360 * 60 = 21,600 "minutes" of longitude around the earth. This was taken as the basis for the nautical mile; thus, by definition, 1 minute of longitude at the equator is equal to 1 nautical mile. So the earth is ideally, by definition, 21,600 nautical miles (and 21,600 "minutes" of longitude) in

		circumference at the equator. If anyone ever asks you how far is it around the earth, you can quickly do the math in your head (360 degrees * 60 minutes per degree) and answer "about 21,600 nautical miles!"
Naval Open Architecture	NOA	The Navy OA is a systems design approach supported by verifiable governmental testing platforms, such as the OACE, that seeks to implement open specifications for interfaces, services and supporting formats. It enables software components to work across a range of systems and interoperate with other software components on local and remote systems
Naval Postgraduate School	NPS	Advanced studies focused on DoD
Naval Sea Systems Command	NAVSEA	Surface Command of Fleet for DoD
Naval Surface Warfare Center	NSWC	Field activity that is part of Naval Sea Systems Command
Naval Surface Warfare Center Port Hueneme Division	NSWC PHD	Field activity that is part of Naval Sea Systems Command Port Hueneme.
Navigation	NAV	Navigation is the process of reading, and controlling the movement of a craft or vehicle from one place to another.(Wikipedia)
Navy Marine Corps Intranet	NMCI	United States Department of the Navy outsourcing program, in which an outside contractor provides a vast majority of information technology services for the entire Department, including the United States Navy and Marine Corps. (Wikipedia)
Navy Modeling and Simulation Office	NMSO	The Navy Modeling and Simulation Office (NMSO) is now operating as the “action arm” of the Navy Modeling & Simulation Governance Board. (Wikipedia)
Navy Network Warfare Command	NETWAR	Navy's type commander for Information Operations, Intelligence, Networks and Space. NETWARCOM is charged with operating a secure and interoperable naval network that enables effects-based operations and innovation.
Net-Centric Architectures	NCA	Enables information sharing by connecting people and systems that have information with people/systems that need information. It includes situational awareness, self-synchronizing ops, information pull, collaboration, shared data, bandwidth on demand, diverse routing, Enterprise services
N-Squared	N2	The N2 Chart, also referred to as N2 Diagram, N-Squared Diagram or N Squared Chart, is a diagram in the shape of a matrix, representing functional or physical interfaces

		between system elements. It is used to systematically identify, define, tabulate, design, and analyze functional and physical interfaces. It applies to system interfaces and hardware and/or software interfaces. (Wikipedia)
NULKA		An Australian designed and developed active missile decoy built by Australian/American collaboration.
Number of attack	Na	Part of Threat Profile
Number of cells	Nc	Part of Threat Profile
Object Constraint Language	OCL	The Object Constraint Language (OCL) is a declarative language for describing rules that apply to Unified Modeling Language (UML) models developed at IBM and now part of the UML standard. Initially, OCL was only a formal specification language extension to UML. OCL may now be used with any Meta-Object Facility (MOF) Object Management Group (OMG) meta-model, including UML. The Object Constraint Language is a precise text language that provides constraint and object query expressions on any MOF model or meta-model that cannot otherwise be expressed by diagrammatic notation. OCL is a key component of the new OMG standard recommendation for transforming models, the Queries/Views/Transformations (QVT) specification. (Wikipedia)
Object Management Group Systems Modeling Language	OMG SysML	The OMG systems Modeling Language (OMG SysML™) is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities. In particular, the language provides graphical representations with a semantic foundation for modeling system requirements, behavior, structure, and parametrics, which is used to integrate with other engineering analysis models. SysML represents a subset of UML 2 with extensions needed to satisfy the requirements of the UML™ for Systems Engineering RFP as indicated in Figure 1. SysML leverages the OMG XML Metadata Interchange (XMI®) to exchange modeling data between tools, and is also intended to be compatible with the evolving ISO 10303-233 systems engineering data interchange standard. (OMGSsyML Official Website)
Object Oriented	OO	Programming techniques may include features such as information hiding, data abstraction, encapsulation, modularity, polymorphism, and inheritance. (Wikipedia)
Office of Management and Budget	OMB	OMB's predominant mission is to assist the President in overseeing the preparation of the federal budget and to supervise its administration in Executive Branch agencies. In helping to formulate the President's spending plans,

		OMB evaluates the effectiveness of agency programs, policies, and procedures, assesses competing funding demands among agencies, and sets funding priorities. OMB ensures that agency reports, rules, testimony, and proposed legislation are consistent with the President's Budget and with Administration policies. (OMB Website)
Office of the Chief of Naval Operations	OPNAV	The Chief of Naval Operations (CNO) is the senior military officer in the Navy. The CNO is a four-star admiral and is responsible to the Secretary of the Navy for the command, utilization of resources and operating efficiency of the operating forces of the Navy and of the Navy shore activities assigned by the Secretary.
Office of the Secretary of Defense	OSD	The Office of the Secretary of Defense (OSD) is the principal staff element of the Secretary of Defense in the exercise of policy development, planning, resource management, fiscal, and program evaluation responsibilities. OSD includes the immediate offices of the Secretary and Deputy Secretary of Defense, Under Secretaries of Defense, Director of Defense Research and Engineering, Assistant Secretaries of Defense, General Counsel, Director of Operational Test and Evaluation, Assistants to the Secretary of Defense, Director of Administration and Management, and such other staff offices as the Secretary establishes to assist in carrying out assigned responsibilities. (US Department of Defense)
Office of the Under Secretary of Defense, Acquisition, Technology & Logistics	OSD AT&L	Innovate and collaborate to engage the war fighting, requirements, and resourcing communities on behalf of the taxpayer
OMG SysML		After a series of competing SysML specification proposals, a SysML Merge Team was proposed to the Object Management Group (OMG) in April 2006. This proposal was voted upon and adopted by the OMG in July 2006 as OMG SysML, to differentiate it from the original open source specification from which it was derived. Because OMG SysML is derived from open source SysML, it also includes an open source license for distribution and use.
Open Architecture	OA	Open Architecture is based on the principle that the government should be allowed to share the data it purchases to all government programs, any qualified vendors, and all service branches.
Open Architecture Computing Environment	OACE	Guidance is provided for overall computing system architecture, system-wide design principles, computing equipment and support software infrastructure technologies, standards, application functional partitioning

		principles, computer program characteristics, and development strategy. (DTIC Online)
Open Architecture Enterprise Team	OAET	The Enterprise Team shall define an overarching OA acquisition strategy and develop guidance that addresses incentives, intellectual property issues, contracting strategies, and funding alternatives. The acquisition strategy will be utilized in future OA procurements tailored as necessary to incorporate domain specific requirements. (DTIC Online)
Open Source Software		An approach to design, development, and distribution offering practical accessibility to a product's source (goods and knowledge). Some consider open source as one of various possible design approaches, while others consider it a critical strategic element of their operations.
Open Systems Joint Task Force	OSJTF	The Under Secretary of Defense for Acquisition, Technology and Logistics chartered the Open Systems Joint Task Force to champion the establishment of a Modular Open Systems Approach (MOSA) and ensure implementation by all DoD acquisition programs. Specifically the OSJTF will: Make MOSA an integral part of the acquisition process, Provide expert assistance in applying MOSA, Ensure application of MOSA by all acquisition programs, and Collaborate with industry to ensure a viable open standards base.
Operational Area	OP Area	An overarching term encompassing more descriptive terms for geographic areas in which military operations are conducted. Operational areas include, but are not limited to, such descriptors as area of responsibility, theater of war, theater of operations, joint operations area, amphibious objective area, joint special operations area, and area of operations. See also amphibious objective area; area of operations; area of responsibility; joint operations area; joint special operations area; theater of operations; theater of war. (Dictionary of Military and Associated Terms)
Operational Availability	Ao	The degree (expressed as a decimal between 0 and 1, or the percentage equivalent) to which one can expect a piece of equipment or weapon system to work properly when it is required, that is, the percent of time the equipment or weapon system is available for use. AO represents system "uptime" and considers the effect of reliability, maintainability, and mean logistics delay time. AO may be calculated by dividing Mean Time Between Maintenance by the sum of the Mean Time Between Maintenance, Mean Maintenance Time, and Mean Logistics Delay Time (MLDT), that is, AO = MTBM / (MTBM + MMT + MLDT). It is the quantitative link between readiness objectives and supportability. See Mean Time Between

		Maintenance, Mean Maintenance Time, and Mean Logistics Delay Time.
Operational Test and Evaluation	OT&E	The field test, under realistic conditions, of any item (or key component) of weapons, equipment, or munitions for the purpose of determining the effectiveness and suitability of the weapons, equipment, or munitions for use in combat by typical military users; and the evaluation of the results of such tests.
Operational View	OV	Architecture view that describes the joint capabilities that the user seeks and how to employ them. The OVs also identify operational nodes, the critical information needed to support the piece of the process associated with the nodes, and the organizational relationships. (CJCSM 3170.01B)
Opportunistic Reuse	OR	Developed software *might* be reusable in future systems
Optimization Process	OP	The discipline of adjusting a process so as to optimize some specified set of parameters without violating some constraint. The most common goals are minimizing cost, maximizing throughput, and/or efficiency. This is one of the major quantitative tools in industrial decision making.
Overarching Integrated Product Team	OIPT	An Integrated Product Team (IPT) led by the appropriate Office of the Secretary of Defense (OSD) director, and composed of the Program Manager (PM), Program Executive Officer (PEO), Component staff, user/user representative, and OSD staff involved in the oversight and review of a particular Acquisition Category (ACAT) ID program.
Package Diagram		The Unified Modeling Language depicts the dependencies between the packages that make up a model.
Parametric Diagram		Used to structure the characteristics of the model into a high-level mathematical diagram that will aid in depicting the flow of calculations for the simulation.
People Ware	PW	People ware is a term used to refer to one of the three core aspects of computer technology: hardware, software, and people ware. People ware can refer to anything that has to do with the role of people in the development or use of computer software and hardware systems, including such issues as developer productivity, teamwork, group dynamics, the psychology of programming, project management, organizational factors, human interface design, and human-machine-interaction. (Wikipedia)
Performance Based Logistics	PBL	The preferred sustainment strategy for weapon system product support that employs the purchase of support as an integrated, affordable performance package designed to optimize system readiness. PBL meets performance goals

		for a weapon system through a support structure based on long-term performance agreements with clear lines of authority and responsibility.
Planned Maintenance System	PMS	The Planned Maintenance System (PMS) provides each command with a simple, standard means for planning, scheduling, controlling, and performing planned equipment maintenance. PMS actions are the minimum actions necessary to maintain equipment in a fully operational condition. (Integrated Publishing)
Point of Contact	POC	Person serving as coordinator, action officer, or focal point for an activity.
Port Hueneme Division	PHD	Division of the NSWC under the Naval Sea Systems Command
Post Independent Analysis	PIA	Post-Independent Analysis. The final step in the JCIDS analysis process is the PIA. In this step, the sponsor will assess the compiled information and analysis results of the FSA (non-materiel and materiel approaches) to ensure the list of approaches with the potential to deliver the capability identified in the FAA and FNA is complete. The sponsor team performing the PIA shall be made up of individuals who were not involved in the FSA. This information will be compiled into an appropriate recommendation and documented in an ICD or joint DCR
Probability of kill	Pk	The Probability of Kill (or Pk) is usually based on a Uniform random number generator. This algorithm creates a number between 0 and 1 that is approximately uniformly distributed in that space. If the Pk of a weapon/target engagement is 30% (or 0.30), then every random number generated that is less that 0.3 is considered a kill. Every number greater than 0.3 is considered a "not kill". When used many times in a simulation, the average result will be that 30% of the weapon/target engagements will be a kill and 70% will not be a kill. (Wikipedia)
Probability Raid Annihilation	P_{RA}	The Navy P_{RA} Test bed implements HLA federated simulations of ship combat system elements against independent, reactive threat raids in a common environment to formulate an overall combat system assessment.
Process Specification	PSPEC	PSPECs are primitive specification provided as structured English descriptions that use “shall” statements to describe the way in which they transform input information into desired outputs.
Program Executive Office; Program Executive Officer	PEO	The military or civilian official who has responsibility for directing several Major Defense Acquisition Programs (MDAPs) and for assigned major system and non-major system acquisition programs. A PEO has no other

		command or staff responsibilities within the Component, and only reports to and receives guidance and direction from the DoD Component Acquisition Executive (CAE).
Programmatic Environmental, Safety, and Health Evaluation	PESHE	PESHE incorporates all the environmental, safety and health regulations into a document that is required by Milestone B and is updated through each acquisition phase.
Project Management Plan	PMP	Project management is a carefully planned and organized effort to accomplish a specific (and usually) one-time effort, for example, constructs a building or implements a new computer system. Project management includes developing a project plan, which includes defining project goals and objectives, specifying tasks or how goals will be achieved, what resources are need, and associating budgets and timelines for completion. It also includes implementing the project plan, along with careful controls to stay on the "critical path", that is, to ensure the plan is being managed according to plan. Project management usually follows major phases (with various titles for these phases), including feasibility study, project planning, implementation, evaluation and support/maintenance
Radar Cross Section	RCS	Radar cross section (RCS) is a measure of how detectable an object is with radar. When radar waves are beamed at a target, only a certain amount is reflected back. A number of different factors determine how much electromagnetic energy returns to the source, such as the angles created by surface plane intersections. For example, a stealth aircraft (which is designed to be undetectable) will have design features that give it a low RCS, as opposed to a passenger airliner that will have a high RCS. RCS is integral to the development of radar stealth technology, particularly in applications involving aircraft and ballistic missiles. RCS data for current military aircraft are almost all classified. (Wikipedia)
Radar Height	Hr	Analytical expressions are derived for the radiation pattern of a fixed reflector antenna on which there are two or more field horns operated in a monopulse mode to obtain target altitude information. Results are obtained for the radiation pattern of an arbitrary reflector located at an arbitrary height above, the earth, and fed by horns, which may be located at an arbitrary position. (DTIC ADA027300)
Radar Horizon	Rd(km)	The locus of points at which the rays from a radar antenna become tangential to the Earth's surface. On the open sea this locus is horizontal, but on land it varies according to the topographical features of the terrain. (DITIC)
Radar Horizon Range	RHR	The locus of points at which the rays from a radar antenna

		become tangential to the Earth's surface. On the open sea this locus is horizontal, but on land it varies according to the topographical features of the terrain. (DoD Military and Associated Terms)
Radio Frequency	RF	
radius of Closes Point Approach	r_{CPA}	Algorithm calculates closest distance that can be reached by two moving objects. This calculation is widely used in collision avoidance. (Implementation of CPA Algorithm in Multidimensional Risk…Waworek, Burka, Baranowski)
Random Access Memory	RAM	A form of computer data storage. (Wikipedia)
Reaction Time	Tr	Reaction time (RT) is the elapsed time between the presentation of a sensory stimulus and the subsequent behavioral response. RT is often used in experimental psychology to measure the duration of mental operations, an area of research known as mental chronometry. The behavioral response is typically a button press but can also be an eye movement, a vocal response, or some other observable behavior. (Wikipedia)
Reconfigurability		Capability of a system, so that its behavior can be changed by reconfiguration, i.e. by loading different configware code. This static reconfigurability distinguishes between reconfiguration time and run time. Dynamic reconfigurability denotes the capability of a dynamically reconfigurable system that can dynamically change its behavior during run time, usually in response to dynamic changes in its environment. (Wikipedia)
Redundancy		Repetition of parts or subsystems to assure operation if original (primary) part or subsystem fails. (DAU)
Reference Nodes	Ref	Reference nodes store connections between attributes in a scene and attributes in the reference; connections between attributes contained in the reference, but that are not part of the reference file; dynamic attributes on note in the reference file, but where the attributes were not defined in the reference file; set attributes made after the file was referenced; and internal broken reference connections.
Reliability Theory	RT	Reliability theory is a general theory about systems failure. It allows researchers to predict the age-related failure kinetics for a system of given architecture (reliability structure) and given reliability of its components. Reliability theory predicts that even those systems that are entirely composed of non-aging elements (with a constant failure rate) will nevertheless deteriorate (fail more often) with age, if these systems are *redundant* in irreplaceable elements. Aging, therefore, is a direct

		consequence of systems redundancy. Reliability theory also predicts the late-life mortality deceleration with subsequent leveling-off, as well as the late-life mortality plateaus, as an inevitable consequence of *redundancy exhaustion* at extreme old ages.
Reliability, Availability, and Maintainability	RAM	Essential elements of mission capability. Reliability is the probability of an item to perform a required function under stated conditions for a specified period of time. Reliability is further divided into mission reliability and logistics reliability. Availability is the measure of the degree to which an item is in an operable state and can be committed at the start of a mission when the mission is called for at an unknown (random) point in time. Availability as measured by the user is a function of how often failures occur and corrective maintenance is required, how often preventative maintenance is performed, how quickly indicated failures can be isolated and repaired, how quickly preventive maintenance task can be performed, how long logistics support delays contribute to down time. Maintainability is the ability of an item to be retained in, or restored to, a specified condition when maintenance is performed by personnel having specified skill levels, using prescribed procedures and resources, at each prescribed level of maintenance and repair. (DoD Guide for Achieving RAM Handbook)
Remote Monitoring	RM	Support monitoring and protocol analysis of LANs. The original version (sometimes referred to as RMON1) focused on OSI Layer 1 and Layer 2 information in Ethernet and Token Ring networks. It has been extended by RMON2, which adds support for Network- and Application-layer monitoring, and by SMON, which adds support for switched networks. It is an industry standard specification that provides much of the functionality offered by proprietary network analyzers. RMON agents are built into many high-end switches and routers. (Wikipedia)
Request for Proposal	RFP	A solicitation used in negotiated acquisition to communicate government requirements to prospective contractor and to solicit proposals. (DAU)
Requirements Analysis	RA	Encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. (Wikipedia)
Requirements Creep	RC	The tendency of the user (or developer) to add to the original mission responsibilities and/or performance requirements for a system while it is still in development. (DAU)

Requirements Diagram	RD	A custom diagram used to describe a system's features or requirements as a visual model. (Sparx Systems)
Research, Development & Acquisition	RDA	Research, Development & Acquisition (RD&A) researches, assesses, and models emerging technologies. DAU
Research, Development, Test and Evaluation	RDT&E	Budget Activity (BA) 6 within an RDT&E appropriation account that includes RDT&E efforts and funds to sustain and/or modernize the installations or operations required for general RDT&E. Test ranges, military construction, maintenance support of laboratories, Operation and Maintenance (O&M) of test aircraft and ships, and studies and analysis in support of the DoD RDT&E program are all funded by this BA. (DoD 7000.14-R) See RDT&E Budget Activities
Reusable Application Software	RAS	The whole of an application system may be reused either by incorporating it without change into other systems (COTS reuse) or by developing application families. (Ian Smith, Software Engineering)
Rich Text Format	RTF	A document language used for exchanging text between different word processors and text-processing applications. RTF is much easier to generate than PDF or PostScript, and is more word-processor friendly than HTML. RTF has been around for over a decade, while hundreds of other binary formats have come and gone. (Interglacial)
Scenario Based Evaluation	SBE	An approach that uses "what if" situations and identifies strengths, weaknesses and potential solutions.
Secretary Of Defense	SECDEF	Current and future threats to America's security. More than leading-edge weapon systems, doctrinal innovation, and the employment of technology, this transformation is clearly about changing our approach to the fundamental business practices and infrastructure "backbone" of the Department of Defense.
Self Defense	SD	Provide a final layer of self-protection against air and surface threats.
Sequence Diagram		A sequence diagram represents the interaction between collaborating parts of a system.
Service Oriented Architecture	SOA	In computing, service-oriented architecture (SOA) provides methods for systems development and integration where systems group functionality around business processes and package these as interoperable services. SOA also describes IT infrastructure, which allows different applications to exchange data with one another as they participate in business processes. Service-orientation aims at a loose coupling of services with

		operating systems, programming languages and other technologies which underlie applications
Ship Self Defense System	SSDS	Ship's Self Defense System (SSDS) is an integrated weapons system used aboard large U.S. Navy ships, such as Nimitz class supercarriers and various amphibious assault ships, such as LHDs and LSDs. SSDS has similar attributes to the combat system used aboard DDGs and CGs, in that it is an integrated Combat Direction System (CDS). The combat direction systems aboard DDGs and CGs, Aegis, are purpose built integrated designs from the outset. SSDS follows a different approach and uses existing shipboard radars and weapons systems, integrated under a COTS framework, to provide a cohesive CDS. The first SSDS designs were back fit onto existing U.S. Navy ships.
SLQ-2		The AN/SLQ-32 (V) provides operational capability for early warning of threat weapon system emitters and emitters associated with targeting platforms, threat information to own ship hard-kill weapons, automatic dispensing of chaff decoys, and Electronic Attack (EA) to alter specific and generic ASCM trajectories. (Military Analysis Network)
Soft-kill	SK	Non-lethal weapons that disable or destroy without causing significant injury or damage. (P. Wolfowitz, '91)
Software	SW	A general term used to describe a collection of computer programs, procedures and documentation that perform some tasks on a computer system. (Wikipedia)
Software Engineering Institute	SEI	The SEI is a federally funded research and development center conducting software engineering research in acquisition, architecture and product lines, process improvement and performance measurement, security, and system interoperability and dependability
Software Product Line	SPL	A software product line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. Def from SW Eng Institute, CMU
Source Lines of Code in Thousands	SLOC	Source lines of code (SLOC) is a software metric used to measure the size of a software program by counting the number of lines in the text of the program's source code. SLOC is typically used to predict the amount of effort that will be required to develop a program, as well as to estimate programming productivity or effort once the software is produced. (Wikipedia)
Space and Naval	SPAWAR	Space and Naval Warfare Systems Center Pacific (SSC

Warfare		Pacific) is responsible for development of the technology to collect, transmits, process, display and, most critically, manage information essential to successful military operations. The Center develops the capabilities that allow decision-makers of the Navy, and increasingly of the joint services, to carry out their operational missions and protect their forces.
Spiral Approaches	SA	The spiral method is similar to the prototyping approach to systems development. However, the Spiral method adds an assessment of the risks inherent in the construction. At each iteration, the development team, in conjunction with the end-user, looks at the risks, costs and alternatives open to the user. In each iteration, the user requirements are refined and a more complex prototype is produced either by building on the existing version or by creating a new one. (4 Consulting)
Stakeholder Requirements	SR	User- or user representative-generated validated needs developed to address mission area deficiencies, evolving threats, emerging technologies or weapon system cost improvements. Operational requirements form the foundation for weapon system unique specifications and contract requirements. (DAU) An enterprise, organization, or individual having an interest or a stake in the outcome of the engineering of a system. (EIA-632, Annex A)
Standard Missile-2	SM-2	The Standard Missile-2 (SM-2) is the Navy's primary surface-to-air fleet defense weapon. The currently deployed SM-2 Block II/III/IIIA/IIIB/IV configurations are all-weather, ship-launched medium-range fleet air defense missiles derived from the SM-1 (RIM-GGB), which is still in allied fleets. SM-2 employs an electronic countermeasures-resistant monopulse receiver for semi-active radar terminal guidance and inertial midcourse guidance capable of receiving midcourse command updates from the shipboard fire control system. SM-2 is launched from the Mk 41 Vertical Launching System (VLS), and from various rail-type Guided Missile Launching Systems (GMLSs) in allied Fleets. SM-2 continues to evolve to counter expanding threat capabilities and improvements in advanced high and low-altitude threat interception, particularly in stressing electronic countermeasure (ECM) environments, which are being implemented through modular changes to the missile sections. (Military Analysis Network)
Starvation	Starv	Inputs do not reach the various locations.
State Machine	SM	A model of behavior composed of a finite number of states, transitions between those states, and actions
State Machine	SMD	The state machine diagram describes the state transitions

Diagram		and actions that a system or its parts perform in response to events. State Machine diagrams are typically used to represent the life cycle of a block.
State Transition Table	STT	A tabular representation of the machine changes from one state another in reaction to the inputs. The state transition table has the same information as the state transition diagram.
Statement of Work	SOW	That portion of a contract which establishes and defines all non-specification requirements for contractor's efforts either directly or with the use of specific cited documents.
Supportability	S	A key component of availability. It includes design, technical support data, and maintenance procedures to facilitate detection, isolation, and timely repair and/or replacement of system anomalies. This includes factors such as diagnostics, prognostics, real time maintenance data collection, and human system integration considerations. (CJCSI 3170.01E)
Supportability of Systems	SoS	The ability to provide for the right level of maintenance, training, test equipment, technical documentation, supply support, facilities, transportability, human systems interfaces and other non-functional requirements need to be brought into the systems engineering process during the initial phases. By considering supportability in design, the stakeholders will have a better opportunity to consider trade-offs related to reliability, maintainability, and usability of the system.
Surface To Air Missile	SAM	Ship-based SAMs are in widespread use. Virtually all-surface warships can be armed with SAMs. In fact, naval SAMs are a necessity for all front-line surface warships. Some warship types specialize in anti-air warfare e.g. *Ticonderoga*-class cruisers equipped with the Aegis combat system or *Kirov* class cruisers with the S-300PMU *Favorite* missile system. (Wikipedia)
Surface Warfare	SUW	Modern surface warfare dates from the mid 20th century, when surface, air, and submarine warfare components were blended together as a tactical unit to achieve strategic objectives. The two most important strategic objectives are interdiction and sea control. Interdiction is the process of preventing enemy forces access to or through a location. Sea control is the dominance of force over a given area that prevents other naval forces from operating successfully. (Wikipedia)
Surveillance		Monitoring Behavior (Wikipedia)
Systems Architecture and Requirements Engineering (Hatley et		Descriptions, including graphics, of systems and interconnections providing for or supporting warfighting functions.

al)		
System Boundary		The system is a ``black-box", with an explicit boundary, describing the behavior of the system by essential use case responsibilities. (Essential Use Cases in Responsibility of OO Development, Biddle, Noble, Tempero)
System Modeling Language Behavior	SysML	The behavior diagrams include the use case diagram, activity diagram, sequence diagram, and state machine diagram.
System of Systems:	SoS	A set or arrangement of interdependent systems that are related or connected to provide a given capability. The loss of any part of the system will significantly degrade the performance or capabilities of the whole. (CJCSI 3170.01E)
System Scenarios	SS	Define the inputs and stimuli planned or expected and the response for each
System Synthesis	SS	An implementation independent specification of a system; this includes functionality and constraints. (System Synthesis of Digital Systems, Eles, Peng)
Systems Engineering	SE	The overarching process that a program team applies to transition from a stated capability to an operationally effective and suitable system. SE encompasses the application of SE processes across the acquisition life cycle (adapted to each and every phase) and is intended to be the integrating mechanism for balanced solutions addressing capability needs, design considerations and constraints, as well as limitations imposed by technology, budget, and schedule. The SE processes are applied early in concept definition, and then continuously throughout the total life cycle. (Defense Acquisition Guidebook)
Systems Engineering Plan	SEP	A description of the programs overall technical approach including processes, resources, metrics, applicable performance incentives, and the timing, conduct, and success criteria of technical reviews. (DAU)
Systems View	SV	An architecture view that identifies the kinds of systems, how to organize them, and the integration needed to achieve the desired operational capability. It will also characterize available technology and systems functionality. (CJCSM 3170.01B)
Target	Tgt	Anything fired at. (Dictionary.com)
Target arrival rate	Tar	Time and rate of target arrival (lb)
Target Maximum Height of target Height	Ht	Estimation of target height of a low altitude in tracking radar or air traffic control radar equipped with monopulse antenna, it is well known that bias error and some large spike error occur because of interference between the direct and reflected signals. (Science Links Japan)

Technical Standards View	TV	Listing of standards that apply to Systems View elements in a given architecture. (DoD Architecture Framework, Steven Dam)
Test and Evaluation	T&E	The TES begins by focusing on Technology Development activities, and describes how the component technologies being developed will be demonstrated in a relevant environment (i.e., an environment of stressors at least as challenging as that envisioned during combat) to support the program's transition into the System Development and Demonstration Phase. It contains hardware and software maturity success criteria used to assess key technology maturity for entry into System Development and Demonstration.
Test and Evaluation Master Plan	TEMP	Documents the overall structure and objectives of the Test and Evaluation (T&E) program. It provides a framework within which to generate detailed T&E plans and it documents schedule and resource implications associated with the T&E program. The TEMP identifies the necessary Developmental Test and Evaluation (DT&E), Operational Test and Evaluation (OT&E), and Live Fire Test and Evaluation (LFT&E) activities. It relates program schedule, test management strategy and structure, and required resources to: Critical Operational Issues (COIs), Critical Technical Parameters (CTPs), objectives and thresholds documented in the Capability Development Document (CDD), evaluation criteria, and milestone decision points. For multiservice or joint programs, a single integrated TEMP is required. Component-unique content requirements, particularly evaluation criteria associated with COIs, can be addressed in a component-prepared annex to the basic TEMP. See Capstone TEMP). (DAU)
Test and Evaluation Plan	TEP	A framework within which to generate detailed T&E plans.
Test and Evaluation Strategy	TES	An early test and evaluation planning document that describes test and evaluation activities starting with Technology Development and continuing through System Development and Demonstration into Production and Deployment. The TES describes how component technologies being developed will be demonstrated in a relevant environment to support the program□s transition into the System Development and Demonstration Phase. Over time, the scope of this document will expand and evolve into the Test and Evaluation Master Plan (TEMP) due at Milestone B. (Defense Acquisition Guidebook
Threat Evaluation Weapon Assessment	TEWA	Providing commanders with advanced decision aids requires a good understanding of the processes involved,

		their information requirements, and the development of formal domain models upon which reasoning processes can be based. (IEEE)
Time of Flight	TOF	The time of flight (TOF) describes the method used to measure the time that it takes for a particle, object or stream to reach a detector while traveling over a known distance. (Wikipedia)
Transmission Control Protocol/Internet Protocol	TCP/IP	The Transmission Control Protocol (TCP) is one of the core protocols of the Internet Protocol Suite. TCP is so central that the entire suite is often referred to as "TCP/IP". Whereas IP handles lower-level transmissions from computer to computer as a message makes its way across the Internet, TCP operates at a higher level, concerned only with the two *end systems*, for example a Web browser and a Web server. In particular, TCP provides reliable, ordered delivery of a stream of bytes from one program on one computer to another program on another computer. Besides the Web, other common applications of TCP include e-mail and file transfer. Among its management tasks, TCP controls message size, the rate at which messages are exchanged, and network traffic congestion. (Wikipedia)
Transmit Receive	T/R	The Tx level is the power in decibels per milliwatt (dBm) at which a modem transmits its signal. The Rx level is the power in dBm of the received signal. The server modems normally transmit at -13 dBm by default. Ideally, the Rx level should be in the range of -18 to -25 dBm. If the Rx level is under -25 dBm, the Signal-to-Noise Ratio (SNR) is likely to decrease, meaning that the speed also decreases. If the Rx level is too high, you may see signal distortion or the receiver's Digital Signal Processor (DSP) being overdriven, and erratic connections are possible. (CISCO)
Under Secretary Of Defense	USD	The Under Secretary of Defense for Policy is the title of a high-level civilian official in the United States Department of Defense. The Under Secretary of Defense for Policy is the principal staff assistant and advisor to both the Secretary of Defense and the Deputy Secretary of Defense for all matters concerning the formation of national security and defense policy. The position is considered the number three office in the Department of Defense, after the Secretary of Defense and Deputy Secretary of Defense.
Unified Modeling Language	UML	The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business

		processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components." (OMG SPEC)
United States	US	Country
United States Navy	USN	Department of Defense Service Branch, responsible for the US Fleet. (lb
Universal Data Access Layer	UDAL	UniDAC offers unified approach to the database-related applications development process. That means you can switch easily between different databases in your projects without going deep into their specifics. (DEVART.com)
Use Case Diagram	UCD	Use case diagrams overview the usage requirements for a system. They are useful for presentations to management and/or project stakeholders, but for actual development you will find that use cases provide significantly more value because they describe "the meat" of the actual requirements. (Agile Modeling, Scott Ambler)
Use Cases	UC	A use case in software engineering and systems engineering is a description of a system's behavior as it responds to a request that originates from outside of that system. The use case technique is used to capture a system's behavioral requirements by detailing scenario-driven threads through the functional requirements. (Wikipedia)
Vee Model	Vee Model	The Vee model is rooted in the project cycle, which is displayed from left to right to represent project time and maturity. Coupled with this depiction is the recognition of levels of decomposition, which are illustrated, in the vertical dimension from top to bottom. The User is at the highest level and parts and lines of code the lowest. The plane orthogonal to the plane of the Vee illustrates the number of entities at each level of decomposition, which relates to the complexity of the system. INCOSE definition
Velocity missile	Vm	Velocity of the missile. (lb)
Velocity of target	V target	A method for measuring the velocity (v) of a moving target (A). Radio signals are transmitted by means of radio transmitter-receiver arrangements of both the measuring station (B) and the target station (A) and received at the opposite stations (A, B), the frequencies of which signal include Doppler shifts (fd) to be observed both at the measuring station (B) and in the measuring target (A). On the basis of the Doppler shifts (fd), the escape and/or approach velocity (v) of the target to be measured is determined relative to the measuring station (B). Radio signals are transmitted from the measuring station (B) and

		the target station (A), the frequencies of which signals deviate from each other by a relatively small frequency difference (fk) or by a Doppler frequency shift (fd), the absolute value of which small frequency difference remains at least as high as the absolute value of the largest expectable Doppler frequency shift (fd_{max}). A difference frequency or difference frequencies (fb, fa) of the frequency including the Doppler shift (fd) and received at the measuring station (B) and the target station (A) and of the reference frequency of essentially the signal frequency range developed locally are formed in such a way that a low-frequency or possibly a zero-frequency signal at the target station is obtained as the difference frequency or frequencies (fb, fa). On the basis of the difference frequencies (fb, fa), the Doppler frequency shift (fd) is determined, by means of which the velocity (v) of the target (A) is obtained
Verification & Validation	V&V	Verification & Validation is the process of checking that a product, service, or system meets specifications and that it fulfils its intended purpose. These are critical components of a quality management system such as ISO 9000. (Wikipedia)
Vertical Launch System	VLS	MK 41 is a fixed, vertical, multi-missile storage and firing system that lets Navy vessels launch significant firepower. The capability of VLS to simultaneously prepare one missile in each half of a launcher module allows for fast reaction to multiple threats with concentrated, continuous firepower. The US Navy currently deploys MK 41 VLS - on AEGIS-equipped Ticonderoga-class cruisers and Spruance- and Arleigh Burke-class destroyers - and plans to use it on next generation of surface ships, the LDP17, DD-21 and CG-21. (Military Analysis Network)
Virtual Private Network	VPN	A virtual private network (VPN) is a network that uses a public telecommunication infrastructure, such as the Internet, to provide remote offices or individual users with secure access to their organization's network. A virtual private network can be contrasted with an expensive system of owned or leased lines that can only be used by one organization. The goal of a VPN is to provide the organization with the same capabilities, but at a much lower cost. (Search Security.Com)
Weapon Engagement Manager	WEM	Real time actions involving use of hard kill and soft kill weapons to counter air and surface threats. (The Smithsonian/NASA Astrophysics Data System)
Windows Metafile	WMF	Windows Metafile (WMF) is a graphics file format on Microsoft Windows systems, originally designed in the early 1990s. Windows Metafiles are intended to be portable between applications and may contain both

		vector and bitmap components. In contrast to raster formats such as JPEG and GIF which are used to store bitmap graphics such as photographs, scans and graphics, Windows Metafiles generally are used to store line-art, illustrations and content created in drawing or presentation applications.
Work Breakdown Structure	WBS	An organized method to break down a project into logical subdivisions or subprojects at lower and lower levels of details. It is very useful in organizing a project. See Military Handbook (MIL-HDBK) 881 for examples of WBSs. (DAU)

APPENDIX D: RESEARCH SUMMARIES

Research Topic: Concept of Operations (ConOps)

Researcher: Paul Wheeler

Title: The Concept of Operations: The Bridge from Operational Requirements to Technical Specifications

Author: Richard E. Fairly, SEMA Inc. and Drexel University

Publisher: IEEE, 1994

Summary:

The History of the ConOps Document

One of the earliest reports on formalizing the description of operational concepts for a software system is contained in a 1980 report entitled "A Structured Approach for Operational Concept Formulation." The importance of a well-defined operational concept to the success of system development is emphasized in the report.

Concept Analysis Process

Concept analysis is the process of analyzing a problem domain and an operational environment for the purpose of specifying the characteristics of a proposed system for the user's perspective. Concept analysis should be the first step taken in the overall system development process. It provides users with a mechanism for defining their needs and desires.

The results of the concept analysis are recorded in the ConOps document, which serves as a checklist to guide the analysis process and becomes the foundation document for all subsequent system development activities. The ConOps document should say everything about the system that the users need to communicate to the system developers.

The ConOps Document

Essential elements of the ConOps document include the following:

- A description of the current system
- A description of the needs that motivate development of a new system or modification of an existing system
- Modes of operation for the proposed system
- User classes and user characteristics
- Operational features of the proposed system

- Priorities among proposed operational features
- Operational scenarios for each operational mode and class of user
- Limitations of the proposed approach
- Impact analysis for the proposed system

The ConOps document should be written in narrative prose, using the language and terminology of the user's application domain. It should be organized in a manner which tells a story and should make use of visual reference and diagrams whenever possible.

Key Points:

- Concept analysis is the process of analyzing a problem domain and an operational environment for the purpose of specifying the characteristics of a proposed system for the user's perspective.
- The ConOps document should say everything about the system that the users need to communicate to the system developers.
- The ConOps has several essential elements, which should be addressed.

Applicability to Capstone: The ConOps document will serve a primary building block for the capstone project and it will contribute to the formulation of multiple capstone products including the CORE DoDAF views, the Extend AAW model, and the requirements document.

Recommendations: Recommend that the essential elements 1, 2, 4 & 9 listed above not be discussed in the context of the ConOps for this capstone project. We are not designing or proposing an AAW system that is better then an existing or current system, but we are building abstract models and views which will allow one to prove out any AAW performance given certain desired parameters and target ranges for P_{RA}. Therefore, mention of a proposed system and how it may or may not address a need that current systems are not addressing will not be discussed.

Research Topic: DoD Architecture Framework (DoDAF)

Researcher: Ruth Matela/Eric Sarabia

Title: *DoD Architecture Framework*

Author: Steven H. Dam

Publisher (Date): SPEC, Marshall VA (March 2006)

Summary: Chapter 2 and 3 defines DoDAF and discusses how it is implemented.

- The DoDAF provides the guidance and rules for developing, representing, and understanding architectures based on a common denominator across DoD, Joint, and multinational boundaries. It provides insight for external stakeholders into how the DoD develops architectures. The DoDAF is intended to ensure that architecture descriptions can be compared and related across programs, mission areas, and, ultimately, the enterprise, thus, establishing the foundation for analyses that supports decision-making processes throughout the DoD.
- DoDAF v1.5 is a transitional version that responds to the DoD's migration towards NCW. It applies essential net-centric concepts1 in transforming the DoDAF and acknowledges that the advances in enabling technologies – such as services within a Service Oriented Architecture (SOA) – are fundamental to realizing the Department's Net-Centric Vision.2 Version 1.5 addresses the immediate net-centric architecture development needs of the Department while maintaining backward compatibility with DoDAF v1.0.
- The Operational View captures the operational nodes, the tasks or activities performed, and the information that must be exchanged to accomplish DoD missions. It conveys the types of information exchanged, the frequency of exchange, which tasks and activities are supported by the information exchanges, and the nature of information exchanges.
- The Systems View captures system, service, and interconnection functionality providing for, or supporting, operational activities. DoD processes include warfighting, business, intelligence, and infrastructure functions. The SV system functions and services resources and components may be linked to the architecture artifacts in the OV. These system functions and service resources support the operational activities and facilitate the exchange of information among operational nodes.
- The Technical Standards View is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements. Its purpose is to ensure that a system satisfies a specified set of operational requirements. The TV provides the technical systems implementation guidelines upon which engineering specifications are based, common building blocks are established, and product lines are developed. It includes a collection of the technical standards, implementation conventions, standards options, rules, and criteria that can be organized into profile(s) that govern systems and system or service elements for a given architecture.

- There are some overarching aspects of an architecture that relate to all three views. These overarching aspects are captured in the AV products. The AV products provide information pertinent to the entire architecture but do not represent a distinct view of the architecture. AV products set the scope and context of the architecture. The scope includes the subject area and time frame for the architecture. The setting in which the architecture exists comprises the interrelated conditions that compose the context for the architecture. These conditions include doctrine; tactics, techniques, and procedures; relevant goals and vision statements; concepts of operations (ConOps); scenarios; and environmental conditions.

Key Points:

- DoDAF definition: "Architecture is a fundamental and unifying structure defined in terms of elements, information, interfaces, processes, constraints, and behaviors."
- DoDAF "defines a common approach for Department of Defense (DoD) architecture description development, presentation, and integration. The Framework is intended to ensure that architecture descriptions can be compared and related across organizational boundaries, including Joint and multinational boundaries."
- Operational View is defined as "...description of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions."
    - OV-1 or the "cartoon", high level concept diagram
    - OV-2 & OV-3, interface diagrams
    - OV-4 is the organizational chart
    - OV-5 & 6, functional analysis products
    - OV-7, logical data model
- Systems View is defined as "...description... of systems and interconnections providing for, or supporting, DoD functions."
    - SV-1, SV-2, SV-3, SV-6; different versions of interface diagrams
    - SV-4, SV-5, SV-10; functional analysis products
    - SV-7; performance matrix
    - SV-8, SV-9; transition planning diagrams
    - SV-11; physical data model
- Technical Standards View is defined as "... minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements..."
    - TV-1; current (or near term projected) standards
    - TV-2; forecast of the standards that may come into play over the life of the architecture
- All Views definition includes the "...overarching aspects of an architecture that relate to all three of the views." Information presented in the all view products link and constrains the three views together.
    - AV-1 provides an executive summary of the architecture
    - AV-2 represents the complete set of architecture definitions

- The Framework does not tell you how to build architectures, only what products/data are needed for evaluation; you have to develop a methodology for developing the architecture.
- After viewing the documentation on DoDAF V1.5, our capstone will utilize the OV-1, OV-2, OV-5, OV-6C, SV-1. SV-2. SV-5a/b, TV-1, AV-1, & AV-2 views.
    - OV-1: High-Level Operational Concept Graphic
    - OV-2: Operational Node Connectivity Description
    - OV-5: Operational Activity Model
    - OV-6c: Operational Event-Trace Description
    - SV-1: Systems Interface Description / Services Interface Description
    - SV-2: Systems Communications Description / Services Communications Description
    - SV-5a: Operational Activity to Systems Function Traceability Matrix
    - SV-5b: Operational Activity to Systems Traceability Matrix
    - TV-1: Technical Standards Profile
    - AV-1: Overview and Summary Information
    - AV-2: Integrated Dictionary

Applicability to Capstone Project: A Model Based Systems Engineering (MBSE) approach used for developing requirements for weapon systems is a new concept for the Navy. Our capstone project will show how this approach will tie our requirements to an existing framework such as the DoDAF. DoDAF views will be generated as part of implementing our proposed MBSE methodology.

Recommendations: The Requirements IPT will apply the MBSE approach as a means of tying DoDAF V1.5's guidance on a "data-centric" approach, which provides a more flexible and adaptable framework for architecting net-centric, integrated, and/or federated architectures. DoDAF views will primarily be generated using CORE.

Research Topic: DoD Architecture Framework (DoDAF)

Researcher: Armando Valdez

Title: *DoD Architecture Framework*

Author: Steven H. Dam, Ph.D.

Publisher (Date): SPEC, Marshall VA (March 2006)

Summary:

- The problem with architectures in the past is that they do not fulfill the primary purpose of architectures, which is to form a bridge between the mission and design. Problems include no end-user involvement in the process, failure to understand the root causes of problems or capability gaps, failure to maintain traceability, no configuration/quality management, and the architecture lacks complete traceability of all elements and requirements.
- There are several techniques for building architectures such as ad hoc viewgraph engineering, structured analysis (IDEF0), object oriented (UML), and Model Based System Engineering (MBSE). Of these techniques, MBSE captures behavior of architectures, systems, or software. MBSE captures control and data flow in a single diagram, Enhanced Function Flow Block Diagram (EFFBD) – A behavior diagram. From the information in the behavior diagram we can create DoDAF products. MBSE allows starting with architecture and drilling down to the systems, subsystems, components, subcomponents (completely specify the physical elements of an architecture). MBSE can develop a traceability hierarchy that can be used to answer the "what if" questions (i.e. complete traceability between all elements).
- Most techniques discussed in this chapter use static diagrams to represent process flows. Static diagrams may or may not work since many of the processes interact with one another and functional decomposition can miss critical interfaces. Simulation enables the execution of these models, thus ensuring that the design is executable. MBSE has a built in discrete event simulation capability (CORESim) that will find execution errors. Most architecture projects never get this far. They begin and end with DoDAF products and never fully model the real architecture, leaving all the problems to the next level of analysis.
- Professor Dam, used seven different evaluation criteria to compare the effectiveness of IDEF, MBSE, and UML techniques. These criteria were 1) addresses your full lifecycle, 2) integrates a set of processes, 3) provides executable results, 4) uses appropriate software tools, 5) communicates well to all audiences, 6) extends the ability to adjust to specific needs, and 7) has been applied successfully. The table presented below is reproduced from Figure 5-25, of Professor Dam's book.

Criteria	**IDEF**	**MBSE**	**UML**
Addresses your full lifecycle	Yes (if all 15 views are developed)	Yes	No (mostly software focused)
Integrates a set of processes	Not directly (i.e., no Risk view)	Yes	No
Provides executable results	No	Yes	No
Uses appropriate software tools	Yes	Yes	No
Communicates well to all audiences	Focused on a technical audience	Provides insight into processes	Well understood by software developers
Extends the ability to adjust to specific needs	No	Yes	Yes
Has been applied successfully	Yes	Yes	No?

Key Points:

- The author compares IDEF, MBSE, and UML and concludes that MBSE is the clear winner.

Applicability to Capstone: The discussion topics of this chapter are very applicable to the Capstone Project.

Recommendations: We should use MBSE to develop our AAW architecture.

Research Topic: DoD Architecture Framework (DoDAF)

Researcher: Armando Valdez

Title: *DoD Architecture Framework*

Author: Steven H. Dam, Ph.D.

Publisher (Date): SPEC, Marshall VA (March 2006)

Title: DoD Architecture Framework Version 1.5, Volume II: Product Descriptions **Author**: N/A

Publisher (Date): DoD (23 April 2007)

Summary:

- The specific DoDAF products developed depend on the intended use of the architecture. In order to arrive at the products for our AAW architecture it is necessary to understand the definitions of the DoDAF artifacts. An **Operational View** is defined as a description of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions. The **Systems View** is defined as a description of systems and interconnections providing for, or supporting, DoD functions. **Technical Standards View** is defined as the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements. A **node** represents an element of architecture that produces, consumes, or processes data. **Needlines** are defined as a requirement that is a logical expression of the need to transfer information among nodes.
- The essential DoDAF products determined necessary for all architecture studies are AV-1, AV-2, OV-1, OV-2, OV-3, SV-1, and TV-1. From this minimum set we can expand the views to include OV-5, OV-6c, SV-2, SV-5a, and SV-5b.
- The following paragraphs provide a brief description of the views.
- ***Overview and Summary Information (AV-1)****:* Scope, purpose, intended users, environment depicted, and analytical finding
- ***Integrated Dictionary (AV-2):*** Architecture data repository with definitions of all terms used in all products
- ***High-level Operational Concept Graphic (OV-1):*** High-level graphical/textual description of operational concept. A textual description accompanying the graphic is crucial. Graphics alone are not sufficient for capturing the necessary architecture data.
- ***Operational Node Connectivity Description (OV-2):*** Operational nodes, connectivity, and information exchange need lines between nodes.
- ***Operational Information Exchange Matrix (OV-3):*** Information exchanged between nodes and the relevant attributes of that exchange
- ***Operational Activity Model (OV-5):*** Capabilities, operational activities, relationships, among activities, inputs, and outputs; overlays can show cost, performing nodes, or other pertinent information.

- ***Operational Event-Trace Description (OV-6c):*** One of three products used to describe operational activity – traces actions in a scenario or sequence of events.
- ***Systems and Services Interface Description (SV-1):*** Identification of systems nodes, systems system items, services, and service items and their interconnections, within and between nodes.
- ***Systems and Services Communications Description (SV-2):*** Systems nodes, systems, system items, services, and service items and their related communications lay-downs
- ***Operational Activity to Systems Function and Activity to Systems Traceability Matrix (SV-5a and b):*** Mapping of system functions back to operational activities and mapping of systems back to capabilities or operational activities
- ***Technical Standards Profile (TV-1):*** Listing of standards that apply to Systems and Services View elements in a given architecture

Key Points:

- The essential DoDAF products determined necessary for all architecture studies are AV-1, AV-2, OV-1, OV-2, OV-3, SV-1, and TV-1. Other products can be added as required.

Applicability to Capstone: The discussion topics of this chapter are very applicable to the Capstone Project.

Recommendations: After having reviewed our Capstone Project Management Plan, DoD Architecture Framework Version 1.5, Volume II, and the DoD Architecture Framework book by Steven H. Dam, I recommend that at a <u>minimum</u> we produce the following DoDAF views for our project:

<u>DoDAF Architecture Products</u>

All Views

AV-1 (Overview and Summary Information)

AV-2 (Integrated Dictionary)

Operational Views

OV-1 (High-Level Operational Concept Graphic)

OV-2 (Operational Node Connectivity Description)

OV-5 (Operational Activity Model)

OV-6c (Operational Event-Trace Description)

System/Services Views

SV-1 (Systems and Services Interface Description)

SV-2 (System and Services Communications Description)

SV-5a/b (Operational Activity to Systems Function and Activity to Systems Traceability Matrix)

Technical Standards Views

TV-1 (Technical Standards Profile)

Relationships

OV-1 to OV-2: Organizations, organization types, and/or human roles, depicted in OV-1 should be traceable to operational nodes in OV-2; relationships in OV-1 should trace to needlines in OV-2.

OV-2 to OV-5: The activities annotating an operational node in an OV-2 map to the activities described in an OV-5. Similarly, OV-5 should document the operational nodes that participate in each operational activity.

OV-2 to OV-6c: Lifelines in OV-6c should map to operational nodes in OV-2.

OV-5 to OV-6c: Events in OV-6c map to inputs and outputs of operational activities.

SV-1 to OV-2: An operational node in OV-2 may be supported by one or more systems in SV-1 (indicating that the operational node owns/uses the system). A needline in OV-2 may map to one or more interfaces in an SV-1, and an interface in SV-1 maps to one or more needlines in OV-2.

SV-1 to SV-4: SV-4 defines system functions that are executed by systems defined in SV-1.

SV-1 to SV-5: Systems in SV-1 match systems in SV-5.

Research Topic: DoD Architecture Framework (DoDAF)

Researcher: Jonathan Mendiola

Title: DoD Architecture Framework Version 1.5 (Volume II: Product Descriptions, Chapter ARCHITECTURE BASICS - VIEWS, PRODUCTS, AND ARCHITECTURE DATA)

Author: Department of Defense

Publisher (Date): Department of Defense (23 April 2007)

Summary:

- As defined in Volume I, the term integrated architecture refers to one in which architecture data elements are uniquely identified and consistently used across all products and views within the architecture. In most cases, an integrated architecture description has an OV, SV, TV, and an All View (AV) that are integrated with each other (i.e., there are common points of reference linking the OV and SV and also linking the SV and TV). The Operational Activity to Systems Functionality Traceability Matrix (SV-5), for example, relates operational activities from the Operational Activity Model (OV-5) to system functions from the Systems Functionality Description (SV-4); the SV-4 system functions are related to systems in the Systems Interface Description (SV-1), thus bridging the OV and SV. An architecture is defined to be an integrated architecture when products and their constituent architecture data elements are developed, such that architecture data elements defined in one view are the same (i.e., same names, definitions, and values) as architecture data elements referenced in another view.

Key Points:

- OV, SV, AV, and TVs should have data elements that are consistent across some are all views. "An architecture is defined to be an *integrated architecture* when products and their constituent architecture data elements are developed, such that architecture data elements defined in one view are the same (i.e., same names, definitions, and values) as architecture data elements referenced in another view."
- The OV captures the operational nodes, the tasks or activities performed, and the information that must be exchanged to accomplish DoD missions.
- The SV captures system, service, and interconnection functionality providing for, or supporting, operational activities. DoD processes include warfighting, business, intelligence, and infrastructure functions. The SV system functions and services resources, and components may be linked to the architecture artifacts in the OV.
- The TV is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements. Its purpose is to ensure that a system satisfies a specified set of operational requirements. The TV provides the technical systems implementation guidelines upon which engineering

specifications are based, common building blocks are established, and product lines are developed.

- The TV is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements. Its purpose is to ensure that a system satisfies a specified set of operational requirements. The TV provides the technical systems implementation guidelines upon which engineering specifications are based, common building blocks are established, and product lines are developed.
- List of products and their applicability are shown in pages 2-3 thru 2-5.
- The Framework does not advocate the use of any one methodology (e.g., structured analysis vs. object orientation) or one notation over another (e.g., IDEF1X or Unified Modeling Language (UML) [2005] notation) to complete this step, but products should contain the required instances of architecture data elements and relationships (i.e., those marked with an asterisk [*] in the data element tables). However, the need for a well-defined and rigorous methodology is acknowledged. There are several candidate methodologies available for consideration, and the choice is ultimately governed by the nature of the architecture being defined, the expertise and preferences of the architecture team, the needs of the customer, and the architecture end users.
- *CORE*
    - The use of an integrated tool or tool suite is highly recommended for developing an integrated architecture for consistency and version control. The selected tool(s) should allow the architect to produce consistent products/views by performing cross product checking. The selected tool(s) should include a mechanism for storing, updating, and retrieving architecture data and their relationships and an ability to automatically generate an integrated dictionary. The tool should be capable of importing/exporting from a CADM conformant database.
    - Before selecting a specific architecture tool-set, the architecture team needs to determine the best method (i.e., object-oriented or structured analysis) to implement the purpose of the architecture. If the purpose of the architecture is to design a system largely for software development, then UML tools are likely the best choice. Alternately, if the purpose of the architecture is to analyze business processes, then IDEF tools are likely a good option. The architecture team must carefully select the best method, because there are no known automated tools to convert one method to another (i.e., IDEF to UML, UML to IDEF). IDEF favors process while UML favors objects.
    - In short, structured analysis (IDEF) emphasizes process and functions, while object-oriented analysis (UML) emphasizes system behavior using objects.
    - An OMG activity that kicked off in 2005 is finalizing the specification of a UML profile for DoDAF [UML Profile For DoDAF/MoDAF RFP (UPDM), Document - dtc/05-09-12, http://syseng.omg.org/UPDM.htm]
- For diagrams that model system decomposition, detail increases as the perspective changes from that of the planner, to the owner, to the designer, and to the builder.

  - A good guide to tracking the level of detail of an architecture is to always ensure that the information is at the level of detail that is meaningful to the intended user of the architecture. A good rule of thumb is to restrict decomposition levels for any one type of diagram within the same perspective to no more than three levels because that is generally sufficient to provide the required level of granularity for a stated objective.
- Depending on the architecture level needed (e.g., high levels of abstraction that hide design and implementation details) and the intended audience, the Framework products may be developed by applying an iterative method. Iterative development crosses all views. OVs can drive SV and TV changes; SVs can drive OV and TV changes, and so forth. Products iterate across views in the same way that they iterate within one view but across levels of detail.
  - The first model may consist of only highly abstract/generic sets of operational nodes, operational activities, and so forth. Later, when new details need to be added and the architecture is expanded to show more design detail, a new model (*Model B*, consisting of modified Model A products plus additional products as necessary) must be developed.
  - The new products that make up Model B will include and trace back to the original group of products (that make up Model A of the architecture).
- UML Actors aggregate to operational nodes useful to express OV-2, OV-3, and OV-4 information
- There are general relationships that logically interconnect the Framework products from one view to products of another view. The architect needs to be continuously aware of these necessary relationships to produce an architecture that is consistent across the four views and to provide clear traceability and connections from one view to another.
- The DoDAF v1.5 is provided as guidance for architects to begin representing net-centric architectural constructs within the DoDAF v1.5 views and products, while remaining backward compatible with DoDAF v1.0 products which may still be sufficient for architectures that have yet to address the NCE.
  - *CONCEPT: Populate the Net-Centric Environment*
  - *CONCEPT: Utilize the Net-Centric Environment*
  - *CONCEPT: Accommodate the Unanticipated User*
  - *CONCEPT: Promote the Use of Communities of Interest*
  - *CONCEPT: Support Shared Infrastructure*
- Description of CADM provided.

Applicability to Capstone Project: Will produce various DoDAF views for our project.

Recommendations: Requirements IPT use the guidance in this article to create required DoDAF views.

Research Topic: DoD Architecture Framework (DoDAF)

Researcher: Alan Wellesley

Title: Validating DoD Architectures: The Promise of Systems Engineering

Author: Laurent Balmelli, PhD, Research Staff Member, T.J. Watson Research Center and Tokyo, Research Laboratory, IBM

Publisher: CCRTS, 2006

Summary: Today's needs for interoperability and portfolio management, across military organizations worldwide, lead to increased focus on architecture development. Architecture frameworks are sponsored by Defense Departments in Australia, Canada, France, Korea, United Kingdom and United States. The majority of today's efforts in architecture development focus on generation of disparate architecture views, seemingly without the benefit and rigor offered by systems engineering.

This paper describes the DoDAF validator–a Model Based Systems Engineering (MBSE) approach to the architecture development process. The approach is an extension to a proven systems engineering process that creates a win-win scenario for executive oversight team as well as the operational and engineering communities. The process also ensures that interoperability based on executable design verification leads to program success and consistent and accurate architecture perspectives for further analysis.

Key Points:

- Meeting the objectives of DoDAF is no trivial task. The development and information required is outside of established mainstream processes.
- Often special working groups are created to generate the products and these groups are often divided into smaller groups that develop specific views based on individual expertise, leading to the creation of inconsistent views, and data and semantic interfacing problems. Reinforcement of inconsistency comes from those who certify the views; in general they too only examine portions in which they are experts.
- Using an architecture product production method independent of mainstream processes is asking for unidentified integration and interoperability problems and management oversight nightmares, let alone meeting the DoDAF objectives between architectures.

Applicability to Capstone Project: Generation of DoDAF views / products must not be a stand alone effort

Recommendations: None

Research Topic DoD Architecture Framework (DoDAF)

Researcher: Wellesley

Title: Ver 1.5 Vol II: Product Descriptions

Author: DoD

Publisher: DoD / 23 April 2007

Summary: Table 7-1 is very informative about the relationships between the various DoDAF views.

Following is a sampling:

Such as: OV-1 to OV-2: Organizations, organization types, and/or human roles, depicted in OV-1 should be traceable to operational nodes in OV-2; relationships in OV-1 should trace to needlines in OV-2; OV-2 to OV-5: The activities annotating an operational node in an OV-2 map to the activities described in an OV-5. Similarly, OV-5 should document the operational nodes that participate in each operational activity; OV-2 to OV-6c: Lifelines in OV-6c should map to operational nodes in OV-2; OV-5 to OV-6c: Events in OV-6c map to inputs and outputs of operational activities; SV-1 to OV-2: An operational node in OV-2 may be supported by one or more systems in SV-1 (indicating that the operational node owns/uses the system). A needline in OV-2 may map to one or more interfaces in an SV-1, and an interface in SV-1 maps to one or more needlines in OV-2; SV-1 to SV-4: SV-4 defines system functions that are executed by systems defined in SV-1; SV-1 to SV-5; Systems in SV-1 match systems in SV-5;

Key Points:

- Relationships between views.

Applicability to Capstone Project:

- This table should be consulted to help validate the DoDAF products generated by the capstone team.
- Should also use in beginning to verify that correct links between views have been identified.

Recommendations: None

Research Topic: Logistics Support
Reviewed by Tim Carpenter
Title: White Paper – Open Source Five Nines

Author: Gregory B. Wilson, NSWC, PHD

Publisher: NSWC, PHD, December 11, 2007

Summary: This paper describes technologies and approaches used by private industry to achieve ultra high availabilities in the presence of faults while controlling labor costs. Naval systems are among the most complex, and complexity is rising rapidly. Increase failure rate due to growing system complexity contributes to performance decline at a rate greater than size increase. Open standards and open source products incorporate private industry lessons-learned to gain insight into solutions needed for high availability naval systems. Private industry has achieved ultra high availability systems that continue to operate when failed.

Key Points:

- Addressing availability needs provides means to identify technology components and human actions required to achieve high availability systems with 99.999% up time (five nines). Technology solutions can automate many labor-intensive processes needed by an organization to keep systems available.
- There are six basic factors to consider for open source and open standards involve testing and adoption of compatible organizational practices. Use of open technology components controls life cycle costs, including future upgrades funded by other organizations. Open source products allow high availability customizations that may be impossible in proprietary products. Proprietary technology establishes traceability to organizations outside customer organization, while open source may require traceability to expert groups within the customer organization. There are six basic factors are as follow:
    - Availability Drivers
    - Fault Reporting
    - System Configuration Management
    - Standardized open hardware
    - Environmental Monitoring
    - Boot and Protection

Recommendations: Recommend pursuing each of the six identified focus areas to identify COTS infrastructure needs as a part of customer acquisition strategy to maximize system availability at minimum cost. This will increase resources available for mission essential features, while maximizing the availability of those features.

Research Topic: Logistics Support

Researcher: Mike Kinberg

Title: A Program Managers Guide to Technical Data Rights

Author: Dr. Raj Iyer

Date: April 2007

Summary: This paper was written as a result of a process designed to assess the risk, value, and benefits associated with the procurement of technical data rights. Dr. Ayer was a part of an IPT that that utilized LEAN principles to determine their conclusions and recommendations. The research compiled a number of issues and concerns regarding technical data rights.

Key Points:

- Issues Identified:
  - Confusing policy on PBL and technical data
  - Difficulty in determining the type of rights to be pursued.
  - Cost of Data management
  - Inappropriate Data formats
  - Technical Data unsuitable for re-procurement
  - Confusion over technical data requirements and logistics support strategy
  - Commercial vs. non-developmental vs. Developmental items
  - Challenging contractor claims
  - Access versus delivery of technical data
  - Use of DID and CDRL
  - System Technical Support (STS) contracts and work directives
  - Reverse engineering of technical data

Recommendations:

- The IPT concluded with a risk analysis method, which attempts to answer the key points identified above. The decision matrix and process chart is to be utilized to determine requirements for data rights.
- The paper goes into great depth as to the necessity of technical data in order to implement logistics support strategy. The decision to procure technical data needs to be accomplished early in the acquisition process.
- Technical data rights can be costly to procure. The risk needs to be assessed to determine the level of proprietary information and whether or not the government would have to be locked into one contractor for the life of the system or systems.
- The decision matrix is a powerful tool to be utilized in the acquisition process and the timely and accurate determination to procure technical data rights can save a program significant dollars.

Research Topic: Logistics Support

Researcher: Mike Kinberg

Title: Life Cycle Sustainment Outcome Metrics

Author: Deputy Under Secretary of Defense for Logistics and Material Readiness

Publisher: DoD, 10 Mar 2007

Summary: This article defines Life Cycle Sustainment Outcome Metrics. The purpose was to address the goals for theses metrics (4) that should be initially set and refined throughout the development process of new acquisitions. 14 key Life Cycle Sustainment "enablers" which are to be considered throughout a programs' life cycle are defined as well. For collection and reporting purposes the four Material Readiness metrics' (KKP/KKS) goals and LCS enablers should be considered and incorporated.

Key Points/Recommendations:

- Materiel Availability (KKP)
- Material reliability (KSA)
- Ownership Cost (KSA)
- Mean Down Time
- Performance Based Logistics (PBL)
- Corrosion Prevention
- Item Unique Identification (IUID)/ Serialized Item Management (SIM)
- Technical Data/ IETM
- Condition Based Maintenance (CBM+)
    - Prognostics & Diagnostics
    - Reliability Centered Maintenance
- Continuous Process Improvement (CPI)
- Title 10 Requirements/ 50/50, Partnering
- Depot Maintenance Plan
- Diminishing Manufacturing Sources and Material Shortages (DMSMS)/ Obsolescence Plan
- Training
- Integrated Supply Chain Management (SCM)
- Radio Frequency Identification (RFID)
- Predictive Modeling
- Long Term Performance Based Agreements (PBA)

Research Topic: Logistics Support

Researcher: Tim Carpenter

Article: White Paper – Ontologies for Data Mining and Knowledge Discovery to Support Diagnostic Maturation

Author: Timothy J. Wilmering, Boeing Company and John W. Sheppard, Johns Hopkins University

Date: August 28, 2006

Summary: This white paper explains an architecture framework for utilizing ontology's combined with machine learning to mature diagnostic models. The approach focuses on using the ontology's to restrict data sets in a meaningful way to manage the curse of dimensionality and still yield useful model revisions. While the research is still too early to report experimental results, theoretical analysis suggests the approach has promise.

Key Points/Recommendations:

Diagnostic Maturation Process: The paper point out that initial test and maintenance solutions that are deployed to support new complex systems are generally imperfect and are initially liable to contribute substantially to system ownership costs. It then explains that this is because development of effective health management solutions requires prediction of complex systemic interactions and the effect of presupposed external stimuli. The authors suggest that model-based approaches provide an excellent representational tool for developing and documenting system design choices that support traceability of design decisions and can be re-factored during the corrective action analysis and development process. This represents an iterative closed loop process of root cause analysis, corrective action deployment and reevaluation called a Maturation Cycle, or more formally in some circles, the FRACAS (Failure Reporting and Corrective Action System) process.

The author states that maturation is difficult because of two fundamental issues:

- The product data that is typically required for maturation analysis is generally stored in disparate heterogeneous data systems - this makes access, retrieval, and integration of the requisite information a costly and often incomplete process.
- There are several categories of analysis required to support maturation: correlation, root cause analysis, and analysis of the support system. The issue is that the data required to perform these analyses is scattered among many different data sources and systems.

He explains that Knowledge Discovery (KD) techniques may offer significant benefit to the diagnostic maturation process. The KD process is then outlined as follows: **Data collection and consolidation:** Identify relevant data and factors, find relevant data sources, locate data in sources and formulate queries; **Prepare Data (Data Integration):**

Develop data correlation keys, retrieve data, integrate data, clean data, examine data; **Data Mining:** Data mining is the heart of the KD process. Select the appropriate methods for pattern extraction from the large data sets collected in the previous step, then apply any of numerous classification and pattern matching techniques; **Interpretation and Evaluation (Analysis of results):** Interpretation of results may employ further data reduction mechanisms, use of visualization techniques, or other methods to enhance presentation and interpretation of the data for human consumption.

Research Topic: Logistics Support

Researcher: Linda Banner-Bacin

Title: Part 3; Guidance to the Application of LSA to Software Aspects of Systems

Author: Ministry of Defense UK Defense Standard 00-60 (part 3)

Date: 24 September 2004

Summary: The document is the United Kingdom Ministry of Defense (MOD) guidelines for Logistics Support Analysis for application to software aspects of systems. The document identifies steps that should be taken to ensure continued use of software in a system. Software support is considered an essential element and the support will vary depending on the type of system and much software is procured. It goes on to say "experience has shown that for many systems the cost of initial software development has been greatly exceeded by the cost of supporting the software during the system operational life".

The document identifies key factors that need to be supported during the development and life cycle phase of Software Supportability. The key factors are:

- Change Traffic: measure of the rate at which S/W modification is required
- Safety Integrity: determined by the safety criticality of the functions that it provides. Criticality is the anomalies in the system with varying severity.
- Expansion Capability: degree which S/W can be modified, physical limitations are included such as available memory, processor performance, mass storage capacity, input/output bandwidth
- Fleet Size and Disposition: number of equipments in use, locations at which software support is conducted, large fleets will have higher equipment usage
- Modularity: low-level structure of software design, dependent on engineering design up from
- Size: metrics are available to quantify software size. Size influences supportability and change traffic expected.
- Security: Classification of S/W items will be dependent on application
- Skills: S/W modification will require personnel with S/W engineering skills,
- Standardization: applies to the computing environment which the S/W executes
- Technology: considered with the S/W engineering methods and tools used in development and implementation, with the H/W and S/W of the host and target platforms
- Tools and Methods: selection of tools and methods is dependent on technologies used to develop and implement the system
- Documentation: includes all records, electronic or hardcopy that relate to the requirements, spec analysis, design, implementation, testing and operation

Logistics Supportability Analysis for software is done in the pre-design phases of the project for supportability and Life Cycle Costs benefits. The development of supportability will be direct results of equipment technical requirements that include the functional requirements, test, testability, system monitoring, and data recording and non-functional requirements such as growth capacity.

Application of LSA tasks in the early project phases will be application software and the system architecture. The reason for early LSA tasking is to clearly identify the requirements, constraints and issues requiring further analysis.

Figure 2 identifies both requirements and analysis issues that need to be developed during the early phase. Example: Prime Equipment Technical Requirements include Functional aspects such as Built-in test, diagnostics and monitoring, recovery, reconfiguration, degradation, mode reversion, data recording/access/upload/download. Non functional aspects: growth capacity, memory, processing, data communications.

Development standards include design, development methods, tolls, requirements capture, systems analysis, design, code, test, implementation standards, language, data communications, firmware, security, documentation, Management: project, quality and CM standards and last Support system requirements that include readiness/responsiveness, release frequencies, intellectual property rights, use of customer resources and support concepts.

The document identifies the use of Failure Modes Effects and criticality Analysis (FMECA) and Reliability Centered Maintenance (RCM) to Software. FMECA analysis is applied to the overall equipment and should identify particular failure modes associated to the functionality provided by or dependent on software. Although RCM is not applicable in the direct sense as failures will be unintended, RCM may identify the safety integrity level.

Process models should be developed for each deliverable and take into account support concept constraints and requirements.

Software support tasks for the system operation will respond to software-induced failures during the mission. The tasks fall into 2 categories: action to reload/restart software to bring the system back to operating capability and the second is to record information about the system condition/configuration at the time of the failure. Post mission tasks include data extraction, fault investigation and functions such as sanitization of classified software data. When software has been modified it suggested that a level of repair analysis (LORA) be done to ensure trade-off analysis of maintenance level for hardware and software support has been addressed.

The documents further outlines issues such as location of support, Software Support Documentation, Configuration Management, Post Mission Recovery, System Support Package Component List that includes development platform, test equipment, facilities, hardware and software tools, documentation and source code, Licenses, design rights, replication and transfer, processes, procedures, controls, consumables, parts, training and skills and personnel requirements.

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Eric Vasquez

Title: DoD Architecture Framework

Author: Steven H. Dam, Ph.D.

Publisher: 2006, SPEC, Marshall, VA

Title: Model-Based Systems Engineering: A New Paradigm

Author: Jerry Fisher, Editor

Publisher: FALL 1998, INCOSE, Seattle, WA

Summary:

- Model-Based System Engineering (MBSE) is a modeling approach to system engineering. This relatively new discipline strays from the conventional document centered approach towards one that is model driven for developing, designing, and verifying a system. The older document driven approach produced physical artifacts such as trade studies, specifications, and test plans. These documents were the primary means for communicating requirements to the system designers. This communication method led to incomplete, incorrect, or sometimes unnecessary requirements, which then led to design errors. The new MBSE approach addresses this issue by translating abstract definitions and statements into a central design repository and relaying information back to the designer in a structured language that can be utilized to trace problem solutions to derived requirements.

  MBSE primarily defines a set of entities, relationships, and attributes for the system and translates these language elements into graphical views, diagrams, and/or drawings. The three primary model types that are required for specifying the system architecture are the control model, interface model, and the physical architecture model. The control model is combined with the interface model via a special case input called a trigger, which leads to the creation of a behavior diagram. The behavior diagram is then used to verify the logical correctness of the system.

  The advances in technology and personal computers have made it possible to completely design the system by using Computer-Aided System Engineering (CASE) tools. CASE tools are the primary means for employing MBSE. This allows the developer to upload all aspects of the system requirements, behavior analysis, architecture analysis, and verification into a main system definition repository. Information can easily be pushed or pulled from this repository in the form of source documentation, diagrams, graphical constructs, simulation models, or even DoDAF views. The utilization of a CASE tool for MBSE makes it very

simple for the system developer or designer to quickly access any aspect of the system domain.

Key Points:

- MBSE is a superior approach for system engineering over the document driven approach. All aspects of the system can be uploaded into a main repository. Changes or modification in any phase of system developments are automatically reflected across all system engineering domains (requirements, behavior, architecture, and verification/validation) in the repository. CASE tools are the primary means for employing MBSE.

Applicability to Capstone Project: MBSE is the primary technique/method that we are using to verify our AAW thread example. The CASE tool that we are utilizing for our project is CORE. This project will verify the applicability and advantages of employing MBSE in creating complex combat system architectures.

Recommendations: The limitations for employing the MBSE approach primarily relates to possessing the necessary technology for the CASE tool, gaining access to a CASE tool that utilizes MBSE (such as CORE), and obtaining the necessary expertise and training to effectively utilize the CASE tool. If a program can overcome the tool accessibility and expertise/training barriers, then MBSE is the optimal approach for system engineering and should be implemented for all phases of system development.

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Tan Pham

Title: Modeling and Simulation (M&S) Strategy
Author: LT COL Posadas
Publisher (Date): Lesson briefs from LT COL 'Serg' Posadas, USMC (Fall AY 2008: 24 September 2007 – 10 December 2007)

Summary: Modeling and Simulation (M&S) is the method for verifying and validating the design of a system. The combat system methodology developed in SI 0180 (Naval Postgraduate School Capstone course) uses multiple levels of successfully predict the outcome of a Test and Evaluation (T&E) event. Models are developed to represent elements in an architecture and put into mathematical simulations to produce an output. In Figure 1 (Model Formation), inputs consist of variables, parameters, constraints, and assumptions. This input is then put into a mathematical model, which can comprise of equations, simulations, or spreadsheets. The resulting output is a measure of the input being used in the model.

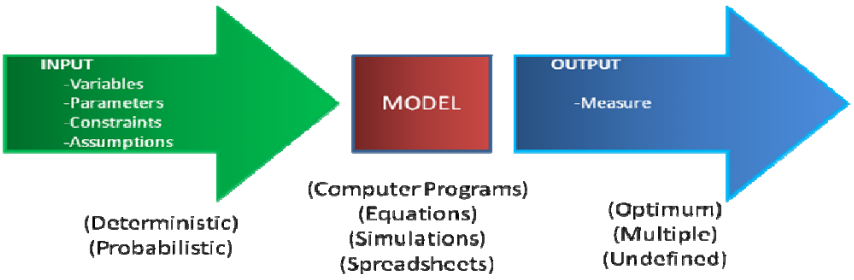


Figure 1. Model Formulation (from Posadas 2007). This is the process of using given inputs to create a model that dispenses a measurable output.

M&S strategy for the Capstone course will be comprised at two levels; the first is the high level model of whether the probability of success (P[success]) can be reached at 0.9. Based on the below first-level model, the initial requirements fail to produce a combat system with P(success) equaling 0.9. See the following algorithms for the breakdown.

GOAL: P(success) = P(AAW) = 0.9

Probability Limited Area Defense = P(LAD) = 0.9

Probability Self Defense = P(SD) = 0.9

Probability Surveillance P(S) = 0.9

P(LAD) * P(SD) * P(S) = P(AAW)

0.9 * 0.9 * 0.9 = 0.729 value too small (failed survivability)

0.729 < 0.9

P(LAD)	P(SD)	P(SURV)	P(AAW)	
0.9	0.9	0.9	0.729	fail

Table 1. Level One Model. This is a high-level model using equations to form a hypothesis.

Steps in Modeling begin with identifying (defining) a problem. From there, a model will need to develop to represent the process. After that, the IPT will acquire input data from other Capstone IPTs. To formulate a solution, the model is developed and tested. Analyzing data will interpret the results to see if the originating requirements for the Capstone project have been met.

Simulation of the models utilizes assumptions. The assumptions M&S has formulated are what the platform will be, type of targets, and type of weapons. The most common platform servicing the Naval Fleet is the Aegis Combat System Baseline 5.3.8. The weapon system provides a means of defending own ship against subsonic and supersonic targets using medium range weapons (SM-2 Blk IIIB and ESSM).

M&S in the second level will utilize the ExtendSim software. This software provides the capability to design scenarios that will stress the weapon system architecture. Scenarios are designed to represent real-life tactical operations of the combat system.

Key Points: Modeling and simulation is used when a real system is difficult to grasp in its entirety. It is extremely helpful when dealing with the development of a new design or changes to an existing design. Modeling is a mathematical representation of the system. Simulation is a model of the actual system and an abstract representation of real-world process.

Applicability to Capstone: M&S is the key to successful prediction in the design and development of a naval combat system. The tools used to model the combat system in multiple layers of defense demonstrate the fidelity and robustness of the combat system. The results of simulations will be verified versus the initial requirements.

Recommendations: A recommendation to using modeling and simulation is to fully understand the flexibility of the tools available. Simulations can handle large and complex systems, but only if the user takes advantage of the software tools. Developers must also have a full understanding of the requirements to every degree. This means that the design should be built in increments of succession (similar to spiral development). This will allow for additional growth of the system upon proven combat system baselines.

Research Topic: Model-Based System Engineering (MBSE)

Researcher: James Kong

Title: ExtendSim User Guide

Author: Imagine That Inc.

Date: 2007, Imagine That Inc., San Jose, CA 95119

Summary:

- ExtendSim is a user-friendly tool for modeling and simulating processes. It helps you understand complex systems and produce better results faster. Modeling with ExtendSim can:
  - Predict the course and results of certain actions
  - Visualize your processes logically or in a virtual environment
  - Identify problem areas before implementation
  - Explore the potential effects of modifications
  - Confirm that all variables are known
  - Optimize your operations
  - Evaluate ideas and identify inefficiencies
  - Understand why observed events occur
  - Communicate the integrity and feasibility of your plans

ExtendSim dynamically models continuous, discrete event, discrete rate, and mixed-mode systems. Systems or processes are simulated by creating logical representation of a model using libraries of many pre-designed building blocks that allow you to build models rapidly. The given building blocks can be customized to replicate a specific model design. Additionally, ExtendSim has the ability to create new building blocks with custom dialogs and icons. New building blocks can be saved into libraries and easily reused for other models.

Using ExtendSim, model properties (item with attributes and values) are adjustable while simulation is running. This feature is critical for items with dynamic properties (properties that are updated and changes based on the outcome of the processes within the model) commonly found in discrete models. The properties may be updated and changed by a decision event, an equation function output, and random probability assignment to the values of the attribute.

ExtendSim presents the results and in-depth analysis of a simulation in customizable reports and plotters graphs.

Key Points:

- Most systems are composed of real-world elements and resources that interact when specific events occur (Discrete event systems and processes). ExtendSim's Item library simulates those systems using blocks that mimic industrial and commercial operations and timing that represents the actual occurrence of events.

  Blocks from Item library are useful to create simulations of business operations, manufacturing processes, networks, service industry flows, information processing, material handling, transportation systems, and so forth.

- Discrete event systems have several things in common:
  - They involve a combination of elements such as people, procedures, materials, equipment, information, space, and energy (called *items* in ExtendSim) together with system *resources* such as equipment, tools, and personnel.
  - Each process is a series of logically related *activities* undertaken to achieve a specified outcome, typically either a product or a service. Activities have duration and usually involve the use of process elements and resources.
  - Processes are organized around *events*, such as the receipt of parts, a request for service, or the ringing of a telephone. Events occur at random but somewhat predictable intervals and can be economic or noneconomic. Events are what drive most businesses.

Applicability to Capstone Project: ExtendSim is M&S IPT's primary modeling and simulation tool to analyze and verify the RIPT and AAW IPT's requirements and specification. Our overall AAW mission concept of operation (Self-Defense, Limited Area Defense and Surveillance) can be model as a discrete event. ExtendSim has the features to module our combat system.

Recommendations: Expert or self-training in ExtendSim is required to overcome the new software application learning curve. Although ExtendSim was designed to be user friendly and includes a users guide, but many of its features are not explain thoroughly. Brief description of the purpose of each building block is given but a new ExtendSim user will have to experiment with the properties of building blocks to fully understand its capability and limitations.

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Armando Valdez

Title: DoD Architecture Framework

Author: Steven H. Dam, Ph.D.

Publisher: SPEC, 2006, Marshall, VA

Title: CORE® 5, Systems Engineering Guided Tour

Author: Vitech Corporation

Publisher: Vitech Corporation, 1992-2007

Summary: Based on research conducted by Professor Dam and documented in his book, the members of Cohort 6 elected to use the Model Based System Engineering (MBSE) technique to develop the methodology for this project. Having selected MBSE, the next step in developing the methodology was to select a tool to implement the technique.

PROs

- There are several tools to choose from such as Proforma, CORE, Metis, System Architect, Rational Rose, NetViz, DOORS/TAU, SLATE, ARIS, and Rhapsody. All of these tools have some capabilities to produce DoDAF products; some even have special facilities, templates, reports, or user interfaces tailored to the DoDAF. Many have built-in modeling and simulation capabilities (e.g., Proforma, CORE, SLATE). Many use UML as a basis (e.g., Rational Rose, TAU, Rhapsody). Of these, CORE implements MBSE.

- Professor Dam, developed seven evaluation criteria for selecting the correct tool. In the table below, DOORS/SA, CORE, and Rational Rose are compared using the seven criteria. The table below is reproduced from Figure 5-29, from Professor Dam's book.

Criteria	**DOORS/SA**	**CORE**	**Rational Rose**
Supports your chosen methodology	IDEF	MBSE	UML
Provides several integrated functions	Yes	Yes	Yes (software focused)
Employs rule-based standards	Yes	Yes	Yes
Enforces Consistency	Yes	Yes	Yes

Uses an integrated, common database	No	Yes	Yes
Permits simulation capability	No (except when adding a third tool)	Yes	Yes (with software coding)
Facilitates exporting data and products	Yes (but limited)	Yes	Yes
Enables flexible reconfiguration	Yes (but limited to 'User Properties file)	Yes (using built-in schema extender)	No?
Applies to multiple lifecycle phases	Yes	Yes	No?
Supports multiple disciplines	No (Architecture focused)	Yes (Architecture to Software, CM, T&E)	No (software focused)

- CORE was designed as a system-engineering tool that focuses on functional analysis. CORE provides an integrated design repository that enables traceability between requirements, functional models, and system design elements. CORE's database schema is modifiable to customize the tool to support customer needs and facilitate tool integration. The executable diagrams provide a rapid way to verify the logic of the design. CORE has a tailored DoDAF schema and reports for just about all the DoDAF products. CORE can generate Enhanced Functional Flow Block Diagrams (EFFBD), FFBDs, N2 diagrams, and IDEF0 diagrams.

CONs

- There is very little User Manual documentation available with CORE. There are no step-by-step instructions on how to develop the DoDAF views. I contacted the developer of CORE, Vitech Corporation, requesting additional User documentation that provides a tutorial for the DoDAF schema and was told that there they had not developed a DoDAF tutorial. The bottom line here is that the lack of documentation makes the learning curve steeper than it needs to be.

  The CORE software is only available via a remote link to SE lab computers at NPS. A VPN connection must be established in order to access the NPS computers. VPN software must be installed on our computers, which means that

we can't access the NPS computers from our NMCI workstations. This severely limits the amount of collaboration between M&S IPT members developing the DoDAF products in CORE.

Conducted research into whether or not CORE and Extend software programs are available and approved for installation on our NMCI computers. I checked to see if the Department of the Navy Application Database Management System (DADMS) lists both of these software programs. They are approved (with a waiver) for NMCI installation but since we do not have a PHD sponsor for either of these programs we can't get NMCI to push the software to our workstations. Bottom line here is that we can't have these programs installed on any government owned computer at PHD including developer stations.

Key Points:

- Based on a comparison of DOORS/SA, CORE, and Rational Rose it becomes clear that CORE offers more benefits than the other software tools.
- CORE implements MBSE and is an SE tool that focuses on functional analysis, provides an integrated design repository that enables traceability between requirements, functional models, and system design elements.
- Most DoDAF products can be generated by CORE.
- There is limited User Manual documentation for CORE.
- Can't access NPS computers from our NMCI workstations (no collaboration possible while on the PHD campus).

Applicability to Capstone: The discussion topics are very applicable to the Capstone Project.

Recommendations: Even though there are some limitations associated with CORE, we should still press forward with using the program to capture and develop our AAW architecture requirements and functionality. We can then use CORE to generate all required diagrams and DoDAF products and diagrams.

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Alan Wellesley

Title: Model-Based Systems Engineering: A New Paradigm

Author: Jerry Fisher

Publisher: International Council On Systems Engineering (INCOSE) Insight, Volume 1 Issue 3, Fall 1998

Summary: This article kicks off the fall 1998 quarterly issue of INCOSE publication, which focuses on "Model Based Systems Engineering: A New Paradigm." As such it introduces several articles on MBSE – so it highlights some of the benefits of MBSE and shortfalls of the "paper-driven" approach to SE.

Key Points:

- Ambiguous and incorrect requirements have long been recognized as primary causes of design errors.
- Intuitively, it should be known that system design using a model-driven paradigm is going to save money and improve quality.

Applicability to Capstone Project: Some of the benefits you will realize with a model-driven approach to systems engineering are:

- All requirements, and the rationale behind them, will be accessible to the designers.
- The completeness of your design can be assessed by tracing the requirements to functions and their allocation to physical components.
- All views of the requirements are saying the same thing. There are no disconnects among the representations of the data.
- The corporate memory of a project can be retained when the staff is reassigned. Some projects may last ten years, and have several systems engineers on the team.
- A simulator must be part of the CASE tool and, therefore, be instrumental in executing the actual behavior.

There are three primary problems with the paper-driven approach:

- Specifications are often written after the design is complete and merely used to record the results.
- Written words tend to be ambiguous.
- Requirements generation are perceived as just words. No one has the time to learn other, more meaningful, methods to fully define the requirements. Within the last decade, and primarily as a result of the expanded capabilities of personal computers (PC), it has become possible to design a system completely using a computer- aided systems engineering (CASE) tool.

Recommendations:

- Good reference to justify use of MBSE
- Gives advantages of MBSE and disadvantages of paper-driven approaches
- Touches on problems with requirements ambiguity and incorrectness, and that MBSE will save money and improve quality.

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Alan Wellesley

Title: The INCOSE Model Driven System Design Interest Group

Author: Howard Lykins and Bob Cohen

Publisher: International Council On Systems Engineering (INCOSE) Insight, Volume 1 Issue 3, Fall 1998

Summary: This article comes from the fall 1998 quarterly issue of INCOSE publication, which focuses on "Model Based Systems Engineering: A New Paradigm." This article describes the focus and activities of the Model Driven System Design Interest Group within the INCOSE organization.

Key Points:

- Modeling tools can assist in bridging gaps between engineering disciplines and between technical and non-technical stakeholders in the development process.
- In the future, automated tools should be able to help integrate disciplines by maintaining "meta-data" that describes the information maintained by various models. This meta-data will identify information available in the models of one discipline (such as electrical engineering) needed by engineers in other areas (such as S/W or Mechanical engineering).
- In addition, model-based tools will abstract, from detailed engineering models, the information that non-technical decision-makers need to support the engineering process.

Applicability to Capstone Project: Modeling tools and MBSE approach assist in bridging the divide between engineering disciplines and logistics / supportability teams.

Recommendations: Use this reference to support argument for MBSE and as a means to integrate engineering and logistics efforts.

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Alan Wellesley

Title: Information Models as a Prerequisite to Software Tool Interoperability

Author: Byron Purses and Loyd Baker

Publisher: International Council On Systems Engineering (INCOSE) Insight, Volume 1 Issue 3, Fall 1998

Summary: This article comes from the fall 1998 quarterly issue of INCOSE publication, which focuses on "Model Based Systems Engineering: A New Paradigm." This article describes software tool interoperability problems and solutions during systems development.

Key Points:

- Presents key reasons to use / advantages of MBSE.

Applicability to Capstone Project: In the Spring '98 ***INSIGHT***, the INCOSE Model Driven System Design (MDSD) Working Group discussed reasons why program managers are adopting a model-based approach to systems and business process engineering. The key reasons presented were:

- Improved communication of ideas and concepts between project personnel when using information models over the traditional document-based approaches. As we all know, a "picture" or "graphic" provides greater visibility and quicker understanding over textual descriptions.
- Improved analysis and verification capabilities through automated interrogation of the system information models.
- Ability to automatically generate, and re-generate, documentation based on the semantic rules embedded in the various kinds of system/ process information models.

Recommendations: Good reference to justify use of MBSE.

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Alan Wellesley

Title: The Benefits of Model Based Engineering

Author: David W. Oliver

Publisher: International Council On Systems Engineering (INCOSE) Insight, Volume 1 Issue 3, Fall 1998

Summary: This article comes from the fall 1998 quarterly issue of INCOSE publication, which focuses on "Model Based Systems Engineering: A New Paradigm." This article describes benefits of MBSE.

Key Points:

- Presents key reasons to use / advantages of MBSE.
- Provides an estimated cost improvement from use of MBSE product development.

Applicability to Capstone Project: MBSE must do the system job better, reduce risk, save money in product or service development, and better match product to marketplace. This can happen because the MBSE process works better and is much more efficient and cost effective than the present use of vernacular text. This article describes the benefits of MBSE and estimates the magnitude of improvement possible by reducing the work of creating requirements, tracing them, creating designs, calculating system performance of designs, transforming information for design engineering and management, evaluating requirements changes, and generating test and validation scenarios.

Magnitude of Benefits. Based on actual projects seen in disciplines like CAD/CAM, integrated chip design, and circuit board design, where work from concept analysis through detailed design has been automated, the estimated cost improvement of carrying out model based product development ***can be as large as 20 to 1.***

Recommendations:

- Good reference to justify use of MBSE.
- Gives advantages of MBSE and provides cost improvement estimation from using MBSE.

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Alan Wellesley

Title: Lessons Learned Applying Model-Based System Engineering Methods to a Strategic Planning Activity

Author: Loyd Baker, Jr.

Publisher: Vitech Corporation, 1996

Summary: This document examines a strategic planning effort at the Savannah River Nuclear Site, which used a MBSE approach to develop a plan for the disposition of stored nuclear waste material. In the beginning, many participants thought use of SE methods was overkill. This paper presents the lessons learned during the effort.

Key Points:

- The lessons learned using this model-based approach were as follows:
  - Models provide an improved view of traceability and its evolution
  - Graphical presentation of process flows enable group participation in concurrent engineering
  - Models provide improved insight in the derivation of interfaces
  - Automated documentation from models produces up-to-date documentation on demand at low cost

Applicability to Capstone Project: Very supportive of MBSE and provides advantages of this approach.

Recommendations: None

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Alan Wellesley

Title: Systems Engineering & Architecting with CORE®: An Overview for University Program Instructors

Author: Vitech Customer Support

Publisher: Vitech Corp © 2007 / August 2008

Summary: The objectives for this briefing are to familiarize the user with key concepts of Model-Based Systems Engineering (MBSE) and introduce capabilities of CORE

"Common SE 'Tool Suite' Architecture" - Multiple products utilizing independent databases forces extraordinary data management – and complicates the original SE effort.

"Preferred SE Tool Architecture" - Integrated, consistent analysis: complete specifications, project documentation, queries and models

Key Points:

- As expected, very "pro" Vitech / CORE.
- An overview for university program instructors after all.

Applicability to Capstone Project: Describes CORE's capabilities and the advantages of MBSE.

Recommendations: None

Research Topic: Model-Based Systems Engineering (MBSE)

Researcher: Alan Wellesley

Title: Model Based Systems Engineering *and How it Aids DoD Acquisition & Systems Engineering*

Author: Dwayne Hardy

Publisher: Enterprise Development Systems and Software Engineering Directorate Office of the USD (A&T), 24 OCT 2006

Summary: Presentation with objectives to provide: Vision for Model-Based Systems Engineering (MBSE) (Primarily from INCOSE's perspective) and How MBSE enables better and faster system trade-offs and related DoD acquisition decisions (From my perspective)

Key Points:

- MBSE enhances the ability to capture, analyze, share, and manage the information associated with the complete specification of a product, resulting in the following benefits:
  - Improved communications among the development stakeholders.
  - Increased ability to manage system complexity by enabling a system model to be viewed from multiple perspectives, and to analyze the impact of changes.
  - Improved product quality by providing an unambiguous and precise model of the system that can be evaluated for consistency, correctness, and completeness.
  - Enhanced knowledge capture and reuse of the information by capturing information in more standardized ways and leveraging built in abstraction mechanisms inherent in model driven approaches.
  - Improved ability to teach and learn systems engineering fundamentals by providing a clear and unambiguous representation of the concepts.

- Identifies DoD acquisitions Top 10 emerging systemic issues and touches on how MBSE can help solve these.

Applicability to Capstone Project: Very supportive of MBSE.

Recommandations: None

Research topic: Model-Based System Engineering (MBSE)

Researcher: Mindy Wentland

Title: Engineering Complex Systems with Models and Objects

Author: David W. Oliver, Timothy P. Kelliher, James G. Keegan, Jr.

Publisher: McGraw-Hill Companies (January 1997)

Summary:

- Describes how to combine text descriptions and rigorous modeling to analyze and describe large or small complex systems. One thesis of this book is that modeling results in higher quality systems, designed and produced at lower cost and in a shorter time, with a better fit to the market. A second thesis of the book is that with modeling the system specification can be executed to show what will occur and can be transformed efficiently and rigorously into the several different languages and forms useful to both the system stakeholders and to the design, engineering, and manufacturing disciplines. A third thesis of the book is that with the same modeling applied to systems engineering itself you get a well defined discipline, improved capability to train, and essential definitions needed for building automation and infrastructure for efficient and creative systems engineering.
- Systems engineering information needs to be rigorously transformed to the multiple different models, notations and views of the downstream engineers who create designs. The Gap is the void between needs expressed in informal, natural language and component specifications described in the multiple engineering notations. To date this gap has been bridged by good systems engineering practices and by hard work. This work results in huge text documents detailing the component specifications for designers. Engineers in each downstream discipline must read and interpret the text, transform it into their own models and terminology, and then enter it into their computer tools. They must remove the ambiguity and inconsistencies between what has been written and what they know will work correctly. Clearly, this process is time-consuming and error-prone.
- Modeling can fill the gap. Capture of the modeling information for modern complex systems is important both for productivity in the engineering work and for checking information for inconsistencies, omissions, and errors. The gap for the engineering of systems can be filled by extending the modeling techniques that are applicable to the definition of systems described in this book.
- Rigorous, executable models of behavior (what things do) and structure (how things are built) mean the capture of system requirements and specifications in models that are computer executable and unambiguous. It is possible to use automatic transformations of the system models into exactly the views and notations needed by the supporting engineering disciplines. This rigor cannot be obtained with non-executable specifications written in natural language text alone. That is not to say that text is unimportant. It is used to accompany the models and provide explanations of them.

- Modeling is used to reduce the time and effort expended by engineers shortening the design cycle time. It is used to check the information for consistency and completeness reducing the error rate. It is used to preserve the current engineering results for use during later maintenance, product upgrade, or product replacement efforts. It is used to describe unambiguously; every symbol and number such that each has one and only one meaning. The models ensure that at the end of the process all necessary information is available and correct.

Key Points:

- Elements of Behavior
  - Functions
    - Accept inputs and transform them to outputs
      - Duration
        - Execution time
    - Generation Rate
      - The speed outputs are generated
    - Consumption Rate
      - The speed inputs are processed
  - Define inputs and outputs to functions
  - Control Operators
    - Define order/sequence of functions
- Ways to model behavior
  - Functional Flow Block Diagrams (FFBD)
    - Including input/output information in an FFBD yields a behavior diagram.
  - State Charts
    - Can be used when complex trade-offs are not needed.
    - Hierarchical
    - Well-defined relationship with functions
    - "AND" states represent concurrency.
      - Composite states, which group together several sub states into a single entity.

Applicability to Capstone Project:

- Provides rationale for Model-Based Systems Engineering.
- Provides a "how-to" for creating behavior models for software functions in both the architecture and repository.

Recommendations:

- Use modeling rationale to explain why MBSE is a better process.

1. AAW Architecture and SW IPTs use for building behavior models.
   - SW IPT use for building behavior models for architecture and repository.

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Tim Carpenter

Title: SIMBASE Executive Summary – Issue 1.0

Author: Dr. Timothy KING, LSC Group

Publisher: LSC Group, November 27th, 2007

Summary: This project report identifies the value of modeling and simulation techniques in a process known as either Synthetic Environment Based Acquisition (SeBA) or Simulation Based Acquisition (SBA) (SeBA/SBA). The SIMBASE project addressed the fundamental research question as to whether the modeling and simulation community should embrace product data standards as a necessary part of the infrastructure to make SeBA/SBA an effective and efficient enabler of acquisition

Key Points/Recommendations:

- The project has been able to achieve the following results:
  - Execution of a research and technology project, with strong linkage to business requirements and benefits
  - Evidence of the feasibility and requirements of SeBA/SBA
  - A robust approach to systematic enterprise integration
  - Application of several key technologies, including STEP, PLCS, UML (Unified Modeling Language), workflow/BPM (business process management), a synthetic environment and logistics modeling

- The project resulted in the following deliverables:
  - Requirements, architecture and case study approach for the SIMBASE capability;
  - Designs for the SIMBASE repository and workflow capability;
  - Designs for systems engineering, logistics engineering and synthetic environment toolsets and their integration into the SIMBASE repository;
  - Developed software to implement the designed capability;
  - A one-day demonstration of the SIMBASE capability to an invited international audience.

The project worked with the following core definitions:

- Synthetic Environment Based Acquisition is the consistent and coherent application of modeling, simulation and synthetic environment technology within, and across, both acquisition phases and programs to facilitate the attainment of Smart Acquisition goals of faster, cheaper and better.
- Simulation Based Acquisition is an iterative, integrated product and process approach to acquisition, using modeling and simulation, that enables the war-

fighting, resource allocation, and acquisition communities to fulfill the war-fighter's materiel needs, while maintaining cost as an independent variable over the system's entire life cycle and within the DoD's system of systems.

- Both of these terms have become better understood and richer over time, resulting in an effective equivalence between the two. Thus, the SIMBASE project has used the term "SeBA/SBA" to cover the target capability.
- The identified context for SIMBASE is the acquisition process that the UK MoD calls "Smart Acquisition", which embodies principles that are applicable to ongoing acquisition transformation in all WEU nations. This process encompasses true affordability, which is faster, better, cheaper, more integrated and in a context of managed risk.
- The Smart Acquisition challenge is to achieve holistic, through-life systems engineering in the face of ever increasing complexity and interdependence of solution functions. The consequences are complexity and interdependence both in technical and organizational aspects of the acquisition enterprise. In turn, demonstrating affordability becomes ever more difficult. The accurate assessment of affordability is only possible with the right information of the right quality.
- The SIMBASE participants identified several components to the holistic systems engineering process. ISO 15288 ("Systems life cycle processes") provides an overarching framework for these processes. Integrated logistic support (ILS) addresses the through-life support aspects of a complex system. The PLCS Application Activity Model (AAM) lays out the through-life support processes in the broader enterprise context. The Synthetic Environment Development and Exploitation Process (SEDEP) provided a systematic approach to developing synthetic environments.
- The bottom-line imperative is for projects such as SIMBASE to provide solutions addressing real business objectives, the most significant of which is ensuring the reduced total cost of ownership for the long-life, high-value assets that are typically the focus of defense acquisition programs.

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Alan Wellesley

Title: Survey of Model-Based Systems Engineering (MBSE) Methodologies

Author: Jeff A. Estefan

Publisher: Jet Propulsion Laboratory, California Institute of Technology

Pasadena, California, U.S.A., May 23, 2008

Summary: The purpose of this report is to provide a cursory description of some of the leading Model-Based Systems Engineering (MBSE) methodologies used in industry today. It is intended that the material described herein provides a direct response to the INCOSE MBSE Roadmap element for a "Catalog of MBSE lifecycle methodologies" [1].

A cursory review of the following "**Leading MBSE Methodologies"** is provided**:** IBM Telelogic Harmony-SE, INCOSE Object-Oriented Systems Engineering Method (OOSEM), IBM Rational Unified Process for Systems Engineering (RUP SE) for Model-Driven Systems Development (MDSD), Vitech MBSE Methodology, JPL State Analysis, and Dori Object-Process Methodology (OPM)

Key Points: In this report, a methodology is defined as a collection of related processes, methods, and tools [2]. A MBSE methodology can be characterized as the collection of related processes, methods, and tools used to support the discipline of systems engineering in a "model-based" or "model-driven" context.

Applicability to Capstone Project:

- Defines "methodology"
- Defines "process," "method," and "tool"
- Provides good descriptions of how MBSE industry leaders do their job.

Recommendations: Very informative, especially the Vitech and INCOSE methodologies.

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Alan Wellesley

Title: SE practices for Describing Systems

Author: Object Management Group, INCOSE

Publisher: © 2006 Object Management Group/ 11 July

Summary: Provides the following

- MBSE Benefits
- SysML Description and its relation to UML
- An example about designing a distiller.

Key Points:

- SysML is a critical enabler of model driven SE
- SysML does not include timing, interaction overview, and communications diagram.

Applicability to Capstone Project: Support for using SysML.

Recommendations: None

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Alan Wellesley

Title: Lessons Learned Applying Model-Based System Engineering Methods to a Strategic Planning Activity

Author: Loyd Baker, Jr.

Publisher: Vitech Corporation, 1996

Summary: This document examines a strategic planning effort at the Savannah River Nuclear Site, which used a MBSE approach to develop a plan for the disposition of stored nuclear waste material. In the beginning, many participants thought use of SE methods was overkill. This paper presents the lessons learned during the effort.

Key Points:

- The lessons learned using this model-based approach were as follows:
  - Models provide an improved view of traceability and its evolution
  - Graphical presentation of process flows enable group participation in concurrent engineering
  - Models provide improved insight in the derivation of interfaces
  - Automated documentation from models produces up-to-date documentation on demand at low cost

Applicability to Capstone Project: Very supportive of MBSE and provides advantages of this approach.

Recommendations: None

Research Topic: Model-Based System Engineering (MBSE)

Researcher: Mindy Wentland

Title: Model Driven Engineering

Author: Douglas C. Schmidt

Publisher: IEEE Computer Society (Feb 2006)

Summary:

- Model-driven engineering technologies offer a promising approach to address the inability of third-generation languages to alleviate the complexity of platforms and express domain concepts effectively to develop Model-Driven Engineering (MDE) technologies
- Domain-specific modeling languages (DSMLs) whose type systems formalize the application structure, behavior, and requirements within particular domains, such as software-defined radios, avionics mission computing. DSMLs are described using metamodels, which define the relationships among concepts in a domain and precisely specify the key semantics and constraints associated with these domain concepts.
- Developers use DSMLs to build applications using elements of the type system captured by metamodels and express design intent declaratively rather than imperatively.

Key Points:

- The idea of using models to alleviate software complexity has been around for many years. However, researchers have largely applied models to selected elements of the development process, particularly structural and compositional aspects in the design phase and model checking and verification in the testing phase.
- Avoiding a one-language-does-all approach. This approach uses domain-specific modeling languages (DSMLs) to represent "aspects of interest" such data access, end-to-end message delay, and resource contention.
- Automated generation of partial implementation artifacts. The mapping between elements in a model and corresponding elements of implementation is well defined. Rather than being restricted to program skeletons, partial implementations also can include functionality and specifications for software.
- Large-scale systems inherently require the incorporation of legacy implementation assets. Reverse engineering is used to build models from existing source code. Many previous attempts to reverse-engineer models from source code have failed due to a lack in constraining aspects of interest. Model verification and checking. Developers can use static analysis as well as rapid-prototype generation in combination with runtime performance analysis to evaluate designs.

Applicability to Capstone Project: Will be used by SW IPT to model the AAW mission using the MBSE approach.

Recommendations: SW IPT to define approach of how MBSE applied to the AAW architecture.

Research Topic: Object Oriented (OO) methodologies

Researcher: Alan Wellesley

Title: A Process for Combining Object Oriented and Structured Analysis and Design (Article and Brief)

Author: Dale M. Rickman,

Publisher: Date 8/9/2000 Raytheon Systems Company, 1768 Business Center Drive, Reston, VA 20190; (703) 759-1216; dmrickman@raytheon.com

Summary: While Object Oriented (OO) methodologies have some advantages over structured methods, OO is not as mature as structured analysis and design and does not contain all of the tools/ techniques needed to support a large system design. By using both OO models and structured models (e.g., data flow diagrams, control flow diagrams, state transition tables) during systems analysis, a more complete understanding of the system requirements can be developed. During the design process, the software architecture components can be designed and built either as OO modules or structured modules depending upon the requirements of the module. Since both views of the system (OO and structured) have been built during the analysis phase, there is no "translation / conversion" from one methodology to the other.

Key Points:

- **Advantages and Disadvantages of Object Oriented Methods**
  - Object Oriented (OO) methodologies have two main strengths. First, they do an excellent job of supporting COTS and reuse. The OO approach inherently makes each software object a stand alone component that can be reused not only within this problem domain, but also in completely different problem domains.
  - The other main advantage of OO is the focus on data relationships. To develop a large system, the data relationships must be well understood.
  - A weakness is that OO methods only build functional models within the objects. There is no place in the methodology to build a complete functional model.
  - Another weakness of the OO methodology is in system modeling for performance and sizing. The OO models do not easily describe the communications between objects.

- **Advantages and Disadvantages of Structured Methods**
  - The main advantage of structured analysis is in the development of a complete system requirements model. Having a complete requirements model on one diagram helps ensure that all requirements are allocated to architecture components. The primary (fatal) flaw with structured

methods is that they do not readily support the use of COTS or reusable modules.

Applicability to Capstone Project: Combining models and approaches from both OO and structured methods in one process allows taking advantage of the strengths of both methodologies.

Recommendations: None

Research Topic: Object Oriented (OO) methodologies

Researcher: Alan Wellesley

Title: The A Process for Combining Object Oriented and Structured Analysis and Design, article and presentation

Author: Dale M. Rickman, Raytheon Systems Company

Publisher: Raytheon Systems Company, 8/9/2000

Summary: This article provides advantages and disadvantages of Object Oriented (OO) methodologies and Structured Methods and describes a process of how to combine OO into Hatley – Perbhai (H-P) to combine the Methods into one Process to Take Advantage of the Strengths of Both Methodologies.

Key Points:

- Object Oriented (OO) Methodologies have some Advantages over Structured Methods, but do not contain all of the Tools and Techniques Needed to Support a Large Information System Design
- Standard Structured Analysis Methods do not Support Reuse as Well as Object Oriented Methods
- Advantages and Disadvantages of Object Oriented Methodologies
    - Advantages: Supports Inclusion of COTS and Reusable Components; Supports Development of Reusable Components; Provides Insight into Data Relationships
- Disadvantages: Lack of a System Functional Model can lead to Missed Requirements; Difficult to Model System Performance and Sizing; and Little Support for Identifying the Objects for an Optimal Design (Difficult to "See" Interface Complexity)
- Advantages and Disadvantages of Structured Methods
    - Advantages: – Mature Discipline for Large Systems Design; Better Understood by Developers and Customers; Contract Requirements are Specified in Terms of Functions; and Provides Complete (Functional) Requirements Model
    - Disadvantages: **Does Not Readily Support COTS/Reuse (**Requirements Developed Top-Down Do Not Map Well to Reusable Components)

Applicability to Capstone Project:

- Provides support for SysML (OO), structured methods, & H-P
- Shows that a "hybrid" SE approach may be best.

Recommendations: This may describe / support our chosen methodology, or combination of methodologies.

Research Topic: Object Oriented (OO) methodologies

Researcher: Alan Wellesley

Title: Relationships between Common Graphical Representations in System Engineering

Author: Jim Long, President, Vitech Corporation

Publisher: Vitech Corporation**,** 2002

Summary: This article describes relationships between diagrams commonly used during the SE process to communicate functional and data requirements. This paper discusses the characteristics of each and shows how they are related.

Key Points: Over the past several years, systems engineers have evolved to a few graphical representations to present the functional and data flow characteristics of their system design. The most common of these are the Function Flow Block Diagram (FFBD), Data Flow Diagram (DFD), N2 (N-Squared) Chart, IDEF0 Diagram, Use Case, Sequence Diagram, Enhanced Function Flow Block Diagram, and Behavior Diagram (BD). Most of these graphical representations allow the engineer to decompose the functional and/or data models hierarchically. The objective of this paper is to analyze the representation capability of these graphic "languages" to see if there is a unifying view available.

The author did conclude, "… the Enhanced Function Flow Diagram and the Behavior Diagram features comprise a "parent/unified" set of graphical system representations. To achieve the same level of specification completeness, you would have to use an integrated set of the FFBD and one of the data models or augment the FFBD with a graphical representation of the data model…"

Applicability to Capstone Project: Not much. It is a good review of each diagram and its pros and cons / strengths / weaknesses.

Recommendations: None

Research Topic: Open Architecture (OA)

Researcher: Alan Wellesley

Title: SE ASNE Paper, Applying Open Architecture Concepts to Mission and Ship Systems

Author: John M. Green

Publisher: TBD

Summary: As the Navy moves towards the implementation of an open architecture acquisition strategy it is clear that a well defined surface domain combat system objective architecture is crucial to the development of the next generation of combat systems. This architecture construct is made more important by the trend towards integration of the C4ISR elements with the combat system. It is also important because it presents an opportunity to develop a truly network-centric architecture thus implementing a true at-sea sensor-to-shooter capability.

Presents an approach to domain modeling that will facilitate the development of a surface domain combat system objective architecture and its requirements. … the methodology uses a process modeling approach based on Finite State Machine theory implemented in an advanced simulation language such as Extend™.

The result is an architecture model that can be used to evaluate mission specific software product lines across diverse platforms as well as providing a framework for the integration of multiple mission areas with a single platform.

Key Points:

- While SOA has demonstrated value in the business world, it is not proven in the Combat Systems or C4ISR realms. The emphasis on this paradigm can force the architecture to take on a form that may not be suitable when real-time system response is required such as supporting as a weapons system error budget. (Taken from page 2)
- A Domain is a functional area shared across a group of products. (Pg 3)
- This leads to the concept of a software product line i.e., a family of products that share a common, managed set of features that satisfy specific needs of a selected warfare area. (Pg 3)

Applicability to Capstone Project: Supports MBSE.

Recommendations:

- Specifically, the model driven architecture approach has the following advantages: It is a formal method for tying the architecture requirements process to the architecture verification process; it is consistent with acquisition policy; and it provides a methodology to test Network Centric Operations concepts such as MDA, CMD, and TCT.

- The use of a simulation-based methodology will result in most of the requisite DODAF artifacts required for both requirements capture and the description of the system functional behavior. In addition, it supports the development of architectures that incorporate modular design and the identification of reusable and interoperable modules/applications **Article / Document:** Using Systems Engineering Standards In an Architecture Framework

Research Topic: Open Architecture (OA)

Title: Establishing a Naval Enterprise Reuse Repository: An Analysis of the Interaction between Engineering and Public Policy

Author: Scott Otto [ENCE 688N: MEPP Scholarly Practicum]

Publisher (Date): December 11, 2007

Summary: The United States Department of Navy (DoN) is facing a challenge to reduce shipbuilding costs and maintaining existing weapon infrastructures, DoN is adopting open architecture to reduce the life cycle costs of equipment, increase interoperability, reduce technical risk, and speed up improvements in capability. To promote the adoption of OA across the Naval Enterprise, the Assistant Secretary of Navy Research, Development and Acquisition (ASN-RD&A) created the Open Architecture Enterprise Team (OAET) and chaired by the Program Executive Office Integrated Warfare Systems (PEO-IWS). PEO-IWS is responsible for moving the various communities like Surface, Air, Submarine, C4I, Space, and Marine Corp to an open architecture environment. The most important principle to open architecture is the reuse of application software. The initial prototype repository established was a Software, Hardware, and Asset Reuse Enterprise (SHARE) for the Surface Domain.

Key Points: There are five key principles of Open Architecture that was established by Rear Admiral M.J. Edwards stated that "modular design and design disclosure, reusable software applications, interoperable war fighting applications, life cycle affordability, and competition & collaboration are requirements for all future projects done across DoN"[10].

- 1: Modular Design and Design Disclosure: The needs to build a modular design a disclosing data to permit evolutionary designs, technology insertion, competitive innovation, and alternative competitive approaches from multiple qualified sources.
- 2: Reusable Application Software: Software application needs to be identified or develop thru open competition that must satisfy the reusable application and operational requirements.
- 3: Interoperability Joint War fighting Applications and Secure Information Exchange:
- Joint War fighting is a must. Therefore weapons must be built to operate with all war fighting applications and ensure secure information exchange using common services, common war fighting applications, and information assurance as intrinsic design elements in order to achieve interoperability.
- 4: Life Cycle Affordability:

[10] Rear Admiral M.J.Edwards, 9010 Ser N6N7/5U916276

- To ensure life cycle affordability, DoN wants to implement Open Architecture within system design to mitigate COTS obsolescence by exploiting Rapid Capability Insertion Process/Advanced Processor Build Methodology.
- 5: Encouraging Competition and Collaboration: The Navy wants non-proprietary solution through open sources and development of alternative solutions. However, the overall goal is to lower cost and increase ideas.

Applicability to Capstone:

- Understand the basic concepts of existing OA infrastructure and the Navy mandated help eliminate the initial work process.
- Leverage the existing work on software reuse and expand on our what needed to be done to move the Navy in the right direction

Recommendation: SW and AAW IPT to use this as a basic foundation for establishing a good SW reuse and open architecture.

Research Topic: Open Architecture (OA)

Researcher: Mindy Wentland

Title: Architectural Principles of Open Architecture

Author: Eric M Nelson

Publisher: Unknown, 12 Dec 2007

Summary: Many of the Open Architecture (OA) requirements apply just as much to a Service-Oriented Architecture (SOA). The difference between them is primarily a matter of focus. The focus for SOA is across systems, while the focus of OA is within a system. At the highest level, SOA is defined in the OASIS SOA Reference Model as "... a paradigm for organizing and utilizing distributed *capabilities* that may be under the control of different ownership domains." Table 1 contrasts the two.

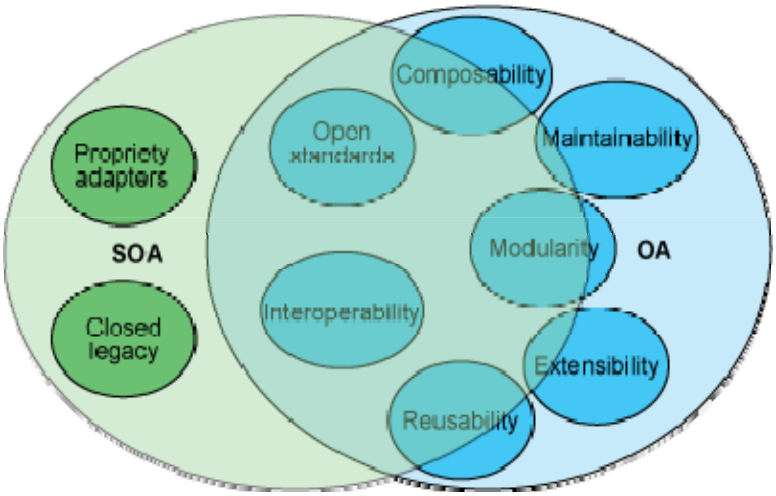
Table 1 – Differentiating SOA and OA

SOA Concerns	OA Concerns
Interoperability, both syntactic and semantic, among systems.	Open and modular design within a system.
Software architecture.	Software and hardware architecture.
A subset of all operations in the system are exposed as services.	The system and all its key components should be open.

Key Points:

- The differentiation is not absolute; both architectural approaches are concerned with interoperability. At the boundaries you get differences due to the different emphases. For example, a Web services adapter could be used to expose a closed system's capabilities to the network without making the system itself open. There is no way that the system itself could be considered open. On the other hand, a system that has no externally visible services could still be very open. In fact, a system that is open already has met many of the interoperability and composability requirements needed to create good services in an SOA.
- Figure 1 shows (not to scale) how SOA and OA share some concern with all the core OA requirements. In some cases, such as maintainability, software maintainability would apply to SOA, but hardware maintainability would not be relevant.

Figure 1 – OA and SOA overlap



Applicability to Capstone:

- It is very crucial to understand the important between Service Oriented Architecture and Open Architecture.
- They are both different and similar in many ways. In order to implement and consider SOA and OA into the design one has to fully understand its characteristics and behaviors.

Recommendation: SW IPT will address SOA and OA as an open literature of fact-findings and recommendation.

Research Topic: Open Architecture (OA)

Researcher: Mindy Wentland

Title: Naval Open Architecture

Author: Program Executive Officer Integrated Warfare Systems (PEO IWS)
Publisher: Defense Daily, August 2008

Summary: A Navy initiative for: A multi-faceted strategy providing a framework for developing joint interoperable systems that adapt and exploit open-system design principles and architectures.

Key Points:

- This framework includes a set of principles, processes, & best practices that:
  - Provide more opportunities for competition and innovation
  - Rapidly field affordable, interoperable systems
  - Minimize total ownership cost
  - Optimize total system performance
  - Yield systems that are easily developed & upgradeable
  - Achieve component software reuse

- The initial goal of OA, Rear Adm. Terry Benedict stated was to separate the software from dependency on specific hardware and migrate to commercial-off-the-shelf (COTS) computing environments. "The goal was to break the ties between hardware and software and then put the hardware and software on different refresh cycles. You want the software on the faster cycle but you want it to work on the latest available COTS hardware platforms," Benedict explained. "And then, periodically, you upgrade the hardware to take advantage of the technology leaps in COTS processing."
- Newer combat systems were designed from the beginning to be modular and to run on COTS processors and networks. The challenge has been to modernize the AEGIS fleet, which has an older architecture based on militarized equipment. Back in late 2004, the goal was set to move toward breaking the latest Aegis software baseline from its legacy hardware by 2008. That effort would be known as Advanced Capability Build (ACB) 08. "We have met this goal. Our next goal is to start developing modular software applications, where you break apart the monolithic software into modules with well-defined interfaces which will ultimately allow you to start competing across platforms for a common software object or software function," Benedict said. "Ultimately in 2012 and beyond you get to fleet-wide introductions of common combat system modules in Aegis, SSDS and DDG-1000."

Applicability to Capstone:

- Naval open architecture is the future of Navy's operation.
- Software IPT has to be address in the AAW thread.

Recommendation:

- Since it is a research study and NOA cannot be produce as a real life artifact. Therefore, NOA will be an important supporting literature and fact finding to the overall report.

Research Topic: Open Architecture (OA)

Researcher: Dave Chacon

Title: Open Architecture

Author: Dr. Wayne Meeks, PEO IWS

Publisher: PEO IWS, 29 Oct 2003

Summary: This presentation highlights why Open Architecture (OA) is required to improve warfighting capability and is fundamental to enabling future warfighting. It loosely defines OA as a Navy-wide technical and functional architecture using common and reusable software applications across platforms to make operational and programmatic improvements and increase affordability. The presentation identifies relevant OA standards and design guidance, and states that OA is required to enable SEA POWER 21, FORCEnet, joint interoperability, and affordable modernization.

Key Points/Recommendations:

- In-service computing architectures are unaffordable.
- Obsolescence is here to stay.
- OA has been embraced by commercial industry.
- Surface ship computing systems reached performance capacity years ago, and have inherent limits on capability increases.
- Current warfighting concepts-to-capability is ~ five years (to IOC).
- Commercial computing technologies and modern software languages are required to meet computer throughput and speed requirements.
- OA enables the establishment of common links between FORCEnet, the GIG, SIAP, CUP, and the Fleet.
- The goal of using OA is to create a computing environment based on international commercial standards while maximizing use of common components and critical interfaces.
- Expected benefits of the use of OA include:
    - Joint interoperability
    - Increased capability and performance
    - Streamlining of software development and maintenance
    - Faster time to market
    - Significantly more affordability
    - Reduction in manning requirements
    - Lower testing and certification costs

Research Topic: Open Architecture (OA)

Reviewed by Tim Carpenter

Title: FORCEnet / Open Architecture Alignment Status

Author: Danny R. Stevenson, SAIC

Publisher: Danny R. Stevenson, SAIC (Power point slides)**,** 18 May 2004

Summary: This presentation gives status on a FORCEnet (Fn)/Open Architecture (OA) EXCOMM action item, which was for a Tiger Team focused on documentation to identify a Fn implementation approach **"**to ensure integration of C4I and warfare systems across FORCEnet implementation initiatives**."** The stated objective was to "guarantee interoperability between many multiple Fn Domains" using a two-step approach. The presentation reminds us "FORCEnet Architecture & Standards is the Naval single technical reference to move towards the GIG." It depicts and explains a FN functional reference model and a model of the GIG, and states that joint integrated architectures require a different process to produce than traditional processes. The presentation discusses the documentation scope and de-confliction plan, depicts the documentation set tree, and outlines the way ahead.

Key Points/Recommendations:

- Two-step approach to ensuring interoperability – 1) Guarantee commercial and service-specific standards, and 2) Provide non-conflicting guidance based on Step 1.
- Net-centric Enterprise Solutions (NESI) provides the technical architecture, implementation guidance, technical criteria, and reusable software components that can facilitate the design, development and usage of C4I systems**.**
- Document de-confliction is performed to ensure interfaces will have a single POC, and other documents will reference, eliminating duplication.

Research Topic: Open Architecture (OA)

Researcher: Dave Chacon

Title: Applying Open Architecture Concepts to Mission and Ship Systems

Author: John M. (Mike) Green

Publisher: TBD, 2008

Summary: The paper makes the case that a well-defined surface domain combat system objective architecture is critical to the development of next-generation combat systems. It advocates an approach to domain modeling that facilitates development of a surface domain combat system objective architecture and its requirements. The recommended methodology uses a process modeling approach based on Finite State Machine theory implemented in an advanced simulation language like Extend™. The stated purpose of the paper is to serve as an introduction to a simulation-based methodology to facilitate development of a software product line architecture concept for Navy combat and C4ISR systems. Recurring themes throughout the paper are emphasis on use of open architecture (OA), the difficulty of trying to adapt older "stove-piped" architecture paradigms to current and future system requirements, domain modeling, formal methods, and the Hatley-Pirbhai process, or Process for System Architecture and Requirements Engineering (PSARE). Domain is defined as a *functional area shared across a group of products.*

Key Points/Recommendations: The discussion of OA cites five key principles advocated by the Navy, with the first two (modular design/design disclosure and reusable application software) called out as being especially relevant to the paper. However, from the perspective of the Supportability IPT, perhaps the more relevant principle is that of life cycle affordability. Using OA modular design and spiral development to create reusable application software is more efficient and affordable during life cycle than having to start from scratch to incorporate technology enhancements and advance product evolution.

Other key points and recommendations cited are:

- Making good design decisions early in the process using OA drives down life cycle cost and system development time.
- Abandonment of "stove-piped" architectures and adoption of a simulation-based methodology to facilitate understanding of how key concepts relate and interact from a functional perspective is crucial to the development of future combat and C4ISR systems.
- OA cannot be separated from concepts such as Enterprise Architecture and DODAF, and needs to include services-oriented constructs where appropriate.
- Use of a simulation-based methodology will result in most of the required DODAF artifacts for both requirements capture and system functional behavior description, and is consistent and compatible with spiral development and the "Vee" systems engineering model.

Research Topic: Open Architecture (OA)

Researcher: Tim Carpenter

Title: Acoustic Rapid COTS Insertion: A Case Study in Spiral Development

Author: Michael Boudreau, Senior Lecturer, Naval Postgraduate School

Date: October 30, 2006

Summary: This paper uses the submarine community's Acoustic Rapid COTS Insertion (A-RCI) as a case study in the successful implementation of Modular Open Systems Approach (MOSA)/Open Architecture (OA). It identifies best practices and lessons learned from the A-RCI experience. Its purpose is to provide a learning vehicle for training and educating acquisition personnel in the application of MOSA/OA.

Key Points:

- Establish an environment that enables modular open system architecture (MOSA)
  - The PM needs to establish requirements, business practices and technology development, acquisition, test and evaluation and product support strategies that support effective development of open systems
  - Ensure appropriate experience and training in MOSA
  - Conduct market research
  - Be proactive in identifying and overcoming barriers and/or obstacles that hinder effective MOSA implementation
- Employ modular design. Modular designs are characterized by the following:
  - Functionally partitioned into discrete, scalable, reusable modules consisting of isolated, self-contained functional elements
  - Rigorous use of disciplined definition of modular interfaces, to include object-oriented descriptions of module functionality
  - Designed for ease of change to achieve technology transparency and, to the largest extent possible, to make use of commonly used industry standards for key interfaces.
- Manage interfaces by grouping them into "key" and "non-key" interfaces. MOSA distinguishes among interfaces that are between technologically stable and volatile modules, between highly reliable and more frequently failing modules, and between modules with least interoperability impact and those that pass vital interoperability information. Key interfaces should utilize open standards in order to produce the largest lifecycle cost benefits.
- Select interface standards based on maturity, market acceptance and allowance for future technology insertion.
- The program manager, in coordination with the user, should prepare validation and verification mechanisms such as conformance certification and test plans to ensure that the system and its component modules conform to the external and internal open interfaces—allowing plug-and-play of modules, net-centric information exchange, and re-configuration of mission capability in response to

new threats and technologies. Open systems verification and validation must become an integral part of the overall organization change and configuration management processes. They should also ensure that the system components and selected commercial products avoid utilization of vendor-unique extensions to interface standards and can easily be substituted with similar components from competitive sources.

- Set up a competitive "playing field" to attract innovative contractors who might be new to DOD contracting or intimidated by large contractors.
  - Focus competition on best ideas and best performance rather than on politics or a preordained hierarchy of competitors.
  - Let the best technical solution "win."
  - Select best solutions using peer review.
- Intellectual property rights should be made available as part of the price of entering into competition so that code and design information can be shared and not become a hindrance to progress.
- Systems Engineering Process (SEP) is a potential risk area. For future programs, SEP responsibility might reside with the Government materiel developer (or be separately contracted) during testing and Peer Review before being handed off to a prime system integrator. On one side of the balance, the Government PM office might not have the necessary staffing for managing SEP; on the other side, contracting out SEP might damage the necessary sense of trust and confidence in a competitive level playing field.
- Spiral Acquisition must be rooted in the JCIDS process to ensure proposed acquisitions address required capabilities, avoid unnecessary redundancy, and provide interoperability with other war-fighting systems. The JCIDS process requires reviews and approvals that are important, but also are time-consuming and may contribute to program delays. There appears to be a risk that rapid op tempo spiral developments potentially may collide with slower-moving JCIDS processes, resulting in incomplete reviews, inadequate user direction, or program delay.
- End-to-end operational testing has not synchronized well with the A-RCI/APB process as testing is time consuming, expensive and may not always be necessary with spiral upgrades. End-to-end operational testing has its place, but possibly not in every spiral of an evolutionary development.

Research Topic: Open Architecture (OA)

Researcher: Mike Kinberg

Title: Naval Open Architecture – Overview on OA

Author: CAPT James Shannon, Program Manager, Naval Open Architecture

Publisher: TBD (Presentation), February 14, 2006

Summary: This presentation provides an overview of the Navy's OA Enterprise Initiative, including the strategy, current state, future state and lessons learned. In addition, it provides an overview of an OA assessment model and tool for assessing the "openness" of a program.

Key Points/Recommendations:

- Establish Enterprise Communities of Interest (COIs)
  - Base COIs on mission areas (Strike, ISR, AAW, ASW, etc.)
  - Include the Warfighter at EVERY step
  - Plan for enterprise-wide reuse of government owned software
  - Use MOSA (Modular Open System Approach) principles - modular design, open standards, key interfaces
  - Incentivize Program Managers for enterprise vice platform/program success
  - Use Business Case Analyses to determine OA priorities

- Contracts
  - Incentivize cooperation among integrators and developers
  - Develop award fees based on group success
  - Maintain continuous competition for application development
  - Conduct independent peer review of products using real data
  - Ensure data rights support open architecture and 3rd party use
  - Full disclosure – Early and Often

Research Topic: Open Architecture (OA)

Researcher: Bob Howard

Title: White Paper – Navy Enterprise Implementation of Open Architecture, An Assessment from the Small Business Perspective v1.1

Author: Harley Garrett, Global Technical Systems, Inc.

Publisher: Global Technical Systems, Inc., December 11, 2007

Summary: This white paper addresses two aspects: 1) the necessity of including small business in the acquisition process in order for the Navy to achieve the cost effectiveness benefits of OA implementation. The agility of the small business community enables them to provide innovative system architectural alternatives. 2) Recent trends in software and hardware development and their impact on the Navy's OA goals.

Key Points/Recommendations:

- Unanimity regarding the meaning of the term "open architecture" is required for effective implementation. A common understanding across all Navy organizations is still evolving.
- Promoting a healthy competitive environment by including small business in the acquisition process will result in a wide array of design alternatives. Conduct an enterprise-wide small business assessment to determine how well current programs of record are acquiring and assimilating OA technologies from the small business community. Assessment should emphasize whether or not the current level of small business participation is providing OA technologies that have, or are having, a measurable impact on reducing combat system acquisition and modernization costs while increasing net-centricity and interoperability. Other metrics may include modularity and reuse of software, platform independence of computer operations, and impact on ownership costs over the lifecycle.
- Consider more small business set-asides to acquire design alternatives at the system and subsystem levels. This should include an independent analysis of existing designs and recommendations on how to improve designs.
- Apply the submarine community's A-RCI (Advanced Rapid COTS Insertion) experience to other domains. Create Peer Review Groups to include subject matter experts from the small business community and academia. Implement the Advanced Capability Build (ACB) process.
- Require source code developed for weapon systems that is funded by Navy contracts to be delivered by the contractor with unlimited rights or GPR (Government Purpose Rights) for deposit into SHARE (Software-Hardware Asset Reuse Enterprise).

Research Topic: Open Architecture (OA)

Researcher: Bob Howard

Title: Open Architecture, The Critical Network Centric Warfare Enabler

Author: Captain Richard T. Rushton

Date: 18 Mar 2004

Summary: This paper describes imperatives of modern battlefield that demands Network Centric Warfare and why Open Architecture (OA) is the most critical enabler. It also puts architectural constructs of Global Information Grid (GIG) and FORCEnet into warrior context and terms that relate. In addition it describes how the current integrated combat systems (ICS) are being transformed so they can be maintained and improved with flexibility. The reason for embracing modern open systems architecture is driven by conditions required to fully support Network Centric Warfighting (NCW) capabilities. OA allows ICS to achieve full joint interoperability and provide seamless information to the Global Information Grid (GIG). NCW is an imperative condition required to meet the challenges encountered today and into the future.

Key Points/Recommendations:

- Open Architecture in a Warrior's Terms
- Open Architecture Computing Environment (OACE)
- OACE/OAFA Relationship
- Aegis Integrated Weapon System v. modern technology and network centric warfighting imperatives.
- FORCEnet as an integrated part of the Global information Grid

Research Topic: Open Architecture (OA)

Researcher: Bob Howard

Title: Open Architecture in Naval Combat System Computing of the 21st Century

Author: Captain Thomas J. Strei

Date: 01 Apr 2003

Summary: This paper specifically describes and defines the Navy's Open Architecture (OA) strategy to address weapon systems affordability, interoperability, and performance for the Navy's 21st Century. OA is based on the concept of COTS information and computing systems. The intent of this architecture is to enable common functions across multiple systems. A key requirement is to take advantage of commercial standards and products that are to be selected depending on performance, cost, and upgrade potential. The initiative of this architecture is to reduce multiple infrastructures and generate economic efficiency. Sea Power 21 published in 2002 by Chief of Naval operations Admiral Vernon E. Clark described his vision of the 21st century Navy that embodied three operational concepts Sea strike, Sea shield, and Sea basin to be enabled by "FORCEnet" which is the "glue" that binds these concepts together. FORCEnet requires OA to achieve standard joint protocols, common data packaging, seamless interoperability, and strengthened security.

Key Points/Recommendations:

- Challenges of Navy's Legacy Systems
  - Limited computational and processing capability; Difficult or unable to add new war fighting missions; Bypassed by commercial industrial base, making software upkeep cost prohibitive
- Open Architecture
  - Use of public, consensus-base standards; Adoption of standard interfaces and protocols; Adoption of standard services and defined functions; Use of product types supported by multiple vendors; Selection of stable vendors with broad customer base and large market share; Interoperability with minimal integration requirements; Ease of scalability and upgradeability; Portability of applications
- Integrated Warfare Approach
  - Maximizes fundamental interoperability across warships, aircraft, weapons, and sensors
  - Key metrics: portability, scalability, extensibility, and flexibility
- Engineering Development Model (EDM) goals
  - Combat system, weapon system, command support systems, and hull, mechanical, and electrical capabilities that continue to pace the threat
  - System design and common components that foster affordable development and life-cycle maintenance and maximizes reuse

- System design and common components that reduce upgrade cycle time and time to deployment for new features
- Architecture that is technologically refreshable despite rapid COTS obsolescence
- Improvements in naval warfare system human systems integration

Research Topic: Service Oriented Architecture (SOA)

Researcher: Mindy Wentland

Title: Data Integration in Service-Orient Architecture

Author: Informatics Corporation

Publisher: Informatics Corporation, October 2005

Summary: This paper examines how an enterprise data integration platform enriches service oriented architecture-and how an SOA provides an ideal framework for data integration technology.

Service-oriented architecture offers an elegant remedy. SOA has gained widespread acceptance with open, industry-standard Web services protocols such as Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), and Web Services Description Language (WSDL). These standards offer a highly flexible layer of abstraction that can reduce development time and cost through the promoting reuse of components.

SOA principle concept cover:

- "Loosely coupled" services: A layer of abstraction between the technical service implementation and client that eliminates the need for customized, "tightly coupled" interoperability
- Leverage of IT assets: Component-based services may be wrapped and reused for deployment across multiple projects and applications to reduce development time and cost
- Use of open standards: Web services standards such as XML, SOAP, and WSDL provides the interoperability that enables an SOA to work across heterogeneous platforms

Key Points:

- The SOA platform is comprised of three functional components a universal data access layer, a metadata repository, and an enterprise data bus. This trio operates in concert to coordinate and deliver a range of data services.
- Universal Data Access
  - It should provide nearly unlimited data access via both traditional physical and virtual data integration approaches, while minimizing the cost and complexity of accessing data regardless of where it resides.
- Metadata-Driven Architecture
  - The data integration platform will extend beyond data to its metadata—the "glue" that describes data values and their semantic meaning. It provides a drag-and-drop user interface that enables developers to rapidly build

processes, and transformations for data and make them available for reuse. It will have at its core a scalable metadata repository that stores and manages data models, and other artifacts

- The metadata repository serves as a universal data interaction framework that brokers the translations between high-level service definitions and more granular data definitions and mappings
- Enterprise Data Bus: Ensuring Enterprise-Class Deployment

  The heart of a data integration platform is an enterprise data bus that offers a variety of flexible data delivery mechanisms and scalability for large-volume data extracts. Leading data integration platforms will provide:

  - Conventional batch mode movement
  - Changed data capture (moves only updated data for improved performance)
  - Real-time capture and movement (e.g., financial markets, just-in-time distribution)
  - Support for distributed, multi-node grid systems
  - High availability, fault tolerance, and failover

Applicability to Capstone Project:

- The ability of SOA to provide reusability of data via application
- The ability to access data (SPL) wherever and whatever form in a consistent and accurate manner
- Universal data access and metadata-driven architecture provides data integrity services
- Identify the project in scope and offer reuse of assets developed in future follow-on projects
- As follow-on projects reuse the same assets and leverage a common infrastructure, it's important to track metrics and be able to show the reuse and cost savings

Recommendations: The Software IPT will investigate if implementing SOA into the Software Architecture is feasible for the Capstone Project.

Research Topic: Service Oriented Architecture (SOA)

Researcher: Mindy Wentland

Title: Implementing SOA in DoDAF 1.5,

Author: Alexander H. Lewis George Mason University Jan 2008

Publisher: George Mason University, Jan 2008

Summary: Service Oriented Architecture (SOA) describes an architectural style that supports loosely coupled services that are interoperable and technology agnostic.

Key Points:

- The unique characteristic for systems conforming to a SOA is adoption of an information technology infrastructure layer which
  - Uses protocol standards common to all the participants…
  - Commercial standards are good
  - Promotes "loose coupling" among participants
  - Causes meaningful information and application access among participants to be established by conformance to the standards (I.e. unique pair-wise information exchange requirements are neither desired nor necessary)
  - Each service provides an interface based description to support flexible and dynamically reconfigurable processes
  - A composite set of services, under SOA, is capable of realizing an end-to-end process or provide a capability

- In order to migrate to a Net-Centric Warfare (NCW) environment, the DoD is focusing on networking the Warfighter enterprise by making essential information available to authorized users. That is, data must be visible, accessible, understandable and trusted.
- NCW concepts are embodied in DoDAF v1.5, which is a transitional version that applies enabling technologies such as services within a Service Oriented Architecture (SOA). A SOA supports an information environment built upon loosely coupled, reusable, standards-based services. A service is a function that is well defined, self-contained, and does not depend on the context or state of other services.

Applicability to Capstone: SOA is complex concept that has not yet been fully implemented. However, SOA will be a great assess if its can be incorporated into the AAW piece because within a Net-Centric SOA, the Warfighter has the capacity to discover applications from any domain.

Recommendation: Software IPT will investigate the possibility of incorporating SOA in the Software architecture domain.

Research Topic: Service Oriented Architecture (SOA)

Researcher: Linda Banner-Bacin

Title: Data Integration in Service Oriented Architecture

Author: Informatica -White Paper

Date: October 2005

Summary: The paper discusses the value of Service Oriented Architecture (SOA), however using SOA comes costs and integrations issues. The author calls it an "integrated architecture foundation". As companies, businesses, and government agencies merge into the global communication network where requirements such as supply, customer access, information and materials are the demand, IT structures must be able to handle the infrastructure to meet the demand. The authors use the term data fragmentation and brittle point-to-point connectivity. A recommended solution is use an Enterprise Data Integration Infrastructure. When developing the SOA the Enterprise Application Integration (EAI) and the enterprise data integration work together process transactions and execute complex data functions. When developing the following should be considered: Data Semantics; Data Quality; Data Governance; Data Access; Data Transformation; and Bulk Data Process

Three Key Principals of SOA are:

- Loose Coupling: access to data where ever it resides, in whatever form and is consistent across the enterprise
- Leverage of IT standards: Component- based data integration services may be wrapped, reused across multiple systems
- Use of open standards: Use of XML, SOAP and WSDL enable data integration to operate seamlessly

In summary prior to using SOA, thoughtful plans must be identified, Enterprise Application Integration must be solidly developed and the data integration must be well established or the results will be "brittle point to point connectivity". Below is an illustration of the concept from the article, which provides a good over view of the concept.

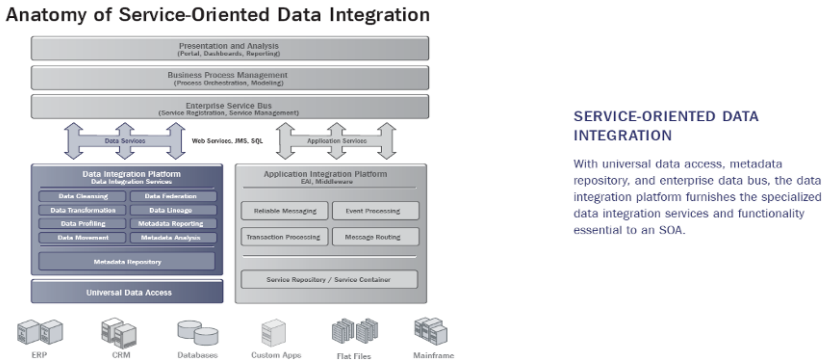


Figure 1 – Service-Oriented Data Integration.

Research Topic: Software Product Lines (SPL)

Researcher: Paul Wheeler

Research Title: Design and Use of Software Architectures: Adopting and evolving a product-line approach

Author: Jan Bosch

Publisher: Pearson Education Limited, 2000.

Summary:

Decomposing Software Product Lines (SPLs)

Three dimensions in which the concepts included in SPLs can be decomposed are as follows:

Architecture – the main activity is to design architecture for the product line that covers all the products in the product line and includes the features that are shared between products.

Component – the product line architecture identifies the components and the variability required from the components.

System – this activity requires an adaptation of the product line architecture to fit the system architecture.

Functionality-Based Architectural Design

Functionality-based architectural design is concerned with the definition of the product context, the identification of the archetypes and the description of the product instantiations.

Defining the Product Context

The first step in functionality-based architectural design is defining the product context.

The products that are part of the product line may be very diverse in terms of their context, in terms of the underlying hardware, the external products that it communicates with and the user interface.

Identifying Archetypes

The next step is the identification and definition of archetypes. Archetypes represent the core concepts used for **modeling the software architecture** and for describing the software instantiations.

Describing the Product Line Instantiations

The final step is describing the product line instantiations. The goal here is to verify the suitability of the selected archetypes and the ability of the current archetype to represent all the variations of the product.

Key Points:

The Initiation Process for SPLs

First, it must be determined whether to take an **evolutionary** approach or a revolutionary approach to the introduction process.

Second, the product line approach may be applied to **an existing line of products** or a new system or product family.

The choice must be made to proceed with one of the following approaches:

- **Evolve an existing set of products into a product line**
- Replace an existing set of products with a software product line
- Evolve a new software product line
- Develop a new software product line

Designing a Product Line Architecture

Changes to product line architectures are generally small and relatively infrequent since major changes prohibit the use of product line components.

Product-specific features need to be considered even when designing the product-line architecture.

Applicability to Capstone:

A Product-line approach to the software architecture will be integrated into the overall AAW hardware architecture, which will compose the final product for capstone. The correct design approaches for developing a software product line from existing software products and new software products will need to be applied to our AAW architecture.

Recommendations: Recommend that the capstone project make use primarily of existing software architectures.

Research Topic: Software Product Lines (SPL)

Researcher: Mindy Wentland

Title: Design and Use of Software Architectures

Author: Jan Bosch

Publisher: ACM Press Book

Summary: The author stated that in order to achieve successful reuse programmer are required to employ a top down approach and bottom up approach in the system engineering process. It is important to design and develop a component that fit into the high level structure that defined in the software architecture. By definition, Software Product Lines (SPL) is a set of software intensive product systems that share a common, managed feature set satisfying a particular market segment's specific needs. SPL can be decomposed to architecture, component, and system. Architecture is the main activity is to design architecture for the product line that covers all the products in the product line and includes the features that are shared between products. Component is the product line architecture identifies the components and the variability required from the components. System is this activity requires an adaptation of the product line architecture to fit the system architecture. There are three steps in SPL; the first step is to identify the functionality architecture. The second step is to define the product context. The last step is the identification and definition of archetypes. Archetypes represent the core concepts used for modeling the software architecture and for describing the software instantiations. (Chapter 7 and follow on).

Key Points: It is important to identify the functionality-based architecture design because without the product context being identified; things can be unclear and misunderstood. Also, products that are part of the product line may be very diverse in terms of their context, in terms of the underlying hardware, the external products that it communicates with and the user interface. The product line approach may be applied to an existing line of products or a new system or product family. In addition, product-specific features need to be considered even when designing the product-line architecture, the choice must be made to proceed with one of the following approaches:

- Evolve an existing set of products into a product line
- Replace an existing set of products with a software product line
- Evolve a new software product line
- Develop a new software product line

Applicability to Capstone: Minimal

Recommendation: Helpful references guidelines for the SW IPT.

Research Topic: Software Product Lines (SPL)

Researcher: Mindy Wentland

Title: An Introduction to Software Architecture

Author: David Garlan and Mary Shaw

Publisher: Carnegie Mellon University (1994)

Summary:

- As the size of software systems increases, the algorithms and data structures of the computation no longer constitute the major design problems. When systems are constructed from many components, the organization of the overall system—the software architecture—presents a new set of design problems. This level of design has been addressed in a number of ways including informal diagrams and descriptive terms, module interconnection languages, templates and frameworks for systems that serve the needs of specific domains, and formal models of component integration mechanisms.
- This paper provides an introduction to the emerging field of software architecture. It begins by considering a number of common architectural styles upon which many systems are currently based and show how different styles can be combined in a single design. Then it presents six case studies to illustrate how architectural representations can improve understanding of complex software systems. Finally, it surveys some of the outstanding problems in the field, and considers a few of the promising research directions.

Key Points:

- Presents several common software architecture styles:
  - Pipes and Filters
  - Data Abstraction and Object-Oriented Organization
  - Event-based, Implicit Invocation
  - Layered Systems
  - Repositories
  - Table Driven Interpreters
  - Other Familiar Architectures
  - Heterogeneous Architectures

Applicability to Capstone Project: Will be used by SW IPT to construct the software architecture that demonstrates the AAW mission.

Recommendations: SW IPT to decide between Event-based, Implicit Invocation and Layered Systems for software architecture to implement OA for AAW-based system.

Research Topic: Software Product Lines (SPL)

Researcher: Mindy Wentland

Title: An Introduction to Software Product Line Development

Author: Magnus Eriksson

Publisher : Alvis Hägglunds, 1982

Summary:

- Software product line development supports large intra-organization software reuse. Product line practice in the software industry is a relatively new concept. Studies have shown that organizations can achieve improvements in productivity, time to market, product quality and customer satisfaction by applying this approach.
- SPL development involves software reuse. SPL line development is a lot more elaborate than traditional software reuse. In traditional software reuse, organizations create repositories where the output of practically all development efforts is stored. The repository would typically contain some reuse library with components, modules and algorithms that developers are urged to use. The problem with this type of reuse is, that it usually takes longer to find the desired functionality and adapting it to current application than it would to build it from scratch.
- In product line development reuse is planned. The reuse repository of an SPL is known as the core assets of the product line. The core assets include all the artifacts that are the most costly to develop; domain models, requirements, architecture, components, test cases, and performance models, etc. Furthermore, these core assets are from the beginning developed to be (re)used in several products. This means that the asset customization to the current product does typically not include any major code writing as it would in traditional approaches. Product instantiation is instead accomplished using variability mechanisms built into the core assets.

Key Points:

- Two important qualities attributes that address the flexibility of product line architectures are modifiability and configurability.
- The component development is the part of the development process where the in-house operational software that is needed by the products is created. The product line components are specified by the product line architecture and implement the required predictability to fulfill expected product requirements. The resultant components can either be a part of the core assets of the product line or they can be developed for product specific reasons.
- There are two levels of system integration in product line development. The first level concerns the installation of core assets into the asset base. The second level concerns the building of the individual products within the product line.

Applicability to Capstone Project: Will be use by the SW IPT to develop SPL framework for the repository.

Recommendations: SW IPT to implement software product line for the AAW architecture.

Research Topic: Software Product Lines (SPL)

Researcher: Mindy Wentland

Title: Introduction to Software Product Lines

Author: Charles W. Krueger, PhD

Publisher: <http://www.softwareproductlines.com>

Summary:

- Software product lines refer to engineering techniques for creating a portfolio of similar software systems from a shared set of software assets using a common means of production.
- The source of the order-of-magnitude improvements from software product line techniques is strategic software reuse: consolidate commonality throughout the product line, strategically manage all product line variation, and aggressively eliminate all duplication of engineering effort.
- Can yield order of magnitude improvements in time-to-market, quality, portfolio scalability and software engineering cost. The result is often a discontinuous jump in competitive business advantage, similar to that seen when manufacturers adopt mass production and mass customization paradigms.

Key Points:

- The characteristic that distinguishes software product lines from previous efforts is predictive versus opportunistic software reuse. Rather than put general software components into a library in hopes that opportunities for reuse will arise, software product lines only call for software artifacts to be created when reuse is predicted in one or more products in a well defined product line.
- Software product lines can be described in terms of four simple concepts, as illustrated in the figure below:
    - Software asset inputs: a collection of software assets – such as requirements, source code components, test cases, architecture, and documentation – that can be configured and composed in different ways to create all of the products in a product line. Each of the assets has a well-defined role within a common architecture for the product line. To accommodate variation among the products, some of the assets may be optional and some of the assets may have internal variation points that can be configured in different ways to provide different behavior.
    - Decision model and product decisions: The decision model describes optional and variable features for the products in the product line. Each product in the product line is uniquely defined by its product decisions - choices for each of the optional and variable features in the decision model.
    - Production mechanism and process: the means for composing and configuring products from the software asset inputs. Product decisions are

used during production to determine which software asset inputs to use and how to configure the variation points within those assets.

- Software product outputs: the collection of all products that can be produced for the product line. The scope of the product line is determined by the set of software product outputs that can be produced from the software assets and decision model.

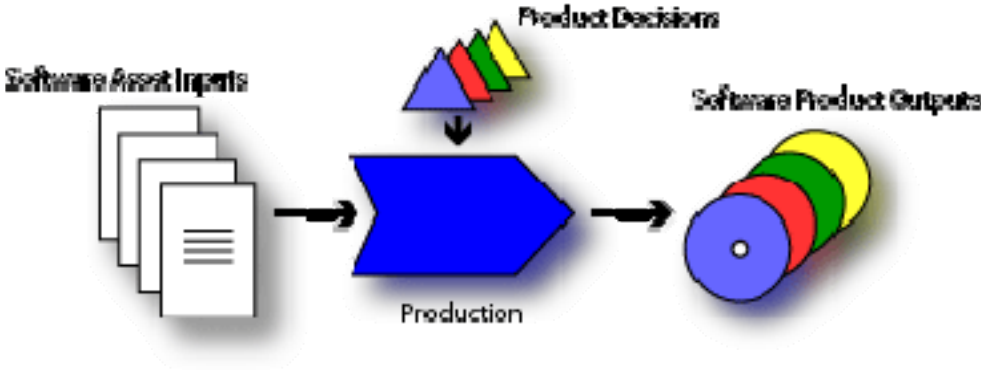


Figure 1- Basic Software Product Line Concepts

- These concepts illustrate the key objectives of software product lines: to capitalize on commonality and manage variation in order to reduce the time, effort, cost and complexity of creating and maintaining a product line of similar software systems.

  - Capitalize on commonality through consolidation and sharing within the software asset inputs, thereby avoiding duplication and divergence.
  - Manage variation by clearly defining the variation points and decision model, thereby making the location, rationale, and dependencies for variation explicit

- Early case studies of software product line transitions reported typical adoption times of 2 to 5 years. For most organizations, this time and effort represents a significant barrier to adopting a product line approach, regardless of the potential benefits. Recently, advances have been made in lightweight approaches that lower the required transition effort, with some case studies reporting adoption efforts as low as 2 months. Lightweight techniques employed include:
  - Minimize differences between single-system and product line engineering in order to minimize impact on organization, processes, software, and infrastructure
  - Utilize an incremental adoption strategy to initially transition a small subset of assets, products, subsystems or people
  - Offer off-the-shelf software product line tools and technology
  - Use reactive approaches to defer product development and deployment efforts
  - Structure production to minimize the need for complex and costly merging, feedback, and product-specific configuration management overhead

- The following steps, each of which has its own section in this chapter, will help you get started down the software product line path:
  - Become informed. Utilize the resources provided or linked to on this site to advance your knowledge of software product lines.
  - Assess your situation. Characterize the pain, the benefits, and the urgency so that you can determine whether software product lines are right for you.
  - Build the team. Assemble the right team to lead your transition to a software product line approach.
  - Create the long-term vision. Identify your ideal "success story".
  - Find the quick wins. Define your first steps to provide some big wins early.

Applicability to Capstone:

- What software does NAVSEA produce? We can develop a roadmap for this, but must research what systems this will affect. This would be a new way of doing things, so there is no way to model success/failure. Implement and re-assess somewhere down the line.
- Missile features and functions have a lot of commonality (SM and ESSM) (maybe Raytheon already uses this and charges the government the full amount for development from scratch?). If they aren't already doing this, can we force contractors to adopt SPLs?

Recommendation: SW IPT to explore the use of SPLs further and develop a software repository based on this concept.

Research Topic: Software Product Lines (SPL)

Researcher: Wentland

Title: Software Product Line

Author: Carnegie Mellon University

Publisher: NOA Contract Guidebook V1.1 October 25,

Summary: A software product line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

Key Points:

- Open source is emerging as a new global paradigm challenging the conventional approach in software development. The fact that product line is a natural evolution in the maturity process of software development is leading to the adoption of related practices by open source communities.
- On June 5, 2007, the Department of the Navy (DoN) Chief Information Officer (CIO) directed DoN commands to treat Open Source Software (OSS) as COTS when it meets the definition of a commercial item. This will allow the DoN to utilize OSS throughout the enterprise when acquiring capabilities to meet DoN business and Warfighter requirements. As with any COTS solution, the use of OSS must adhere to all Federal, DoD, and DoN policies and be based on open standards to support the DoD's goals of net-centricity and interoperability. In addition, DoN commands must work with their intellectual property general counsel to ensure compliance with the OSS license agreement.

Application to Capstone:

- SPL is one of the most important key points of the capstone project.
- SPL is a new and emergent method of reusing software and model based system engineering.

Recommendation: The Software IPT is researching and producing artifact that can incorporate methodology of SPL into a weapon centric environment where software development can reduce cost, increase productivity and quality.

Research Topic: Software Product Lines (SPL)

Researcher: Bob Howard

Title: Design & Use of Software Architectures - adopting & evolving a product line approach

Author: Jan Bosch

Date: 6/22/1995

Length: 335 pages

Summary: The book provides examples that an explicit design and first class representation can allow a paradigm shift away from traditional software approaches. Of key interest from a supportability and TOC is identification of Quality Attributes, which help a SPL to meet its quality objectives in conjunction with its functional objectives. Assessment methods are discussed to provide a path for future growth and to provide a method to assess tradeoff between quality attributes.

Research Topic: Systems Modeling Language (SysML)

Researcher: Ruth Matela / Tuyen Hoang

Title: "Requirements Specification and Modeling through SysML"

Author: Michel dos Santos Soares, Jos Vrancken

Publisher (Date): IEEE (March 2007)

Summary: The article is about applying the SysML requirements diagram to the specification of system requirements. SysML is a new systems modeling language that supports specification, analysis, design, verification and validation of a broad range of complex systems. It is expected that SysML will become a *de facto* standard for Systems Engineering, just like the UML is for Software Engineering.

Key Points:

- Requirements are presented in a graphical and tabular form, and are modeled instead of just written in natural language.
- It is presented that the SysML requirements diagram can fill the gap between natural language based specifications and UML use case diagrams. Another advantage is that there's a defined semantics to associate SysML requirements diagram to other models created during system design.
- Requirements traceability is "the ability to describe and follow the life of a requirement, in both a forward and backward direction, i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases."
- Requirements traceability helps in identifying the sources, destinations and links between requirements and models created during system development. Traceability also provides a possibility to ensure that all requirements are fulfilled by the system and sub-system components.
- The type of requirements relationship can be shown using a tabular matrix, which allows an agile way to identify, prioritize and improve requirements traceability. The SysML requirements table is a good manner to improve the traceability between requirements.
- The SysML use case diagram is derived without modifications from UML. The purpose is to describe the usage of the system by its actors in order to achieve a goal. The use case can also be viewed as functionality and/or capabilities that are accomplished through the interaction between the subject and its actors.

Applicability to Capstone Project: SysML is the language chosen for the Requirements IPT to represent the system requirements.

Recommendations: The Methodology IPT recommends for the Requirements IPT to utilize SysML in generating the requirements artifacts (e.g., requirements diagram).

Research Topic: Systems Modeling Language (SysML)

Researcher: Caleb Vajdos

Title: Supporting Building Bridges Between Systems and Software with SysML and UML

Author: Matthew Hause and Francis Thom

Publisher: Unknown

Summary: Systems are becoming increasingly reliant on software. One of the roles of the systems engineer is to perform a trade-off analysis of the different architectural solutions to a problem, and allocate requirements to different engineering domains within that solution, including software. It is important to investigate effective ways of establishing traceability from the system definition to the software and other requirements. The Systems Engineering Language, (SysML), which is based on the Unified Modeling language (UML), is being increasingly used by systems engineers to model systems. As well as providing system requirements, SysML models can be used to define the system architecture to be used by the software engineers. In this paper, we will demonstrate how SysML and UML can effectively work together to provide an effective handover between systems and software.

Key Points:

- Discusses the use of Object-Oriented Modeling, the use of SysML to define software requirements and ensure traceability, the use of SysML to define the environment in which the software will be deployed.
- SysML provides a model-centric approach to formal requirements handover between systems engineers and software engineers providing software engineering with the necessary contextual information.

Applicability to Capstone Project:

- This proposes a good way to tie in software to our overall architecture development method.
- Promotes tractability of requirements between system and software engineering. It also reinforces the use of SysML.

Recommendations: See applicability to Capstone!

Research Topic: Systems Modeling Language (SysML)

Researcher: Alan Wellesley

Title: Requirements Specification and Modeling through SysML

Author: Michel dos Santos Soares, Jos Vrancken

Publisher: IEEE, 2007

Summary: Abstract—Use Case diagrams are well known for their use to specify and describe system requirements. From initial system requirements documents, use cases can be derived representing several scenarios. These scenarios can later be detailed in different ways, as for example, through informal descriptions. In this paper, system requirements are first specified using the SysML requirements diagram and later by use cases. The main goal is to fill the gap between documents written in natural language and use cases by modeling requirements in a graphical and tabular way, which can improve the requirements representation. Also, the relationship between requirements is enhanced. An example of a real time distributed system is given to illustrate the approach.

Key Points:

- Systems requirements are commonly written using natural language. The principal problem of this approach is the ambiguity of natural languages.
- One good example of a semi-formal notation for requirements specification is the UML use case diagram. In [6], it is presented how use cases are applied with a scenario-based requirements approach in the development of safety critical systems. But two problems still arise: the natural language is still needed, and the diagrams are sometimes not user-friendly.
- SysML may facilitate the communication between heterogeneous teams (for instance, mechanical, electrical and software engineers) that work together to design a system.

Applicability to Capstone Project: Supports use of SysML during requirements development.

Recommendations: None

Research Topic: Systems Modeling Language (SysML)

Researcher: Wellesley

Title: OMG Systems Modeling Language (OMG SysML™), V1.0., OMG Available Specification

Author: Object Management Group (OMG)

Publisher: Object Management Group (OMG), Sep 2007

Summary: The purpose of this document is to specify the Systems Modeling Language (SysML), a new general-purpose modeling language for systems engineering that satisfies the requirements of the UML for Systems Engineering RFP. Its intent is to specify the language so that systems engineering modelers may learn to apply and use SysML, modeling tool vendors may implement and support SysML, and both can provide feedback to improve future versions.

This specification defines a general-purpose modeling language for systems engineering applications, called the OMG Systems Modeling Language (OMG SysML™).

In a manner similar to how UML unified the modeling languages used in the software industry, SysML is intended to unify the diverse modeling languages currently used by systems engineers. Since SysML uses UML 2 as its foundation, systems engineers modeling with SysML and software engineers modeling with UML 2 will be able to collaborate.

The specification also provides examples of how the language can be used to solve common systems engineering problems. SysML is designed to provide simple but powerful constructs for modeling a wide range of systems engineering problems. It is particularly effective in specifying requirements, structure, behavior, and allocations and constraints on system properties to support engineering analysis.

Key Points:

- SysML will improve communication among the various stakeholders who participate in the systems development process and promote interoperability among modeling tools.
- It is anticipated that SysML will be customized to model domain-specific applications, such as automotive, aerospace, communications, and information systems.
- The language is intended to support multiple processes and methods such as structured, object-oriented, and others, but each methodology may impose additional constraints on how a construct or diagram kind may be used.

Applicability to Capstone Project: Lists many advantages and uses of SysML.

Recommendations: None

Research Topic: Systems Modeling Language (SysML)

Researcher: Alan Wellesley

Title: An overview of the Systems Modeling Language for product and systems development -- Part 1: Requirements, use case, and test-case modeling

Author: Laurent Balmelli, PhD, Research Staff Member, T.J. Watson Research Center and Tokyo, Research Laboratory, IBM

Publisher: IBM, 15 Aug 2006

Summary: This article introduces the Systems modeling Language (SysML), a general-purpose, graphical modeling language for product and systems development.

Key Points:

- The SysML standard gives systems engineers and architects a much-needed way to collaborate using a common language that is specifically designed to support this process.
- As a standard modeling language for systems engineering, SysML will enable improved communications across development teams, while greatly enhancing our ability to manage ever-growing system complexity.
- Further, by enabling an electronic representation of the product design, SysML opens the door to analytics for faster and more effective decision making across the entire systems development lifecycle.

Applicability to Capstone Project: Provides advantages of SysML

- improves collaboration
- communication
- analytics

Recommendations: None

Research Topic: Systems Modeling Language (SysML)

Researcher: Alan Wellesley

Title: Project Quicklook Final Presentation Tactical Satellite – 3 System Design

Author: Team lead; David Alexander, Members: Soroush (Kevin) Sadeghian, Siroos Sekhavat, Thomas Saltysiak; Faculty advisor: Prof. Kathryn Laskey; External sponsor: Shana Lloyd (Aerospace Corporation)

Publisher: George Mason University / May 11, 2007

Summary: This presentation describes a project with the purpose of evaluating SysML as a Modeling Language and SysML's contribution to more efficient and effective performance analysis.

Key Points:

- Knowledge of Unified Modeling Language (UML) makes SysML easier to learn:
  - Takes advantage of Object-Oriented design; Provides bi-directional traceability between design and requirements, and Reduces efforts involved with verification of requirements and validation of system behavior.
- Modeling in SysML could be improved by using well-developed modeling tools that allow:
  - Creation of a unified data dictionary, which makes it easy to translate the design model to executable models
  - Automation of updating the model based on modifications realized after performing design trade-off
- Applying a hierarchical design process to define systems at the right level of abstraction

Applicability to Capstone Project: Provides a good evaluation of SysML.

Recommendations: None.

Research Topic: Systems Modeling Language (SysML)

Researcher: Ruth Matela/Tuyen Hoang

Title: "OMG Systems Modeling Language Tutorial)

Author: Sanford Friedenthal, Alan Moore, Rick Steiner

Publisher (Date): INCOSE (19 June 2008)

Summary: OMG SysML tutorial with the following topics: Motivation and Background, Diagram Overview and Language Concepts, SysML Modeling as Part of SE Process, SysML in a Standards Framework, Transitioning to SysML

Key Points:

- MBSE Benefits
  - Shared understanding of system requirements and design
  - Assists in managing complex system development
  - Supports early and on-going verification and validation to reduce risk
  - Enhances knowledge capture
- SysML is a graphical modeling language in response to the UML for Systems Engineering RFP developed by the OMG, INCOSE, and AP233
  - Supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities
  - SysML is critical enabler for model driven SE
  - SysML is a visual modeling language that provides semantics (= meaning) and notation (= representation of meaning)
  - SysML is not a methodology or tool; it is methodology and tool independent
- Use cases provide means for describing basic functionality in terms of usages/goals of the system by actors; use is methodology dependent and is often accompanied by use case descriptions
- SysML provides a general purpose modeling language to support specification, analysis, design and verification of complex systems

Applicability to Capstone Project: SysML is the language chosen for the Requirements IPT to represent the system requirements.

Recommendations: The Methodology IPT recommends for the Requirements IPT to utilize SysML in generating the requirements artifacts (e.g., requirements diagram).

Research Topic: Systems Modeling Language (SysML)

Researcher: Ruth Matela

Title: "An overview of the Systems Modeling Language for product and systems development"

Author: Laurent Balmelli, PhD

Publisher (Date): IBM (15 Aug 2006)

Summary: Introduces the SysML, a general-purpose, graphical modeling language for product and systems development.

Key Points:

- The SysML standard gives systems engineers and architects a much-needed way to collaborate using a common language that is specifically designed to support this process
- SysML will enable improved communications across development teams, while greatly enhancing our ability to manage ever-growing system complexity
- SysML is a modeling language for representing systems and product architectures, as well as their behavior and functionality
- Requirements have traditionally been represented as text, often accompanied by figures and drawings, and stored in files or databases; the requirements describe all the product functions, as well as the constraints under which these functions should be realized
- SysML allows the representation of requirements as model elements, therefore making requirements become an integral part of the product architecture
    - Offers a flexible means by which to represent text-based requirements of any nature (e.g., functional or non-functional) as well as the relationships between them
    - Requirements diagram contains both functional and non-functional requirements

Applicability to Capstone Project: SysML is the language chosen for the Requirements IPT to represent the system requirements.

Recommendations: The Methodology IPT recommends for the Requirements IPT to utilize SysML in generating the requirements artifacts (e.g., requirements diagram).

Research Topic: Systems Modeling Language (SysML)

Researcher: Eric Sarabia

Title: Using Systems Engineering Standards in an Architecture Framework

Author: Ian Bailey, Eurostep; Fatma Dandashi and Huei-Wan Ang, Mitre Corp; Dwayne Hardy, American Systems Corp

Publisher (Date): N/A

Summary:

- The Systems Modeling Language™ (SysML™) is a general-purpose systems modeling language (graphical) that will support specification, analysis, design, verification and validation of complex systems. It is a key enabler for transitioning the practice of systems engineering from being document-centric to a model-centric approach – i.e. model driven systems engineering. It is being developed by the SysML Partners as a joint initiative of INCOSE and the Object Management Group (OMG), and is defining extensions to the Unified Modeling Language™ (UML™). The requirements for SysML were developed as a cooperative effort between the OMG, INCOSE, and the ISO AP233 team, resulting in the issuance of the UML for Systems Engineering RFP in March 2003[11]. The SysML Partners group was formed to respond to these requirements, and includes broad representation from end-users, tool vendors, and liaisons with related initiatives.
- SysML is based on UML™ version 2. SysML will reuse and extend a subset of UML™ to provide a comprehensive set of concepts to model structure, behavior, properties, requirements, verification and other systems aspects of interest to systems engineers. Since SysML is being developed as a customization of UML™, it will define both visual (concrete) syntax and repository (meta model) semantics. SysML version 1.0 will address many of the requirements in the RFP and is projected for adoption by the OMG in Q4 2004. Future revisions are planned to address the full spectrum of requirements as well as lessons learned from its use. Additional information on SysML can be found at the SysML Partners website (http://www.sysml.org) and at the OMG SE DSIG site (http://syseng.omg.org).
- The principle for combining the SysML and DoDAF standards is relatively obvious. SysML provides the modeling notation, backed with the formal semantics of its meta model. The various DoDAF views and products are used to classify and present the operational and system descriptions. AP233 provides a neutral data exchange format for the data presented in the architecture framework including –the operational and systems modeling information, and the supporting text.

- Figure 1 illustrates a simple case of three DoDAF views, which are modeled in SysML and exchanged from one tool to another as an AP233 file.

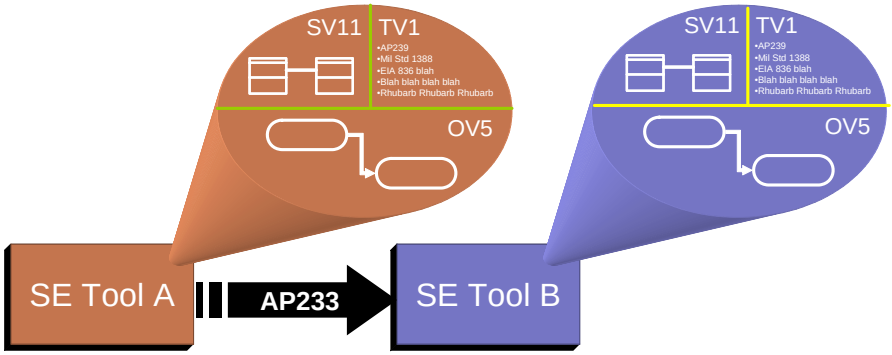


Figure 1 – AP233, DoDAF and SysML together

NOTE: REPLACE AP233 with AP233 FORMAT

- SysML offers the capabilities of UML and other models and representations that are required for DoDAF. The SysML and DoDAF specifications are underpinned by "meta-models". A meta-model defines the meaning of each element of the specification and the permissible relationships between those elements. The contents of the meta-models are comparable with the AP233 specification, and are seen as key drivers in the development of the AP233 ISO Standard.
- AP233 is independent of any systems modeling approach. Therefore, the AP233 format can be used for exchanging models between tools, which use different notations – e.g. an IDEF0 activity diagram can be exported as AP233 and re-imported into a UML tool as a SysML activity diagram. This enables collaborating team companies to all use their own preferred notations, but still be able to exchange information and prepare their DoD Architecture Framework in one common notation (e.g. SysML).

Key Points:

- DoDAF and SysML are indeed highly complimentary standards. The semantic overlap between them is quite significant, and where there are gaps, there are opportunities to enhance each standard.
- The enhanced capabilities of SysML help reduce ambiguity and add a richness of semantics to many DoDAF products.
- SysML emphasizes the SE domain
- SysML is NOT a specific tool or methodology

Applicability to Capstone Project: SysML will be used to showcase our capstone's Model Based Systems Engineering (MBSE) approach. This document also shows the traceability between MBSE, SysML and DoDAF.

Recommendations: The Requirements IPT will utilize SysML in developing our Requirements flow diagram as well as illustrating the use of MBSE to create requirements vs. text based requirements documentation.

Research Topic: Systems Modeling Language (SysML)

Researcher: Eric Sarabia

Title: OMG Systems Modeling Language (OMG SysML™), V1.0, OMG Available Specification DoD Architecture Framework

Author: Object Management Group, Inc.

Publisher / Date: September 2007

Summary:

- The SysML specification defines a general-purpose modeling language for systems engineering applications, called the OMG Systems Modeling Language (OMG SysML™). SysML supports the specification, analysis, design, and verification and validation of a broad range of complex systems. These systems may include hardware, software, information, processes, personnel, and facilities.
- Currently it is common practice for systems engineers to use a wide range of modeling languages, tools and techniques on large systems projects. In a manner similar to how UML unified the modeling languages used in the software industry, SysML is intended to unify the diverse modeling languages currently used by systems engineers. SysML reuses a subset of UML 2 and provides additional extensions needed to address the requirements in the UML for SE RFP. SysML uses the UML 2 extension mechanisms as further elaborated in Chapter 17, "Profiles & Model Libraries" of this specification as the primary mechanism to specify the extensions to UML 2. Since SysML uses UML 2 as its foundation, systems engineers modeling with SysML and software engineers modeling with UML 2 will be able to collaborate on models of software-intensive systems. This will improve communication among the various stakeholders who participate in the systems development process and promote interoperability among modeling tools. It is anticipated that SysML will be customized to model domain-specific applications, such as automotive, aerospace, communications, and information systems.
- SysML reuses a subset of UML 2 and provides additional extensions to satisfy the requirements of the language. This specification documents the language architecture in terms of the parts of UML 2 that are reused and the extensions to UML 2. The specification includes the concrete syntax (notation) for the complete language and specifies the extensions to UML 2. The reusable portion of the UML 2 specification is not included directly in the specification but is included by reference. The specification also provides examples of how the language can be used to solve common systems engineering problems.

Below are definitions of Activity and Use Case Diagrams used within SysML.

- **Activity Diagrams:** Activity modeling emphasizes the inputs, outputs, sequences, and conditions for coordinating other behaviors. It provides a flexible link to blocks owning those behaviors.

SysML extends control in activity diagrams as follows.

- In UML 2.1 Activities, control can only enable actions to start. SysML extends control to support disabling of actions that are already executing. This is accomplished by providing a model library with a type for control values that are treated like data

- A control value is an input or output of a control operator, which is how control acts as data. A control operator can represent a complex logical operation that transforms its inputs to produce an output that controls other actions

- **Use Case Diagrams:** The use case diagram describes the usage of a system (subject) by its actors (environment) to achieve a goal that is realized by the subject providing a set of services to selected actors. The use case can also be viewed as functionality and/ or capabilities that are accomplished through the interaction between the subject and its actors. Use case diagrams include the use case and actors and the associated communications between them. Actors represent classifier roles that are external to the system that may correspond to users, systems, and or other environmental entities. They may interact either directly or indirectly with the system. The actors are often specialized to represent a taxonomy of user types or external systems.
- The use case diagram is a method for describing the usages of the system. The association between the actors and the use case represent the communications that occurs between the actors and the subject to accomplish the functionality associated with the use case. The subject of the use case can be represented via a system boundary. The use cases that are enclosed in the system boundary represent functionality that is realized by behaviors such as activity diagrams, sequence diagrams, and state machine diagrams.
- The use case relationships are "communication," "include," "extend," and "generalization." Actors are connected to use cases via communication paths, which are represented by an association relationship. The "include" relationship provides a mechanism for factoring out common functionality that is shared among multiple use cases, and is always performed as part of the base use case. The "extend" relationship provides optional functionality, which extends the base use case at defined extension points under specified conditions. The "generalization" relationship provides a mechanism to specify variants of the base use case. The use cases are often organized into packages with the corresponding dependencies between the use cases in the packages.

Key Points:

- SysML provides modeling constructs to represent text-based requirements and relate them to other modeling elements. The requirements diagram described in this chapter can depict the requirements in graphical, tabular, or tree structure format. A requirement can also appear on other diagrams to show its relationship to other modeling elements. The requirements modeling constructs are intended to

provide a bridge between traditional requirements management tools and the other SysML models.

- There is a real need for requirement reuse across product families and projects. Typical scenarios are regulatory, statutory, or contractual requirements that are applicable across products and/or projects and requirements that are reused across product families (versions/variants). In these cases, one would like to be able to reference a requirement, or requirement set in multiple contexts with updates to the original requirements propagated to the reused requirement(s).
- The use case diagram is a method for describing the usages of the system. The association between the actors and the use case represent the communications that occurs between the actors and the subject to accomplish the functionality associated with the use case.

Applicability to Capstone Project: SysML will be used to develop our Model Based Systems Engineering "models" which will showcase our requirements as models vs. using text based documentation.

Recommendations: The Requirements IPT will develop Use Case diagrams to show a top level view of the three mission threads (Self Defense, Limited Area Defense, & Surveillance) and the functionality and/ or capabilities that are accomplished through the interaction between the subject and its actors.

Research Topic: Systems Modeling Language (SysML)

Researcher: Jonathan Mendiola

Title: An overview of the Systems Modeling Language for product and systems development -- Part 1: Requirements, use case, and test-case modeling

Author: Laurent Balmelli, PhD

Publisher (Date): T.J. Watson Research Center and Tokyo Research Laboratory, IBM (15 August 2006)

<http://www.ibm.com/developerworks/rational/library/aug06/balmelli/>

Summary:

- Today's competitive pressures and other market forces drive manufacturing companies to improve the efficiency with which they design and manufacture products and systems. Across the product lifecycle, one area where there has been a notorious lack of efficiency support is the conceptual stage, during which the functional architecture (and sometimes the physical architecture) is decided upon.
- The conceptual stage follows the transformation of customer needs into product functions and use cases, and precedes the design of these functions across the engineering disciplines (for example, mechanical, electrical, software, etc.). A lack of support during product conceptualization makes it difficult to efficiently trace the realization of requirements in the product. The lack of a formal representation for concepts also results in an inadequate ability to make decisions at the level of systems in the product, such as during feasibility studies. Moreover, the lack of a clear vision of the product architecture hinders team understanding and communication, which in turn often increases the risk of integration issues. It is these and other challenges confronted during the conceptual phase of product and system development that SysML is designed to mitigate.
- In this Part 1 of a three-part article, the author explains the basic purpose and value of SysML, relate it to Unified Modeling Language (UML), and describe its Requirements diagram, Use-Case diagram, and test-case representations.

Key Points:

- When modeling a system, an important primary task is to decide what belongs to the system and what does not. The Context diagram is an informal (in the sense that it does not carry precise semantics) way to represent the boundaries of the system.
- SysML allows the representation of requirements as model elements. Hence, requirements become an integral part of the product architecture. The language offers a flexible means by which to represent text-based requirements of any nature (e.g., functional or non-functional) as well as the relationships between them.

- SysML provides a Use-Case diagram that is inherited from UML 2.0 without modifications.

Applicability to Capstone Project: Will be used in development of SysML requirements diagrams that will be eventually input into CORE and provide high-level requirements to the rest of the IPTs.

Recommendations: Requirements IPT to use this as a guide to building requirements diagrams for use by other IPTs.

Research Topic: Systems Modeling Language (SysML)

Researcher: Jonathan Mendiola

Title: Requirements Specification and Modeling through SysML (JM)

Author: Michel dos Santos Soares, Jos Vrancken

Publisher (Date): IEEE (2007)

Summary:

- Use Case diagrams are well known for their use to specify and describe system requirements. From initial system requirements documents, use cases can be derived representing several scenarios. These scenarios can later be detailed in different ways, as for example, through informal descriptions. In this paper, system requirements are first specified using the SysML requirements diagram and later by use cases. The main goal is to fill the gap between documents written in natural language and use cases by modeling requirements in a graphical and tabular way, which can improve the requirements representation. Also, the relationship between requirements is enhanced. An example of a real time distributed system is given to illustrate the approach.

Key Points:

- Principle problem of gathering requirements is the ambiguity of natural languages.
- One good example of a semi-formal notation for requirements specification is the UML use case diagram.
    - But two problems still arise: the natural language is still needed, and the diagrams are sometimes not user-friendly.
- SysML is a new systems modeling language that supports specification, analysis, design, verification and validation of a broad range of complex systems.
- The basic difference is that SysML was built from scratch to support System Engineering, which means that some specific software oriented constructs, not necessary to systems modeling, were avoided.
- It is presented that the SysML requirements diagram can fill the gap between natural language based specifications, too ambiguous and informal, and UML use case diagrams.
- Use cases are visual, which is good to system analysts and the users to better comprehend the system. Also, a use case can be detailed, and its several scenarios described in natural language, pseudo-code or other UML diagrams.
- The SysML requirements diagram helps in better organizing requirements, and also shows explicitly the various kinds of relationships between different requirements.
- One important quality factor in systems design is to know what happens to a requirement during system modeling and specification. This activity is known as requirements traceability.

  - One way to manage the requirements traceability in SysML is within requirements tables.
- SysML allows the representation of requirements as model elements, which mean that requirements are part of the system architecture.
- The SysML requirements diagram allows several ways to represent requirements relationships. These include relationships for defining requirements hierarchy, deriving requirements, satisfying requirements, verifying requirements and refining requirements.
- The purpose of the use case diagram is to describe the usage of a system by its actors in order to achieve a goal. The use case can also be viewed as functionality and/or capabilities that are accomplished through the interaction between the subject and its actors.

Applicability to Capstone Project: Will be used in development of SysML requirements diagrams that will be eventually input into CORE and provide high-level requirements to the rest of the IPTs.

Recommendations: Requirements IPT use the guidance in this article to create requirements diagrams and use case diagrams.

Research Topic: Systems Modeling Language (SysML)

Researcher: Heng Sysavath

Title: The SysML Modeling Language

Author: Matthew Hause

Publisher (Date): 18-20 September 2006

Summary:

- The article provides a good summary of SysML extension into UML 2.0. The author outlines some of the additions and modifications made to UML 2.0 to support OMG SysML. They include the structure, behavior, requirement, and parametric diagrams. Examples of each type of SysML diagram and its usage are illustrated in the articles.

Key Points

- The requirements model is not meant to replace external requirements management tools, but is meant to be used in conjunction with them to increase traceability within UML models.
- There are two structured diagram types: Block Definition Diagram (BDD), and Internal Block Diagram (IBD) is used to describe block internals.
- Parametric diagrams are used to describe constraints on system properties to support engineering analysis.
- Behavioral diagrams include the activity diagram, sequence diagram, state machine diagram and use case diagram.

Applicability to Capstone Project: SysML requirements diagrams will be developed which in turn will be input into CORE to provide high-level requirements to the other IPTs.

Recommendations: The requirement IPT can use this article as guidance in the development of use case diagrams, requirement diagrams, and context diagrams.

Research Topic: Systems Modeling Language (SysML)

Researcher: Heng Sysavath

Title: OMG Systems Modeling Language (OMG SysML™), V1.0

Author: Object Management Group, Inc. (OMG)

Publisher (Date): September 2007

Summary:

- SysML is designed to provide simple but powerful constructs for modeling a wide range of systems engineering problems. It is particularly effective in specifying requirements, structure, behavior, and allocations and constraints on system properties to support engineering analysis. The language is intended to support multiple processes and methods such as structured, object-oriented, and others, but each methodology may impose additional constraints on how a construct or diagram kind may be used. The purpose of this document is to specify the Systems Modeling Language (SysML), a new general-purpose modeling language for systems engineering that satisfies the requirements of the UML for Systems Engineering. Its intent is to specify the language so that systems engineering modelers may learn to apply and use SysML

Key Points:

- SysML is intended to be supported by two evolving interoperability standards including the OMG XMI 2.1 model interchange standard for UML 2 modeling tools and the ISO 10303 STEP AP233 data interchange standard for systems engineering tools.
- SysML reuses a subset of UML 2 and provides additional extensions needed to address requirements in the UML for Systems Engineering
- The SysML language reuses and extends many of the packages from UML.
- The SysML user model is created by instantiating the metaclasses and applying the stereotypes specified in the SysML profile and subclassing the model elements in the SysML model library.

Applicability to Capstone Project: SysML requirements diagrams will be developed which in turn will be input into CORE to provide high-level requirements to the other IPTs.

Recommendations: The requirement IPT can use this article as guidance in the development of use case diagrams, requirement diagrams, and context diagrams.

Research Topic: Systems Modeling Language (SysML)

Researcher: Mindy Wentland

Title: An Overview of the Systems Modeling Language for product and systems development -- Part 3: Modeling system behavior

Author: Laurent Balmelli, PhD

Publisher: T.J. Watson Research Center and Tokyo Research Laboratory, IBM (16 Oct 2006)

Summary:

- This multipart article introduces SysML, a standard modeling language for systems engineering. SysML gives systems engineers and architects a much-needed way to collaborate using a common language. By enabling an electronic representation of product design, SysML improves communication among development teams; helps manage system complexity, and can serve as the basis for analytics to drive faster and more effective decision-making across all phases of the systems development lifecycle.
- In Part 3, the author explains how SysML can be used to model the operating behavior of a product. The author makes reference to a real-life example of an embedded system, a Rain Sensing Wiper (RSW) for an automotive application.

Key Points:

- SysML Behavior Models
  - Behavior of a system equates to realizing its use cases under a specified set of nonfunctional constraints
  - SysML offers three types of behavioral constructs: *Interactions*, *State Machine*, and *Activities*
  - Several behavioral models from UML are not reused in SysML, either for the sake of simplicity or because of some maturity concerns

Applicability to Capstone Project: Will be used in designing SysML behavior models for software architecture and repository.

Recommendations: SW IPT to use in creating SysML behavior models for software architecture and repository.

Research Topic: System Standards

Researcher: Caleb Vajdos

Title: SE ASNE Paper, Applying Open Architecture Concepts to Mission and Ship Systems

Author: Ian Bailey, Eurostep; Fatma Dandashi and Huei-Wan Ang, Mitre Corp;

Dwayne Hardy, American Systems Corp

Publisher: Unknown / Unknown (appears to be >2006)

Summary: In recent years, three standards have begun to emerge which support the systems engineering process. The standards are concerned with the information that systems engineers work with – requirements, architecture, behavioral models, interfacing, verification, validation, etc. The standards are complementary, and the purpose of this paper is to examine how they can be used together. The standards are: AP233, DoDAF, and SysML. Developing today's complex systems typically requires engineering teams that are distributed in time and space and that are often composed of many companies, each with their own culture, methods and tools. Effective collaboration requires agreement and a thorough understanding of the various work assignments and resulting products. Many of these products pertain to important systems engineering considerations such as requirements and architectures that apply throughout the entire life cycle of the system of interest. So it is critical that the system information contained in these work products is accurately captured and 'readable' by appropriate team members in a timely manner. Today, this information is generally captured in an array of tools where each is only concerned with a portion of systems engineering data and can't share its data with other tools. To mitigate this situation, collaborating organizations are usually forced to either adopt a common set of tools or develop a unique, bi-directional interface between many of the tools that each organization normally uses. This can be an expensive and untimely approach to data exchange between team members. The standards permit an alternate approach that should be more affordable and timely. In addition, if the tools that each participating organization is currently using implement the standards discussed in this paper, this approach should allow: data exchanges between tools of different types, common representations and improved communications among systems engineers and other engineering disciplines, and consistent descriptions of system architectures

Key Points: Explains why the use of standards is import and how these AP233, DoDAF, and SysML can work together to improve communications in systems engineering and remove the dependence on specific software applications.

Applicability to Capstone Project:

- Justifies the use of SysML
- Shows explicit links between SysML and DoDAF products

- Reinforces the use of MBSE

Recommendations: Very Informative however is slightly dated. SysML and ISO10303 AP233 are both approved standards now. Additional info from these standards could be useful to accompany this article.

Research Topic: System Standards

Researcher: Caleb Vajdos

Title: Supporting Systems Engineering with Methods and Tools: A Case Study

Author: Jock Rader and Leslie Haggerty

Publisher: Unknown

Summary: Many projects have applied the Hatley-Pirbhai real-time structured analysis method to the definition and analysis of system/software requirements. Application of the method results in the generation of a functional requirements model, which includes data/control flow diagrams, plus process and control specifications. Although infrequently applied, the method also defines an architecture model, which specifies the physical components and the physical interconnect channels for the system. After first providing a brief overview of the architecture and requirements models, we discuss how the Hatley-Pirbhai methodology is being used by system engineers on a real-time embedded avionics program. We also discuss the set of automated tools used to support the methods, and how both methods and tools were tailored and enhanced. Lastly, we describe operational experience and some difficult lessons learned.

Key Points: Discusses the Hatley-Pirbhai method and some lessons learned.

Applicability to Capstone Project:

- Gives examples of the use of the Hatley-Pirbhai method.
- Shows a good approach to allocate system requirements to hardware and software.
- Provides summarized steps to H-P process

Recommendations: Good additional source to back up use of H-P real-time process.

APPENDIX E: THE DODAF AS A SPECIFICATION MODEL

Abstract:

Modern military operations require interoperability. However, joint forces are comprised of multiple services aided by coalition partners covering a mix of systems, operating procedures, standards, protocols, and languages, which in turn produces an interoperability quandary.

Recognizing the interoperability problems experienced in previous U.S. joint operations in the 1980's and 1990's, the DoD embarked on a path to develop system architectures to address interoperability by developing version 1.0 of the *C4ISR Architecture Framework* in 1996. In order to emphasize the applicability of the framework beyond the C4ISR community, the third version was renamed the DoD Architecture Framework (DoDAF) v. 1.0 and released in August 2003. The DoD approved DoDAF v. 1.0 for official use on February 9, 2004.

During the same initial time period (1994), Secretary of Defense, William Perry initiated acquisition reform doing away with the MIL-STD specifications for systems acquisition in favor of "performance based" requirements and Commercial Off The Self (COTS) technology. The ultimate goal of MIL-SPEC & MIL-STD reform was intended to reduce the current (1990's) weapons systems development time of 12-15 years in order to keep pace with commercial technologies.

Nonetheless, "*even though it was argued they* (MIL-STD's) *represented generations of technical best practices, oftentimes painfully learned as a result of design and production mistakes*"[12]. Expressed another way by Jeffery O. Grady in 1995:

> One key element of acquisition reform was to eliminate . . . contractually dictated prescriptive "how-to" instructions or processes used by contractors. For a decade, we have limited and reduced our use of specifications and standards in RFPs, proposal evaluations, contractor performance assessments, and on contracts as compliance documents. The unintentional result was that technical baselines and processes were compromised. With the turnover, consolidations, and retirement of many industry and

[12] Meshel, David, David Davis, William Tosney, and Frank L. Knight, 2007. Implementation of Revitalized Engineering Specifications and Standards for National Security Space Programs, *U.S. Air Force T&E Days 13 – 15 Feb 2007, Destin, Florida*

> government personnel, we have hampered our ability to pass on lessons learned from generation to generation.[13]

To overcome some of the unintentional consequences of the MIL-SPEC & MIL-STD reform, the DoD instituted the Systems Engineering Revitalization Effort whose purpose is to "Help drive good systems engineering practice back into the way we do business."[14] This effort includes revisiting MIL-SPEC & MIL-STD's to aid the Systems Engineering process.

This paper will examine the similarities between the DoDAF and MIL-STD 490A/B. It will argue that, intended or not, the two approaches map directly to each other providing a basis for a current, relevant approach to developing complex systems specifications. This paper will also propose a process that supports this integration of the two concepts.

[13] Grady, Jeffrey O. 1995. *System Engineering Planning and Enterprise Identity*, Boca Raton Fla.: CRC Press

[14] Skalamera, Robert, J. 2004. USD(AT&L) Imperatives, Implementing OSD Systems Engineering Policy, *OUSD (AT&L)*

Interoperability Issues as the driver for System Architectures

U.S. joint operations in the 1980's and 1990's including Operations Urgent Fury in Grenada, Joint Endeavor in Bosnia, and Desert Shield/Desert Storm revealed the shortfalls of interoperability among U.S. forces;

- The need for common and open standards
- Interface systems never imagined to have the need to communicate
- Integration leads to what you get vs. what you need.[15]

To overcome the above difficulties, systems must be fully interconnected and interoperable with a reliable infrastructure.

The Joint Interoperability Test Command (JITC) provides this definition of *interoperability*:

> *"The ability of systems, units, or forces to provide services to and accept services from other systems, units, or forces, and to use the services so exchanged to enable them to operate effectively together."*

The reliable infrastructure is indicative of an architecture. One definition of *architecture* is:

> *"The structure of components, their relationships, and the principles and guidelines governing their design and evolution over time."*[16]

Brief History of DoD Architecture Development

In the mid 1990s the DoD determined that a common approach was needed for describing its architectures, so that DoD systems could efficiently communicate and interoperate during joint and multinational operations.

The Federal Enterprise Architecture Framework (FEAF) was the first attempt to document the ground rules for various government agencies to develop their own architectures to be compliant with the intent of Clinger-Cohen act of 1996 mandate to

[15] DiMario, Michael, Brian Sauser, and Dinesh Verma, 2006. System of Systems Characteristics and Interoperability in Joint Command and Control, *2nd Annual System of Systems Engineering Conference, Ft Belvoir, VA*

[16] CJCSI 3170.01C. 2003.

develop an enterprise information technology architecture. The DoD adopted the FEAF and tailored it into the C4ISR Architecture Framework (AF). Subsequently, the C4ISR Integration Task Force was formed and developed version 1.0 of the *C4ISR Architecture Framework* in 1996. The C4ISR Architecture Working Group completed version 2.0 in 1997.

> *The purpose of C4ISR architectures is to improve capabilities by enabling the quick synthesis of "go-to-war" requirements with sound investments leading to the rapid employment of improved operational capabilities, and enabling the efficient engineering of warrior systems.*[17]

After working with the first two versions of C4ISR framework, and recognizing the need to strengthen it prior to adoption, the DoD began work on a third version. In order to emphasize the applicability of the framework beyond the C4ISR community, the third version was renamed the DoD Architecture Framework (DoDAF) v. 1.0 and released in August 2003. The DoD approved DoDAF v. 1.0 for official use on February 9, 2004.

> *The purpose of the Department of Defense (DoD) Architecture Framework, Version 1.0, is to provide guidance for describing architectures for both warfighting operations and business operations and processes. The Framework provides the guidance, rules, and product descriptions for developing and presenting architecture descriptions that ensure a common denominator for understanding, comparing, and integrating Families of Systems (FoS), Systems of Systems (SoS), and interoperating and interacting architectures.*[18]

All DoD architectures developed or approved subsequent to December 1, 2003 must comply with this framework. Architectures developed prior to this date must comply with DoDAF upon their next version update.[19]

The principal objective of the DoD Architecture Framework (DoDAF) is to define a common approach for DoD architecture description, development, presentation, and integration. By characterizing the form, function, and rules governing systems, architecture frameworks (1) serve as a communication tool to stakeholder communities with different views of the system and (2) facilitate comparative evaluation across architectures.

---

[17] C4ISR Architecture Framework,Version 2.0. 1997.

[18] DoD Architecture Framework,Version 1.0. 2003.

[19] Sibbald, Chris and Cris Kobryn, 2004. Modeling DoDAF Compliant Architectures, The Telelogic Approach for Complying with the DoD Architectural Framework, *A Telelogic White Paper*

The framework is intended to ensure that architecture descriptions can be compared and related across organizational and mission boundaries. Improved interoperability of weapons systems is expected to be achieved through the DoDAF.

Acquisition Reform and improving the Systems Engineering process by tailoring MIL-STD's

During the same initial time period (1994), Secretary of Defense, William Perry initiated acquisition reform ("*Specifications & Standards – A New Way of Doing Business*") doing away with the MIL-STD specifications for systems acquisition in favor of "performance based" requirements and Commercial Off The Self (COTS) technology. The ultimate goal of MIL-SPEC & MIL-STD reform was intended to reduce the current (1990's) weapons systems development time of 12-15 years in order to keep pace with commercial technologies.

Nonetheless, "*even though it was argued they* (MIL-STD's) *represented generations of technical best practices, oftentimes painfully learned as a result of design and production mistakes.*"[20] Expressed another way by Jeffery O. Grady in 1995:

> One key element of acquisition reform was to eliminate . . . contractually dictated prescriptive "how-to" instructions or processes used by contractors. For a decade, we have limited and reduced our use of specifications and standards in RFPs, proposal evaluations, contractor performance assessments, and on contracts as compliance documents. The unintentional result was that technical baselines and processes were compromised. With the turnover, consolidations, and retirement of many industry and government personnel, we have hampered our ability to pass on lessons learned from generation to generation.[21]

To overcome some of the unintentional consequences of the MIL-SPEC & MIL-STD reform, the U.S. Air Force Space & Missile Systems Command (SMC) established the Systems Engineering Revitalization Effort whose purpose is to ""Help drive good systems engineering practice back into the way we do business."[22]

---

[20] Meshel, David, David Davis, William Tosney, and Frank L. Knight, 2007. Implementation of Revitalized Engineering Specifications and Standards for National Security Space Programs, *U.S. Air Force T&E Days 13 – 15 Feb 2007, Destin, Florida*

[21] Grady, Jeffrey O. 1995. *System Engineering Planning and Enterprise Identity.* Boca Raton Fla.: CRC Press

[22] Skalamera, Robert, J. 2004. USD(AT&L) Imperatives, Implementing OSD Systems Engineering Policy, *OUSD (AT&L)*

> In the early to mid 1990s, as a result of the growing problems with military specs and standards, DoD sought to restrict the use of such documents. The resulting effort, which became known as the Military Specifications and Standards Reform Program (MSSRP) soon grew into a wider acquisition reform initiative. However, acquisition reform had unfortunate unintended consequences.
>
> The SMC Systems Engineering Revitalization effort was established to deal with those consequences. As noted in a recent policy letter from the SMC Commander, "One key element of acquisition reform was to eliminate . . . contractually dictated prescriptive "how-to" instructions or processes used by contractors. For a decade, we have limited and reduced our use of specifications and standards in RFPs, proposal evaluations, contractor performance assessments, and on contracts as compliance documents. The unintentional result was that technical baselines and processes were compromised. With the turnover, consolidations, and retirement of many industry and government personnel, we have hampered our ability to pass on lessons learned from generation to generation.[23]

There is no intent to return to the pre-acquisition reform approach of using an excessive number of specifications and standards and prescribing detailed processes. A list of high-priority critical specifications and standards is being reviewed and established for appropriate use in the acquisition process." Many of the specifications and standards selected for the SMC technical baseline have been tailored and in some cases may be updated or revised. All should be reviewed and further tailored as necessary to bring them in line with the objectives of each contractual action. "Tailored specifications and standards and contractor responses must be relevant and hold members of the government industrial partnership appropriately accountable to sound technical disciplines. They should be used in new acquisitions and can be used on legacy programs to modify contracts if benefits can be shown to warrant the changes. They will be used in a less prescriptive manner than in the past. For example, the contractor may propose the listed specification/standard or another government, industry, technical society, international or company version provided it is comparable in vigor and effectiveness. Proof of this comparability must be provided. Acceptable responses will be put on contract as compliance documents with follow-up at program reviews to ensure they are appropriately addressed."[24]

---

[23] SMC Systems Engineering Primer & Handbook. 2005. *Space & Missile Systems Air Force 3rd Edition*. pg 37

[24] SMC Systems Engineering Primer & Handbook. 2005. *Space & Missile Systems Air Force 3rd Edition*. pg 37

Capabilities based process replaces the Requirements Generation System

The Joint Capabilities Integration and Development System (JCIDS) were developed to address shortfalls identified by the Joint Chiefs of Staff in the requirements generation system. These shortfalls were identified as: not considering new programs in the context of other programs, not sufficiently considering combined service requirements and effectively prioritizing joint service requirements, and not accomplishing sufficient analysis.

The Chairman of the Joint Chiefs of Staff (CJCS) approved the new JCIDS process on June 24, 2003 with the release of CJCS Instruction 3170.01C, which provides a top-level description of JCIDS and outlines the organizational responsibilities of key players and deliberative bodies involved in the process. CJCS Manual (CJCSM) 3170.01 defines performance attributes, key performance parameters, validation and approval processes, and associated document content. Subsequent versions of the document continue to refine and evolve the JCIDS process.

JCIDS replaces the Requirements Generation System (RGS), in order to identify needed future joint concepts for the armed services, and changes many of the terms associated with that system. Mission Need Statements (MNS), Operational Requirements Documents (ORDs), and Combat Mission Needs Statements (C-MNS) have been replaced with several new documents to satisfy similar requirements in the new process. An Initial Capabilities Document (ICD) replaces the MNS, a Capability Development Document (CDD) replaces the Milestone B ORD, a Capability Production Document (CPD) replaces the Milestone C ORD, and the Combat Capability Document (CCD) replaces the Combat Mission Needs Statement (C-MNS). CJCS Manual 3170.01 further defines performance attributes, key performance parameters, validation and approval processes, and associated documents.

This process strives to make certain that future capabilities are “born” joint, meaning that systems will enable and enhance joint operations from their inception, whereas the former requirements generation system was Service-centric with joint interoperability as an afterthought. JCIDS is conducted in a top-down manner, with functionally-focused teams centered on future capabilities and effects for the Joint Force. The process is designed to better identify gaps in capabilities and achieve joint solutions to fill those gaps.

The figure below is taken from DoDI 5000.02 (Operation of the Defense Acquisition System). DoDI 5000.02 generates the management framework for translating capability needs and technology opportunities into acquisition programs as shown below.

Figure 1. The Defense Acquisition Management System.

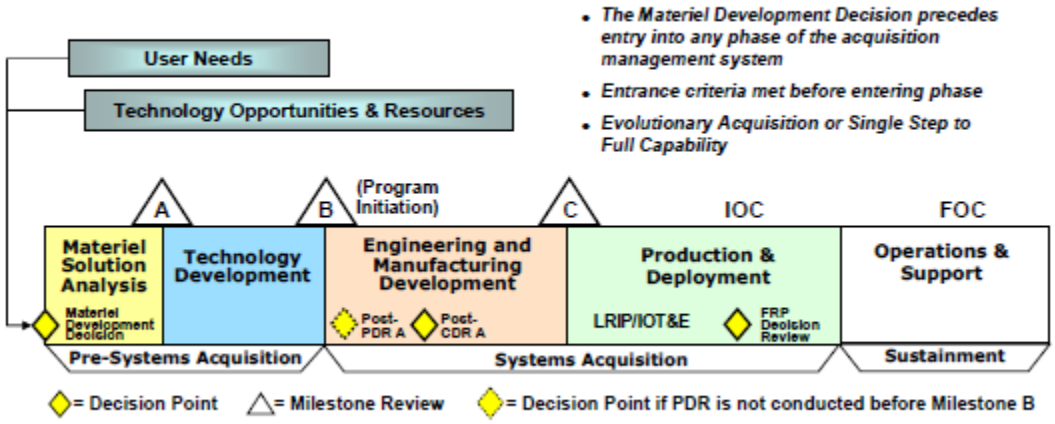


Figure 1: The Defense Acquisition Management System

For such programs, at each acquisition milestone, Joint Capabilities Integration and Development System documents (ICD, CDD and CPD) are provided to guide the subsequent development, production and testing of the program.

The Figure 2[25] on the next page highlights the differences between the "old" Requirements Generation System acquisition process and the "new" acquisition process.

Replacing Milestones 0, I, II and III are Milestones A, B, and C. Milestones A and B are essentially equivalent to the old Milestones 0 and II respectively. The new Milestone C, the Commitment to Low-Rate Initial Production and to Produce and Deploy Systems, occurs ahead of the old Milestone III. The old Milestone III Production and Deployment Decision is no longer a Milestone but is now made at a Full-Rate Production Decision Review that occurs during Phase C, the Production and Deployment Phase.[26]

[25] NAVAIR Acquisition Guide. 2006/2007. pg 29

[26] SPAWAR Acquisition Program Structure Guide Vol I Version 1.0. 2001. pg 10

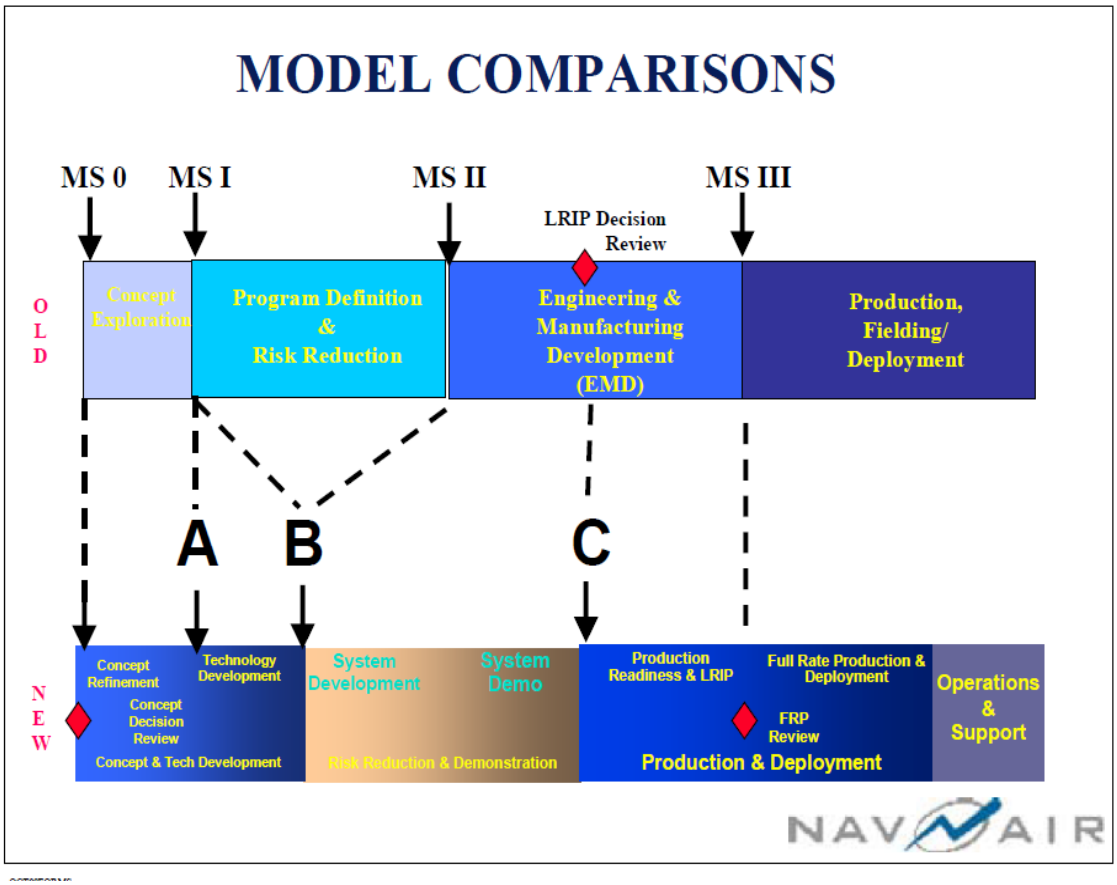


Figure 2: Acquisition Model Comparisons

Baselines govern each level of Development

System development phasing is demarcated by milestones established by DoDI 5000.02. The Defense Acquisition Management System development phasing serves two purposes;

> it controls the design effort and is the major connection between the technical management effort and the overall acquisition effort. It controls the design effort by developing design baselines that govern each level of development. It interfaces with acquisition management by providing key events in the development process, where design viability can be assessed. The viability of the baselines developed is a major input for acquisition management Milestone (MS) decisions. As a result, the timing and coordination between technical development phasing and the acquisition schedule is critical to maintain a healthy acquisition program.[27]

[27] Systems Engineering Fundamentals. 2001. *Defense Acquisition University Press*. pg 4

The relationship between Design Baselines and MIL-STD-490A Specifications

In design baseline terms the MIL-STD-490A specifications A-Spec, B-Spec and C-Spec are designated Functional Baseline, Allocated Baseline and Product Baseline respectively.

The following diagram and definitions from the Defense Acquisition University Systems Engineering Fundamentals illustrates the similarities.

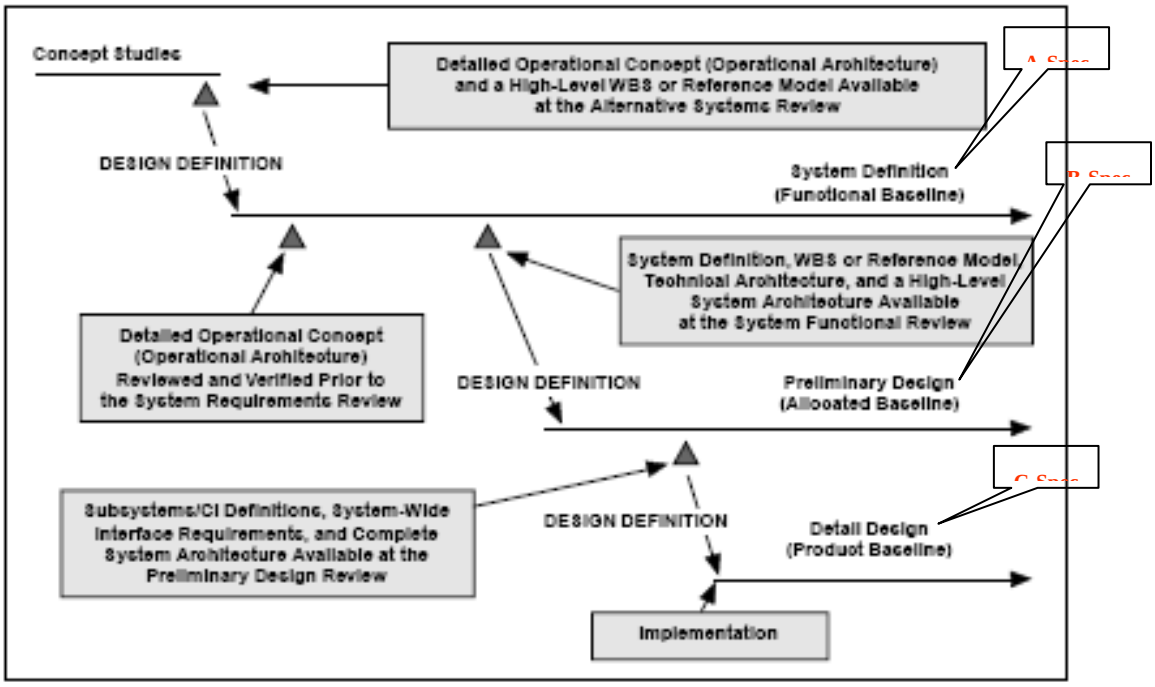


Figure 3: Systems Development Phasing[28]

[28] Systems Engineering Fundamentals. 2001. *Defense Acquisition University Press*. pg 167

Functional Baseline

Documentation describing system/segment functional characteristics and the verification required to demonstrate the achievement of those specified functional characteristics. The system or segment specification establishes the functional baseline. See System Specification.

Item Performance Specification

A program-unique specification usually approved as part of the allocated baseline (formerly called a "B specification" or "development specification"). States all necessary design requirements of a Configuration Item (CI) in terms of performance. Essential physical constraints are included. Item performance specifications state requirements for the development of items below the system level. They specify all of the required item functional characteristics and the tests required to demonstrate achievement of those characteristics.

Item Detail Specification

A program-unique specification usually approved as part of the product baseline (formerly called a "C specification" or "product specification"). Item detail specifications are applicable to any item below the system level, and define performance, functional and physical requirements, and design details of a Configuration Item (CI). Item detail specifications are intended to be used for the procurement of items, including computer programs.

NOTICE OF CANCELLATION

MIL-STD-490A, dated 4 June 1985 cancelled 31 August 1995.

Requirements for the preparation of program-peculiar specifications have been incorporated into MIL-STD-961 "Department of Defense Standard Practice for Defense Specifications."

With the cancellation of MIL-STD-490A, the A-Spec, B-Spec and C-Spec assumed the names; Performance Specification and Detail Specification as shown in the following section from MIL-STD-961E and the Defense Acquisition University Systems Engineering Fundamentals.

NOTE:

The specification definitions of MIL-STD-490A are included as an appendix in this report.

MIL-STD-961E Specifications

MIL-STD-961E establishes uniform practices for specification preparation regarding the format and content of system, configuration item, software, process and material specifications. These program-unique specifications are developed through application of the systems engineering process.

3.35 Performance specification. A specification that states requirements in terms of the required results with criteria for verifying compliance, but without stating the methods for achieving the required results. A performance specification defines the functional requirements for the item, the environment in which it must operate, and interface and interchangeability characteristics. Both defense specifications and program-unique specifications may be designated as a performance specification.

5.8.1 Performance specifications. Requirements in performance specifications shall prescribe the item's required performance, operating requirements, operational environment, interfaces, and interoperability requirements. Performance specifications shall not prescribe how a performance requirement is to be achieved by requiring the use of specific materials or parts or detailed requirements for the design or construction of the item beyond those needed to ensure interchangeability with existing items. For a general specification to be designated as a "Performance Specification," the requirements in its specification sheets or MS sheets shall also be stated as performance requirements. A general specification shall not have a mixture of performance and detail specification sheets. The SD-15 provides guidance on writing performance requirements.

3.13 Detail specification. A specification that specifies design requirements, such as materials to be used, how a requirement is to be achieved, or how an item is to be fabricated or constructed. A specification that contains both performance and detail requirements is still considered a detail specification. Both defense specifications and program-unique specifications may be designated as a detail specification.

5.8.2 Detail specifications. Detail specifications may consist of all detail requirements or a blend of performance and detail requirements. To the greatest extent possible, requirements in detail specifications shall be in terms of performance. Detail specifications shall specify materials, design or construction requirements, or "how to" requirements only to the extent necessary to ensure the adequacy, safety, and interchangeability of the item being acquired.

Specification Definitions from the Defense Acquisition University (Circa 2001)

Performance Specifications

Performance Specifications state requirements in terms of the required results with criteria for verifying compliance, but without stating the methods for achieving the required results. In general, performance specifications define products in terms of functions, performance, and interface requirements. They define the functional requirements for the item, the environment in which it must operate, and interface and interchangeability characteristics. The contractor is provided the flexibility to decide how the requirements are best achieved, subject to the constraints imposed by the government, typically through interface requirements. System Specifications and Item Performance Specifications are examples of performance specifications.

This Defense Acquisition University performance specification description is a higher abstraction of the MIL-STD-490A 'A-Spec' and 'B-Spec'.

NOTE: MIL-STD-961E, which superseded MIL-STD-490A, uses the designator 'PRF' for this type specification.

Detail Specifications

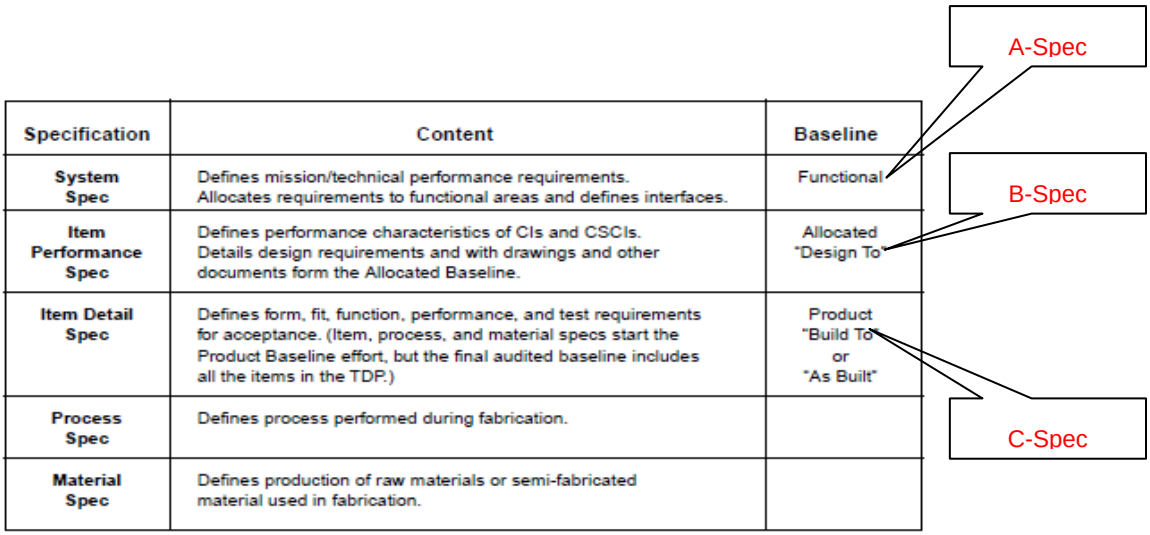
Detail Specifications, such as Item Detail, Material and Process Specifications, provide design requirements. This can include materials to be used, how a requirement is to be achieved, or how an item is to be fabricated or constructed. If a specification contains both performance and detail requirements, it is considered a Detail Specification, with the following exception: Interface and interchangeability requirements in Performance Specifications may be expressed in detailed terms. For example, a Performance Specification for shoes would specify size requirements in detailed terms, but material or method of construction would be stated in performance terms.[29]

This Defense Acquisition University performance specification description is likewise a higher abstraction of the MIL-STD-490A 'C-Spec'.

NOTE: MIL-STD-961E, which superseded MIL-STD-490A, uses the designator 'DTL' for this type specification.

[29] Systems Engineering Fundamentals. 2001. *Defense Acquisition University Press*. Pg 78

Specification Types and their relationship to MIL-STD-490A



Specification	Content	Baseline
System Spec	Defines mission/technical performance requirements. Allocates requirements to functional areas and defines interfaces.	Functional
Item Performance Spec	Defines performance characteristics of CIs and CSCIs. Details design requirements and with drawings and other documents form the Allocated Baseline.	Allocated "Design To"
Item Detail Spec	Defines form, fit, function, performance, and test requirements for acceptance. (Item, process, and material specs start the Product Baseline effort, but the final audited baseline includes all the items in the TDP.)	Product "Build To" or "As Built"
Process Spec	Defines process performed during fabrication.	
Material Spec	Defines production of raw materials or semi-fabricated material used in fabrication.	

Figure 4: Specification Types

The above table[30] describes the various specification types and their associated baseline.

[30] Systems Engineering Fundamentals. 2001. *Defense Acquisition University Press*. Pg 75

The Systems Engineering 'Vee' Process and MIL-STD-490A Specifications

The following is a high-level summary[31] of the Systems Engineering 'Vee' Process and its role in the "new" acquisition environment. The Vee model provides an orderly approach to implementing and integrating the systems engineering processes during each acquisition phase.

- The traditional Systems Engineering process is focused on system design
- At a conceptual level, it is a sequential process with clearly defined steps and milestones (in reality, many interactions and feedback loops)
- It is exemplified by the "V" Process model
- To cope with uncertainty in requirements and the complexity of Systems-of-Systems, an architecture based approach is now mandated
- The architecture based approach, while more complex up front, facilitates integration and interoperability
- It forces users, acquirers, and developers to work together at all stages

Based on the definitions provided in the previous section, and, as illustrated in the figure below, MIL-STD-490A specifications are an integral part of the Systems Engineering process.

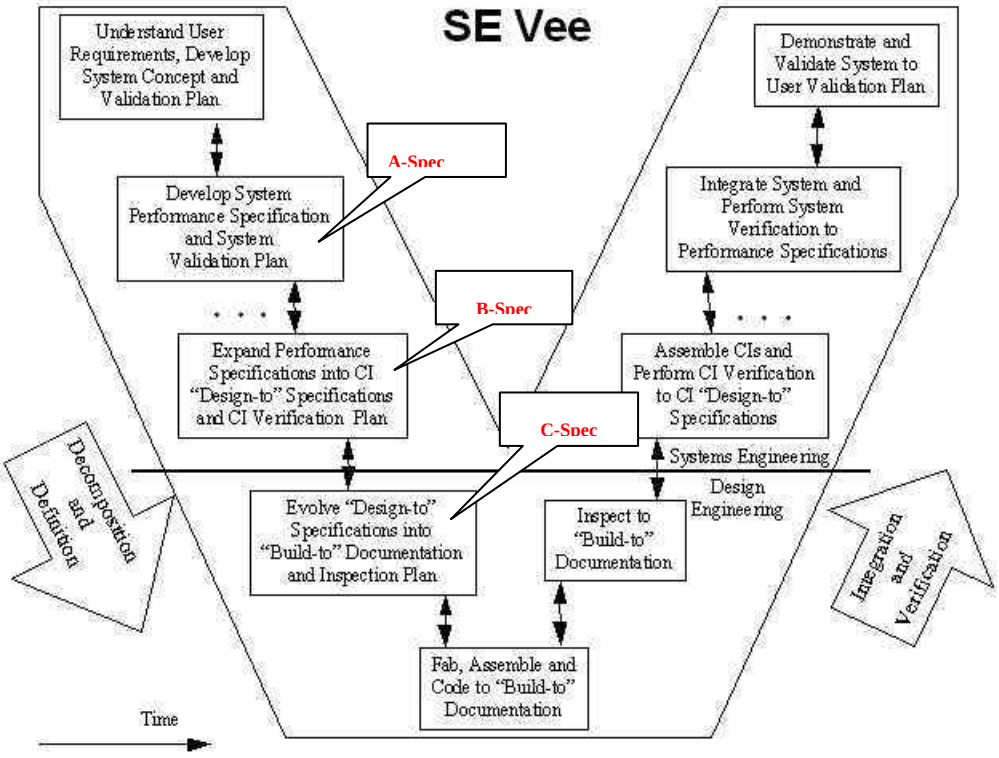


Figure 5: The Systems Engineering Vee Model Process

[31] Levis, Alexander. 2004. Modeling and Simulation for Architecture Assessment, *US Air Force*

The Vee process model was developed by Forberg and Mooz and described by them as "the technical aspect of the project cycle." This model begins with the user's needs on the upper left-hand side and ends with a user validated system on the upper right-hand side. On the left-hand side, decomposition and definition activities resolve the system architecture, thus creating the design details. Integration and verification flows up and to the right as successively higher levels of subsystems are verified, thus culminating at the system level. The Vee process model also shows the verification and validation progress from the component level to the validation of the operational system. At each level of testing, the original specifications and requirements are referenced to ensure that the components, subsystems, and the system itself all meet the original specifications.

Systems Engineering in the System Life Cycle

The acquisition framework established by DoDI 5000.2, is structured in phases separated by milestones with systems engineering processes performed at each phase from concept to disposal. The systems engineering processes are iterated at each phase and subsequently build upon the previous phase to move toward a technical solution. Technical reviews serve to control and manage the technical development from one acquisition phase to the next.

Alignment of MIL-STD-490A with DoDAF Architectural Products

The diagram below and the depiction of Key Systems Engineering activities that follow demonstrate the similarities between MIL-STD-490A and the DoDAF.

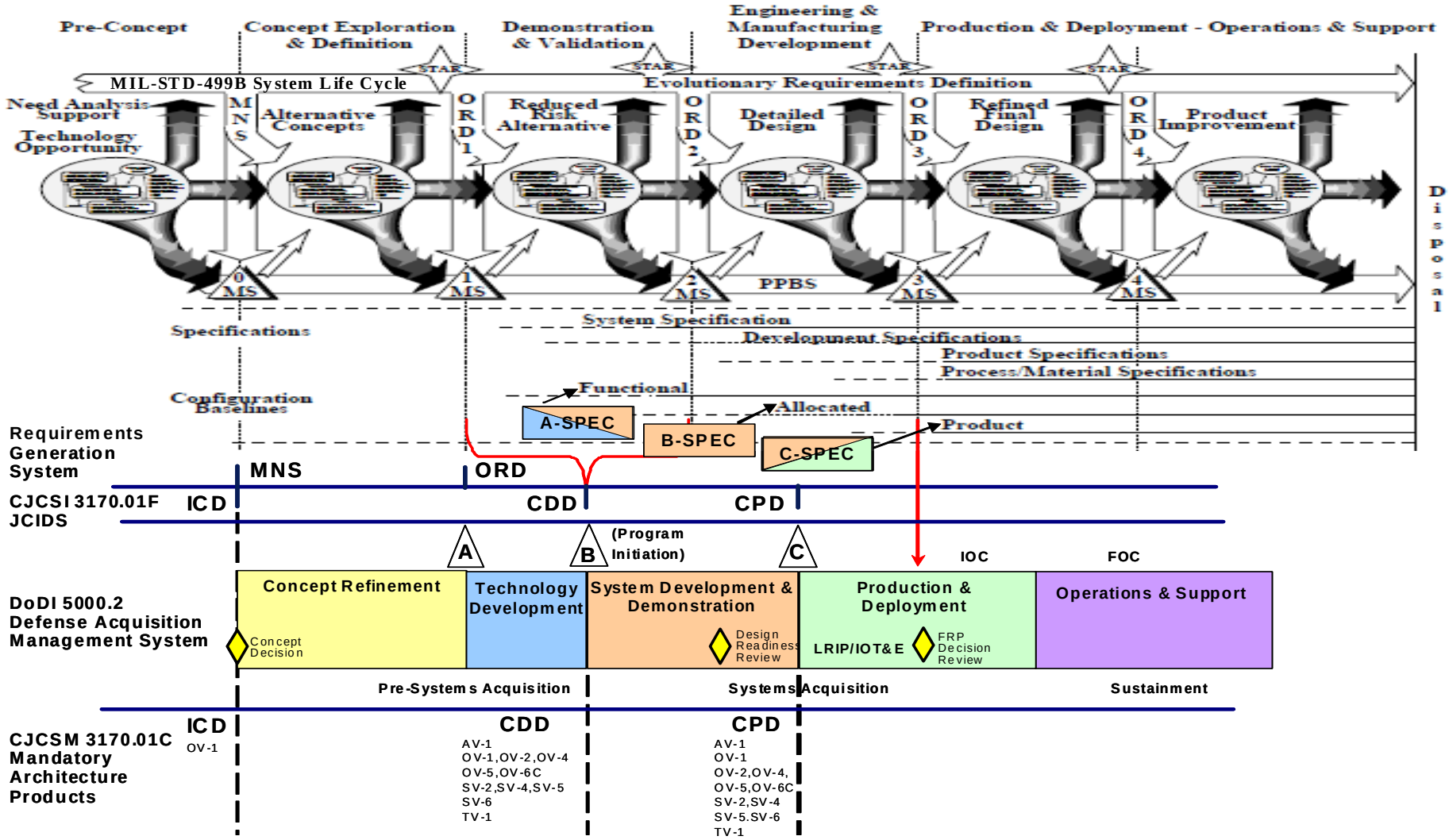


Figure 6: Alignment of MIL-STD-490A with DoDAF Architectural Products

Concept Refinement outputs and the MIL-STD-490A Type A (A-Spec)

An Alternative System Review (ASR) is held at the end of the Concept Refinement Phase to ensure the requirements agree with the customers' needs and expectations and that the system under review can proceed into the Technology Development phase. The ASR should provide a preliminary System Specification that captures the system technical baseline. This preliminary System Specification corresponds to the initial requirements developed during the Concept of Operations phase of the "old" requirements generation system.

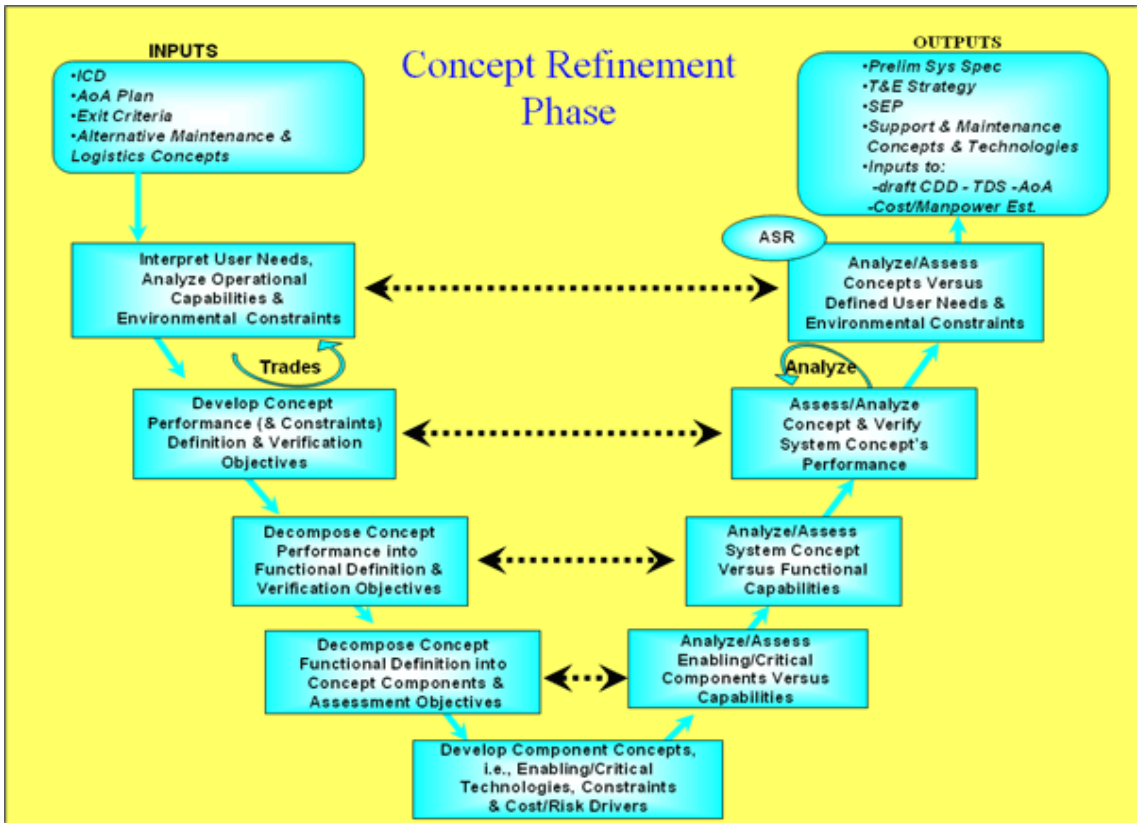


Figure 7: Concept Refinement Phase

The following abridged section is taken from MIL-STD-490A 4 June 1985:

> 3.1.3.1 Type A - System/segment specification. This type of specification states the technical and mission requirements for a system/segment as an entity, allocates requirements to functional areas, documents design constraints, and defines the interfaces between or among the functional areas. Normally, the initial version of a system/segment specification is based on parameters developed during the Concept Exploration phase. This specification (initial version) is used to establish the general nature of the system that is to be further defined and finalized during the Demonstration and Validation phase. The system/segment specification is maintained current during the Demonstration and Validation phase, culminating in a revision that forms the future performance base for the development and production of the prime items and configuration items.

Technology Development outputs and the MIL-STD-490A Type A (A-Spec)

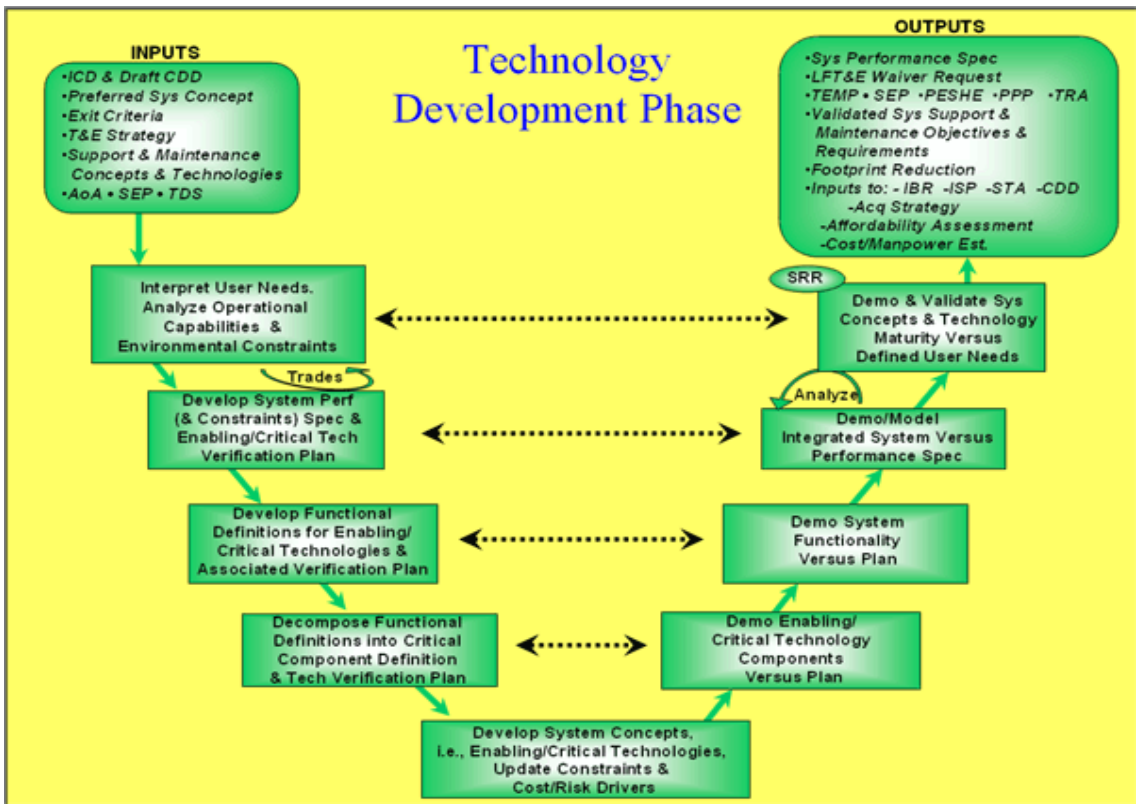


Figure 8: Technology Development Phase

The technology development systems engineering phase converts each required capability into a system performance specification. The System Requirements Review (SRR) produces an approved preliminary system performance specification. From MIL-STD-490A, as noted above, the system performance specification output from the Technology Development phase corresponds to the Demonstration and Validation phase.

System Development outputs and the MIL-STD-490A Type B (B-Spec) and Type C (C-Spec)

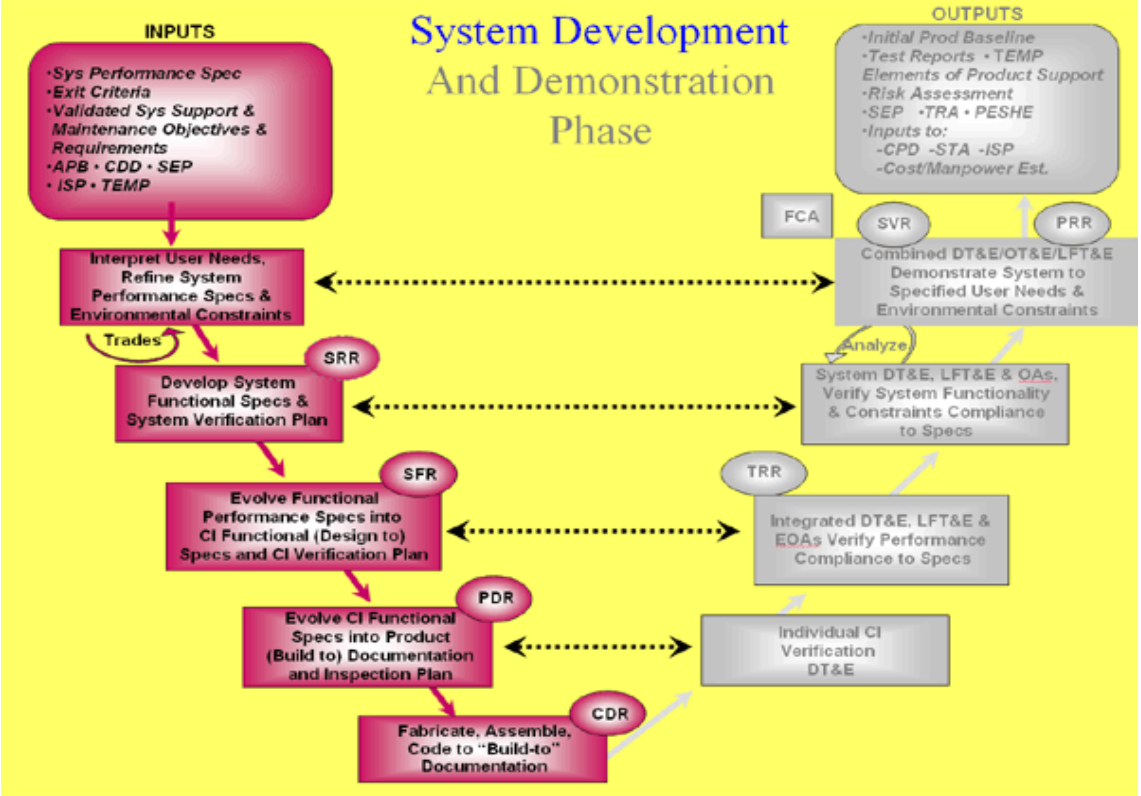


Figure 9: System Development and Demonstration Phase

In System Development and Demonstration, the system elements down to the configuration item level are defined. System design requirements are allocated down to the major subsystem level. This phase involves allocating functional performance specifications into system functional and performance requirements. A Preliminary Design Review (PDR) should approve the allocated "Design to" baseline. The allocated baseline corresponds to MIL-STD-490A Type B (B-Spec). Further decomposition of the functional specification yields the product "Build to" baseline. The product baseline corresponds to MIL-STD-490 Type C (C-Spec).

The following abridged section is taken from MIL-STD-490A

> 3.1.3.2 Type B - Development specifications. Development specifications state the requirements for the design or engineering development of a product during the development period. Each development specification shall be in sufficient detail to describe effectively the performance characteristics that each configuration item is to achieve when a developed configuration item is to evolve into a detail design for production.

> 3.1.3.3 Type C - Product Specifications. Product specifications are applicable to any configuration item below the system level, and may be oriented toward procurement of a product through specification of primarily functional (performance) requirements or primarily fabrication (detailed design) requirements.

CJCSM 3170.01 Mandatory Architecture Products and MIL-STD-490A

As shown in figure 5, the ICD corresponds to MIL-STD-490A A-Spec, the CDD to the B-Spec, CPD to the B-Spec and C-Spec.

CJCSM 3170.01 Mandatory Architecture Products		
ICD /A-Spec	CDD / B-Spec and C-Spec	CPD / C-Spec
OV-1	AV-1	AV-1
	OV-1	OV-1
	OV-2	OV-2
	OV-4	OV-4
	OV-5	OV-5
	OV-6C	OV-6C
	SV-2	SV-2
	SV-4	SV-4
	SV-5	SV-5
	SV-6	SV-6
	TV-1	TV-1

The DoDAF is about acquiring capabilities, using an iterative top-down approach while, as quoted below, MIL-STD-490A is about acquiring products that focuses on a bottom-up integration process.

> 3.1.3.1 Type A - System/segment specification. This type of specification states the technical and mission requirements for a system/segment as an entity,

> allocates requirements to functional areas, documents design constraints, and defines the interfaces between or among the functional areas[32].

The MIL-STD-490A specifications lend themselves well to a structured analysis development of the product. A structured analysis method such as the Integrated Definition for Function Modeling 0 (IDEF0), identifies an overall function and iteratively decomposes the function into smaller functions while preserving input, outputs, controls and mechanisms. One of the weaknesses of structured analysis is it identifies inputs and outputs without source and destination interactions thereby losing any interoperability characteristics.

Conclusions

The system engineering process and products as specified in MIL-STD-490A can be traced to DoDAF views. While MIL-STD-490A is focused on system design with a bottom-up approach, the DoDAF's top-down "born joint" philosophy is focused on interoperability across systems.

[32] MIL-STD-490A 4 June 1985

APPENDIX

MIL-STD-490A Specifications

3.1.3.1 Type A - System/segment specification. This type of specification states the technical and mission requirements for a system/segment as an entity, allocates requirements to functional areas, documents design constraints, and defines the interfaces between or among the functional areas. Normally, the initial version of a system/segment specification is based on parameters developed during the Concept Exploration phase. This specification (initial version) is used to establish the general nature of the system that is to be further defined and finalized during the Demonstration and Validation phase. The system/segment specification is maintained current during the Demonstration and Validation phase, culminating in a revision that forms the future performance base for the development and production of the prime items and configuration items. The System/Segment Specification shall be prepared by the contractor and shall be in accordance with the format and content of the System/ Segment Specification Data Item Description (see 6.2).

3.1.3.2 Type B - Development specifications. Development specifications state the requirements for the design or engineering development of a product during the development period. Each development specification shall be in sufficient detail to describe effectively the performance characteristics that each configuration item is to achieve when a developed configuration item is to evolve into a detail design for production. The development specification should be maintained current during production when it is desired to retain a complete statement of performance requirements. Since the breakdown of a system into its elements involves configuration items of various degrees of complexity which are subject to different engineering disciplines or specification content, it is desirable to classify development specifications by sub-types. The characteristics and some general statements regarding each sub-type are given in the following paragraphs (see 6.2).

3.1.3.2.5 Type B5 - Software development specification (see 6.2). Software development specifications are applicable to the development of computer software and consist of a Software Requirements Specification and Interface Requirements Specification(s).

3.1.3.2.5.1 Software Requirements Specification. This type of specification describes in detail the functional, interface, quality factor, special, and qualification requirements necessary to design, develop, test, evaluate and deliver the required Computer Software Configuration Item (CSCI). The Software Requirements Specification shall be prepared by the contractor and shall be in accordance with the format and content of the Software Requirements Specification Data Item Description (See 6.1).

3.1.3.2.5.2 Interface Requirements Specification. This type of specification describes in detail the requirements for one or more CSCI interfaces in the system, segment, or prime item. The specified requirements are those necessary to design, develop, test, evaluate, and deliver the required CSCI. The interface requirements may be included in the associated Software Requirements Specifications under the following conditions: (1) there are few interfaces, (2) few development groups are involved in implementing the interface requirements, (3) the interfaces are simple, or (4) there is one contractor developing the software. The Interface Requirements Specification shall be prepared by the contractor(s) and shall be in accordance with the format and content of the Interface Requirements Specification Data Item Description (see 6.2).

3.1.3.3 Type C - Product Specifications. Product specifications are applicable to any configuration item below the system level, and may be oriented toward procurement of a product through specification of primarily functional (performance) requirements or primarily fabrication (detailed design) requirements. Sub-types of product specifications to cover equipments of various complexities or requiring different outlines of form are covered in paragraphs 3.1.3.3.1 through 3.1.3.3.5.

a. A product function specification states (1) the complete performance requirements of the product for the intended use, and (2) necessary interface and interchangeability characteristics. It covers form, fit, and function. Complete performance requirements include all essential functional requirements under service environmental conditions or under conditions simulating the service environment. Quality assurance provisions for hardware include one or more of the following inspections: qualification evaluation, preproduction, periodic production, and quality conformance.

b. A product fabrication specification will normally be prepared when both development and production of the HWCI are procured. In those cases where a development specification (Type B) has been prepared, specific

reference to the document containing the performance requirements for the HWCI shall be made in the product fabrication specification. These specifications shall state: (1) a detailed description of the parts and assemblies of the product, usually by prescribing compliance with a set of drawings, and (2) those performance requirements and corresponding tests and inspections necessary to assure proper fabrication, adjustment, and assembly techniques. Tests normally are limited to acceptance tests in the shop environment. Selected performance requirements in the normal shop or test area environment and verifying tests therefore may be included. Preproduction or periodic tests to be performed on a sampling basis and requiring service, or other, environment may reference the associated development specification. Product fabrication specifications may be prepared as Part II of a two-part specification (See 3.1.4) when the contracting agency desires close relationships between the performance and fabrication requirements.

3.1.3.3.5 Type C5 - Software Product Specification (see 6.2).

The Software Product Specification is applicable to the delivered CSCI and is sometimes referred to as the "as built" software specification. This specification consists of the final updated version of the Software Top-Level Design Document, the Software Detailed Design Document, the Data Base Design Document(s), the Interface Design Document(s), and the source and object listings of the software. The Software Product Specification shall be prepared by the contractor and shall be in accordance with the format and content of the Software Product Specification Data Item Description (see 6.2).

3.1.3.3.5.1 Software Top Level Design Document. The Software Top Level Design Document describes how the top-level computer software components (TLCSCs) implement requirements allocated from the Software Requirements Specification and, if applicable, Interface Requirements Specification(s). The Software Top Level Design Document shall be prepared by the contractor and shall be in accordance with the format and content of the Software Top Level Design Document Data Item Description (see 6.2).

3.1.3.3.5.2 Software Detailed Design Document. The Software Detailed Design Document describes the detailed decomposition of TLCSCs to lower level computer software components (LLCSCs) and units. The Software Detailed Design Document shall be prepared by the contractor and shall be in accordance with the format and content of the Software Detailed Design Document Data Item Description (see 6.2).

3.1.3.3.5.3 Data Base Design Document. The Data Base Design Document describes one or more data base(s) used by the CSCI. If there is more than one data base, each data base may be described in a separate Data Base Design Document. The Data Base Design Document(s) shall be prepared by the contractor and shall be in accordance with the format and content of the Data Base Design Document Data Item Description (see 6.2).

3.1.3.3.5.4 Interface Design Document. The Interface Design Document provides the detailed design of one or more CSCI interfaces. When Interface Requirements Specifications have been prepared, associated Interface Design Documents shall be prepared as well. The Interface Design Document shall be prepared by the contractor and shall be in accordance with the format and content of the Interface Design Document Data Item Description (see 6.2).

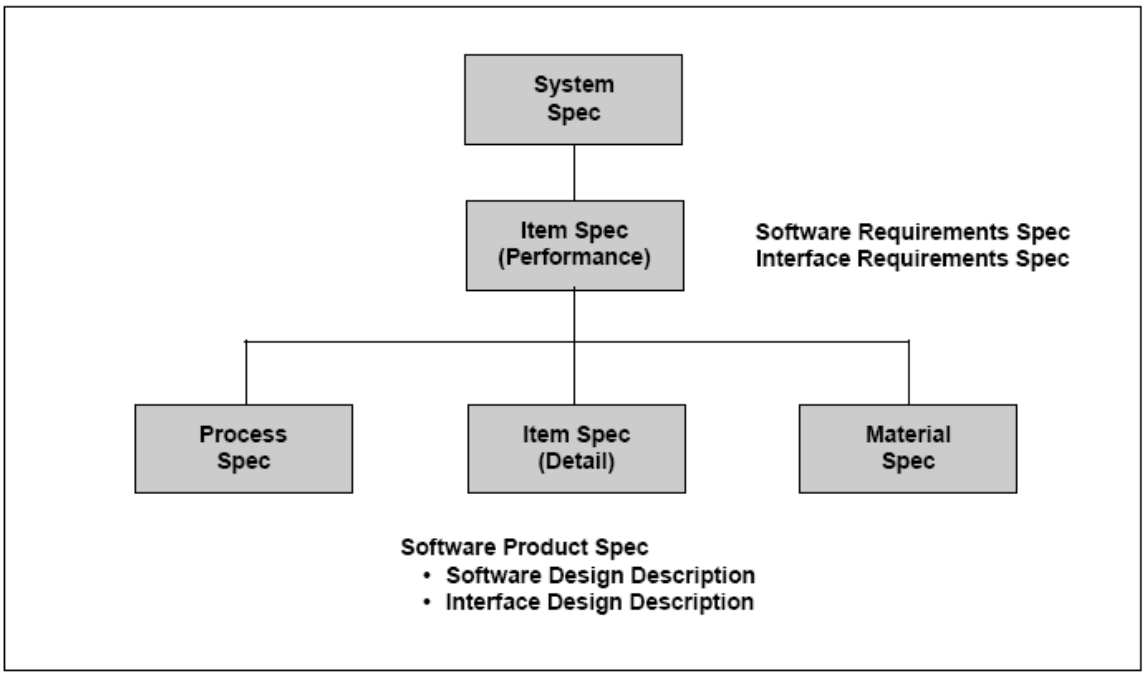


Figure 8–7. Specification Hierarchy

The chart above (Figure 8-7)[33] depicts a MIL-STD-490A style specification hierarchy.

[33] Systems Engineering Fundamentals. 2001. *Defense Acquisition University Press*. pg 79

MIL-STD-973 Specifications

3.49 Functional Baseline (FBL) . The initially approved documentation describing a system's or item's functional, interoperability, and interface characteristics and the verification required to demonstrate the achievement of those specified characteristics.

3.3 Allocated Baseline (ABL) . The initially approved documentation describing an item's functional, interoperability, and interface characteristics that are allocated from those of a system or a higher level configuration item, interface requirements with interfacing configuration items, additional design constraints, and the verification required to demonstrate the achievement of those specified characteristics.

3.74 Product Baseline (PBL]. The initially approved documentation describing all of the necessary functional and physical characteristics of the configuration item and the selected functional and physical characteristics designated for production acceptance testing and tests necessary for support of the configuration item. In addition to this documentation, the product baseline of a configuration item may consist of the actual equipment and software.

References

Meshel, David, David Davis, William Tosney, and Frank L. Knight, 2007. Implementation of Revitalized Engineering Specifications and Standards for National Security Space Programs, *U.S. Air Force T&E Days 13 – 15 Feb 2007, Destin, Florida*

Grady, Jeffrey O. 1995. *System Engineering Planning and Enterprise Identity*, Boca Raton Fla.: CRC Press

Skalamera, Robert, J. 2004. USD(AT&L) Imperatives, Implementing OSD Systems Engineering Policy, *OUSD (AT&L)*

DiMario, Michael, Brian Sauser, and Dinesh Verma, 2006. System of Systems Characteristics and Interoperability in Joint Command and Control, *2nd Annual System of Systems Engineering Conference, Ft Belvoir, VA*

CJCSI 3170.01C. 2003.

C4ISR Architecture Framework,Version 2.0. 1997.

DoD Architecture Framework,Version 1.0. 2003.

Sibbald, Chris and Cris Kobryn, 2004. Modeling DoDAF Compliant Architectures, The Telelogic Approach for Complying with the DoD Architectural Framework, *A Telelogic White Paper*

SMC Systems Engineering Primer & Handbook. 2005. *Space & Missile Systems Air Force 3rd Edition*.

NAVAIR Acquisition Guide. 2006/2007.

SPAWAR Acquisition Program Structure Guide Vol I Version 1.0. 2001.

Systems Engineering Fundamentals. 2001. *Defense Acquisition University Press*.

Levis, Alexander. 2004. Modeling and Simulation for Architecture Assessment, *US Air Force*

MIL-STD-490A 4 June 1985

APPENDIX F: MODELING AND SIMULATION IN NAVY ACQUISITION

Executive Summary

Although modeling and simulation (M&S) tools are being used extensively in requirements generation in many programs throughout the Department of Defense (DoD), their effectiveness is largely undocumented. In the current world political and economical environment, the Department of the Navy (DoN) will be required to carry out a diversified list of missions with a smaller force and reduced budgets. The DoN will have to find more efficient and less costly ways to train troops, refine new weapon system requirements, evaluate solutions and acquire new or modified systems. Advances in M&S capabilities and technologies offer significant opportunities for responding to these challenges.

Modeling and Simulation in Acquisition

The Department of Defense (DoD) M&S acquisition vision is to optimally employ responsive, trustworthy, and cost-effective M&S capabilities to support defining, developing, testing, producing and sustaining America's capabilities that support the spectrum of DoD missions. [34]

Modeling and Simulation (M&S) have long played an essential, but imperfect, role in system acquisition, operations, and support across the system life cycle. Increasingly, capability concepts and system designs are being defined by building models within the synthetic environments provided by systems and software engineering tools and computer-aided design (CAD) tools. M&S tools can be utilized to help manage complexity by tracking system characteristics, functions, relationships and interactions at the most granular level and then presenting aggregated impacts and higher-level measures of merit to decision makers. System capabilities, processes, workloads, performance, logistics and cost can be modeled as well. M&S also allows the immersion of warfighters in realistic operational environments to assess concepts, capabilities and tactics. It can help explore the entire functional capability trade space. Distributed simulation technology allows the flexible mixing of simulations, lab hardware, and real systems into an integrated environment in which to conduct integration and testing.[35]

[34] Paragraph taken verbatim from Acquisition M&S Master Plan [1]

[35] Modeling and Simulation Guidance for the Acquisition Workforce [3]

Effective acquisition requires collaboration among multiple stakeholders. M&S tools and products are an effective means of communication, facilitating a shared understanding and insight among warfighters, sponsors, program staffs and industry at a much earlier point than would otherwise be possible. Therefore M&S tools linked as needed and combined with an information-sharing infrastructure (i.e., integrated data environment – IDE) into a distributed collaborative environment, can enable cost-effective development and sustainment of systems and systems of systems (SoS). If program circumstances (e.g., budgets, threats, technology) change, system design and support changes can be made rapidly while simultaneously allowing all of the stakeholders to play an appropriate role in those decisions.[36]

The comprehensive and integrated use of M&S envisioned under the term simulation-based acquisition (SBA) in the late 1990s has been advanced in various domains under different names. An increasingly widespread instantiation of this concept is termed model-based systems engineering (MBSE). The International Council on Systems Engineering (INCOSE), in its October 2006 version of Systems Engineering Vision 2020, defines MBSE as "the formalized application of modeling to support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases." It goes on to say: MBSE is part of a long-term trend towards model-centric approaches that have evolved in other engineering disciplines, including mechanical, electrical and software. In particular, MBSE is expected to replace the document-centric approach that has been practiced by systems engineers in years past and to become fully integrated into the systems engineering process (SEP).

Acquisition managers should realize that there are still many challenges in achieving effective M&S use. The DoD Acquisition M&S Master Plan, issued under the authority of the DoD Systems Engineering Forum in April 2006, describes the actions being pursued to improve M&S support to acquisition. However, to date, these actions have not all been accomplished, so acquisition managers must consider the impact of the remaining impediments in their M&S planning. For instance, M&S capabilities and limitations are often not adequately understood and M&S is sometimes not planned and managed with sufficient care. Although many things can be modeled quite credibly today, such as physical capabilities, natural phenomena, and physics-based interactions, it is much more difficult to reliably represent things we understand less well, such as human behavior, reliability and emergent behaviors of complex systems. It must also be remembered that M&S capability involves not just the software tools themselves, but the data that feeds them; the computing platforms that execute them; the standards, middleware, and networks that may interconnect them; the encryption capabilities and security constraints that protect them; and, most importantly, the people that plan, develop, integrate, verify, validate, accredit, and use them.

[36] Modeling and Simulation Guidance for the Acquisition Workforce [3]

Evolutionary Acquisition

Special considerations should be made in the context of a program that is based on an evolutionary acquisition approach. The T&E Strategy for a system acquired using evolutionary acquisition shall address each increment intended for fielding. In general, T&E that has previously confirmed the effectiveness and suitability of a previous increment need not be repeated in its entirety to confirm that the subsequent increment still provides those mission capabilities previously confirmed. However, regression testing to reconfirm previously tested operational capabilities and/or suitability might be required if the subsequent increment introduces a significantly changed hardware or software configuration, or introduces new functions, components, or interfaces that could reasonably be expected to alter previously confirmed capabilities or technical/performance specifications.

Modeling and Simulation Policy

There is a general agreement within the test and evaluation (T&E) community that M&S has a complementary role to play with respect to live testing. It may also provide an objective frame of reference for testing individual systems and evaluating joint capabilities. However, implementation has proven difficult due to the lack of definitive policy and guidelines regarding the appropriate role, extent and fidelity of the models and simulations. Although public law has been clear regarding the testing of actual individual systems under realistic conditions, the DoD intent to orient its acquisition activities on functional capabilities does not identify the extent to which evaluation of those capabilities must be accomplished in live testing versus M&S.[37]

Explicit DoD policy is now needed to require improvement of system representations and models to meet acquisition decision needs and to dictate the appropriate use of M&S to plan tests, complement live tests, and evaluate the joint capabilities enabled by a system of systems within a net-centric environment. M&S is integral to the T&E process by complementing testing by aiding in the assessment of a system in various scenarios, climatic and threat environments and specific areas of the mission space and performance envelope where live testing is not cost effective. Modeling and simulation and test tools allow the analyst and tester to focus on what is essential to evaluate, to monitor the activities as they occur and to consolidate and analyze the results of their activities. Testing tools include live tests, stimulators, and laboratory facilities that have supported testing for many years. M&S includes resources that together describe the system characteristics and performance at all levels from engineering models to campaign level war games. They are used in a variety of ways from measuring compliance to design requirements through predicting system performance in an operational environment. M&S plays a significant role in testing a system that is part of a system of systems, families of systems, or utilized in a joint environment.

[37] Paragraph taken verbatim from Modeling and Simulation Guidance for the Acquisition Workforce [3]

There is currently no DoD requirement for formal M&S planning to support acquisition other than T&E. Some Services require acquisition program managers to develop stand-alone M&S support plans. M&S is a key enabler of systems engineering and is best linked from the outset to systems engineering planning to attain effective and efficient M&S across the system life cycle. The acquisition of DoD systems is supported by a Systems Engineering Plan (SEP), a Test and Evaluation Strategy (TES), and a Test and Evaluation Master Plan (TEMP). Currently, only the TEMP requires documenting the use of M&S when all three documents should do so. Most DoD M&S takes a project, vice an enterprise, approach. There is scarce activity across the Department to plan M&S use to support development of a joint capability. A program's M&S plans should be addressed in the afore-mentioned systems engineering and test and evaluation documents.[38]

The M&S strategy may optionally be summarized in a separate section of these documents (or even in a stand-alone document such as a Simulation Support Plan), but the use of M&S to support specific systems engineering or test activities should be embedded in the discussion of those activities. Furthermore, M&S planning should be explicitly addressed at the joint capability level – not just the individual program level. Such plans should address program responsibilities to support others with models and data.

The Defense Acquisition Guidebook mandates that, "The program manager, in concert with the user and test communities, without compromising rigor, is required to integrate modeling and simulation (M&S) activities with government and contractor Developmental Test and Evaluation (DT&E), Operational Test and Evaluation (OT&E), Live Fire Test and Evaluation (LFT&E), system of systems interoperability and performance testing into an efficient continuum." This testing shall be event driven within the program's overall acquisition strategy, and allow for a realistic period of time in which to accomplish the planned T&E events, including report preparation. The program manager should also be charged with developing a robust DT&E effort to ensure that a successful OT&E outcome is achieved. The program manager is required to develop hardware and software metrics in the form of T&E success criteria and OT&E entrance criteria in consultation with the Operational Test Agency (OTA), to use in monitoring program maturity and to support decisions to progress through the development cycle. The T&E Working-level Integrated Product Teams (T&E WIPT), may include representatives from Program Management Offices, T&E agencies, operational users, the OSD staff, DoD Component staffs, the intelligence community and other agencies, as necessary, to assist in this task. [39]

[38] Acquisition Modeling and Simulation Master Plan [1]

[39] Defense Acquisition Guidebook [2]

Prior to Milestone A

The acquisition process begins with the identification of a capability need that requires a materiel solution. Typically, this is done by performing a functional area analysis (FAA) and a functional needs analysis (FNA). M&S can and should be utilized during the performance of the FNA to provide technical support for the proposed solution(s).

Milestone A

The CJCSI 3170.01F states, "The process to identify capability gaps and potential materiel and non-materiel solutions must be supported by a robust analytical process that incorporates innovative practices – including best commercial practices, Human Systems Integration (HSI), collaborative environments, modeling and simulation and electronic business solutions."[40] Accordingly, the use of M&S is endorsed prior to the initiation of a program to aid in the performance of an analysis of alternatives in preparation for entry into Milestone A, the Technology Development Phase. The Test and Evaluation Strategy (TES) is also due at Milestone A.

Milestone B

The application of M&S to solving the identified capability gap should be addressed in the TEMP. The initial version of the TEMP is a requirement for the Milestone B decision that leads to entry into the System Development and Demonstration Phase.

Milestone C

The refined, desired operational capabilities and expected system performance contained in the Capability Production Document (CPD) are used by the T&E/M&S WIPT to update the TEMP for the Milestone C decision and for subsequent updates later in the Production and Deployment Phase, such as the full rate production decision review. At Milestone C, the technical testing begins to focus on production testing, such as Production Qualification Testing, to demonstrate performance of the production system in accordance with the contract. Operational testing focuses on evaluating the system's operational effectiveness, suitability, and survivability.

[40] CJCSI 3170.01F [4]

Modeling and Simulation Planning Guidance

Deliberate and disciplined planning, accomplished early and iteratively throughout the program's lifecycle, is essential for effective use of M&S. Typical programs use many, often hundreds, of synthetic environments and M&S to:

- Develop the system concept.
- Design the system, including its sustainment.
- Assess the merits of alternatives throughout the development cycle
- Integrate the system.
- Test the system to verify it meets requirements.
- Support system introduction, sustainment and evolution.

A program should involve its Operational Test Authority (OTA) in M&S planning to support both developmental test and operational test objectives.

Although M&S is often regarded as an obscure discipline, the M&S planning process is logical and straight forward. Utilizing a logical process will prevent M&S decisions from being made ad-hoc, undisciplined, or biased by the past experiences and economic interests of the program's contractors.

Staff Expertise

Planning should begin by ensuring that the program staff is equipped with adequate knowledge regarding M&S strengths and weaknesses, applicable standards, potentially available reusable resources, lessons learned from other M&S efforts and the available options for obtaining technical assistance. Responsible personnel should review applicable DoD and Component policies and consult with Component M&S management offices, the Navy Modeling and Simulation Office (NMSO), acquisition programs that have faced similar challenges and other known experts. Training courses, reviews of studies and professional papers on M&S support to acquisition, participation in conferences and workshops targeted toward M&S in acquisition and perusal of M&S resource registries and repositories also help equip a program's staff with the required background information that is needed to commence effective M&S planning.

Government-Contractor Responsibility Allocation

One of the important M&S strategy decisions that must be made early in a program is the allocation of M&S responsibility between the government and its contractor(s), with attendant funding and accountability implications. This allocation typically varies by phase, with government M&S activities prominent in the early phases (e.g., Concept Refinement, Technology Development), but the prime contractor assuming a preeminent role after source selection and throughout the Systems Development and Demonstration phase. Government M&S activity typically increases again during Operational Test and

Evaluation (OT&E). The program must decide to what degree it wishes to have an independent M&S-based capability rather than just insight into the contractor's M&S activities. The program must also decide whether it will provide, or facilitate providing, the contractor with government-owned M&S tools and data, and if so, what its limits of obligation will be regarding the functional adequacy, trustworthiness, and evolution of such government-furnished equipment or information (GFE/GFI). Verification, Validation, and Accreditation (VV&A) responsibilities must also be allocated.

Systems Engineering Approach

Synthetic environments and M&S are both systems, which are intended to accomplish particular objectives. It follows that M&S planning represents the initial steps of a disciplined system engineering approach, the elements of which are:

- M&S requirements analysis
- Analysis of alternative solutions
- Selection of best solution
- Procurement or development of the selected M&S capability
- Integration, test and evaluation of the M&S capability
- Application/employment of the modeling and simulation capability

Once realized, the M&S capability then typically evolves over time as the program deems appropriate.

Developing Modeling and Simulation Tools

It is important to consider the potential for reuse when deciding on how to proceed with the development of modeling and simulation tools to support a program throughout the acquisition lifecycle. The Navy stood up an office, the Navy Modeling and Simulation Office (NMSO), to aid in the coordination of such efforts.

Navy Modeling and Simulation Office

For over a decade, the Navy's M&S office has operated via OPNAV N6 or N6/N7 in a collaborative manner with the numerous functional constituents utilizing M&S. As a result of recent DoN M&S assessments, conducted by Task Force SIM, and others, the Corporate Business Council (CBC) has established an SES level, M&S Governance Board (GB) to aid in the oversight of M&S activities for the enterprise. In the spirit of organizational spiral development and transformation, the GB builds upon existing policies and organizational structures. The GB is chaired by the Assistant Secretary of the Navy for Research, Development, and Acquisition (ASN RD&A) Chief Engineer (ASN RDA CHENG).[41]

The GB consolidated the twelve M&S functional areas into four SES Community Leads (CLs) consisting of:

- RDA CL (ASN RDA)
- Analysis CL (OPNAV N8)
- Training CL (U.S. Fleet Forces Command)

NMSO will function as the GB's "action arm" by implementing policy guidance, and providing:

- Administrative support for CLs and the M&S community of users.
- Core services and products to improve M&S within, between, and across organizations in the community of users.
- Assistance to promote and implement cross cutting M&S projects that improve the Navy's M&S capabilities.

The following diagram illustrates the community lead focused construct of the NMSO:

Utilizing the NMSIS

Through the use of the Navy's Modeling & Simulation Information Service (NMSIS), provided by the NMSO, a program manager can find opportunities for modeling and

[41] Navy Modeling and Simulation Office website [4]

simulation tool reuse. The NMSIS collects, maintains, and distributes information about Navy M&S for the use of program managers, engineers, M&S builders and others throughout the M&S community.

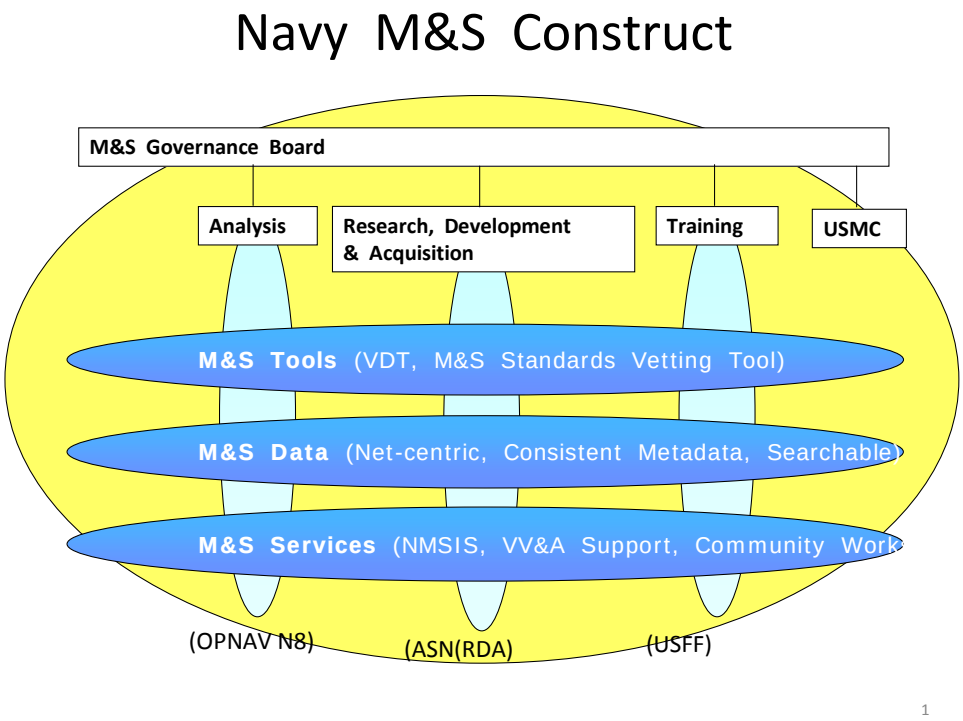


Diagram 1: Navy M&S Construct[42]

The NMSO provides a searchable function within their website that enables various DoD users to seek out and reuse tools and/or models that have been vetted by the NMSO. Although, many valid models and simulations are available for reuse, it is important for those who are seeking to reuse a particular model or simulation, whether in whole or in part, to consider the need to validate the use of the chosen model or simulation for their particular application. Unless the application of the model or simulation is identical to the original intended use, the user should perform validation and verification to ensure that the model can be reused effectively for the desired application or scenario simulation.

[42] id [4]

Recommended Approach

From the policy and guidance that has been presented, the following recommendations are made with regard to the process of implementing M&S in DoN Acquisition: Program managers should ensure that their staff, engineers and contractors posses the required expertise in M&S; Program managers should include M&S in the all stages of the acquisition process; The NMSO should be consulted in the early stages of a program and at each acquisition milestone to determine if any opportunities for reuse of tools or models exist that would support the program's current efforts/direction; The tools or models chosen for use should be validated and verified to be suitable to the desired application; and the chosen M&S capabilities should be revisited and adapted over time as the program deems appropriate.

References

1. Department of Defense. 17 April 2006. Acquisition Modeling and Simulation Master Plan.
2. Department of Defense. November 2006. Defense Acquisition Guidebook.
3. Department of Defense. October 2008. Modeling and Simulation Guidance for the Acquisition Workforce, Version 1.01.
4. U.S. Navy. November 2007. Navy Modeling and Simulation Office – Structure and Function, https://nmso.navy.mil/Communities/NavyMSGovernanceBoard/tabid/68/Default.aspx. (accessed 5 November 2007).
5. Department of Defense. 1 May 2007. CJCSI 3170.01, Joint Capabilities Integration and Development System.
6. Department of Defense. 8 August 2007. DoDD 5000.59, DoD Modeling and Simulation (M&S) Management.

APPENDIX G: AAW CONCEPT OF OPERATIONS

1 INTRODUCTION

The Anti-Air Warfare (AAW) Architecture Concept of Operation (ConOps) goals are to describe and develop a system that satisfies the user's needs in regards to performance against a littoral threat. The user needs were determined to be similar to the analysis stated in problem 4 of the SE 3123 Final Exam. The combat system functionality will be described through an architectural perspective given the requirements and threat analysis stated. Based on the given information, we will also address and discuss scenarios describing combat engagements and layered defense for both self protection and protection of high value units (i.e. Limited Area Defense).

2 SCOPE

The ConOps scope will require the system to ultimately achieve a probability of raid annihilation (P_{RA}) of 0.99 against a littoral threat that consists of eight radially inbound anti-ship missiles (ASMs). We will also be limited to a 30^{o} wedge area and half nautical mile for protection of high value units. Further distance will produce a less confidence in defending against enemy attacks. Additionally, the scope of the problem will be limited to an ideal environmental situation for the area of operation; this will include clear weather and/or "blue ocean" conditions. The given threat situation will also include the following:

3 THREAT CAPABILITIES

- ASMs rate of arrival will be one every four seconds
- ASMs will have an area of 1 meter squared
- ASMs velocity will be Mach 0.9
- ASMs will fly at a height of 9 meters
- ASMs are equipped with an RF seeker that turns on at 7nm

Potential targets that meet these performance criteria are as follows:

- Exocet medium range anti-ship missile – transonic speed range (1134 km/hr), sea-skimming, 1.1 meter wingspan

The AAW Combat system will meet the following required thresholds for both Self Defense and Limited Area Defense:

4 AAW COMBAT SYSTEM REQUIREMENTS

- System reaction time is maximum 6 seconds maximum
- Firm tracks occur 1.5 seconds after target detection
- Combat weapon system will include fire-and-forget and guided weapons
- Fire-and-Forget weapons will be utilized based on performance parameters that will be required to achieve the P_{RA} of 0.99 against the previously stated threat

    - Possible fire-and-forget weapons selection option may include:
        - RAM

    - Possible guided weapons selection options may include:
        - ESSM
        - SM 2

- The ship will support either two 8-cell VLS launchers or 2 RAM launchers
- VLS has a launch rate of 1 missile per second
- RAM can fire the first missile after 5 seconds and has a launch rate of 1 missile per 2 seconds thereafter
- The ordnance requirements are as follows:

    - RAM: Speed - Mach 2.0, Range – (1km-10km), P(k) = 0.85
    - ESSM: Speed - Mach 3.0, Range – (1km-30km), P(k) = 0.8
    - SM 2: Speed - Mach 2.5, Range – (3km-80km), P(k) = 0.7

        - ✓ *Probability of kill, P(k) data account for all the missile performance and fuzing reliability issues*

- There are two x-band illuminators for the ESSM and SM 2 missiles
- The illuminator tie-up time is 6 seconds maximum
- The kill assessment time is 2 seconds
- Soft-kill countermeasures will be utilized in architecture

    - Soft-kill P(k) = 0.6

- Keep-out range is 2km

5 OPERATION CONCEPT

There are three primary scenarios, which can describe the ConOps with the given requirements. The first scenario will describe the long-range surveillance capabilities for conducting a horizon and volume search continuously. The second scenario will describe a Self Defense (SD) architecture that engages and achieves a 0.99 percent P_{RA} against a littoral threat. The SD architecture will protect the ship against enemy attacks through a layered defense strategy that incorporates both hard-kill weapon systems and soft-kill countermeasures.

Generally speaking, a layered defense strategy in AAW breaks down into defending areas or zones around ownship; each zone is defined by its range from ownship. Typically, one could use eight (8) zones (zone 1 would be at the farthest range from the ship, while zone 8 would be the closest) and assign the defense of those zones with hard kill (HK)/soft kill (SK) countermeasure as follows:

Zone 1:	Electronic jamming measures
Zone 2:	Electronic decoy measures
Zone 3:	Medium-range missiles
Zone 4:	Short-range missiles
Zone 5:	Medium-range chaff
Zone 6:	Close-range missiles
Zone 7:	Guns
Zone 8:	Close-in chaff/IR decoys

Due to time constraints, this effort will integrate only HK and SK defenses and will not address a gun system.

The third scenario will describe a situation in which the AAW architecture will achieve the desired P_{RA} in defense of High Value Units (HVUs) or within a Limited Area Defense. This situation describes a scenario in which the AAW protects any/all external HVUs such as aircraft carriers and critical combat support ships within a certain critical combat sector/range.

5.1 Basic Operational Scenario OV-1

Ownship is operating in ideal conditions (i.e. "Blue Ocean"). The ship's radar and EO/IR sensors are operating and performing constant search sweeps around the ship searching for possible enemy attacks. The EO/IR sensor and L-band radar are designed to perform horizon and above horizon surveillance and to work in conjunction with the combat weapon systems to insure a keep-out range of 2km.

The radar and sensor systems suddenly detect and initiate tracks on multiple in-bound threats. The threats consist of eight littoral ASMs, arriving at a rate of one every four seconds. The ASMs are traveling at a speed of Mach 0.9 and possess RF seekers, which turn on at 7nm. The AAW Architecture performs search, detect, track, command/control, and engagements of all eight ASMs and destroys the raid, preventing casualty to ownship.

The ship is adequately outfitted with spares due to the Failure Mode and Effect Analysis (FMEA) and Level of Repair Analysis (LORA) conducted on the ship's combat systems elements. The equipment onboard the ship has the appropriate level of Built in Test (BIT) and Built in Test Equipment (BITE) to troubleshoot failures and correctly identify the fault within the system. The sailors are trained to a level, which gives them the ability to operate, maintain, and sustain their equipment at sea in a hostile environment. Technical documentation has been developed and is available to support Organizational, Intermediate and Depot level repairs.

The OV-1 (figure 1) depicts the operational environment of a layered defense AAW system which is engaging eight (8) ASMs equipped with RF seekers approaching at Mach 0.9 with one every four seconds. The OV-1 also shows that the system would detect and track targets for engagement by implementing horizon and above horizon, short and long range searches. Once the enemy hostile is identified, the system will manage weapon systems, track, engage and destroy the target; thereby protecting itself and other high value units within a limited area. A net centric communication is provided within the strike group and satellite for distance support.

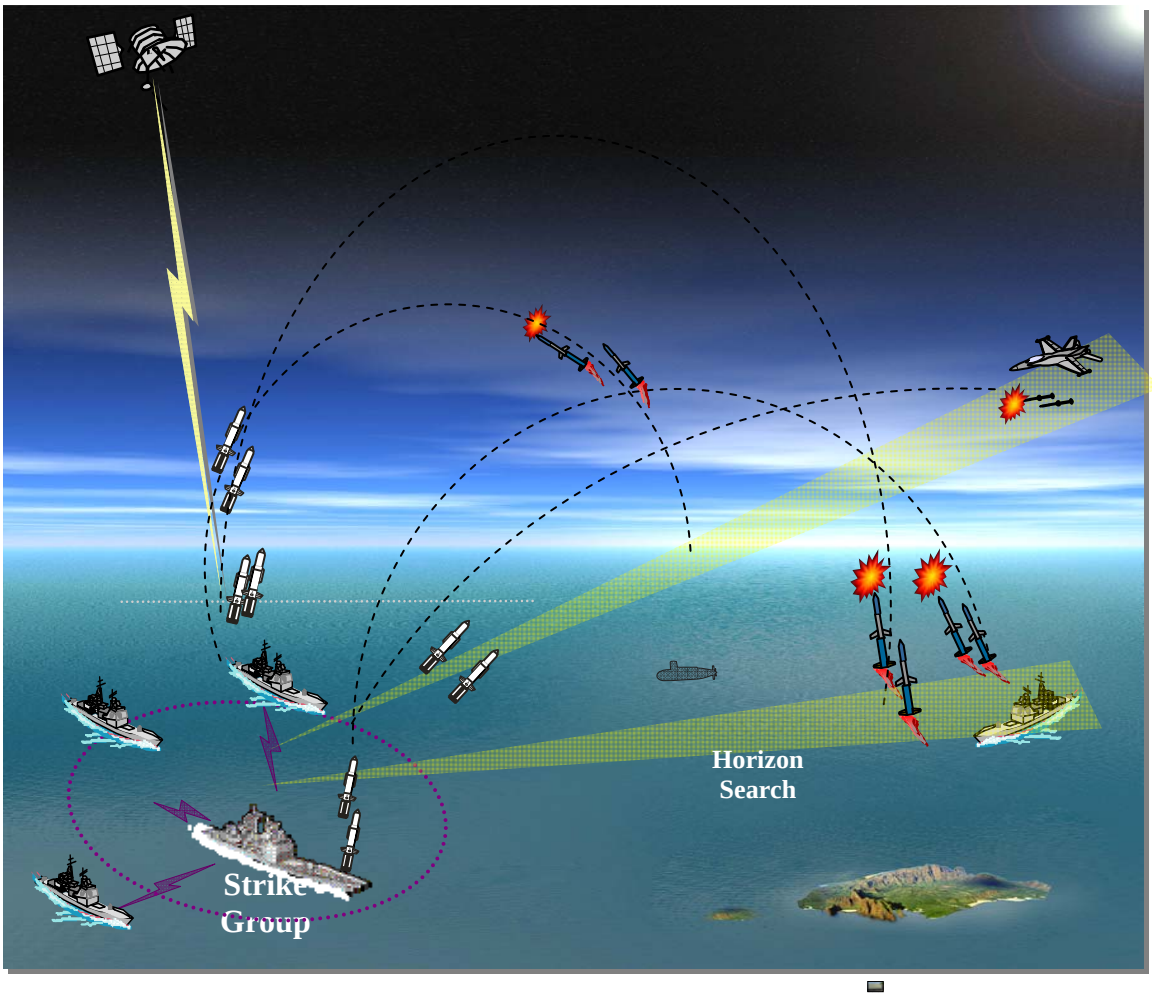


Figure 1: ConOps OV-1

Scenario 1: Surveillance (Search and Detect)

Surveillance and target acquisition are vital components in the overall combat system architecture. It is the first process in the architecture for self-defense and protection of high value units. Surveillance goal is to gather data from multiple sensors, which look independently, as well as collectively to provide the clearest picture possible of an object/s of interest and the ability to share the information with other systems. However, it is difficult to obtain an absolute clear picture because radar track displays detect large quantities of observations within a small section or area often in the form of clutter. It is difficult to recognize an object of interest within the clutter or noise of the radar signal; enemy radar jammers and decoys can also produce false observations. Once an observation is determined, the radar can assign and monitor (track).

Another concept of Surveillance is the Field of Regard (FOR) and Field of View (FOV). FOR is defined as a hemisphere of solid angle that is, the target could be expected to be anywhere above the earth's surface. FOV is a smaller area of interest. It is the area in which an observer can view in one direction. As the observer moves the FOV will be the direction of area in which the observer is looking. Surveillance can also be applied to horizon search patterns or above horizon search patterns[3].

Scenario

AAW system will conduct horizon and volume search capabilities; long range search and detection above the horizon and short range search and detect at horizon. The long range search (L-band) radar operates and conducts volume search capabilities searching for targets above the horizon. The short range EO/IR sensor operates and conducts short range search and detect capabilities to the horizon.

Figure 2 depicts the surveillance scenario in which a ship would conduct long and short-range search and scan at horizon and above horizon for objects of interest. Radar search often produce clutter that are difficult to comprehend. The ships within the strike group use a net centric communication to transmit and receive information about the operational environment to each other for coordination and to optimize efforts. A Satellite also provides a net centric communication to allow for long range C2 communication and distance support.

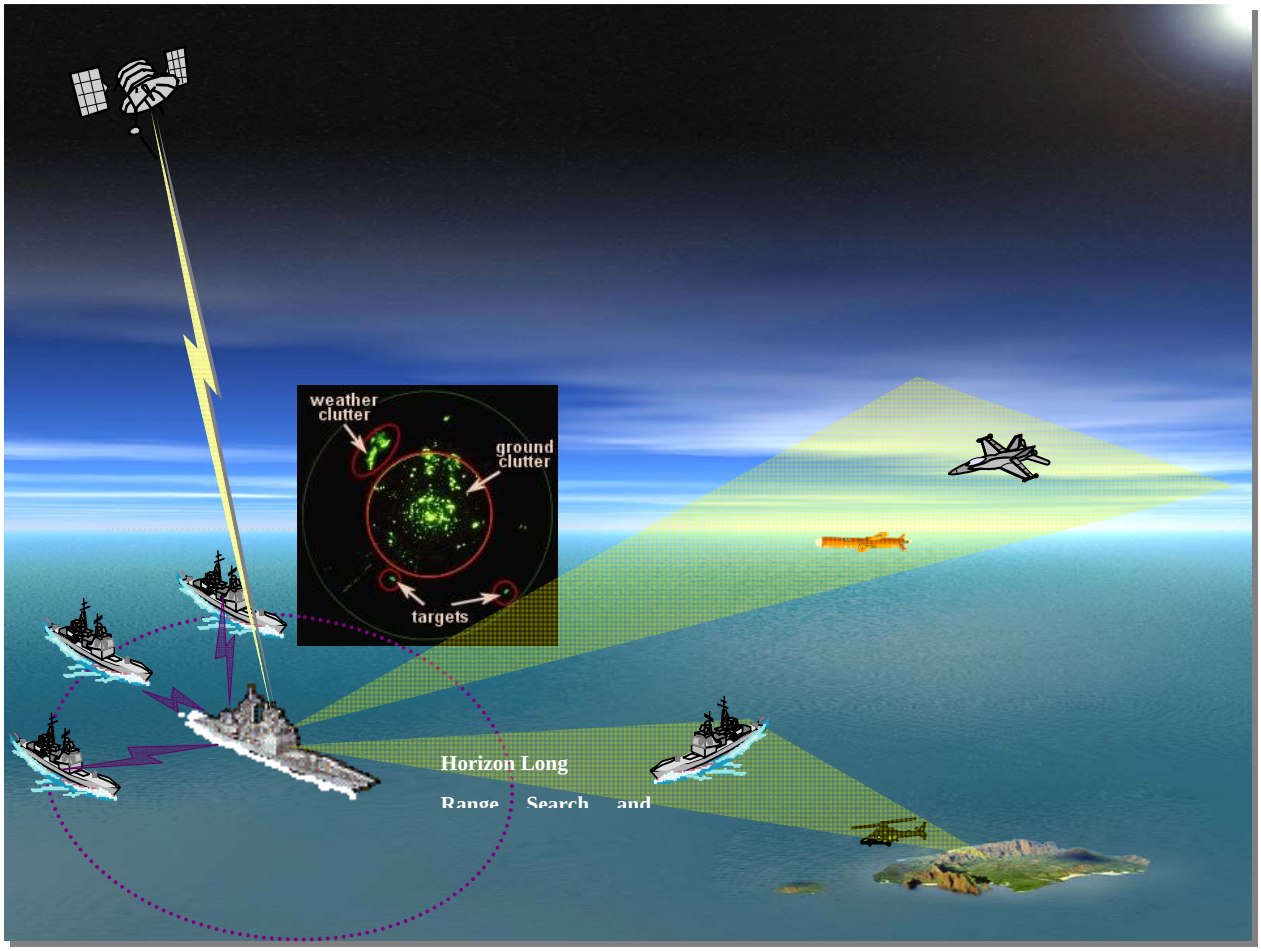


Figure 2: Surveillance ConOps

Scenario 2: Self Defense (SD)

As the ASMs approach the ship, the combat system radar will perform long-range search and detection above the horizon; tracks are subsequently initiated on the ASMs. The AAW system will then conduct a command/control sequence in which the sensors provide feedback and the control system provides weapon assignment for engagement. An initial engagement could be generated from a short-range missile system such as RAM. If new threats are identified, the capability will exist for rapid upgradeability of the Weapon System Baseline software due to the open systems architecture of the associated software. The ship will have the appropriate bandwidth to allow for rapid insertion of software upgrades as well as the subsequent changes to the technical documentation. The technical documentation has been developed in a language, which allows for changes to be incorporated in a timely and accurate manner. The sailor has received computer based training (CBT) operational scenarios and the associated change has been incorporated into the CBT.

Figure 3 depicts the ship self defense capability scenario. A ship on patrol search, detects, and engages 8 ASMs. The ship initially searches at above horizon and long range. It detects 8 incoming ASMs and then engages the attacks with the appropriate counter measures. The satellite communication allows for long range distance support.

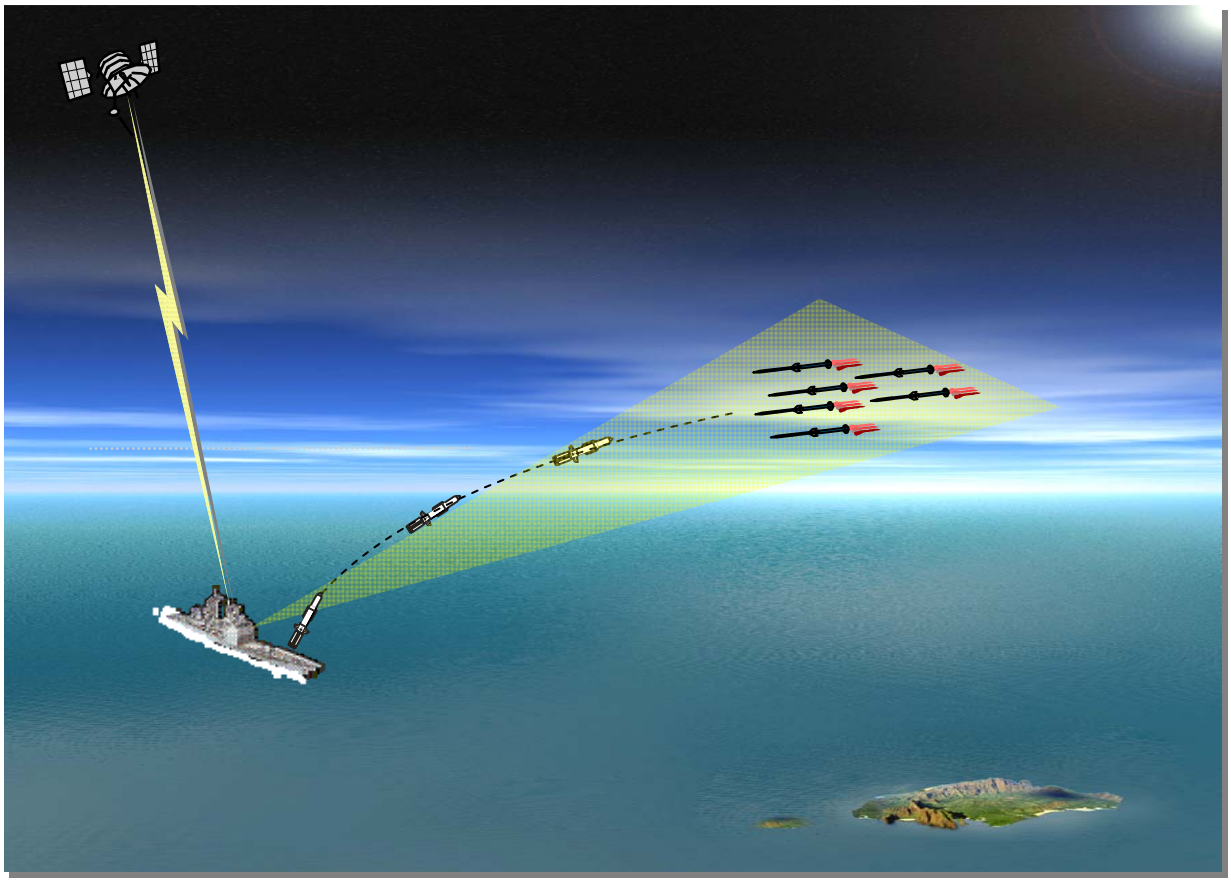


Figure 3 – Ship Self Defense ConOps

Scenario 3: Limited Area Defense (LAD) or Protection of High Value Units (HVUs)

As the ASM raid approach the ship, the AAW architecture's long range radar system detects and tracks the ASMs; detecting the first ASM at maximum range. The AAW system will then conduct a command/control sequence, where the sensors provide feedback and the control system provides weapon assignment for engagement. In the protection of HVUs, a longer and wider range engagement is necessary; the first possible engagement will be to assign a soft-kill weapon system such as decoys or chaff. The next possible engagement would be to deploy a long-range missile system such as ESSM or SM 2 (an x-band illuminator provides real-time kill assessment for ESSM or SM 2). The third line of defense would be to engage the incoming ASM attacks with a short-range missile system such as RAM. The supply system will be adequately stocked to support the needs of the warfighters. The transportation system will support the ship in all areas around the globe. The goal of the supply system will be to maintain the right level of inventory and the right places and to transport those supplies within a prescribed amount of time given the criticality of the ship's requirement.

Figure 4 depicts high value unit scenario. The ship conducts operations in a strike group and performs search patterns to detect and defend enemy attacks towards the other ships within the strike group. The center ship search, detects, track, and engage the 8 incoming ASMs to protect the strike group. Each ship and satellite is also in net centric communication during operations.

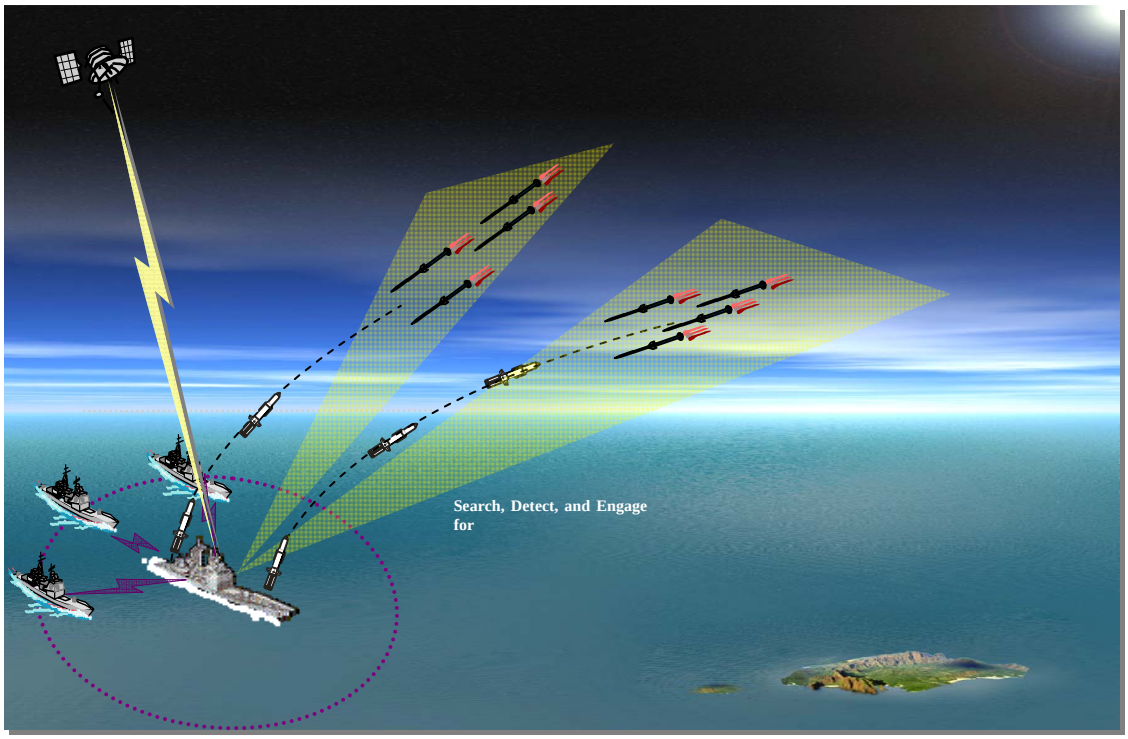


Figure 4 – High Value Units ConOps

References

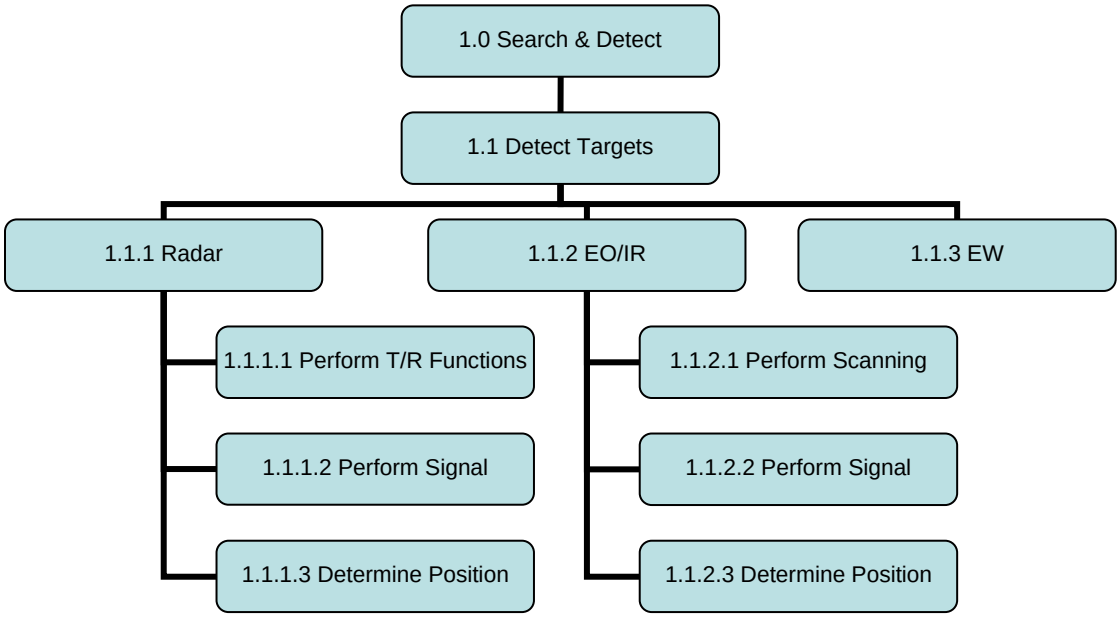
Bjorke, Per, Richard E. Fairley, and Richard H. Thayer. The Bridge from Operational Requirements to Technical Specifications. California State University, Sacramento and Drexel University.

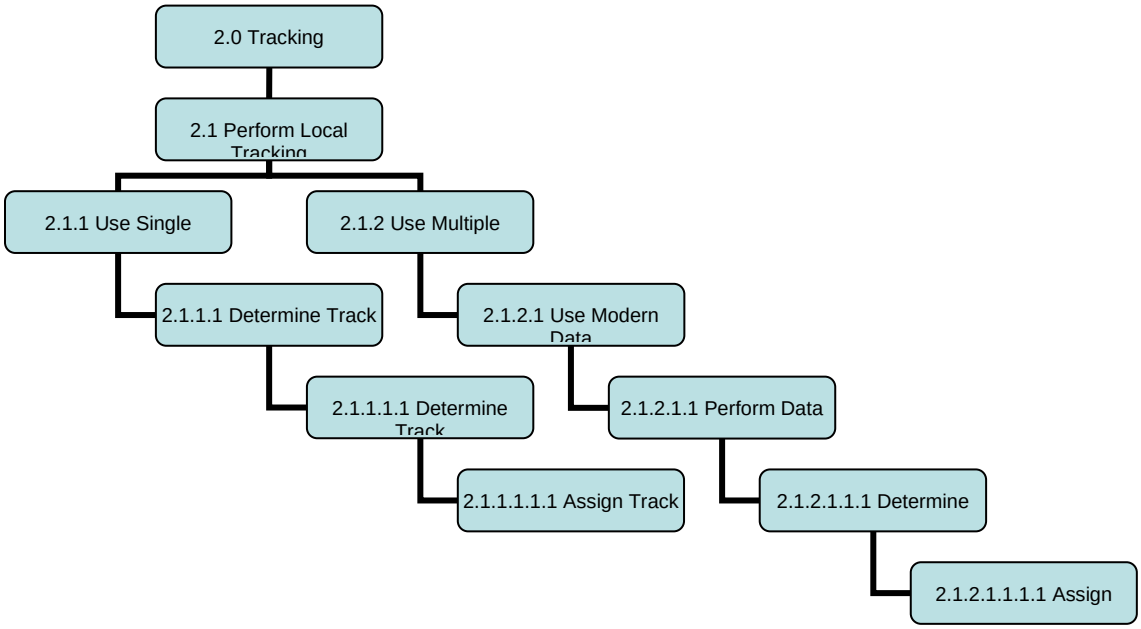
Green, Mike. 2008. SE 3123 Weapon Systems II, Final Exam, Problem 4.

Maier, Mark W. and Eberhardt Rechtin, 2002, The Art of Systems Architecting, Second Edition. CRC Press.

Payne, Craige M., 2006. Principles of Naval Weapon Systems. Annapolis, MD: Naval Institute Press

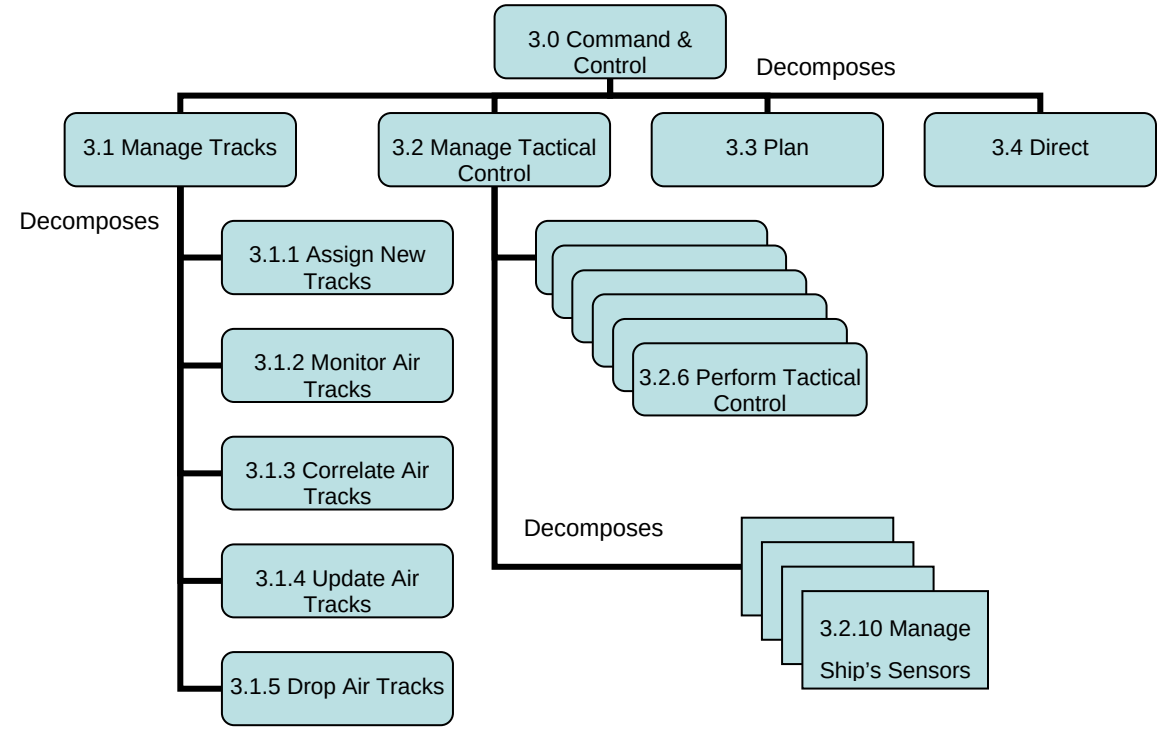
Attachment (1) - Functional Decomposition (Search & Detect, Track, C2 and Engage)

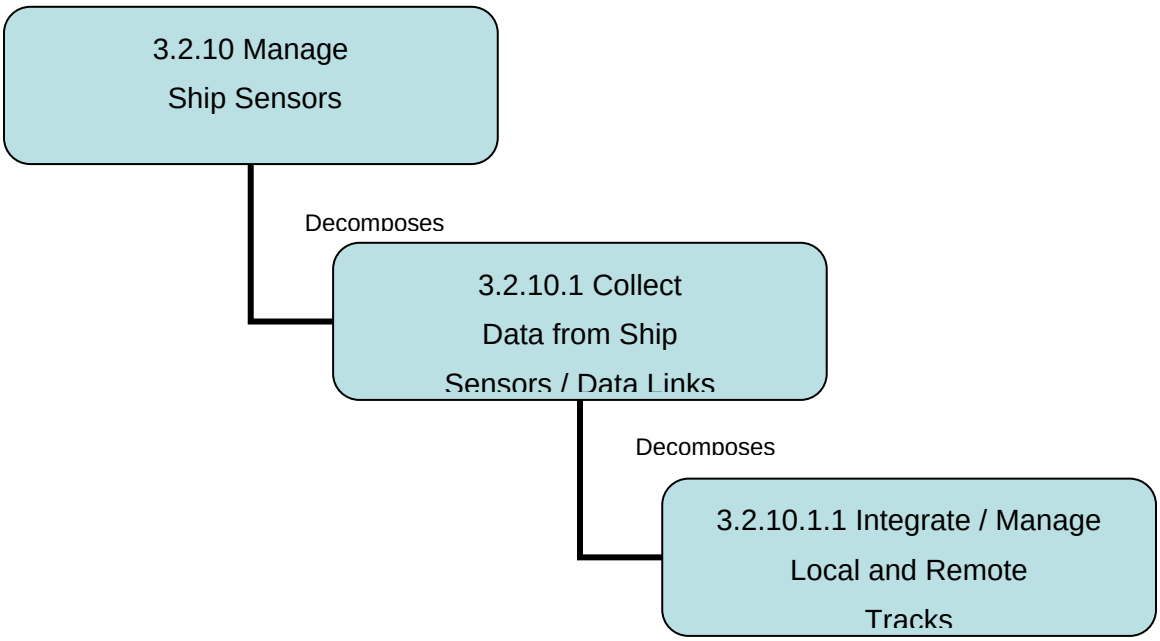




Attachment (1) - Functional Decomposition (Search & Detect, Track, C2 and Engage)

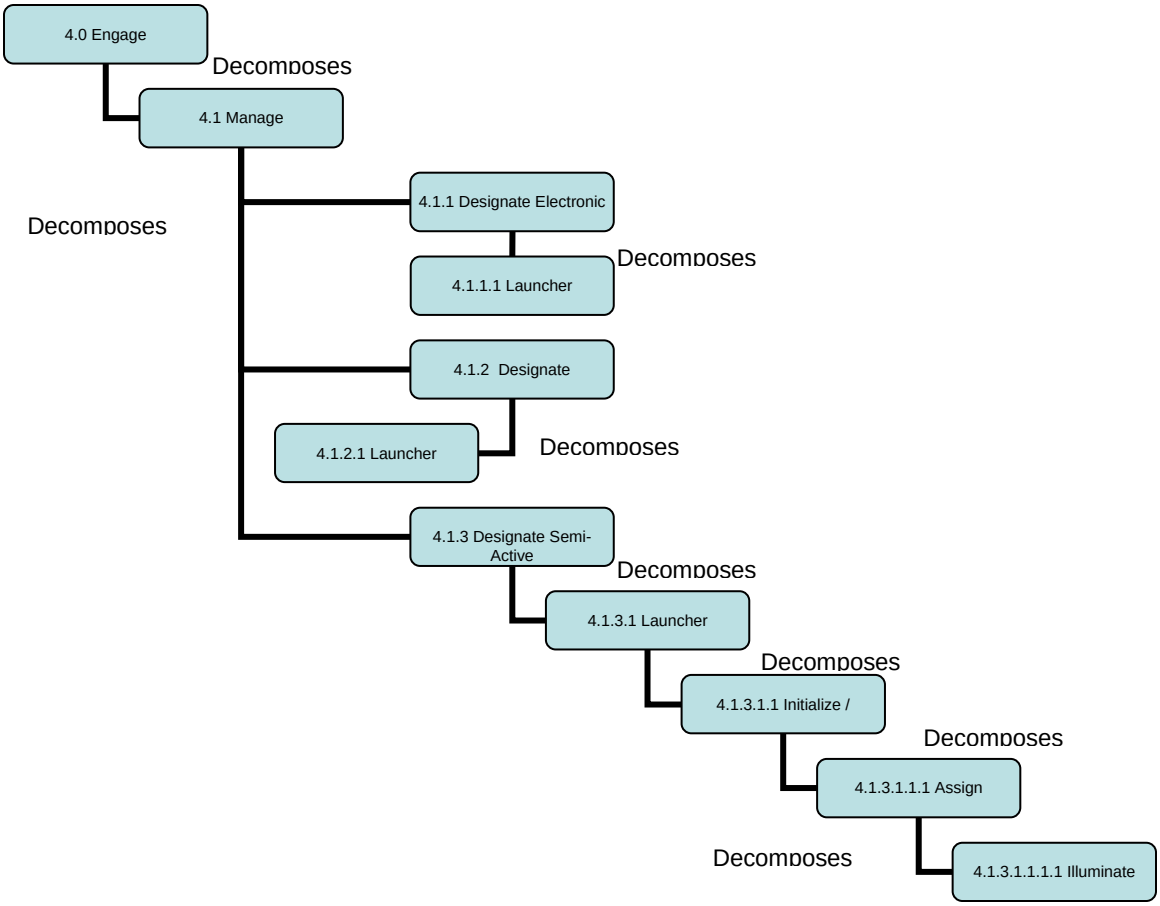
(cont.)



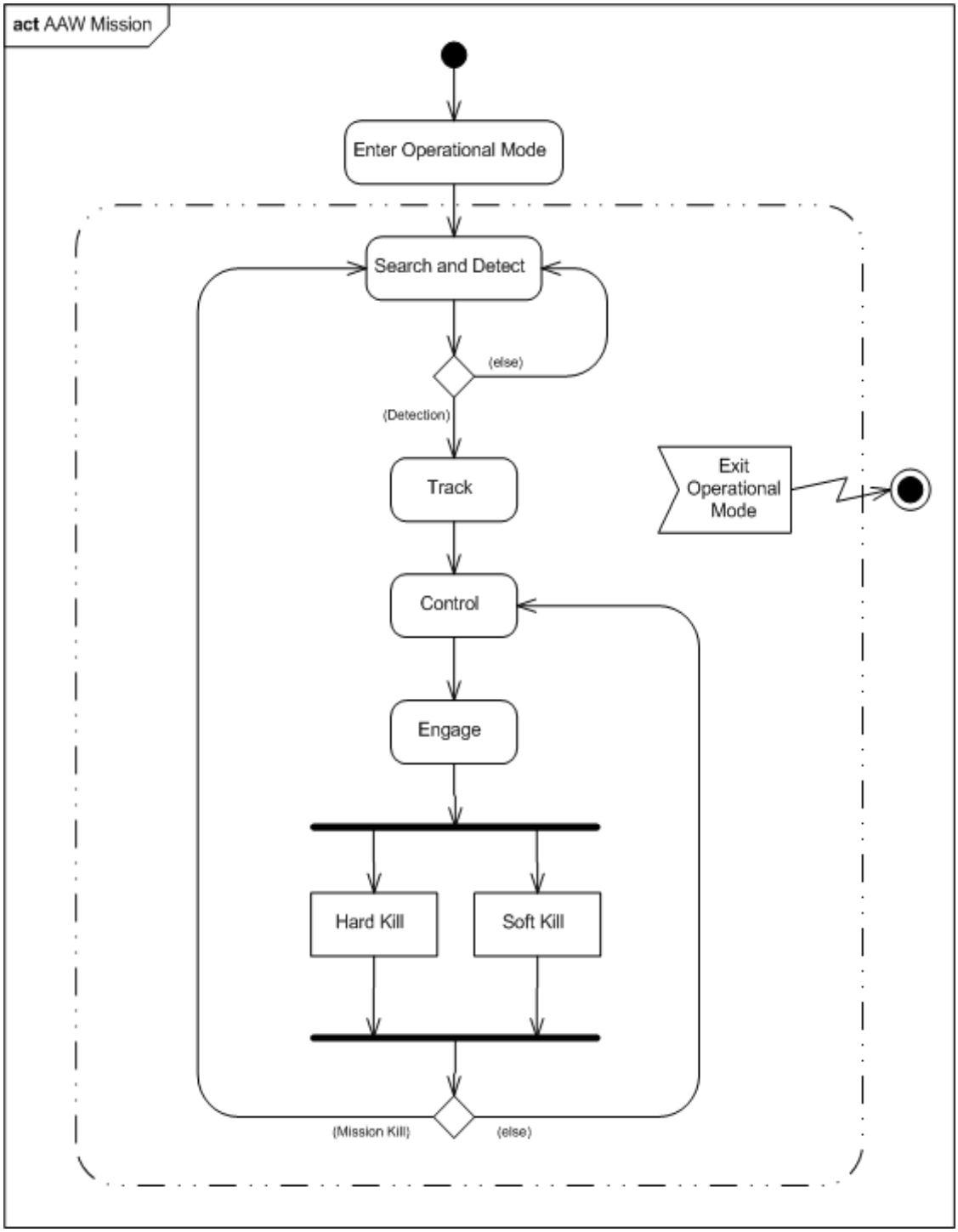


Attachment (1) - Functional Decomposition (Search & Detect, Track, C2 and Engage)

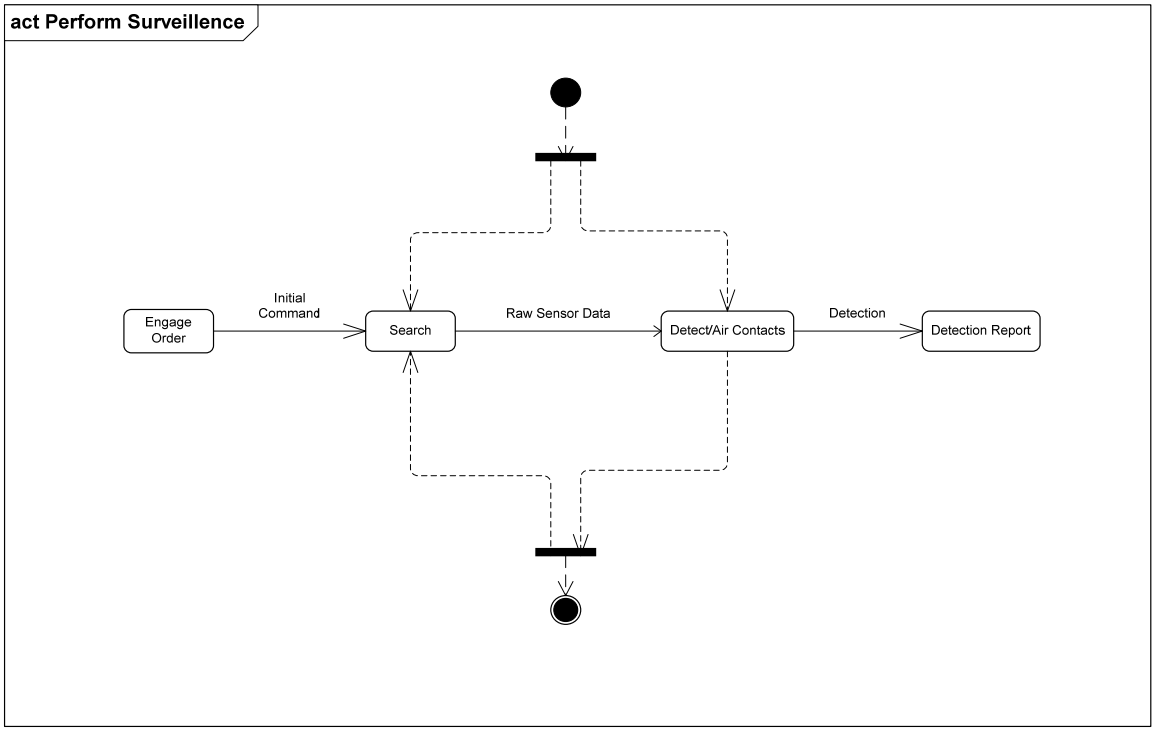
(cont.)

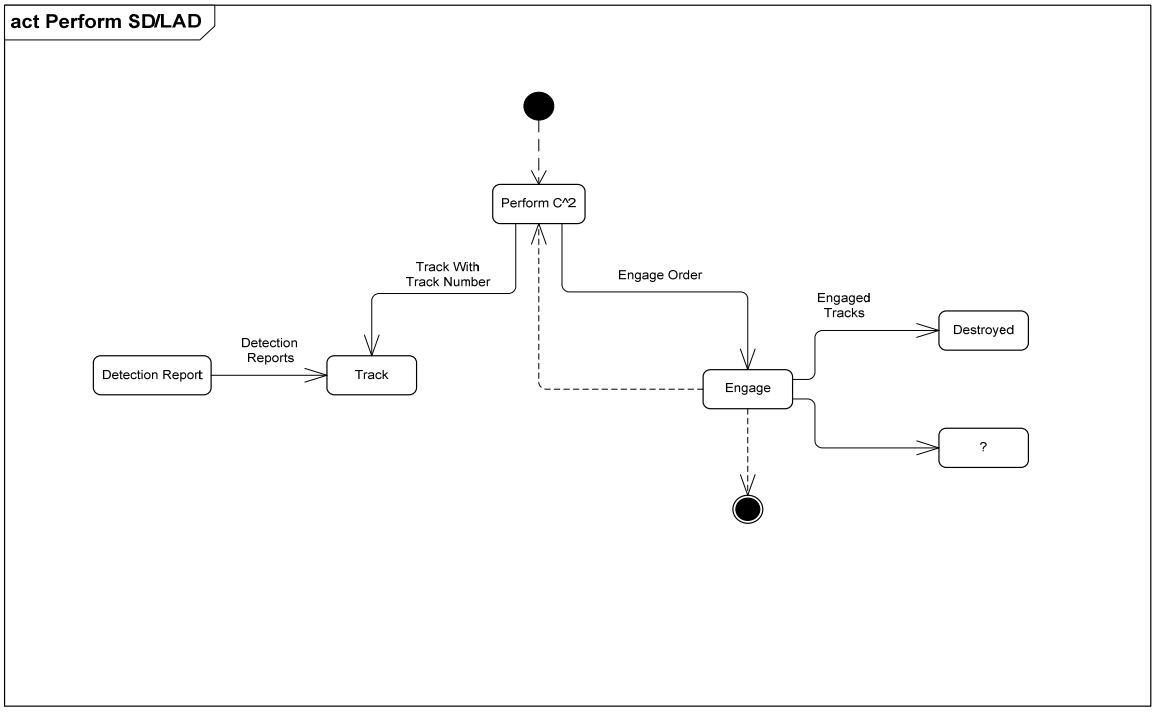


Attachment (2) – AAW Activity Diagrams

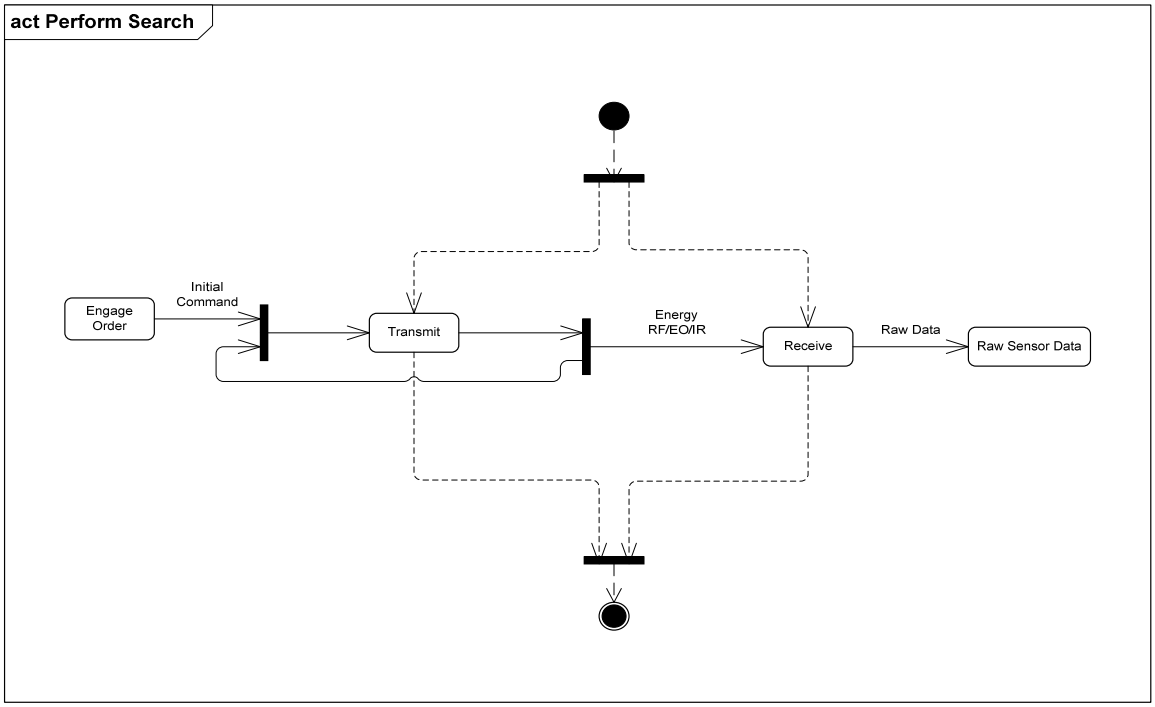


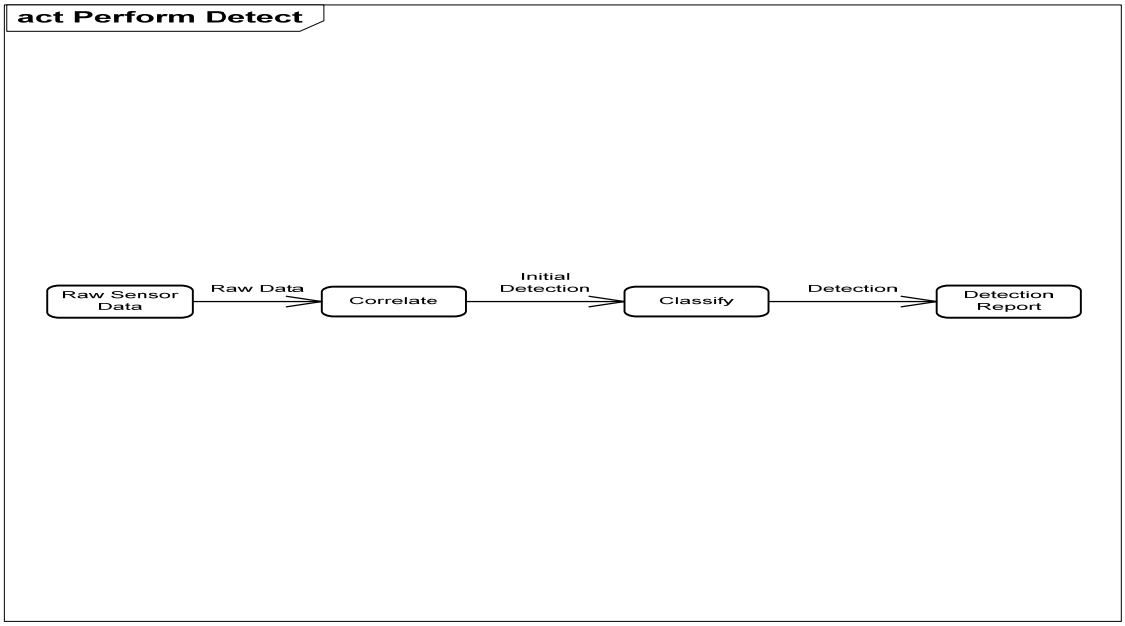
Attachment (2) – AAW Activity Diagrams (cont.)



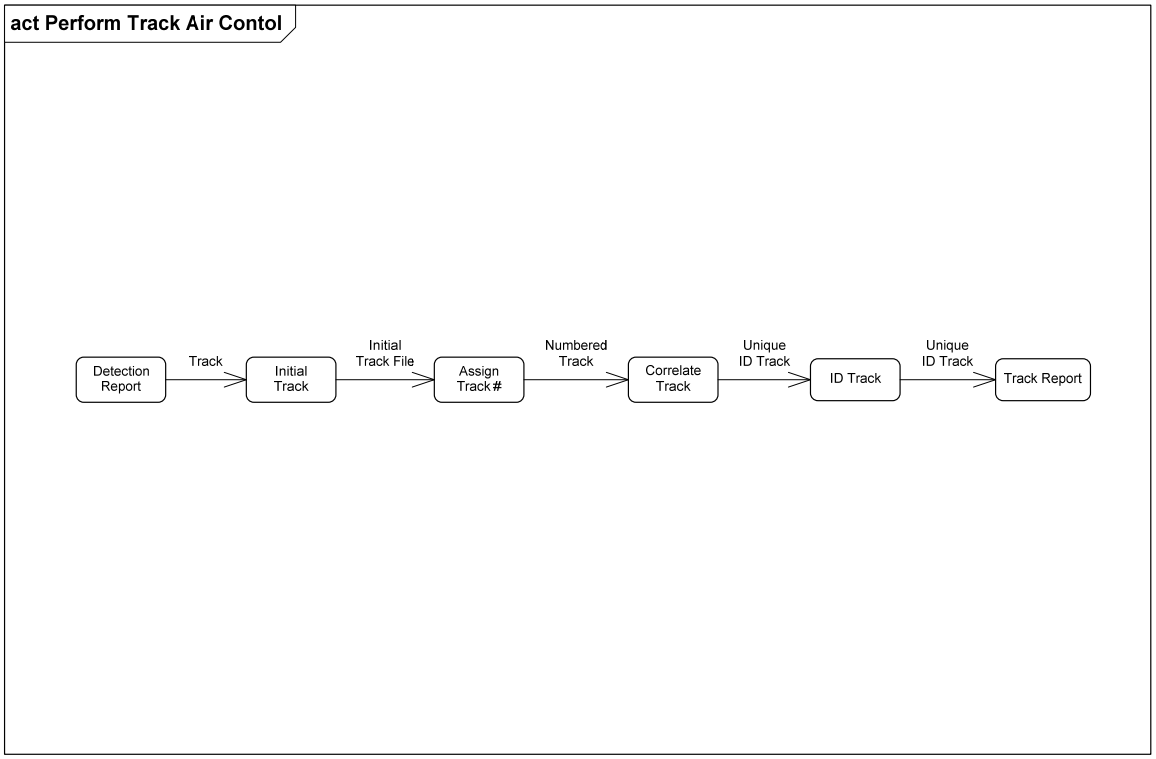


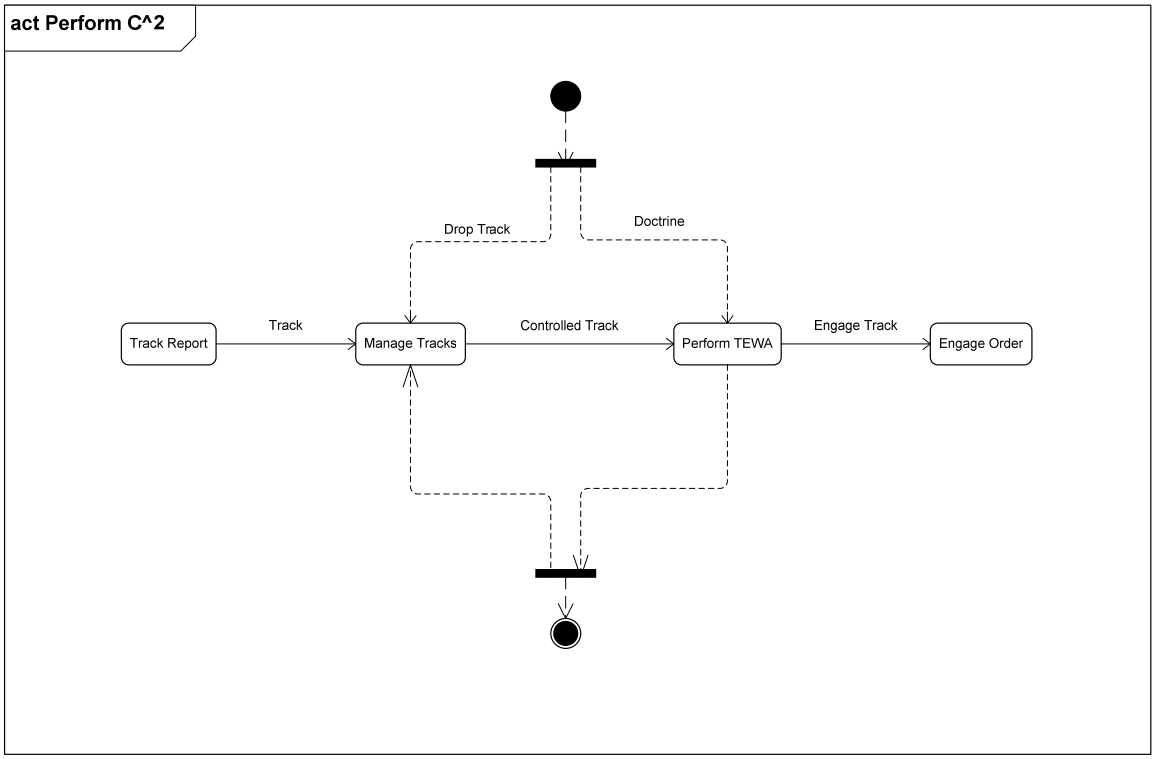
Attachment (2) – AAW Activity Diagrams (cont.)



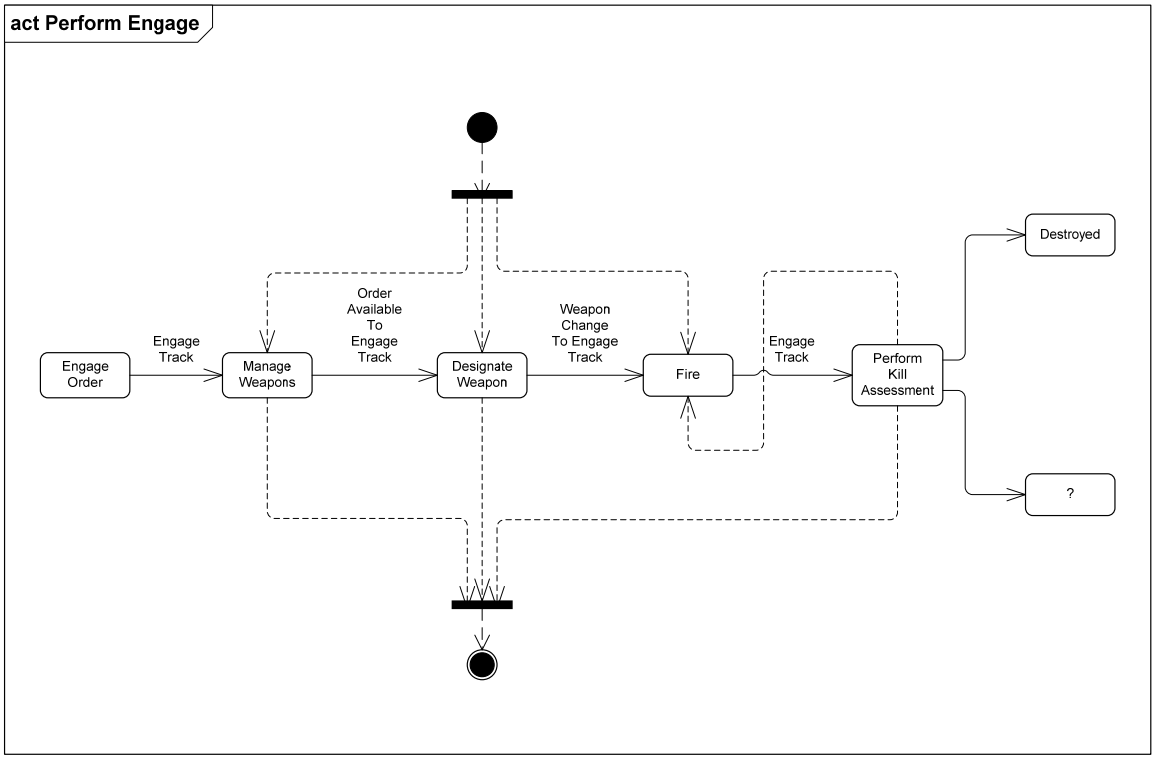


Attachment (2) – AAW Activity Diagrams (cont.)





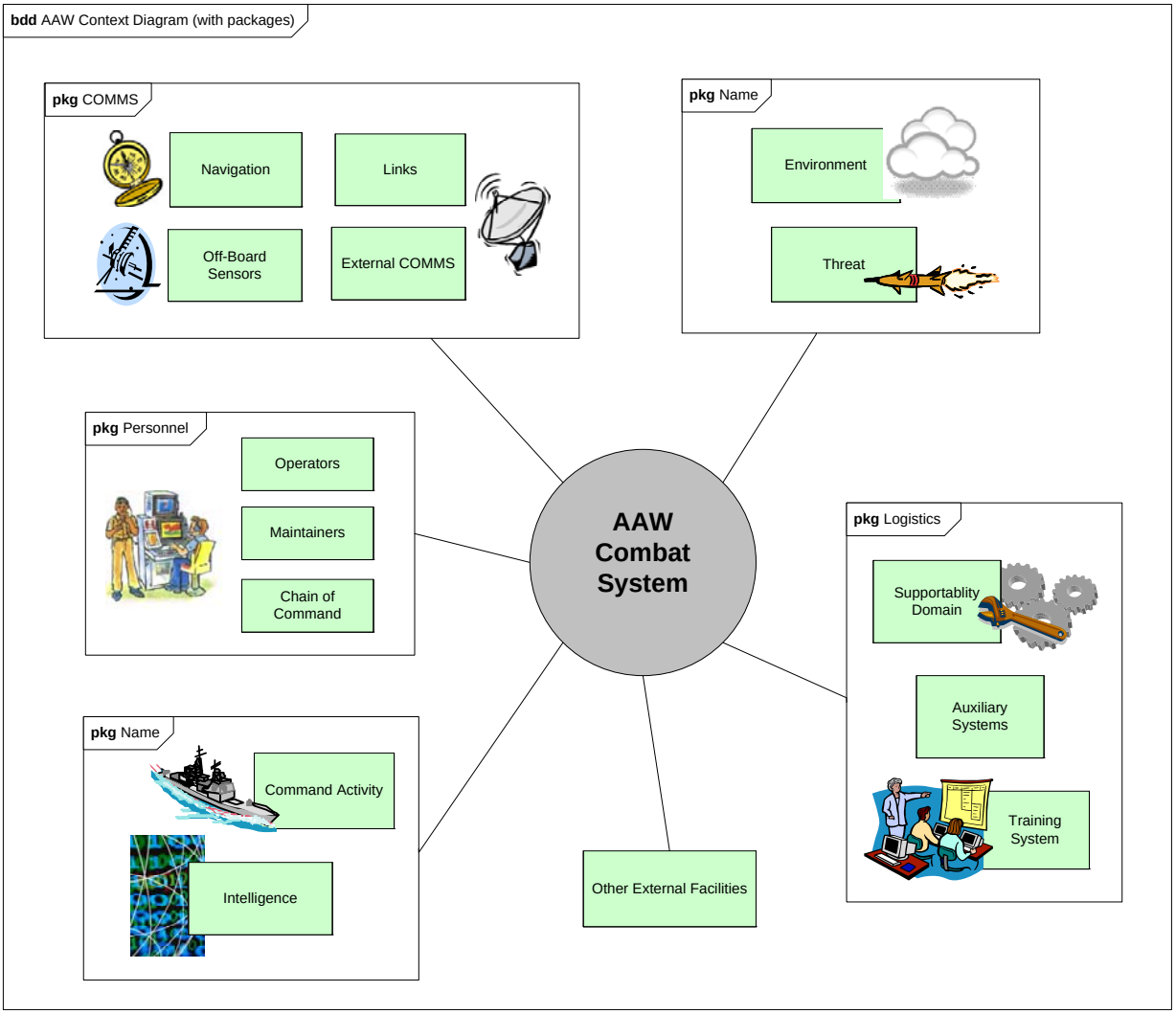
Attachment (2) – AAW Activity Diagrams (cont.)



APPENDIX H: SYSML PRODUCTS

SysML Context Diagrams

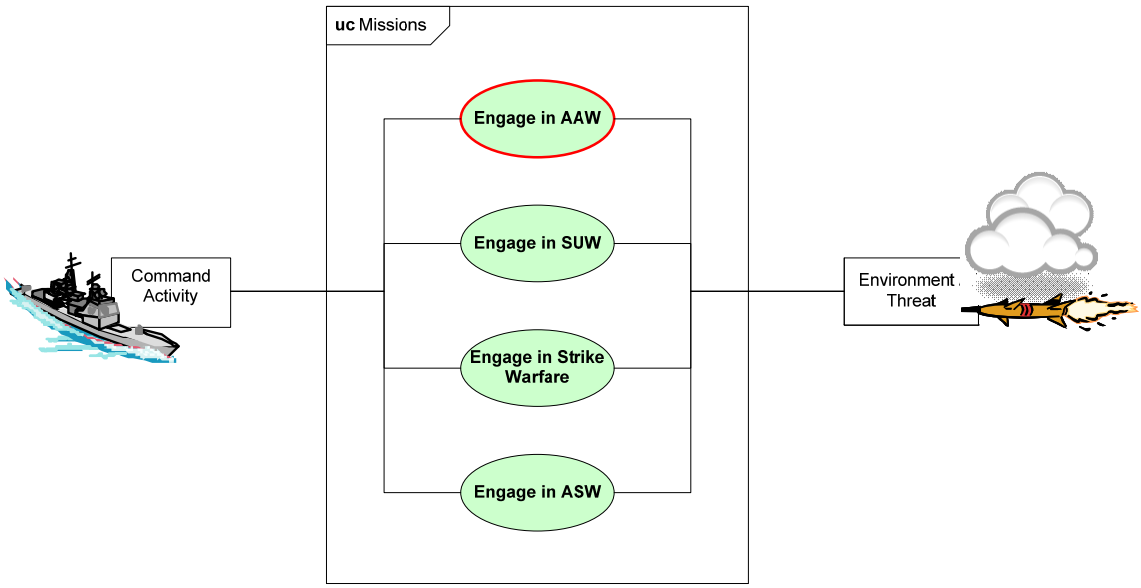
AAW with Packages



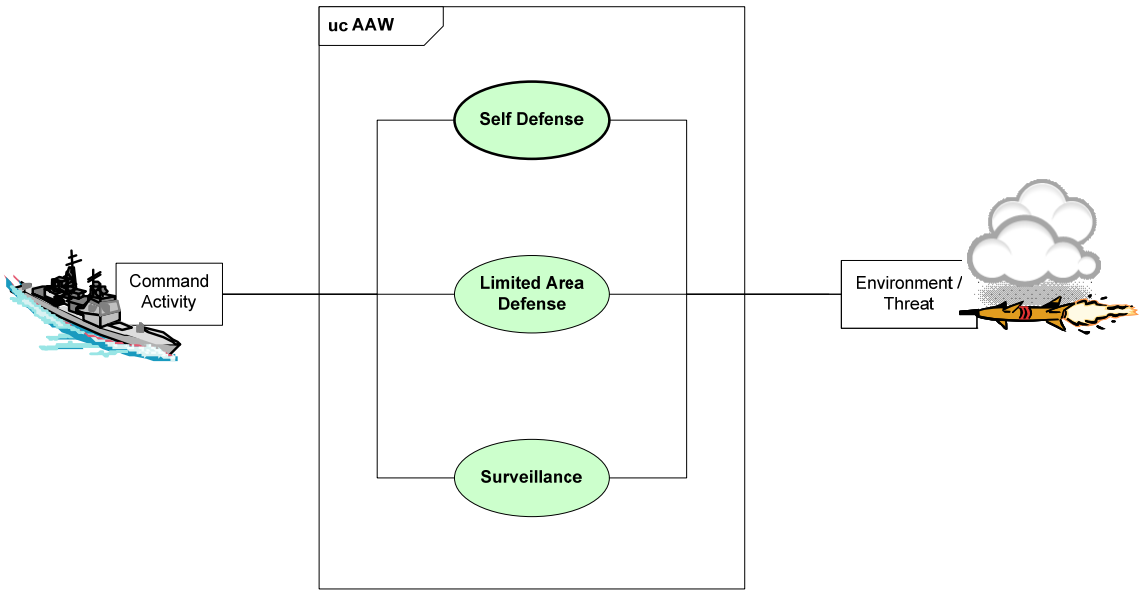
Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

SysML Use Case Diagrams

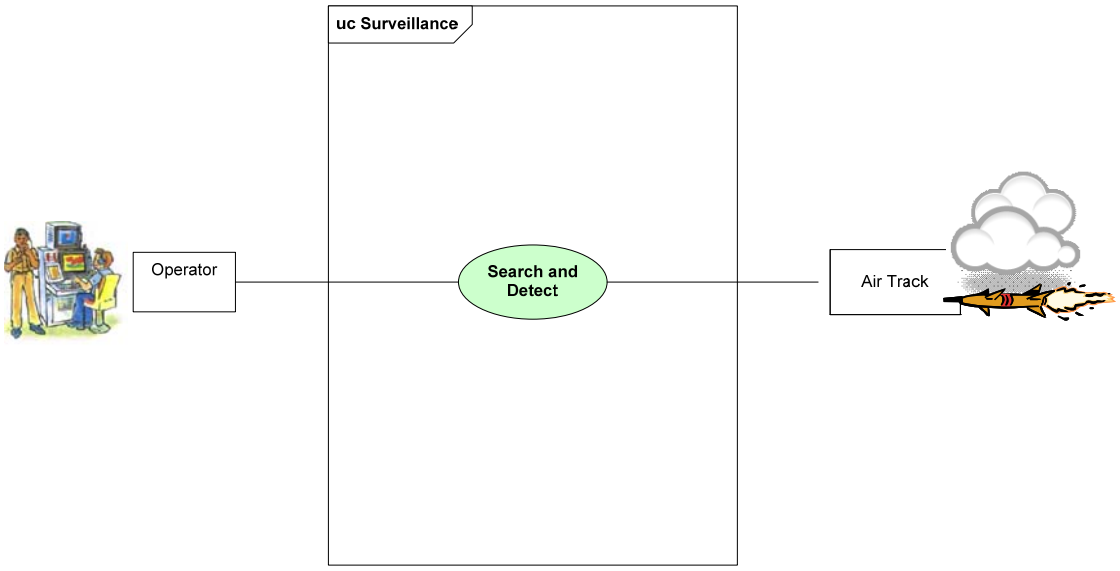
Multi-Mission



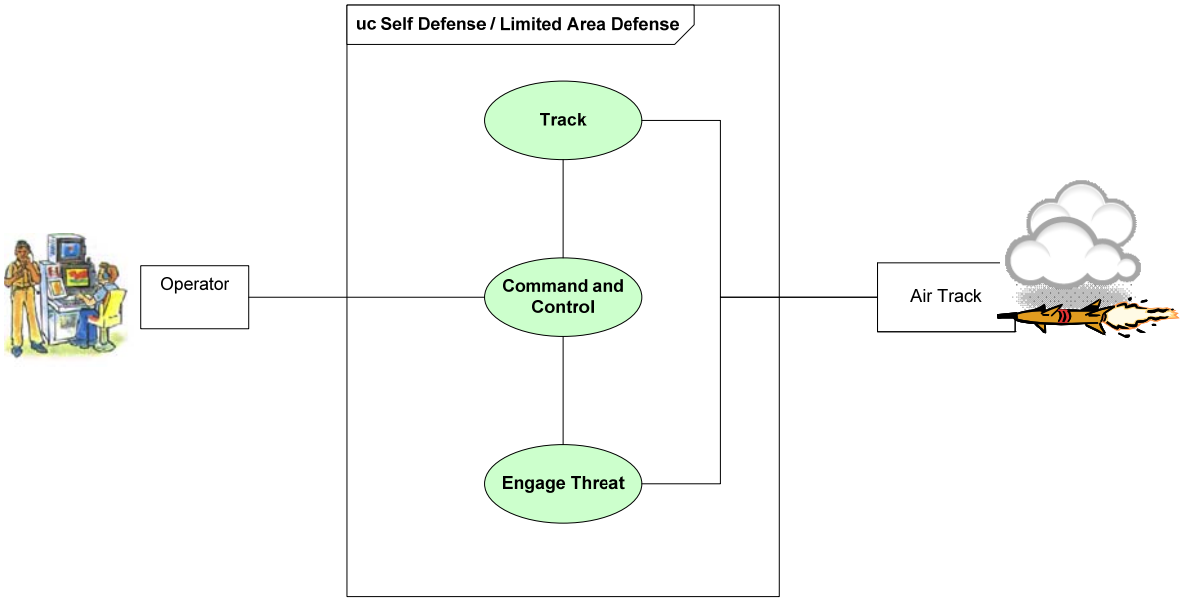
AAW Mission



Surveillance

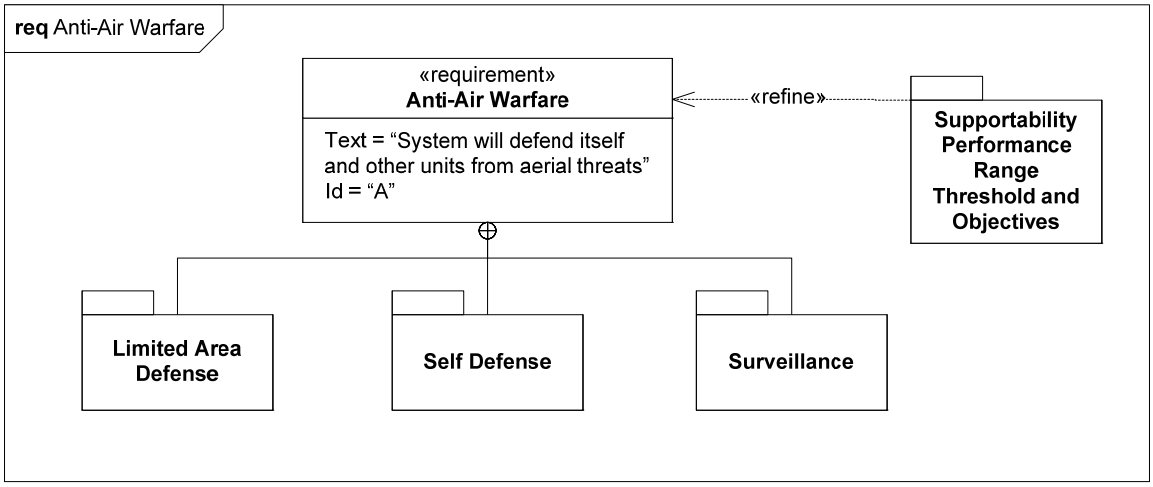


SD/LAD

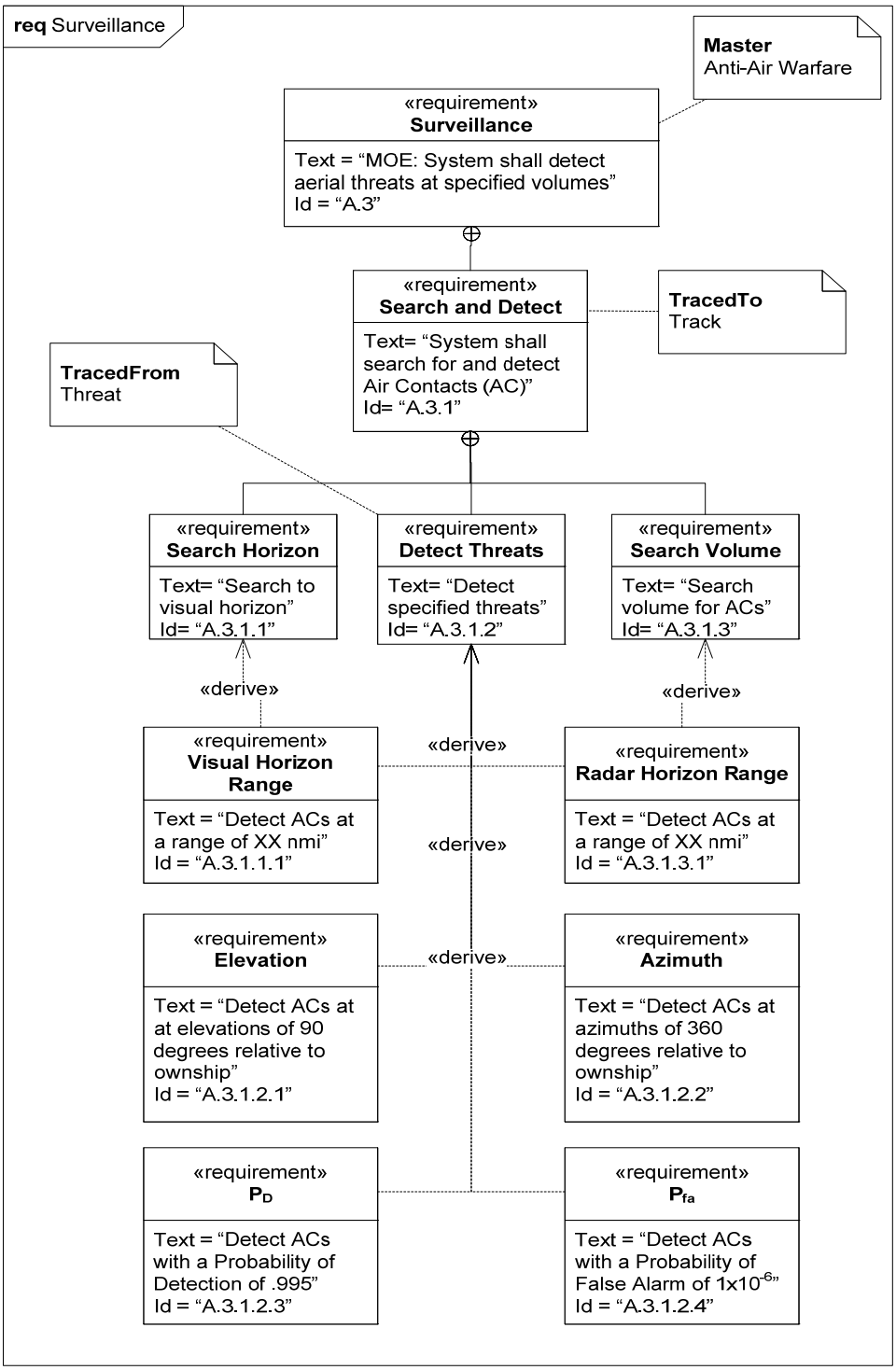


SysML Requirements Diagrams

Anti-Air Warfare

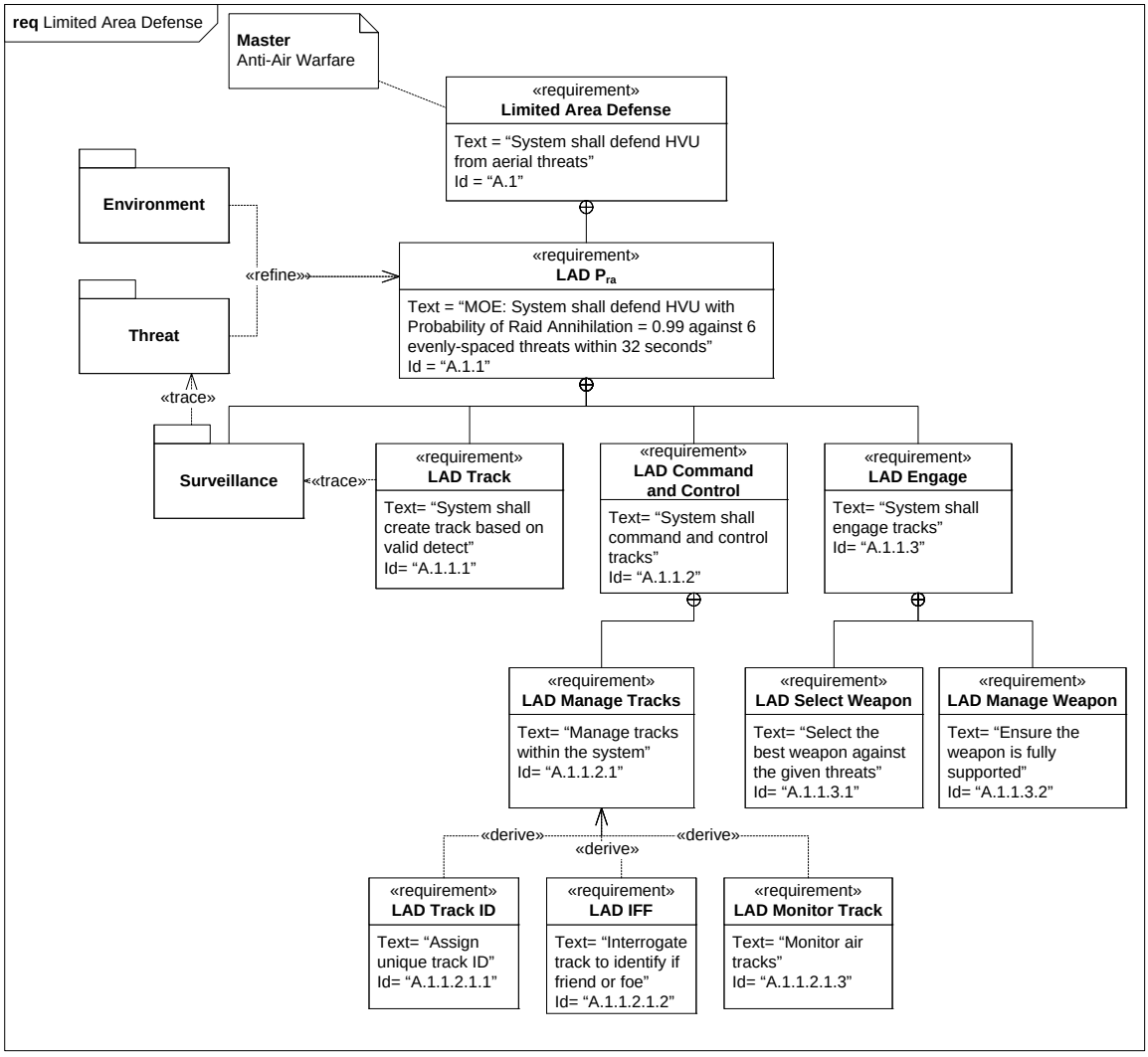


Surveillance Package

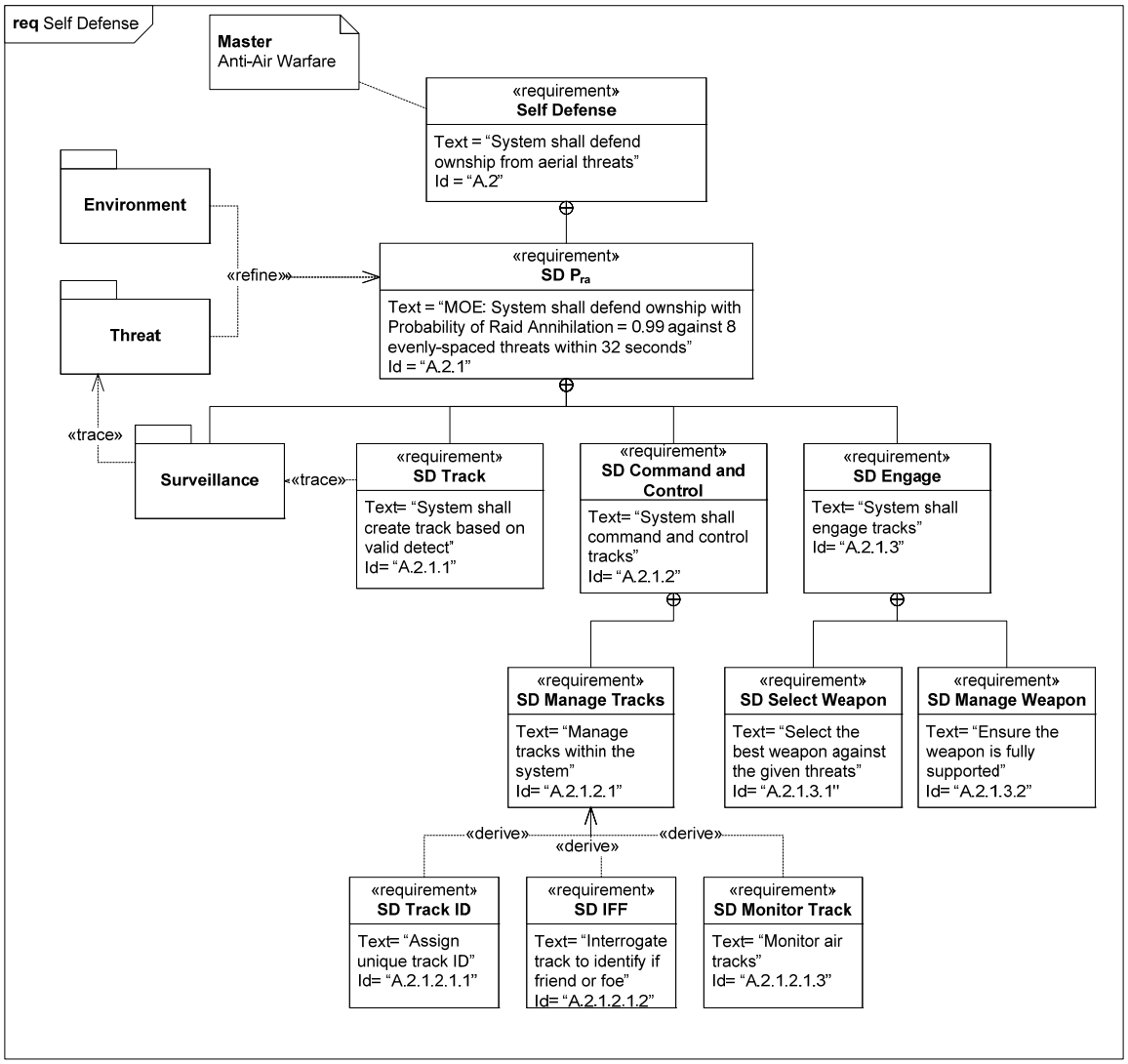


Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

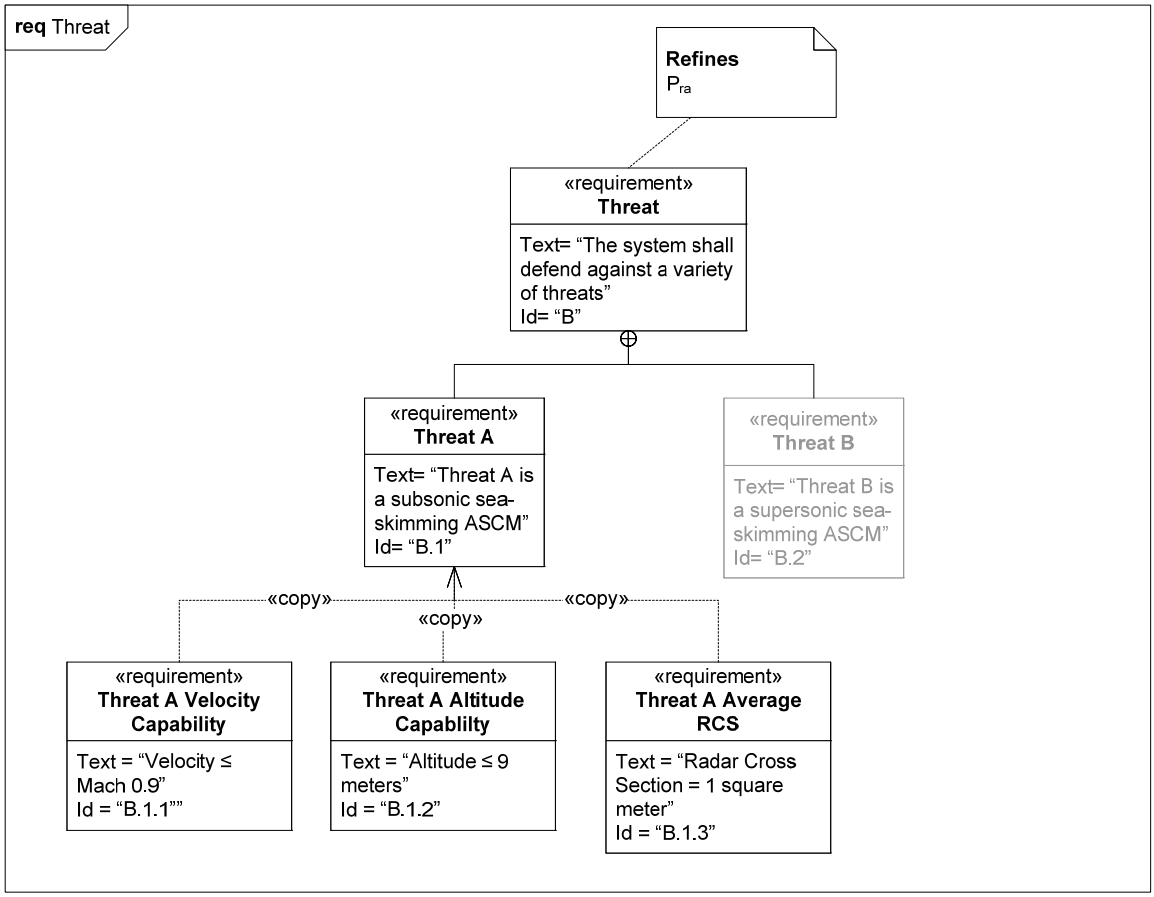
Limited Area Defense Package



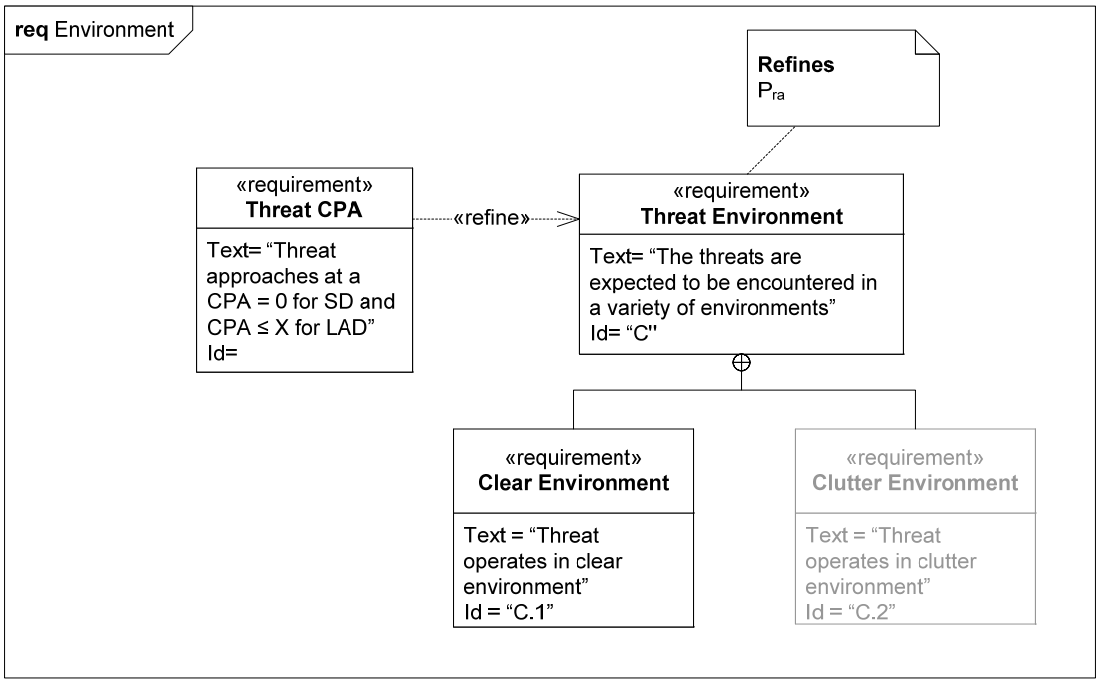
Self Defense Package



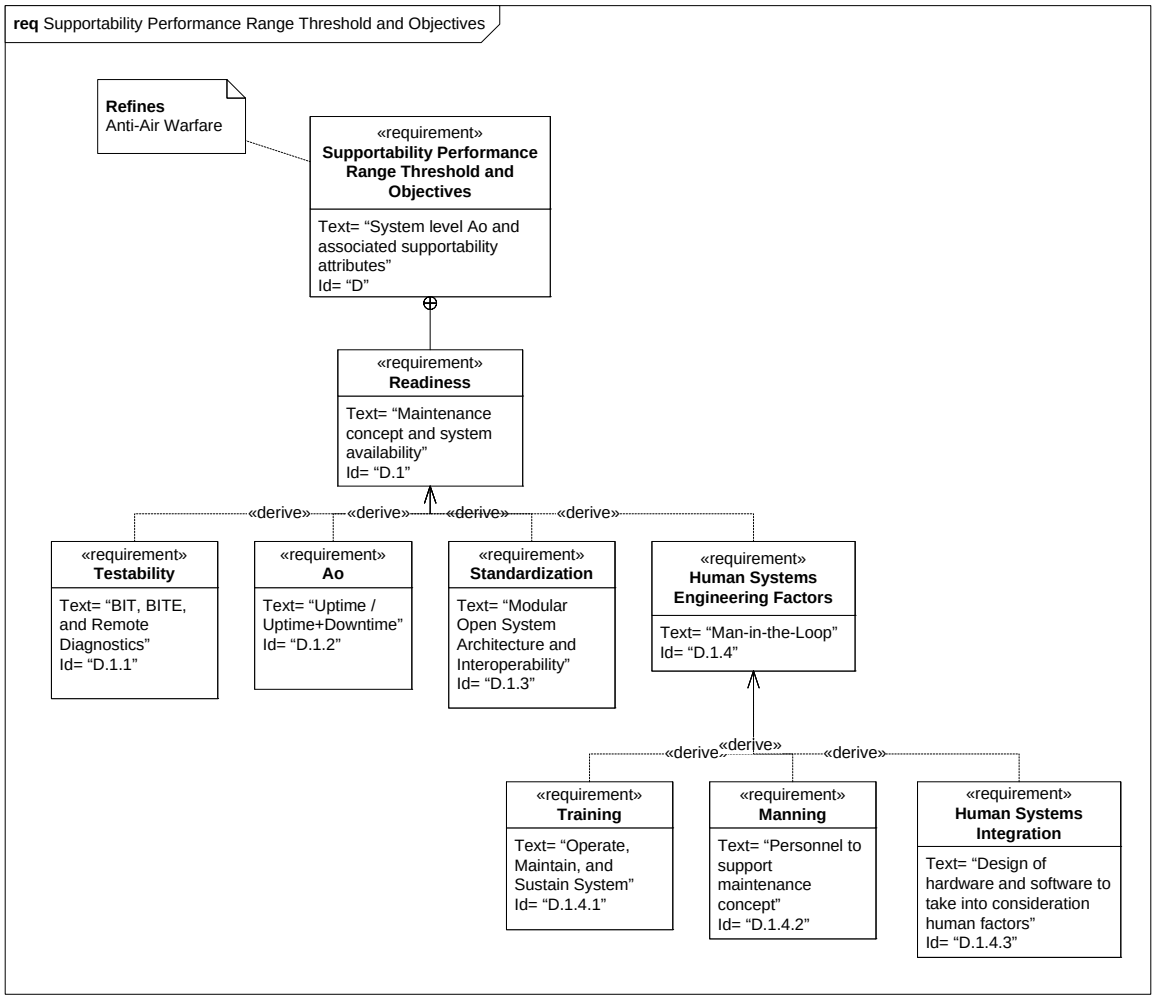
Threat Package



Environment Package

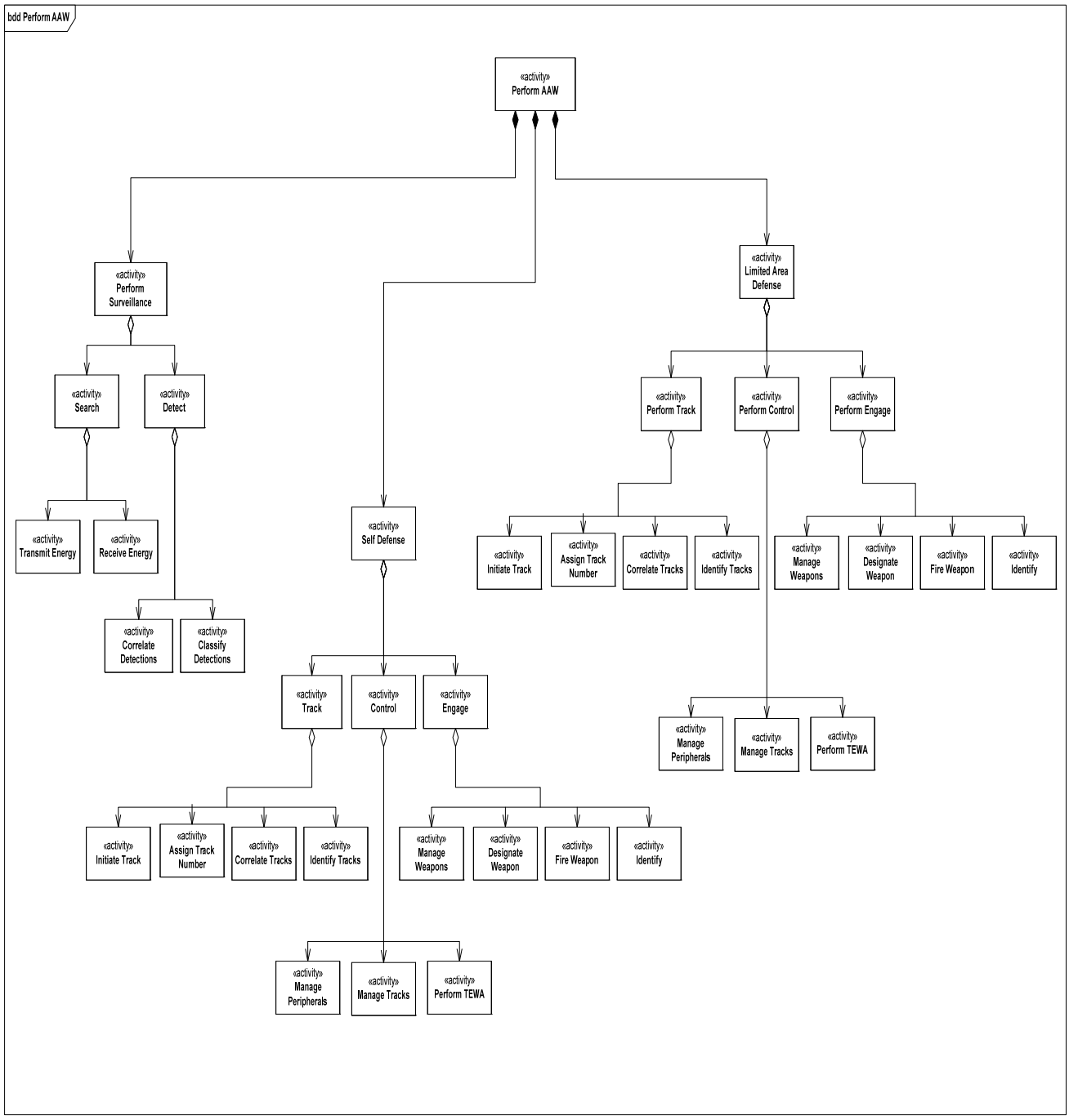


Supportability Package



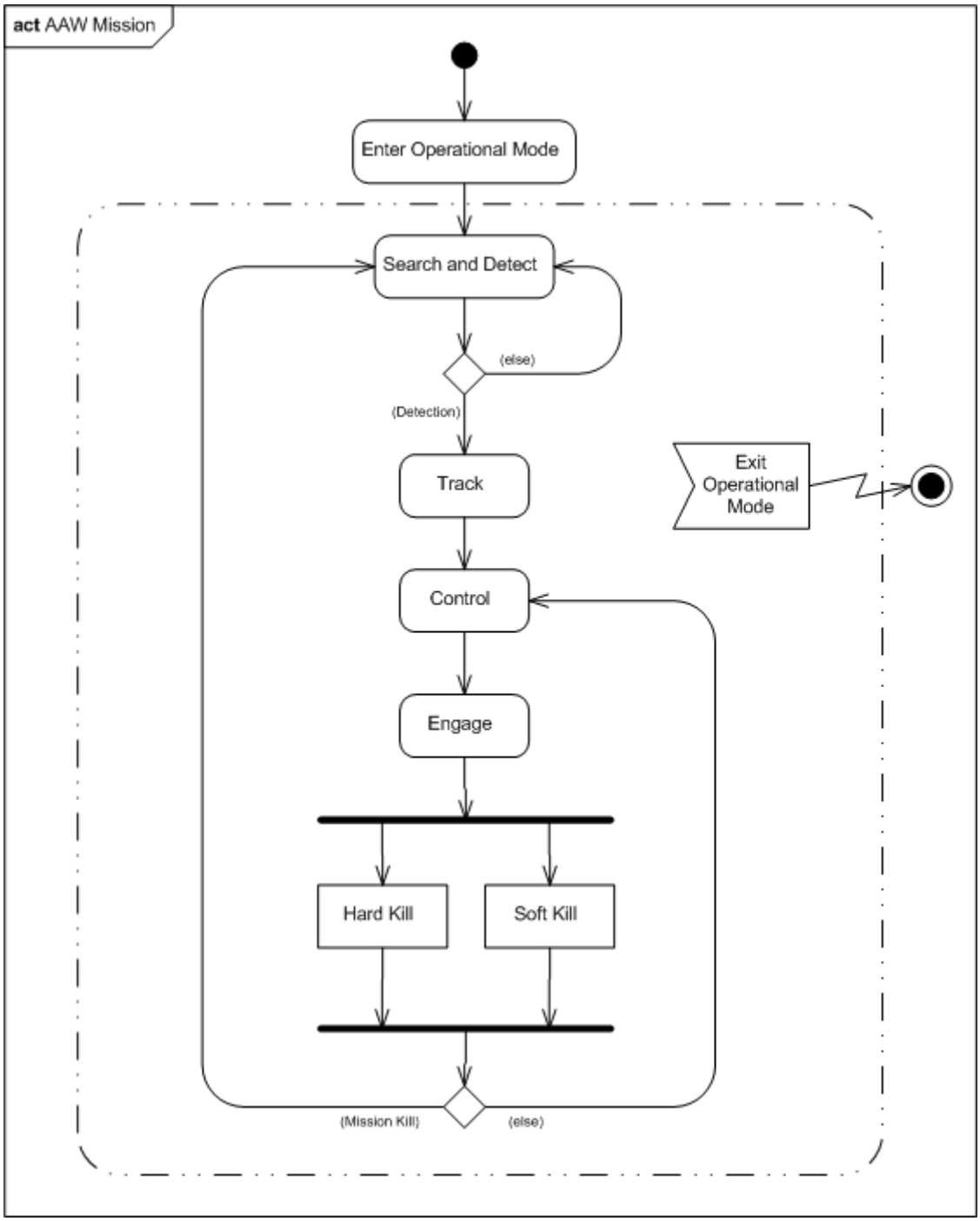
SysML AAW Functional Architecture

Perform AAW



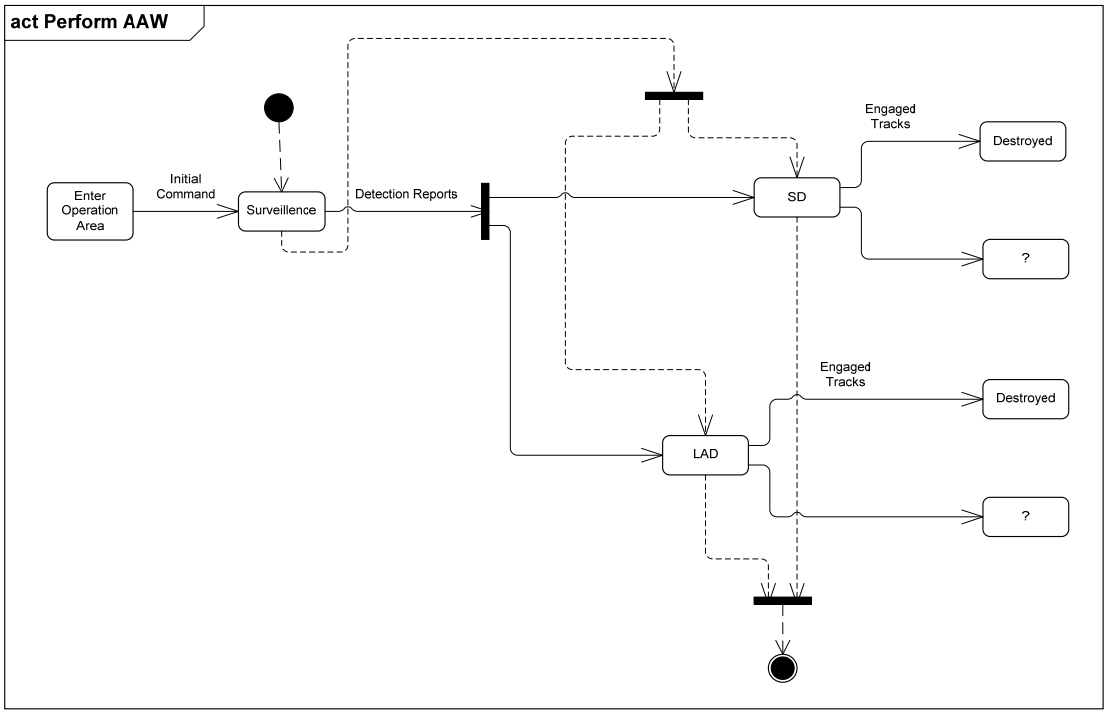
SysML Activity Diagrams

AAW Mission Thread

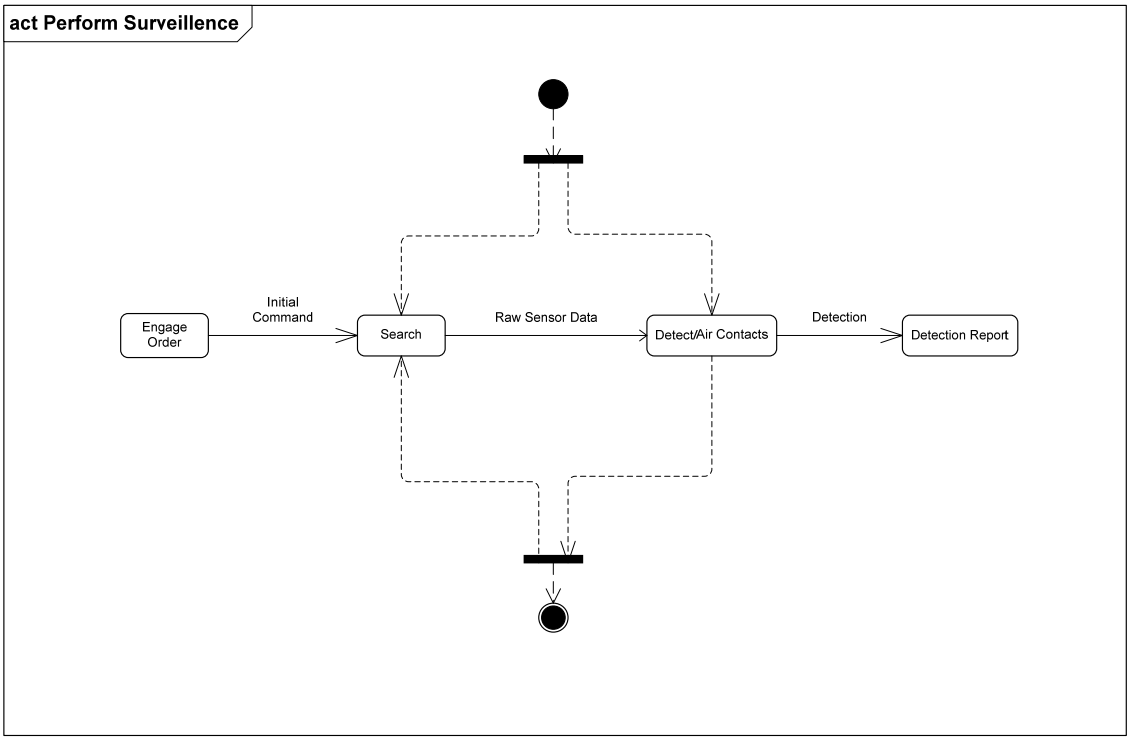


Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

Perform AAW

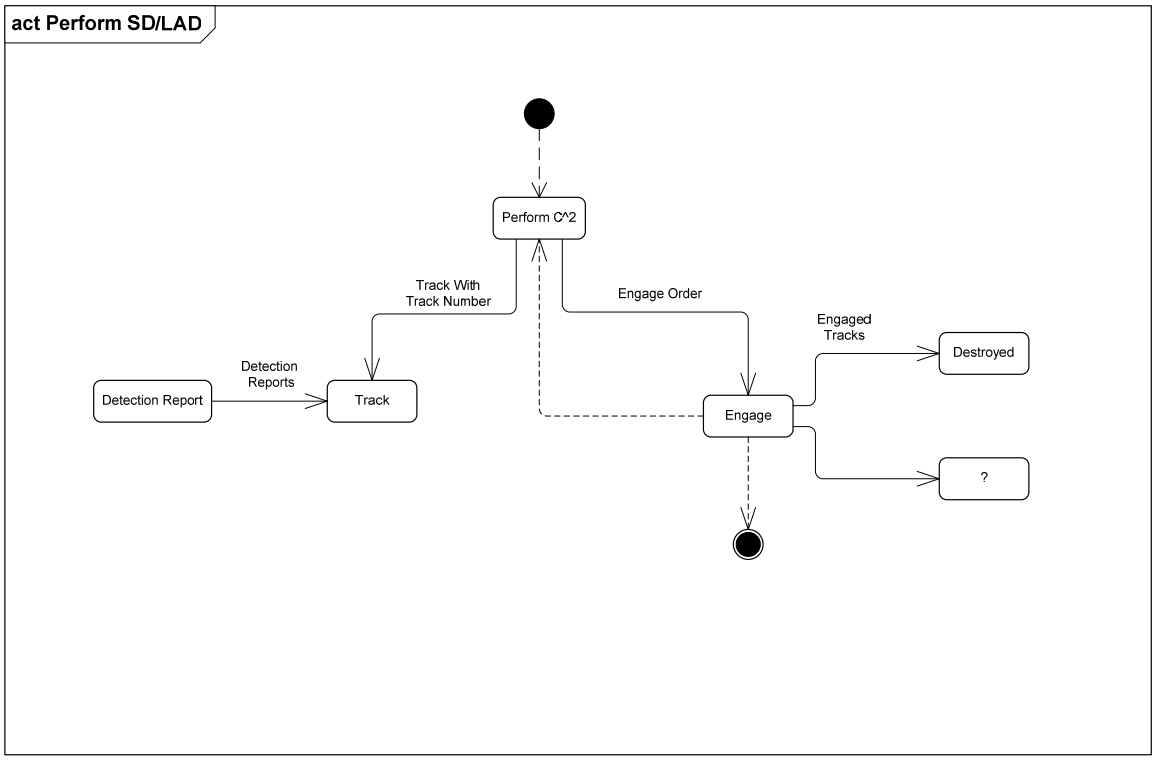


Perform Surveillance

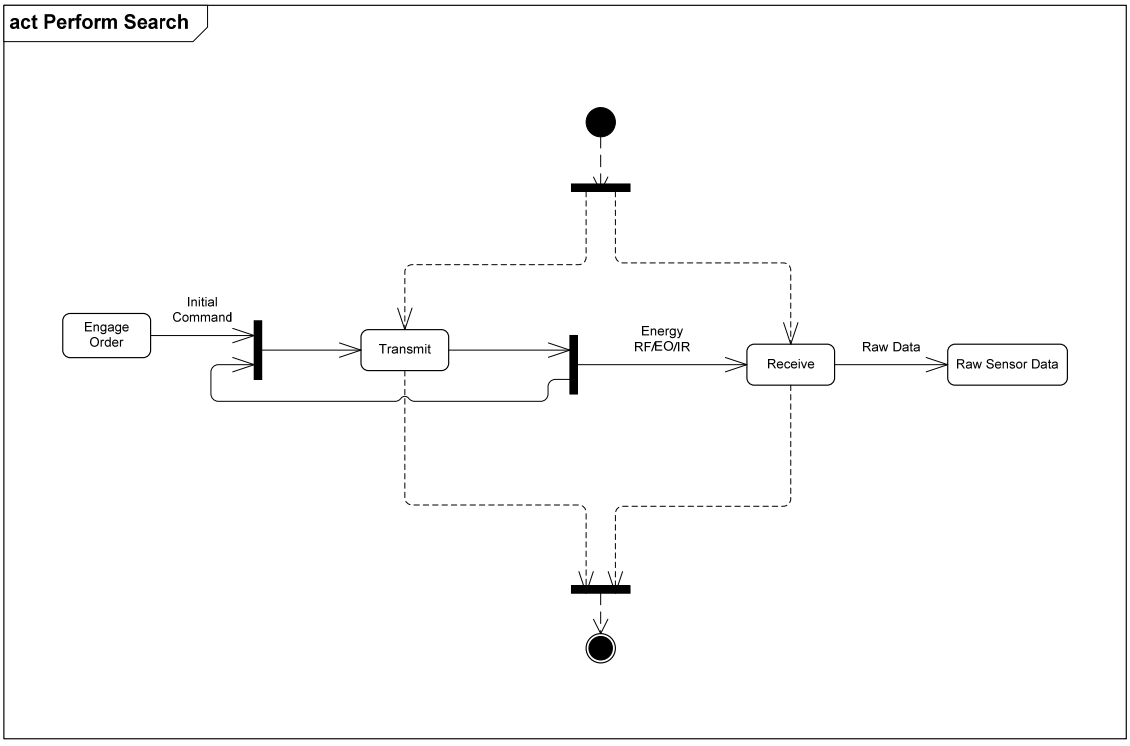


Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

Perform SD/LAD

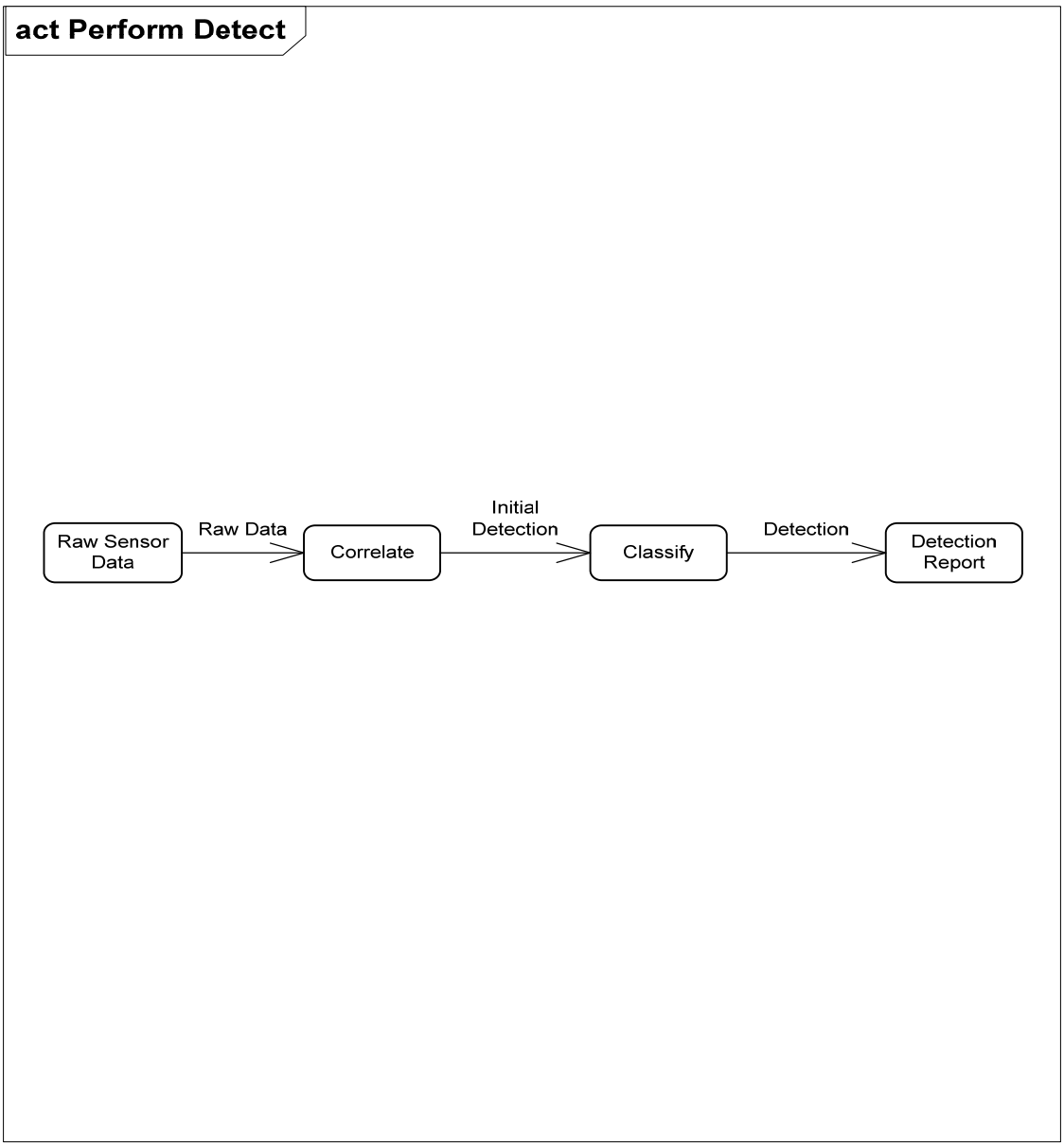


Perform Search

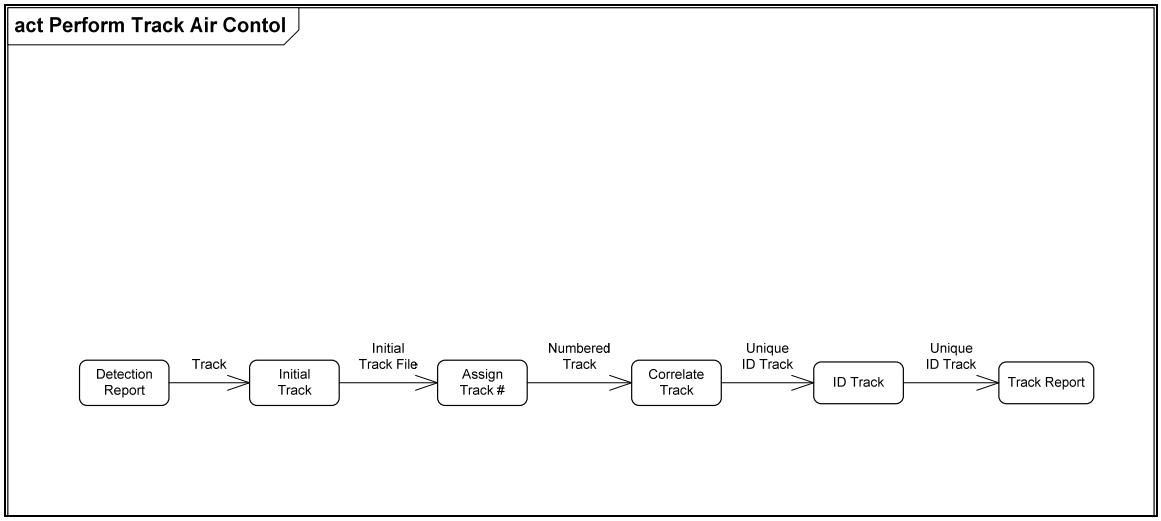


Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

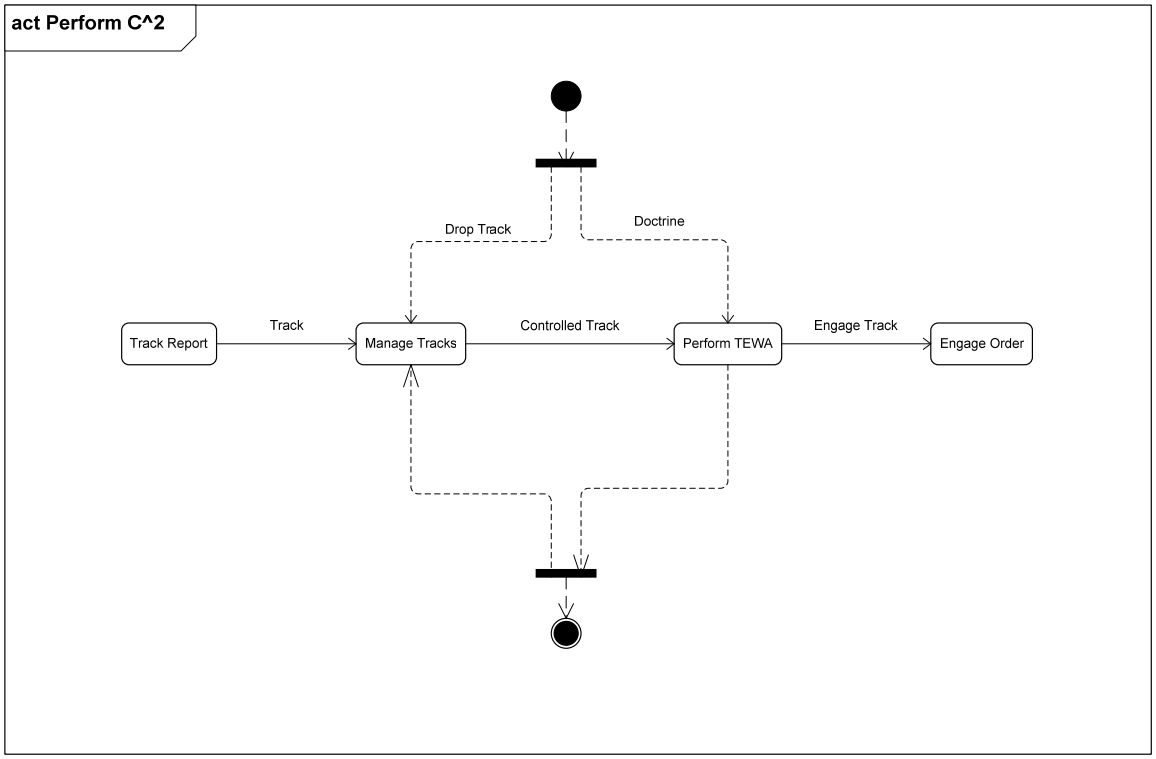
Perform Detect



Perform Track Air Control

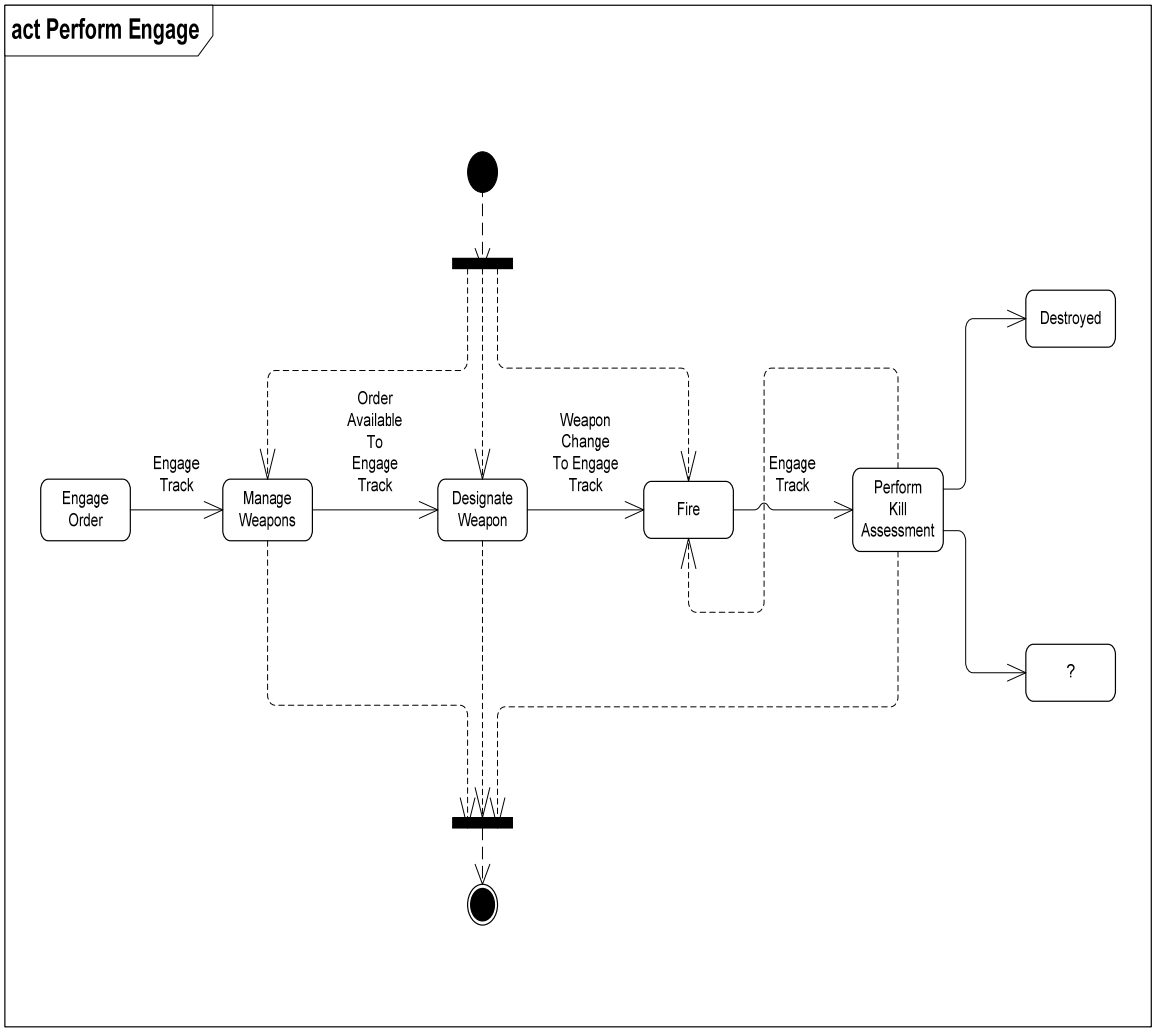


Perform Command & Control

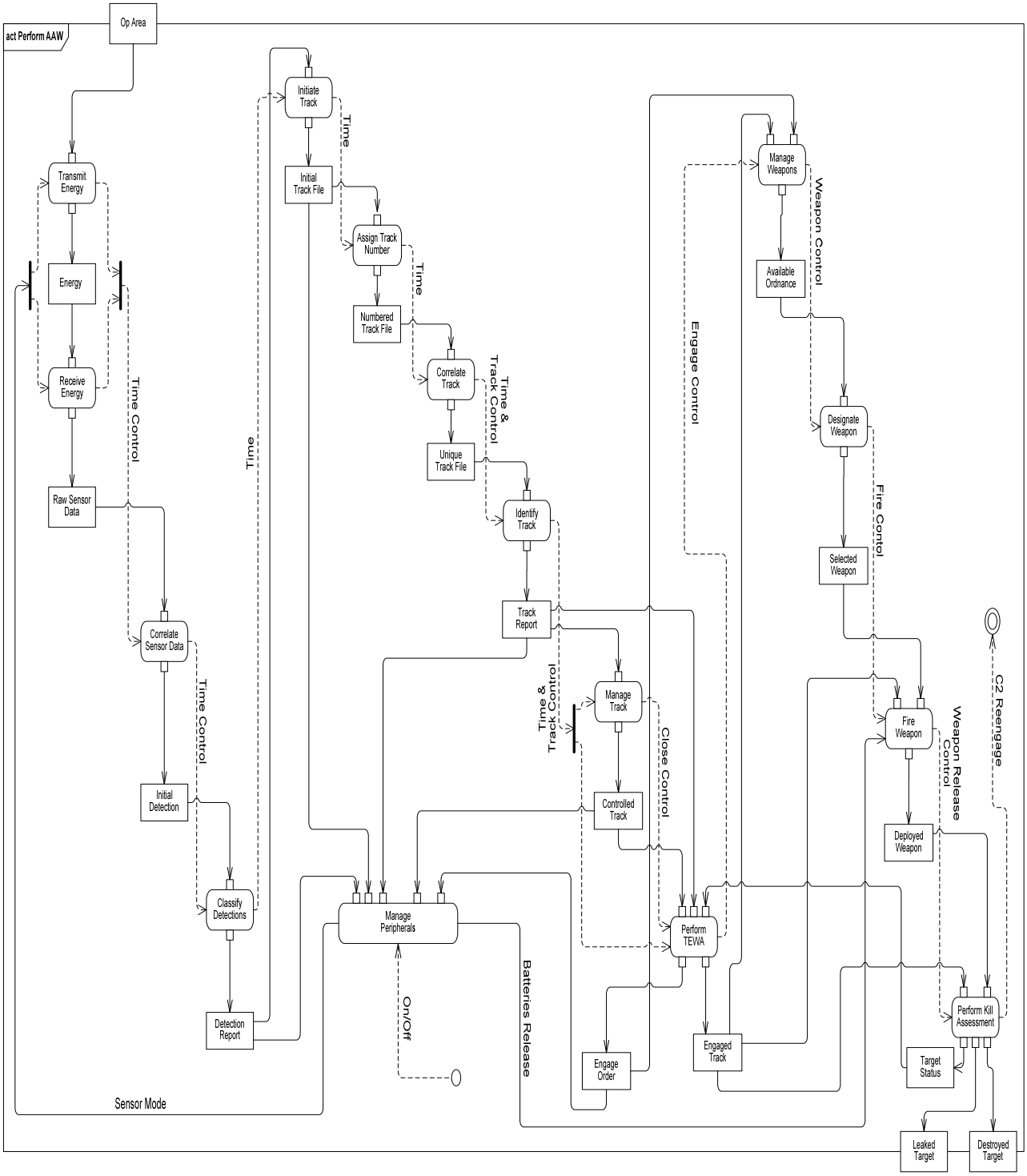


Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

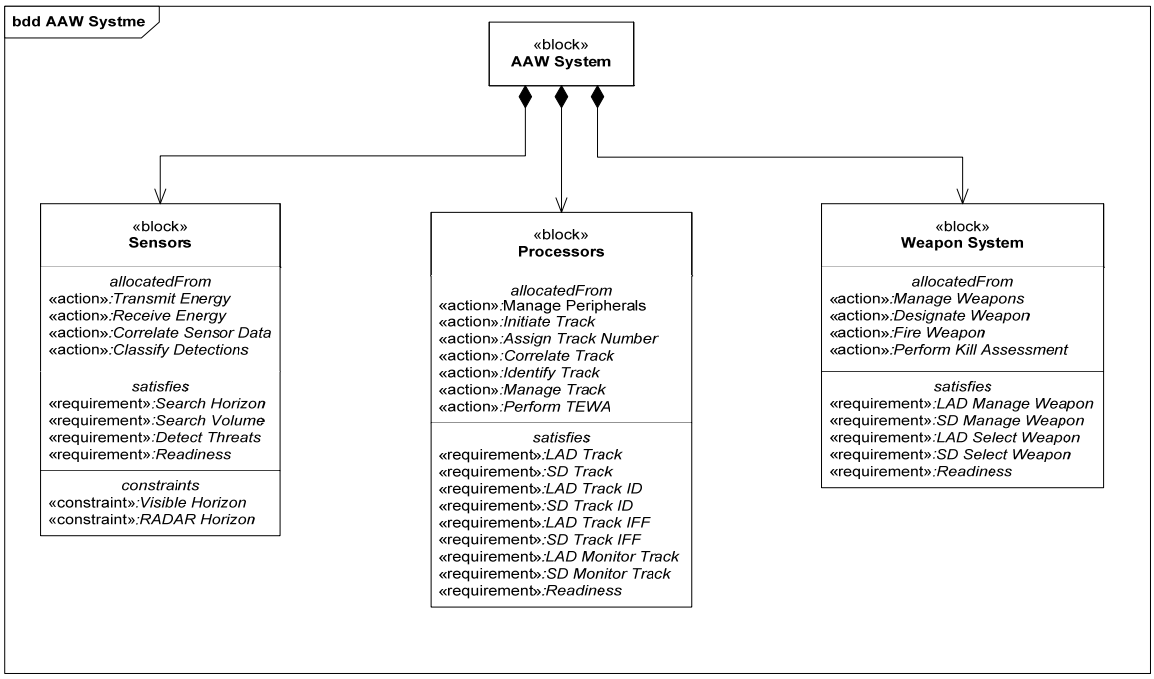
Perform Engage



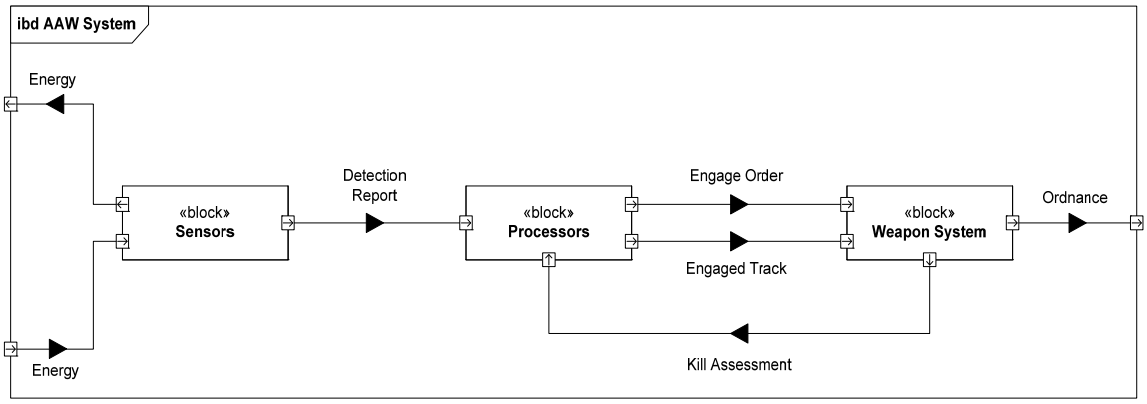
SysML Block Diagrams
Enhanced Functional Flow Block Diagram
Perform AAW



AAW Block Diagram



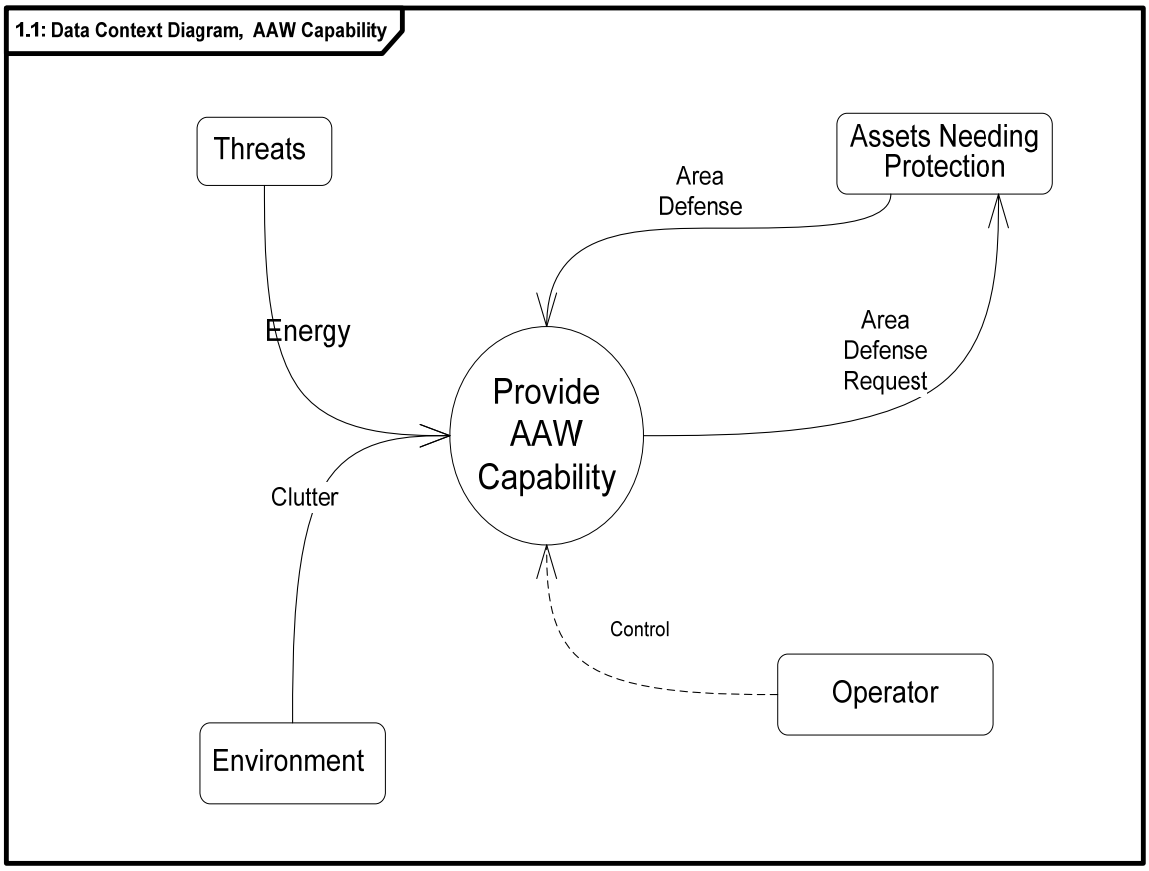
AAW Internal Block Diagram



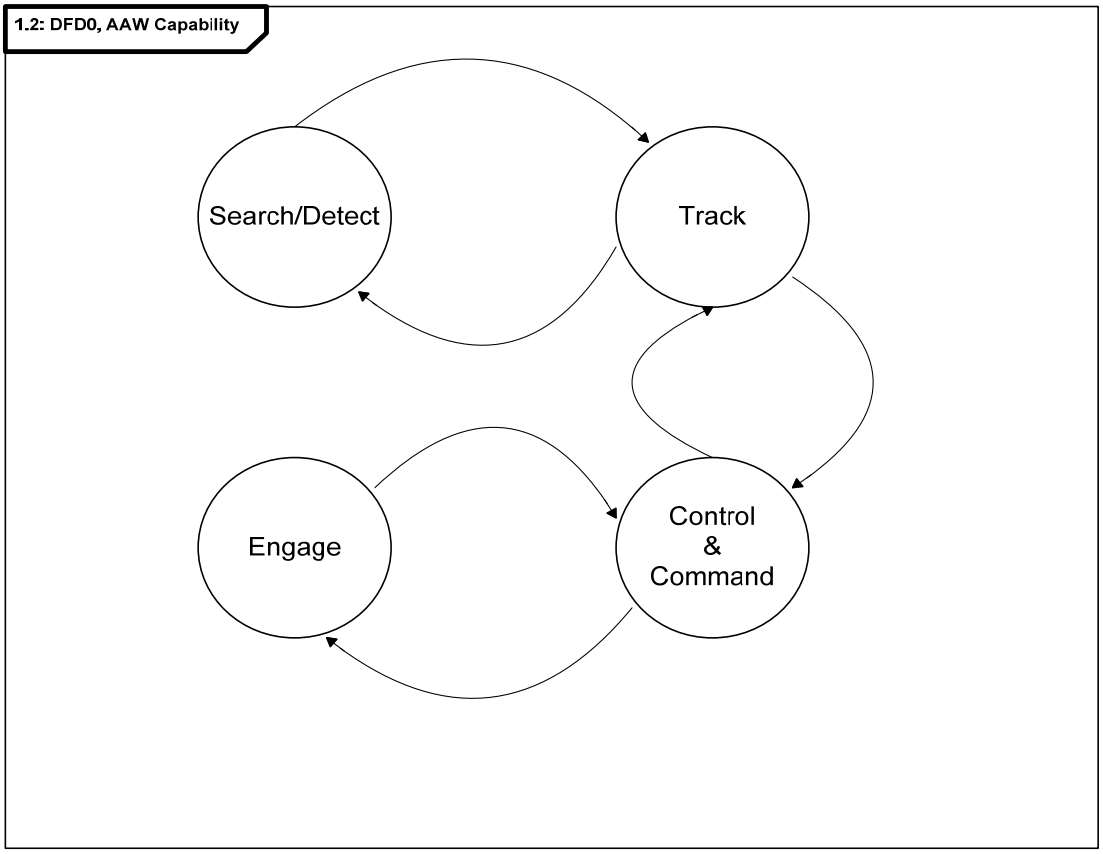
Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

SysML Data Context Diagrams (DCD)

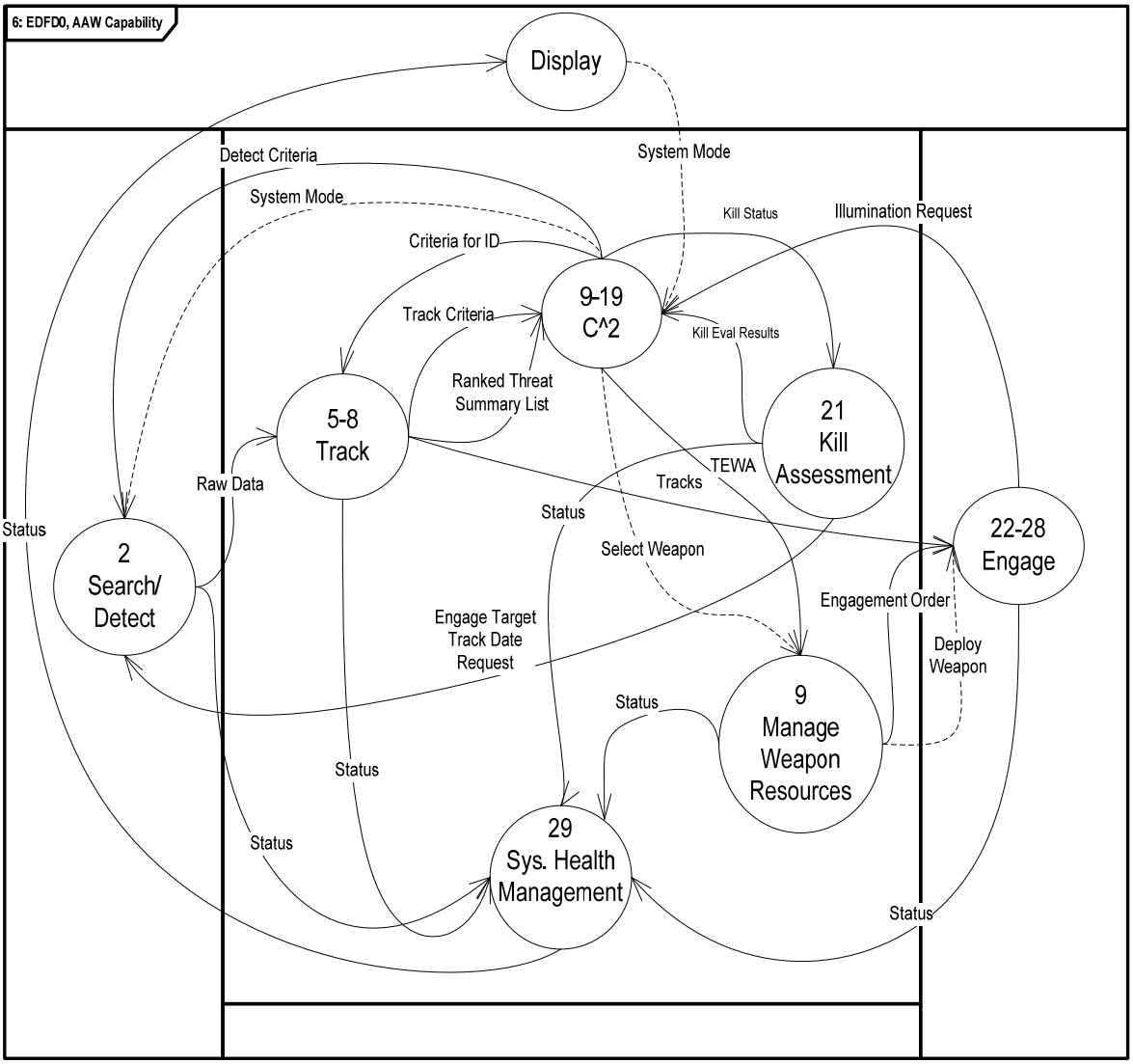
AAW Context Diagram



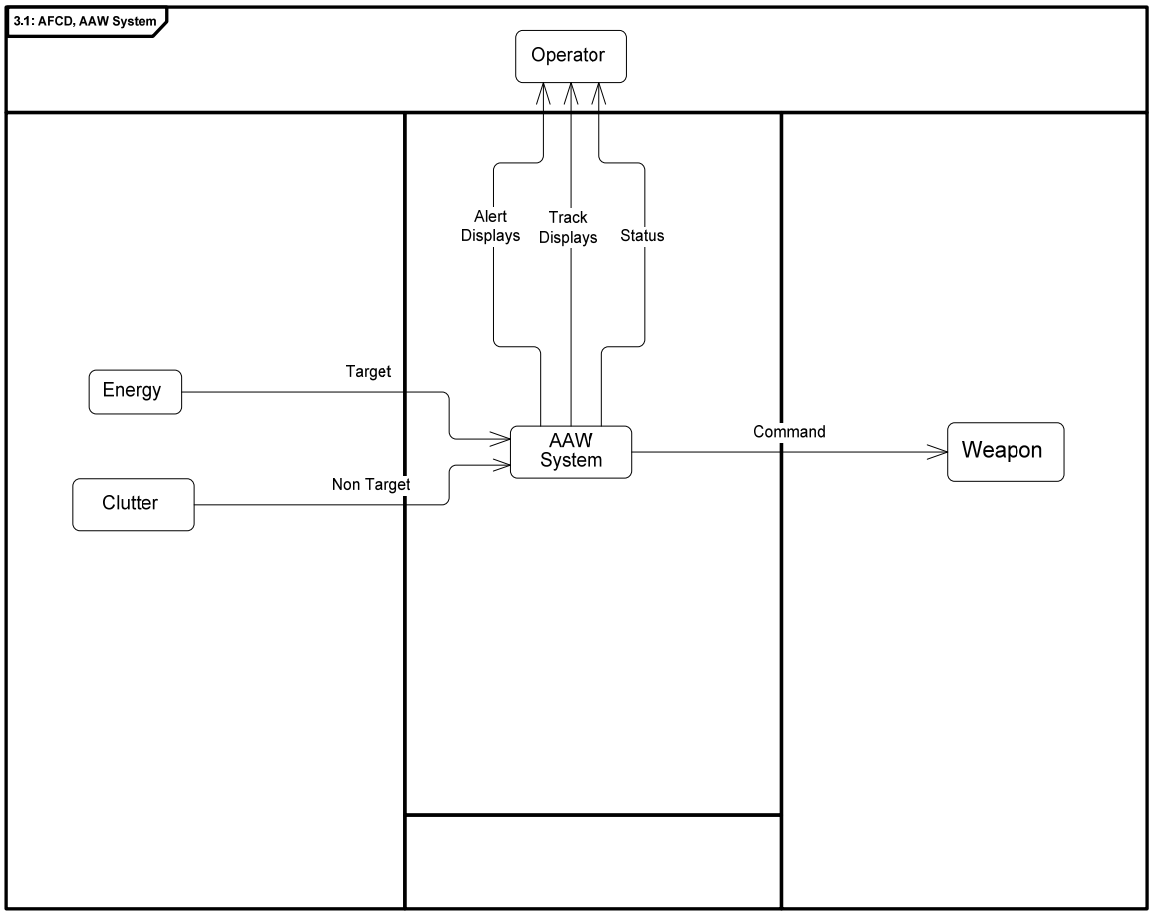
AAW Capability DFD0



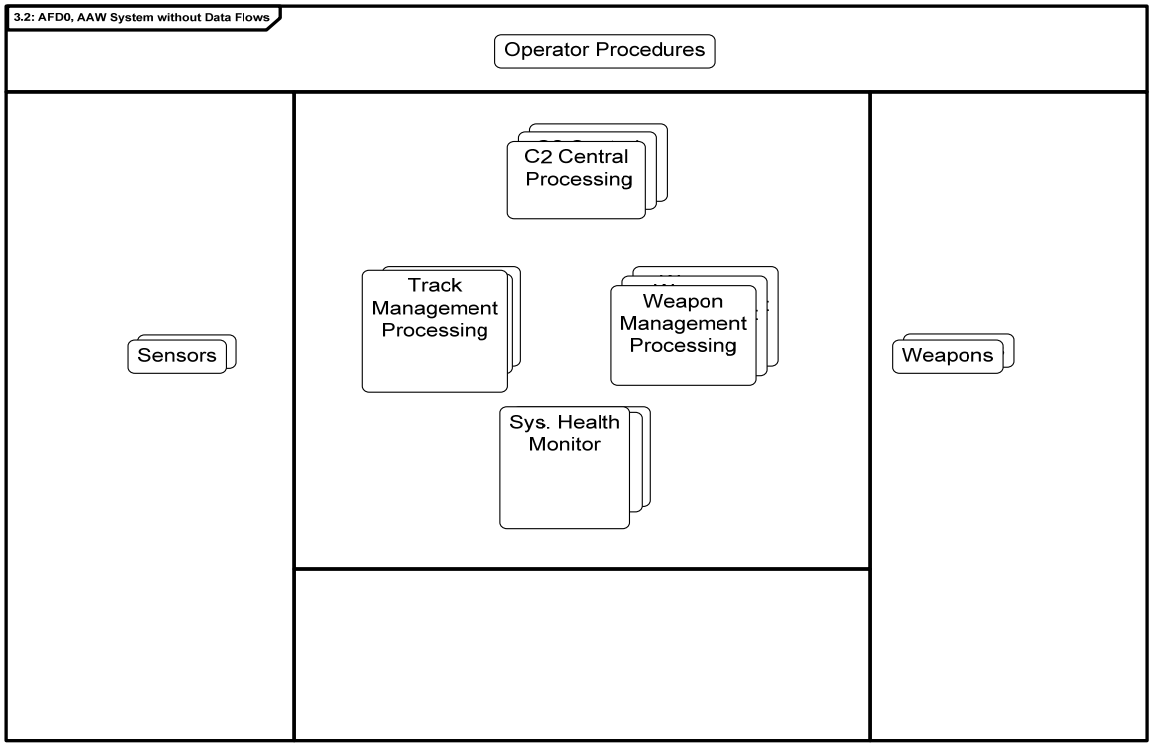
AAW Capability EDFD0



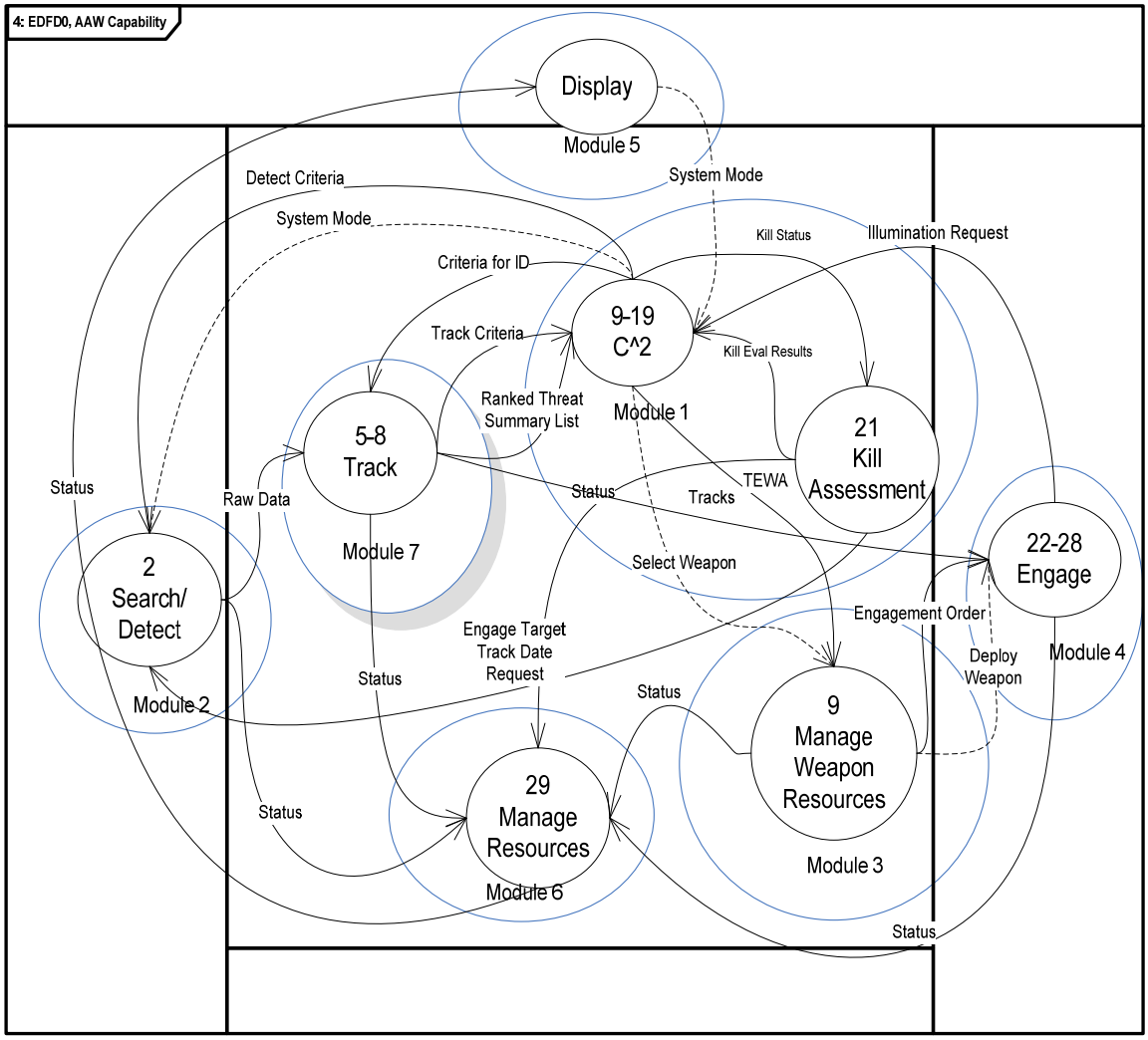
AAW System AFCD



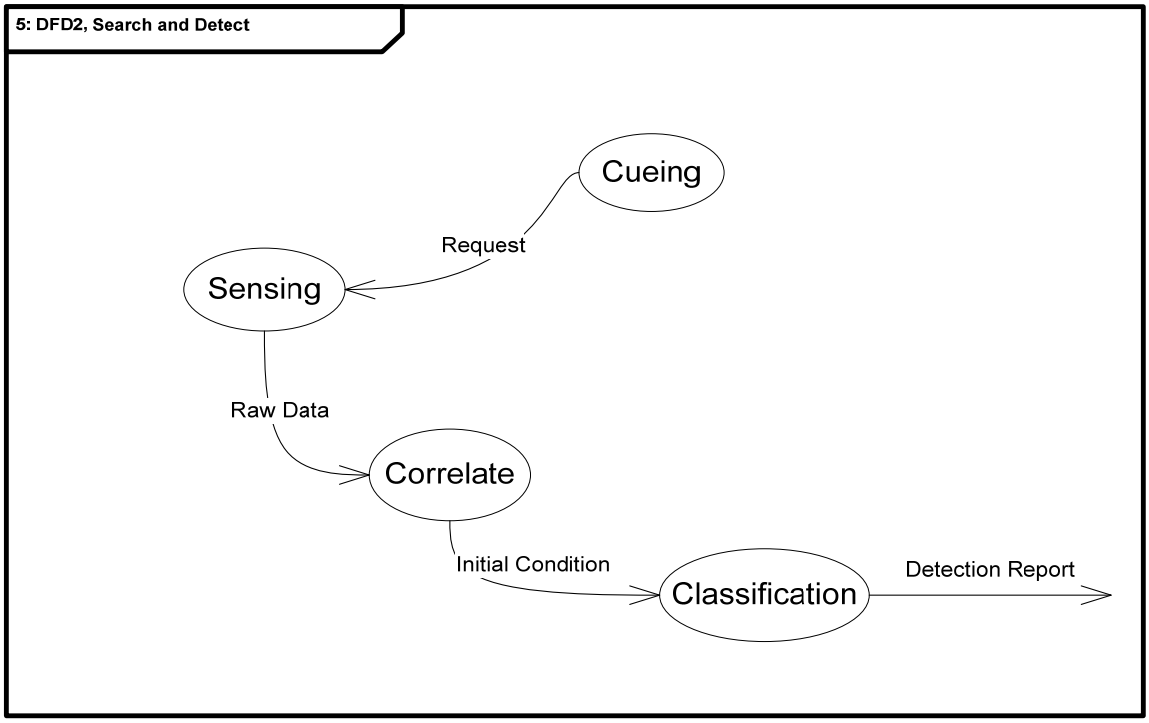
AAW System w/o Data Flows AFD0



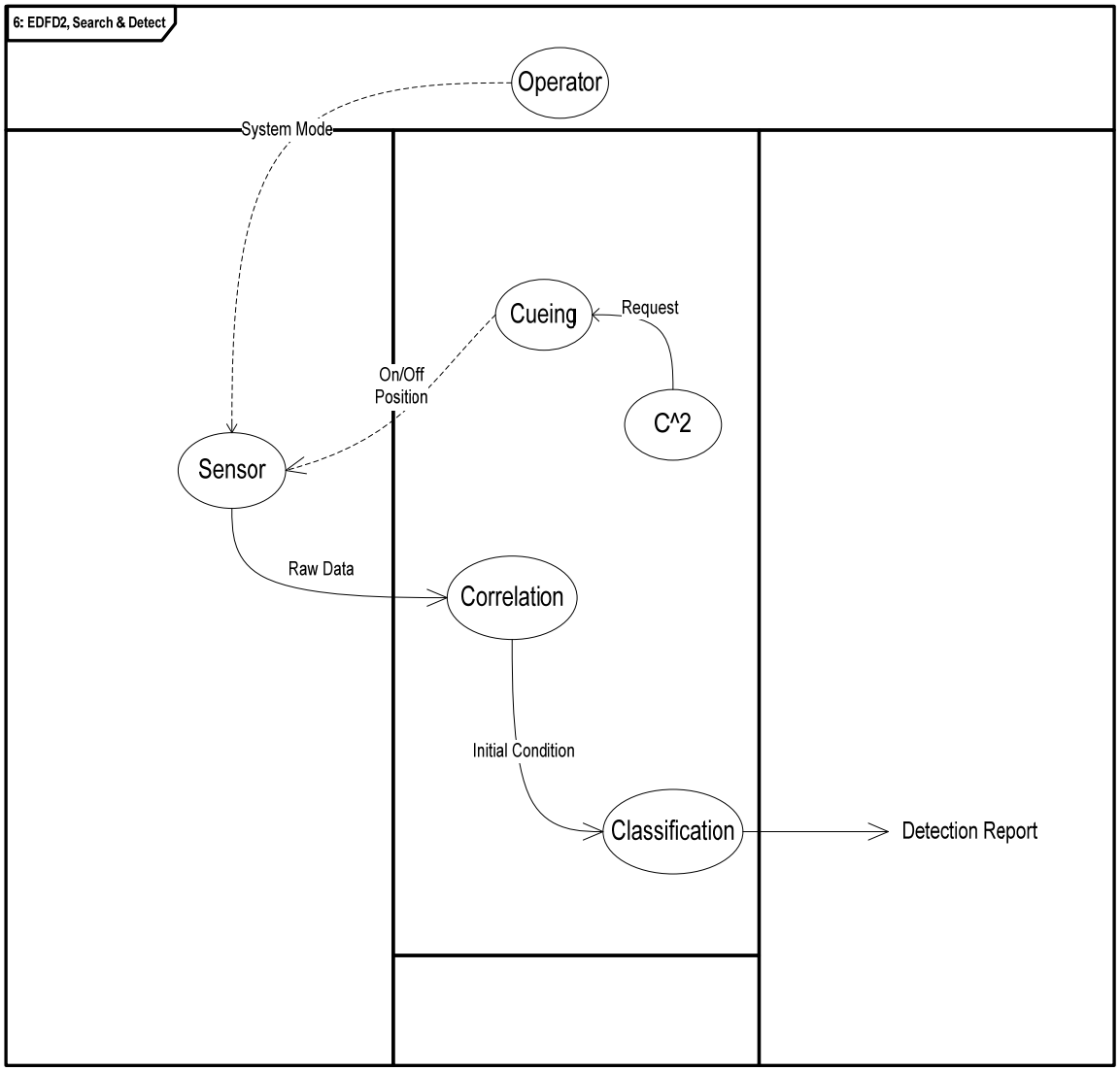
AAW Capability EDFD0



Search & Detect DFD2

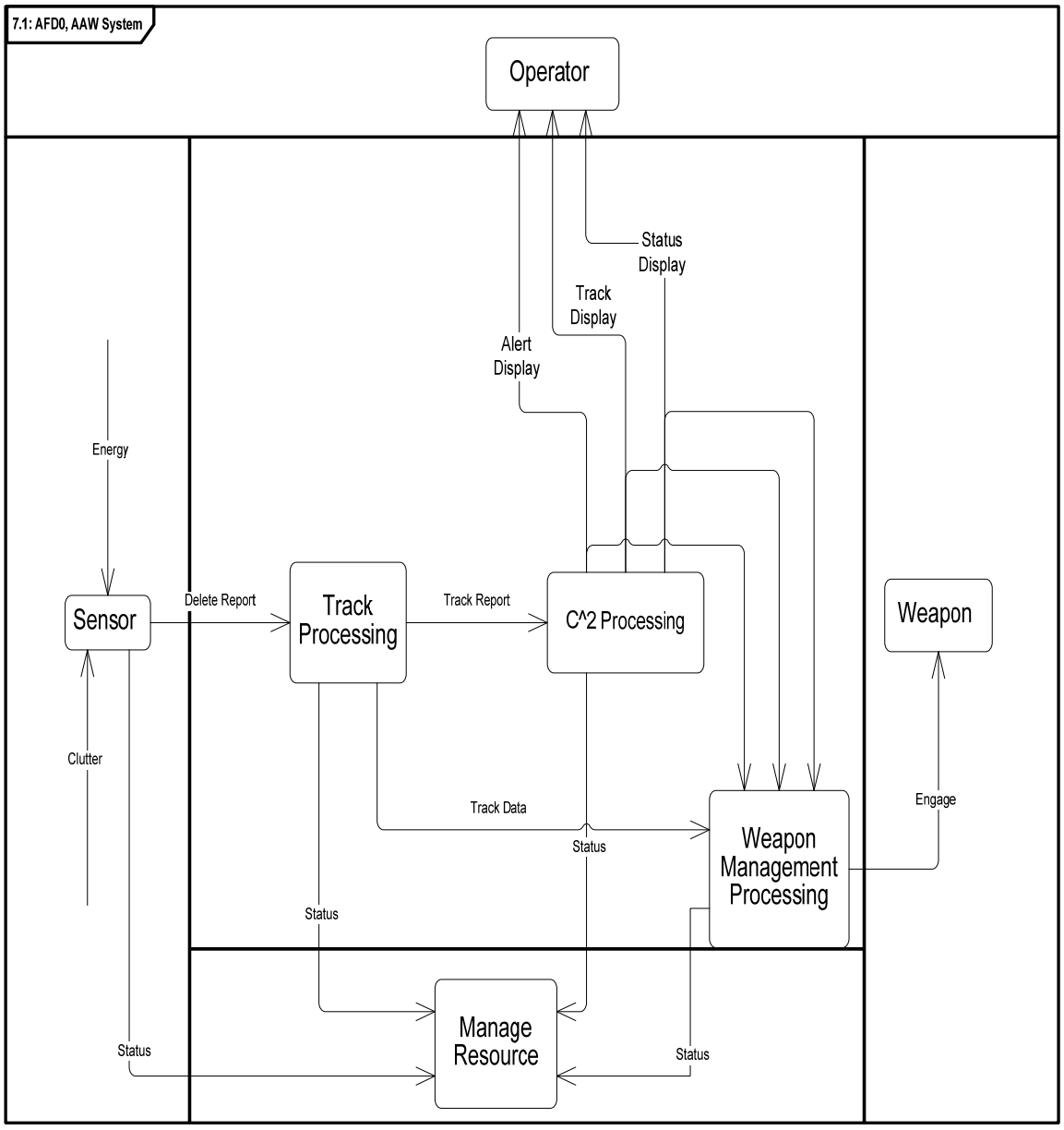


Search & Detect EDFD2

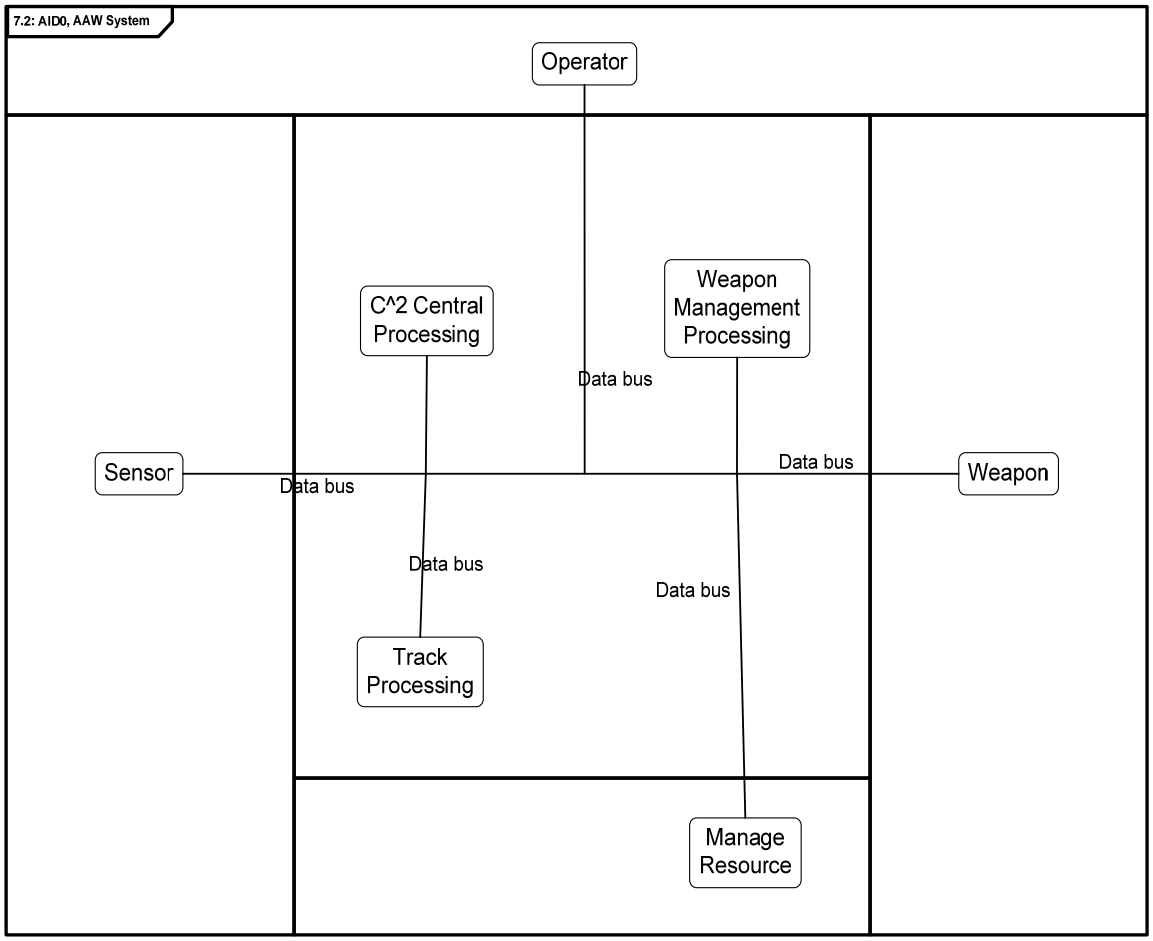


\

AAW System AFD0

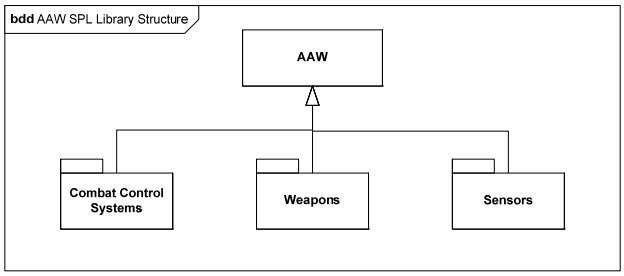


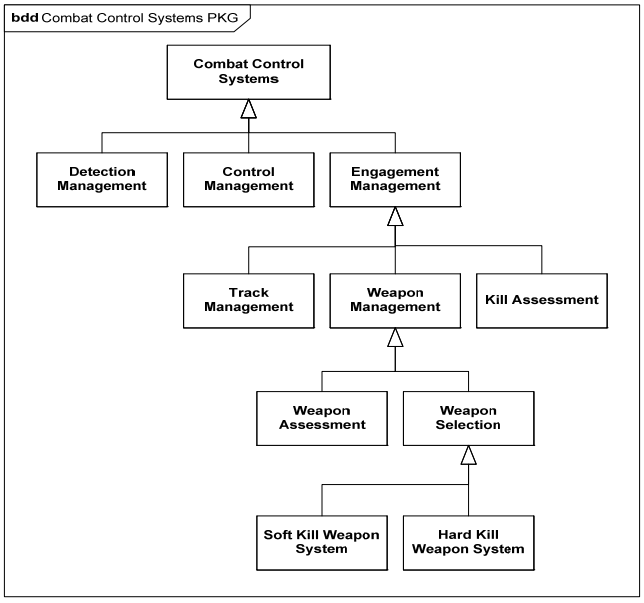
AAW System AID0



AAW SPL Library Structure

Combat Control Systems Package





APPENDIX I: AoA OF SOFTWARE ARCHITECTURE

1 Executive Summary

This paper summarized the methodologies applied to the analysis of alternatives on these two distinctive styles of software architecture: event-base implicit invocation and layered architecture. It detailed the functional aspect of the two architecture styles. It explained the advantages and disadvantage of each style as related to the quality of the architectures. The qualities of the architecture, also known as attributes, facilitate in analysis of selection of the layered architecture style.

2 Introduction

An architectural style encapsulates important decisions about the architectural elements and emphasizes important constraints on the elements and their relationships. (Fielding) This definition allows for styles that focus only on the connectors of an architecture, or on specific aspects of the component interfaces. In practice, software architectures are commonly treated as a collection of components and connectors. Components are the system's functional elements. (Fielding) Connectors are the protocols for communication between components.

The choice of particular software architecture is made on the basis of an overall system organization. This is to say that there is no single-fit, perfect architecture. Two metrics important for consideration in defining the publicly exposed interfaces of architecture's components and connectors are a system's cohesion and coupling. Cohesion is a measure of the degree to which a component has a singular purpose.(Edwards) The greater cohesion a component exhibits, the more focused is the component and the fewer are the assumptions about contexts for reuse. Coupling is the degree of interdependence between components. The less a component relies on other components (the looser its coupling), the more independent and reusable it is. Maximized cohesion (simple components) and minimized coupling (fewer connectors) are hallmarks of a flexible, maintainable architecture. (Edwards)

From extensive literature researched, we choose to evaluate these two architecture styles: event-based implicit invocation and layered architecture in the analysis of alternatives for the AAW based architecture.

The methodology employed in the analysis of alternatives includes the following steep:

1) Research the common implementations of software architectures styles.

2) Selection of two software architecture styles best suited for the AAW system.
3) Conduct trade-off analysis of the two alternatives.
4) List and define the quality attributes associated with software architecture.
5) Develop an objective hierarchy with all the quality attributes defined.
6) Apply weights to all the quality attributes, base on its overall system's requirements/KPPs.
7) Evaluate and apply score to each of the selected software architecture.
8) Select the software architecture with the highest score.

3 Alternative 1: Event-Based Implicit Invocation

Event-Base Implicit Invocation is one of the more broadly accepted architectural styles in software engineering. It is a component based architecture style built upon the concept of high cohesion and loose coupling. The idea behind implicit invocation is that instead of invoking a procedure directly, a component can broadcast one or more events. Other components in the system can register an interest in an event by a associating a procedure that has been registered for the event. Thus an event 'implicitly' causes the invocation of procedures in other modules. (Edwards)

Implicit invocation systems are driven by events. Events are triggered whenever that system needs to do something. Events can take many forms across different types of implantations. For object-based systems, an event is an object whose properties contain any contextual information needed to process the events. When an event is announced, the system looks up listener components for that event. Listeners fit the same criteria for components are functional modules of the system. Components that wish to act as listeners are registered to listen for certain events at configuration time. In the event of a triggered, all registered listener of that event are passed the event by means of a dynamically determined method call. In this way, function is implicitly invoked. This process of notifying listeners of an event is called event announcement. Events and listeners can also trigger other events.

One advantage of implicit invocation is its strong support of reuse. Component can be introduced into a system simply by registering it for the events of that system. Another is its eases in system evolution. (Garlan, Shaw) Components may be replaced by other components without affecting the interfaces of other components in the system. Lastly, its support of standardized components makes it interoperable.

The primary disadvantage of implicit invocation is that components relinquish control over the computation performed by the system. When a component announces an event, it has no idea what other components will respond to it. (Edwards) This reduces its reliability. Components announcing events have no guarantee of getting a response. It cannot rely on the order in which they are invoked. Nor can it know when they are finished. This reduces its simplicity. The difficulty in reasoning about the behavior of a

component making the announcement independently of the component that registers for its events make it difficult to understand. Another problem concerns exchange of data. Sometimes data can be passed with the event. But in other situations event systems must rely on a shared repository for interaction. In these cases global performance and resource management can become a serious issue. Finally, reasoning about correctness can be problematic, since the meaning of a procedure that announces events will depend on the context of bindings in which it is invoked. (Garlan, Shaw)

4 Alternative 2: Layered Architecture

The layered view of architecture is one of the most commonly used views in software architecture. Layered systems have good properties of modifiability and portability. (Garlan, Shaw) Layering, like all architectural structures, reflects a division of the software into units. A layer is a collection of software that together provides a cohesive set of services that other software can utilize without knowing how those services are implemented. Each layer could be implemented using a different architecture style. (Garlan, Shaw)

A layered system architecture style is organized into hierarchy. Each layer provides service to the layer about it and serving as a client to the layer below. In some layered systems inner layers are hidden from all except the adjacent layer, except for certain functions carefully selected for export. The connectors are defined by the protocols that determine how the layers will interact. The lower layers tend to be focused on the computing platform. Lower layers tend to be built with knowledge of the computers, and communications links. These areas are independent of the particular application that runs on them. They do not require changed as the application layers are modified. From a higher layer perspective, the lower layers provide a virtual machine that may be distributed and provide facilities to handle communication issues. (Garlan, Shaw) Higher layers tend to be more independent of the hardware. This means they are not likely to change should there be a change to the computing platform or environment; they are only concerned only with details more native to the application.

Layered systems have several desirable properties. First, they support design based on increasing levels of abstraction. This allows implementers to partition a complex problem into a sequence of incremental steps. This reduces design period as the whole network is broken into several manageable components that can be easily understood. Second, they support enhancement. Because each layer interacts with at most the layers below and above, changes to the function of one layer affect at most two other layers. Third, layered systems reduce coupling across multiple layers by hiding the inner layers from all except the adjacent layer, thus improving system evolution and reusability. (Garlan, Shaw) Different implementations of the same layer can be used interchangeably, provided they support the same interfaces to their adjacent layers. This also provides opportunity for

testing thus increasing its reliability. Fourth, the confidence in the correctness of a layer system is more easily established by testing and an analyzing each layer in turn.

But layered systems also have disadvantages. Not all systems are easily structured in a layered fashion. And even if a system can logically be structured as layers, considerations of performance may require closer coupling between logically high-level functions and their lower-level implementations. (Garlan, Shaw) Additionally, it can be quite difficult to find the right levels of abstraction. This is particularly true for standardized layered models.

5 Comparison of Alternatives

The software architecture properties include all properties that derive from the selection and arrangement of components, connectors, and data within the system. They include functional properties as well as non-functional properties. Functional properties, such as relative ease of modifiability, reusability of components, testability, and portability are often referred to as quality attributes. Nonfunctional qualities attributes falls under the domain of software supportability. The goal of the architectural design is to create architecture with properties that from a superset of the system requirements and KPPs. The relative importance of the various architectural properties depends on the nature of the intended system. The objective hierarchy in Figure 1 quantified the quality attributes for the AAW architecture. Table 1 listed the respective quality attribute and definition.

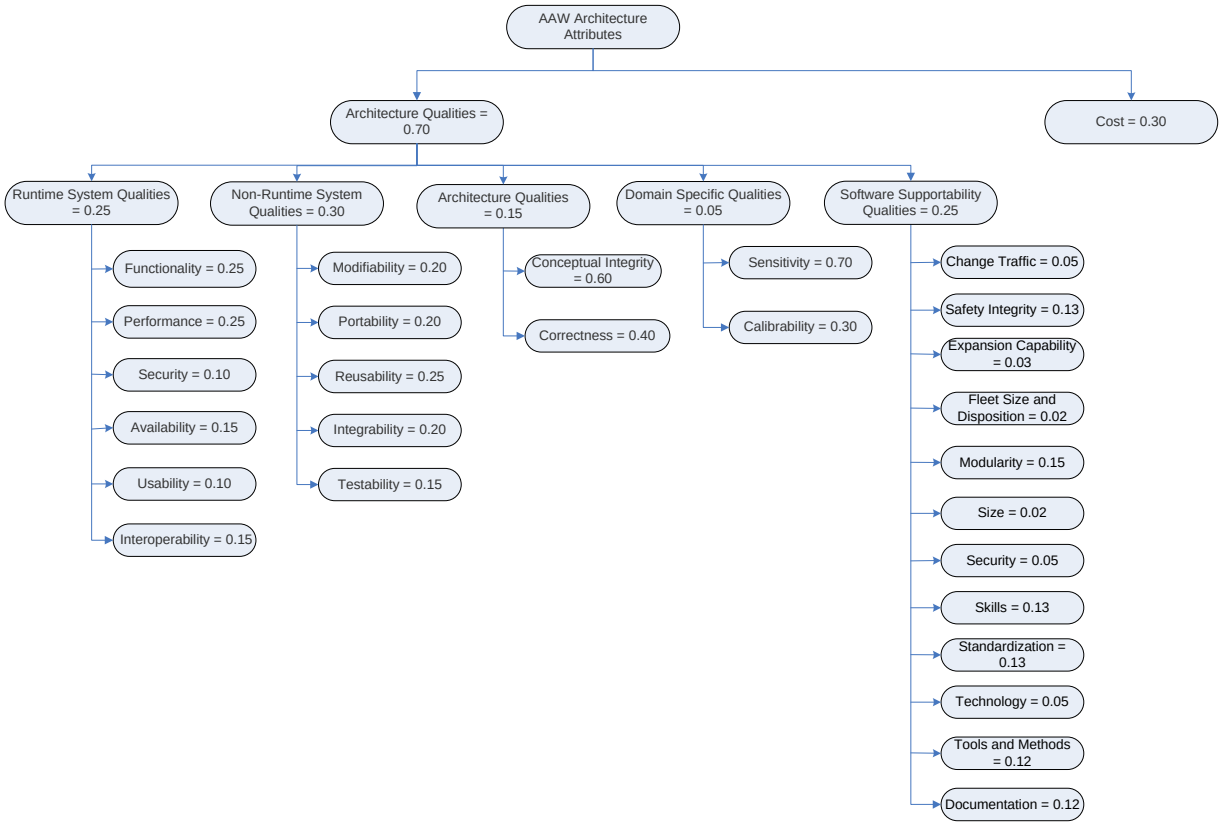


Figure 1: Objective Hierarchy

Quality Attributes	**Definition**
Functionality	The ability of the system to do the work for which it was intended.
Performance	The response time, utilization, and throughput behavior of the system.
Security	A measure of system's ability to resist unauthorized attempts at usage or behavior modification, while still providing service to legitimate users.
Availability	The measure of time that the system is up and running correctly; the length of time between failures and the length of time needed to resume operation after a failure.
Usability	The ease of use and of training the end users of the system.
Interoperability	The ability of two or more systems to cooperate at runtime.

Modifiability	The ease with which a software system can accommodate changes to its software.
Portability	The ability of a system to run under different computing environments. The environment types can be either hardware or software, but is usually a combination of the two.
Reusability	The degree to which existing applications can be reused in new applications.
Integrability	The ability to make the separately developed components of the system work correctly together.
Testability	The ease with which software can be made to demonstrate its faults
Conceptual Integrity	The integrity of the overall structure that is composed from a number of small architectural structures.
Correctness	Accountability for satisfying all requirements of the system.
Sensitivity	The degree to which a system component can pick up something being measured.
Calibrability	The ability of a system to recalibrate itself to some specific working range.
Change Traffic	Change traffic is a measure of the rate at which software modification is required. It is a complex function of requirements stability, software integrity and system operation. Change traffic will affect the volume of software support activity. Higher change traffic will require more software modification work. Change traffic may only be measured during actual use of the system. Before the software is in use, estimates may be made by comparison with similar applications and projections from requirements change and fault detection rate metrics taken during the software and system testing and trials. Any data available from comparable in-service systems on change traffic and effort will also be of significant value.
Safety Integrity	The safety integrity required of a software item will be determined by consideration of the safety criticality of the

	functions that it provides. Safety criticality relates to the likelihood of anomalies in the system causing accidents of varying severity. The overall safety criticality of a system should be established by the application of an appropriate hazard analysis technique. The criticality of particular software items will be consequent upon the partitioning of system functions in the system design. Designs should aim to minimize and isolate software, which implements highly critical functions. System requirements should define safety criticality categories and specify appropriate software safety integrity levels. Various constraints and requirements for software development, testing and modification will be associated with each safety integrity level.
Expansion Capability	Expansion capability is an attribute of system design. It is concerned with the degree to which software may be modified without being limited by constraints on computing resources. Associated physical limitations, such as space, are to be addressed in the context of the parent system. Examples of constraints on computing resources are: (a) Available memory. (b) Processor performance. (c) Mass storage capacity. (d) Input/Output bandwidth. Inadequate expansion capability might limit the scope for software modification or significantly impact on modification costs. Even simple changes might involve significant amounts of rework to overcome system limitations. Limited expansion capability is of particular relevance in the case of embedded, real-time applications. In such cases it is normal to state spare capacity requirements as part of any procurement specification.
Fleet Size and Disposition	The number of equipments in use (fleet size), and locations at which software support is conducted, will have an impact on software supportability requirements and support costs. Significant sub-groups of users might generate requirements for

	variations of the software to suit their specific needs. The number and distribution of equipments will influence the magnitude of the software support task and the optimum location of the software support facilities. Moreover, large fleets are more likely to accumulate higher levels of equipment usage, thereby increasing the probability of fault detection and the identification of corrective change requirements.
Modularity	Modularity is an attribute of the low-level structure of a software design, and relates to the extent to which processes and functions are represented as discrete design elements. The modularity exhibited by a particular design will be a function of the engineering practices applied by the developer, and factors determined by the choice of design method, tools and programming language. However, in general the optimum approach to modularity will be one that balances functional and performance requirements against the need to provide an understandable and supportable design. Poor modularity might result in increased modification costs owing to the need to implement consequential changes in other parts of the software. Requirements for interface control and standardization might be used to influence the modularity of a system design.
Size	A number of metrics are available to quantify software size. The size of a software item might influence its supportability, both in terms of the level of change traffic expected and the resources required to implement a change. The size of the software within a system is dependent upon the application and the design solution Software requirements should state any constraints on the size of run-time software imposed by the system design. Many software support and supportability projections will be based on estimates of software size and complexity. Software development requirements should specify requirements for data collection and analysis to measure software size, and to verify any models or estimates of supportability parameters that depend on software size.
Security	The security classification of data, executable code and

	documentation might impose constraints and demands on the software support activities and/or the Project Support Environment (PSE). The main influence on a prime equipment will be to impose special handling requirements. These might limit access to the software and introduce design requirements, which give rise to specific software support tasks and equipment. The security classification of a software item will be dependent upon the application and the equipment design. Wherever possible systems should be designed such that highly classified software is physically segregated from all other software within a system. System security requirements should provide criteria for security classification of software items and should specify modification and handling constraints associated with such classifications.
Skills	Software modification will require personnel with appropriate software engineering skills. Requirements for particular skills might be associated with the application domain, the technology or the methods used. Skill requirements will be determined by the system design, the software design and the chosen software support policy. Skill requirements will have an impact on personnel and training needs.
Standardization	Standardization may be applied to the computing environment within which the software executes, and to the technologies and engineering processes used to develop the software and the associated software documentation. Standardization will benefit software supportability by reducing the diversity of tools, skills and facilities required. The scope for standardization across a system might be constrained by the overall architectural design. Standardization requirements should be included within system and software requirements. Software standardization requirements might be less rigorously applied to software, which will only be supported by the original developer utilizing existing facilities, personnel and equipment. Software standardization requirements should avoid constraining the design to software

	technology, which has limited life expectancy or no clear evolutionary path.
Technology	Technology should be considered in respect of the software engineering methods and tools used in development and implementation together with the hardware and software aspects of both the host and target platforms. Technology issues might include: specification and software design methods and supporting tools; operating systems, programming languages and compilers; software test methods and environments; project specific tools and techniques; processing architectures. Requirements for the use of specific technologies might impose constraints on the system and software design solutions; they will also affect software engineering productivity and integrity.
Tools and Methods	The selection of tools and methods is dependent on the technologies used to develop and implement the system. The use of particular tools or methods might influence the software productivity and integrity achieved during software modification. The cost of acquiring and supporting tools should be carefully considered, since it might influence the selection of the software support policy. Depending on the level of standardization achieved, the same tools and facilities might be used to support software items from one or more systems. Selection of the tools and methods to be used during software development is a design decision and will form part of the design solution. The selected toolset will normally be incorporated in a PSE, which would also provide, depending on the chosen support policy, the basis for the post-delivery support environment for the software. The tools within a PSE will require support throughout the life of the prime system to which they relate, since the tools themselves will experience change, upgrade and obsolescence. System developers should have a strategy for considering these issues in their initial toolset selection and for on-going management of overall toolset effectiveness and integrity during the system life cycle. Aspects which should be considered in respect of each potential tool supplier include the following:

	(a) Commercial viability and track record. (b) Quality of customer service arrangements. (c) Product upgrade policy, particularly in respect of the maintenance of functional compatibility between succeeding software versions and the continued provision of support for preceding versions.
Documentation	The term documentation refers to all records, electronic or hardcopy, that relate to the requirements, specification, analysis, design, implementation, testing and operation of a software item. In order to ensure software supportability the documentation must be produced to an agreed standard and it must be available to the organization charged with delivering software support. Any software tools used in the creation of documentation must be included in the support facility and arrangements must be defined for their through-life support.

Table 1: Quality Attributes

Table 2 summarized the evaluations of the quality attributes to the selected architectures under consideration. The evaluation criteria are based on the advantages and disadvantages of each architecture style listed in the previous two sections. The pros and cons of each style are assessed by how they pertain to the quality attributes. The architecture style exhibiting a certain quality attributes are given a (+), otherwise it is given a (-).

Quality Attributes	Event-Based Implicit Invocation	Layered Architecture
Architecture Qualities		
Runtime System Qualities		
Functionality	+	+
Performance	+	-
Security	+	+
Availability	+	+

Usability	+	+
Interoperability	+	+
Non-Runtime Qualities		
Modifiability	+	+
Portability	+	+
Reusability	+	+
Integrability	-	+
Testability	-	+
Architecture Qualities		
Conceptual Integrity	+	-
Correctness	-	+
Domain Specific Qualities		
Sensitivity	+	+
Calibrability	+	+
Software Supportability Qualities		
Change Traffic	+	+
Safety Integrity	-	+
Expansion Capability	+	+
Fleet Size and Disposition	+	+
Modularity	+	+
Size	-	-
Security	+	+

Skills	+	+
Standardization	+	+
Technology	+	+
Tools and Methods	+	+
Documentation	-	+
Cost	-	-

Table 2: Comparisons chart of architecture style quality attributes

Table 3 applies the weight factors from the objective hierarchy to the respective quality attributes associated with the architecture styles.

Quality Attributes	Event-Based Implicit Invocation	Layered Architecture
Architecture Qualities		
Runtime System Qualities		
Functionality	0.25	0.25
Performance	0.25	-
Security	0.10	0.10
Availability	0.15	0.15
Usability	0.10	0.10
Interoperability	0.15	0.15
Non-Runtime Qualities		
Modifiability	0.20	0.20
Portability	0.20	0.20
Reusability	0.25	0.25
Integrability	-	0.20

Testability	-	0.15
Architecture Qualities		
Conceptual Integrity	0.60	-
Correctness	-	0.40
Domain Specific Qualities		
Sensitivity	0.70	0.70
Calibrability	0.30	0.30
Software Supportability Qualities		
Change Traffic	0.05	0.05
Safety Integrity	-	0.13
Expansion Capability	0.03	0.03
Fleet Size and Disposition	0.02	0.02
Modularity	0.15	0.15
Size	-	-
Security	0.05	0.05
Skills	0.13	0.13
Standardization	0.13	0.13
Technology	0.05	0.05
Tools and Methods	0.12	0.12
Documentation	-	0.12
Cost	-	-

Table 3: Comparisons chart of architecture style quality attributes with weigh factors

Table 4 sum up the weight factors of each category of quality attributes. The lower level attributes are further factored by the category weigh factor which in turn produced the overall architecture quality score.

Quality Attributes	Event-Based Implicit Invocation	Layered Architecture
Architecture Qualities	0.7675(0.70)= 0.53725	0.8425(0.70)=0.58975
Runtime System Qualities	1.0(0.25)= 0.25	0.75(0.25)=0.1875
Non-Runtime Qualities	0.65(0.30)=0.195	1.0(0.30) = 0.30
Architecture Qualities	0.60(0.15)= 0.09	0.40(0.15)= 0.06
Domain Specific Qualities	1.0(0.05)=0.05	1.0(0.05)=0.05
Software Supportability Qualities	0.73 (0.25) = 0.1825	0.98(0.25) = 0.245
Cost	-	-

Table 4: Calculated weight factor of quality attributes

6 Recommendation

Based upon the result calculated in Table 4, one can surmise the layered architecture is the recommended software architecture style. The layered architecture had an overall score of 0.58975 vs. 0.53725. From the criteria set forth by the AAW architecture selection process, we came to the conclusion that the layered style software architecture has better quality attributes compare to the event-based implicit invocation software architecture.

7 Conclusion

The methodologies applied the trade off analysis of the given choose of alternative software architecture conclusively produced a superior choose for the given application.

The choice of particular software architecture is made on the basis of an overall system organization. This is to say that there is no single-fit, perfect architecture. However, the analysis of alternative from the eight steps methodologies demonstrate the criteria selection of this AAW architecture shown layered style software architecture is a superior choose.

References

1. Edwards, Benjamin. An Introduction to Implicit Invocation Architectures. www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf (accessed November 25, 2009).

2. Fielding, Roy Thomas. 2000. Architectural Styles and the Design of Network-based Software Architectures. Doctoral Dissertation, University of California, Irvine.

3. Garland, David, and Mary Shaw. 1994. An Introduction to Software Architecture, Carnegie Mello University.

4. Kazman, Rick, and Mark Klein. 2000. Attribute-Based Architectural Styles. News@SEI Interactive.

5. Ministry of Defense Standard 00-60(Part 3). 24 September 2004. Integrated Logistic Support Part 3: Guidance for Application of Software Support.

6. 2001. SoftwareArchitectures.com, Quality Attributes, http://www.softwarearchitectures.com/go/Discipline/DesigningArchitecture/QualityAttributes/tabid/64/Default.aspx (accessed November 31, 2008).

APPENDIX J: SOFTWARE IPT REPORT

1 Introduction

As Software becomes an important integrated part of the Navy's Combat System, the size and complexity of software has increase exponentially. The traditional way of designing and developing combat system software tailored to specific platform is no longer a viable solution. It's a costly proposition in term of development, and software logistical support throughout its life cycle. Here is an example of the Window Operating system SLOC from Wikipedia that software line of code has increase exponentially over the years. [http://en.wikipedia.org/wiki/Source_lines_of_code (accessed January 2009].

Year	Operating System	SLOC (Million)
1993	Windows NT 3.1	4-5[1]
1994	Windows NT 3.5	7-8[1]
1996	Windows NT 4.0	11-12[1]
2000	Windows 2000	more than 29[1]
2001	Windows XP	40[1]
2003	Windows Server 2003	50[1]

Currently, general software suites are placed into a repository in hope of opportunity for reuse in future projects. The method of software reuse is cumbersome and costly due to interoperability issues. Software Product Line (SPL) is a method and technique for creating a collection of similar software system from a shared set of assets using a common means of production. Commonalities within software are used to create common product baselines within a product family. The utilization of these commonalities and efficient use of a product line approach can:

- reduce time to create and deploy a new product
- reduce the number of defects per product
- reduce the overall engineering cost per product
- increase total number of products deployed and managed (Krueger)
- reduce life cycle supportability cost

The main tenet of SPL is the selection of a software architecture most suited for SPL implementation. The inherit modularity of the layer architecture is make it a coherent choice.

This paper analyzes the implementations, styles of software architectures and examines which is the most appropriate within DoD framework for software reuse. It also examined cases studies of industry successful applications of SPL. From analysis of research materials, this paper proposed a SPL structure and behavior of the Navy's Software Hardware Asset Reuse Enterprise (SHARE) library. Lastly, the paper made recommendation of integrating supportability requirements throughout the development phases of the SPL product line.

2 Industry Implementation

2.1 Software Architectures

Several software architectural styles exist and are used extensively in industry depending on how designers wish to use computational components and the desired interactions between them. These components and connectors, along with their associated constraints, represent a software architectural style that defines a family of such systems in terms of patterns of structural organization.

Among the architectural styles widely-used in industry and further examined in this paper are:

- Pipes and Filters
- Data Abstraction and Object-Oriented Organization
- Event-based, Implicit Invocation
- Layered Systems

2.1.1 Pipes and Filters

Pipe and filter architectures use components (or "filters") to read streams of data on its inputs, incrementally manipulate the data as needed, and produce streams of data on its outputs. The connectors (or "pipes") simply relay the outputs of one filter to the inputs of another.

There are several advantages for implementing a pipe and filter architecture:

1. The overall behavior of the system is made up of the composition, behavior, and order of the individual filters.
2. Software reuse is supported since any two filters can be connected (provided that the data input into the filter is understood).

3. Software modularity is inherent in the system. Existing filters can be improved or new filters can be added to the system to improve functionality and efficiency.
4. Specialized analysis, such as throughput analysis, is supported.
5. Concurrent execution is supported so that data can be executed in parallel.

Batch processing can be seen as the greatest disadvantage of these types of systems. Each filter is independent and completely transforms input data into output data. This inhibits efficient handling of interactive applications where constant updates are needed in a timely manner. Because individual filters are usually connected in sequence, an update may need to run through the entire gamut of filters.

2.1.2 Data Abstraction and Object-Oriented Organization

This architectural style uses encapsulation to hide data representations and their associated operations into components (abstract data types or objects). Objects interact through function and procedure invocations; the objects themselves must preserve the integrity of its data representation while hiding its representation from other objects.

Object-oriented systems allow an object to hide its representation from its clients, making it possible to change without affecting those clients. The inherent modularity also allows for easy debugging by decomposing issues into collections of interacting agents.

The most significant disadvantage of this architecture is that in order for one object to interact with another, it must know the identity of that other object. Object changes, therefore, may necessitate that all other objects that explicitly invoke it be changed as well.

2.1.3 Event-based, Implicit Invocation

In an implicit invocation system, instead of invoking a procedure directly, a component broadcasts one or more events. Other components in the system listen for all broadcasts, and when one is of interest, a component can associate a procedure with the event. When the event is broadcast, the system itself invokes all procedures that have been registered for the event. The broadcast implicitly causes the invocation of procedures in other modules.

One major advantage of using this type of system is its modularity, which supports reuse and eases system evolution. New components can be added simply by registering it for its interested events. Existing components can be replaced without affecting the interfaces of any other component of the system.

The major disadvantage of implicit invocation is that the broadcasting components have no control as to how the system handles the associated data crunching. As a broadcaster, the component does not know if other components respond, does not know if procedures are followed in the proper order, and does not know if the procedures invoked are finished. If other components rely on data that has not yet been processed, timing will be a serious issue. This puts a lot of computational pressure on the system itself; overly complex systems with concurrent component broadcasts are at an extreme disadvantage.

2.1.4 Layered Systems Style

Layered systems implement a hierarchical approach, in which each layer provides or requests services from adjacent layers. Through predefined levels of extraction, the connectors are defined by the protocols that determine how the layers interact.

Several advantages come with using a layered architecture:

1. Increasing levels of abstraction support system design for complex problems.
2. Software modularity fosters ease of upgrades. Any changes in one module will affect, at most, only two layers.
3. Modularity also fosters software reuse, where new modules can be inserted as long as they support adjacent layers.

Not all systems can be structured in a layered organization, however. Although loose coupling is one of the major tenets of system software for complex systems, tighter coupling may be required when performance is the priority (Garlan and Shaw).

2.2 Existing Application of Software Product Lines

Software reuse has long been viewed by the software engineering community as the "silver bullet" for increasing software design efficiency, deploying software products faster, cheaper, and better. A number of software reuse approaches, such as object-oriented programming, have been presented over the years. Although such techniques have helped software engineers in producing highly modular and, to some extent, reusable code, they have not met the high goals of increased productivity, high quality software that were delivered on schedule and within budget. These techniques support "small-grained" reuse and have not provided the sought-after "silver bullet" (Eriksson, 26).

Software product line (SPL) development has emerged as a new and promising approach to large-grained, intra-organizational software reuse. Commonalities within software parts are used to create common product baselines within a product family. The exploitation of these commonalities and efficient use of a product line approach can:

- reduce time to create and deploy a new product
- reduce the number of defects per product
- reduce the overall engineering cost per product
- increase total number of products deployed and managed (Krueger)

Product lines are not an entirely new concept. The manufacturing industry has long used product lines to reduce costs and increase productivity by exploiting commonalities between products. The advantages of mass production and mass customization in manufacturing can be seen by applying the same techniques to software development. Several companies in industry have had great successes in their use of SPLs.

2.2.1 Medical Applications

The medical industry has seen advances with SPLs. Philips Healthcare produces imaging equipment that is used to support medical diagnosis and interventions with X-ray, Magnetic Resonance Imaging (MRI), Computer Tomography (CT), and Ultrasound. Possibilities for software reuse exist across the diverse set of product lines, specifically dealing with storing, retrieving, and exchanging medical images, and many product lines support image processing and viewing.

Philips implemented an across-the-board initiative to produce common software for all product lines in the imaging platform. "This success story describes the use of a central group producing a medical middleware platform, to be used by other product lines in the company. The platform itself is a product line in itself. Different product groups use different variants and platform configurations. Within many of the product groups, software running on top of the imaging platform is built in software product lines as well. This induces additional variability requirements to the platform" (van der Linden).

The product groups within Philips together fund the software development of the platform. In the end, they get platform software much cheaper than if they had developed it themselves. "Although the platform software is more generic than what they need, it is also of better quality, since it is tested within many distinct software product lines." Utilizing SPLs, Philips has also seen a significant decrease in problem reports from product groups after release of platform components (van der Linden).

2.2.2 Computer Peripheral Applications

Hewlett-Packard (HP) is the world's leading manufacturer of printers. Each printer is comprised of complex firmware that controls all aspects of the printer's operation. In its venture to implement SPLs, HP specifically seeks better-quality software components, delivered more quickly and at a lower research and development (R&D) cost.

Instead of setting up a distinct SPL team as Philips had done, HP organized the firmware teams from several products into what they called a "firmware cooperative." "Thus, the Owen cooperative is an autonomous collection of projects, voluntarily united to meet their common needs and aspirations. The expectation of each participating project is that by pooling resources in an appropriate way, projects will derive significantly greater value from the cooperative than the effort they contribute to it" (Toft).

HP has measured and enjoyed huge successes with the implementation of SPLs. HP's firmware cooperative reported, with a 70% increase in shared code, a 3X improvement in time-to-market, a 4X reduction in overall team size, and a 25X reduction in typical defect densities (Toft).

2.2.3 DoD Applications

In support of the US Navy's open architecture (OA) business model, the Software Hardware Asset Reuse Enterprise (SHARE) has taken significant steps in the direction of the SPL approach. SHARE is a software repository for government-owned systems. It was formed with the ultimate goal of leveling the playing field for companies to compete for government contracts.

Often times, contracts are dominated by certain companies because the expertise is non-existent within the rest of the industry. The idea is that, if companies have access to those existing systems, they can compete in contracts they couldn't before. CAPT James Shannon, in a February 2007 interview with Defense Daily, stated, "we want to widen the competitive base both in terms of cost and intellectual competition. The only way you can increase that competitive base is you make sure that any potential developer understands what is in (our) design and what we are looking for. So if we go out with an RFP (Request for Proposal) to do an improvement to a specific combat system, how is a company going to compete if they don't know what is already out there? What most companies have been telling us is they think they can help us, but they are not sure. If only they can have access to certain things and understand the systems, they will at least be in a better position, so that when we go out with an RFP they will be able to compete in a fair and open way. That was one of the drivers to developing this (repository)."

Unlike private businesses in industry, the Department of Defense (DoD) does not own all software implemented in combat or weapon systems. With DoD's OA approach, open source software may be used but not included in the repository due to open source licensing agreements. This ownership issue represents a significant hurdle the DoD must overcome to implement true SPLs that others within private industry may not, since they own all associated software.

Perhaps the next step in full SPL deployment within the DoD is forming a distinct SPL team, as Philips Healthcare had previously done, to categorize and understand the commonalities between systems and their behaviors. Only then can the DoD even begin to realize the benefits of implementing SPLs for its combat and weapon systems.

3 Software Architecture

The layered architecture style was chosen for the Software Architecture via extensive literature research and studies of various existing architecture styles that are in use today. The initial process was to conduct an analysis of the various types and determine which best fit the problem statement. The second process was to narrow the styles down to two best styles and perform an alternative analysis that assigned quality attributes and weight to determine the best style. The detail analysis of the process is outlined in the Alternative of Analysis (AoA) of Software Architecture Paper. When followed, the process of picking out an architecture style, the next step was to construct a structure that best suited the need. The layered architecture was chosen to satisfy the requirements that drawn from the AAW IPT requirements.

3.1 Layered Software Architecture

A layered system is organized hierarchically. Each layer performs an assigned function to prepare data for adjacent layers. In the proposed AAW software architecture, the layers are logically grouped into three data handling groups: data display group; data processing group; and data collection & execution group. The three groups broke down further into layers by their function in the software system. Figure 1 shows layers in each group.

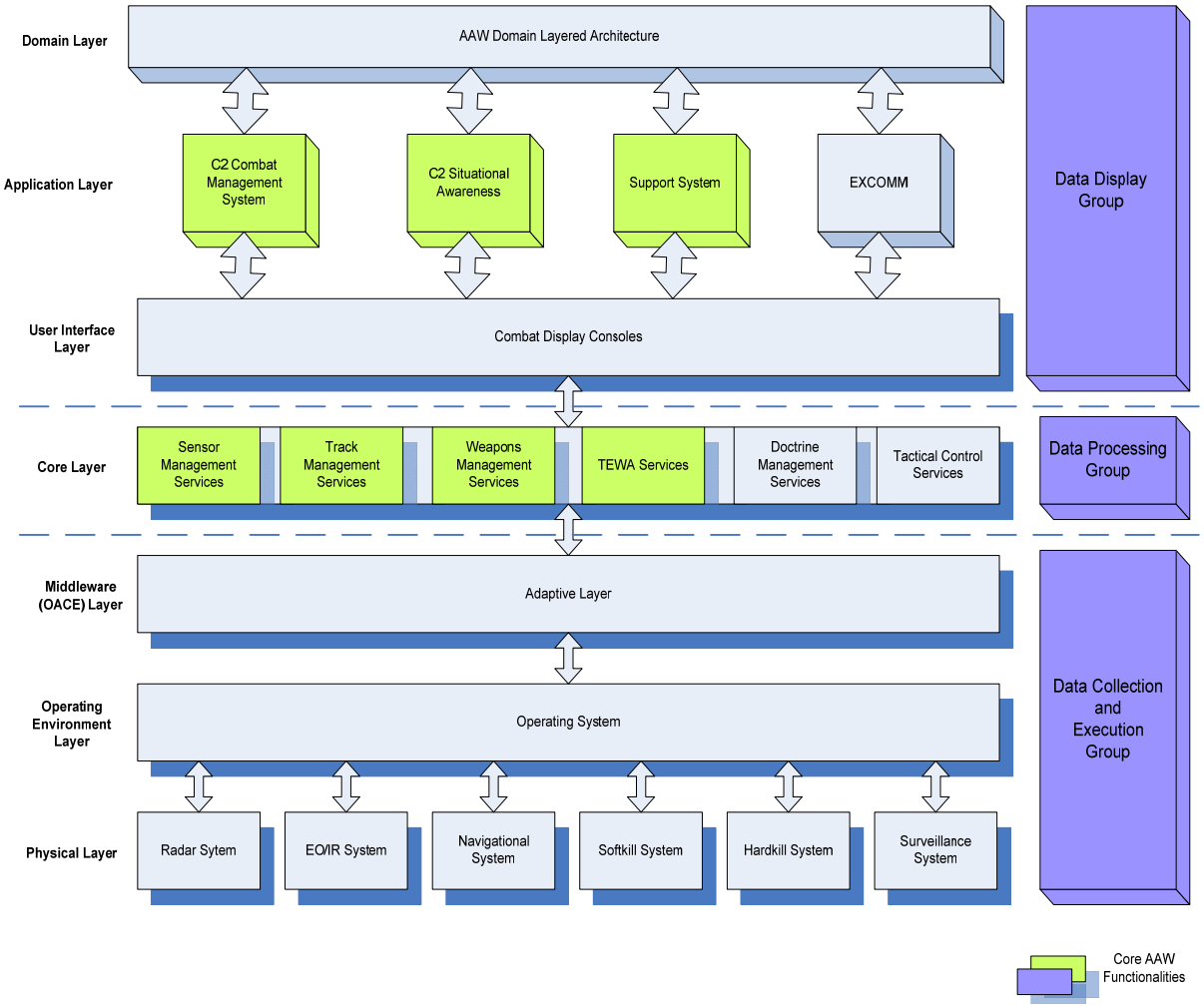


Figure 1: AAW Software Layered Architecture

Data collection & execution group consists of the low level function layers comprised of the physical layer, operating environment layer, and middleware layer. A variety of differing data formats can exist in these low level layers because various hardware and software components can be made available to support the required functions. The hardware and software exists in the physical layer (hardware) and operating environment layer (software) respectively.

Middleware is defined as, "a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems." - [David E. Bakken. 2003. Middleware. Washington State University: Kluwer Academic Press]. The main function of middleware in an OA environment is to isolate computing technology changes. In the proposed AAW software architecture, the middleware layer serves two functions as a gatekeeper to the lower level layers. One is to prep collected data from the physical and operating environment layer for higher-level use because the high level layers support higher functions. For an example, data correlation cannot process data in different

formats from various hardware and software in order to perform their functions correctly and efficiently within the restricted time requirement for AAW. The second function serves to convert execution data from higher level layers back to a format that the physical layer and operating environment layer can process.

Middleware is already recognized as a solution for OA. The following is a quote from an article of a company using middleware technology to upgrade US legacy systems.

> Standards-based interfaces are a key element of the U.S. Navy's long-term planning. "Using open architectures will allow the Navy to afford its future," noted William Johnson, director of Open Architecture, program executive office for Integrated Warfare Systems (PEO-IWS7). "New technology is available every 18 months. Using standards-based interfaces such as DDS allows us to easily introduce upgraded hardware in our operational systems without having to rewrite the application software."
>
> This example of Dot21's use of NDDS involves the U.S. Navy's SPS48 radar system. "We took an existing display based on proprietary Message-Oriented Middleware (MOM) and replaced the interface with NDDS. That will allow our display application to interoperate with other new systems that are DDS-compliant without the cost of maintaining proprietary software," said Bailey. "NDDS will also allow us to create remote interfaces that talk to legacy systems on one side and future mission systems that use DDS on the other side."

- Dot21 Adopts COT Middleware From Real-Time Innovations for U.S. Military Systems Upgrade, September 6, 2005 [http://www.rti.com/company/news/dot21.html]

The data processing group resides in the core layer. The core layer performs data processing and data management for all functions that the system requires. The core layer consists of several data processing/managing components for system's need. For AAW software architecture, there are sensor management services, track management services, weapons management services, and TEWA services. Each service will process and manage data that are assigned to it. For an example, when the system calls for the use of a track service for correlating track information, track management services in the core layer will process all available stored track data to produce correlated track information for the user and broadcast the data to C2 applications in the application layer through the user interface layer.

Data display group includes functions of displaying data, providing security, and sharing data with other domains. Data display group consists of user interface layer, application layer, and domain layer. The user interface layer manages user's access level to data and applications. Application layer allows users to access, display, and analyze data that exist in core layer. By the demand of how the user wants to use data in the system, the user can install applications in the application layer if such applications that meet the user's need exist. The proposed AAW system has three AAW applications, which are C2

Combat Management System, C2 Situational Awareness, and Support System. These AAW applications help operators to use collected data more effectively by displaying and analyzing data in a strategically effective way. The domain layer allows one domain to interface with other domains. Information from AAW domain can be used in SUW domain through the domain layer when the SUW domain requested information. Further detailed studies are left for the reader to investigate.

4 Software Product Line

What is a software product line? According to Carnegie Mellon University, "a software product line is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way." [http://www.sei.cmu.edu/productlines/]. However, there is more to SPL than grouping similar software that has common behavior and function. The process of identifying and determining the usefulness of the SPL is the key element in software architecture.

Thus, it is important to recognize that "the products that are planned to be part of the product line need to exhibit sufficient commonality and it should be possible to handle the variability in a modular fashion" [Design & Use of Software Architectures, Jan Bosch].

4.1 Proposed Software Product Line (SPL) Library Structure Repository

The optimal implementation of SPL required a complementary approach to model how SPL is achieved and retrieved. The following diagrams are examples of the AAW library structure repository. The library contains framework of software components (SPL), all their respective specifications and descriptions required for retrieval for future reuse.

The SPL library structure is outside the scope of what this study focused on. However, realizing that storing SPL materials in a proper manner is essential to take full advantage of software reuse required a cursory understanding of what a library structure could look like.

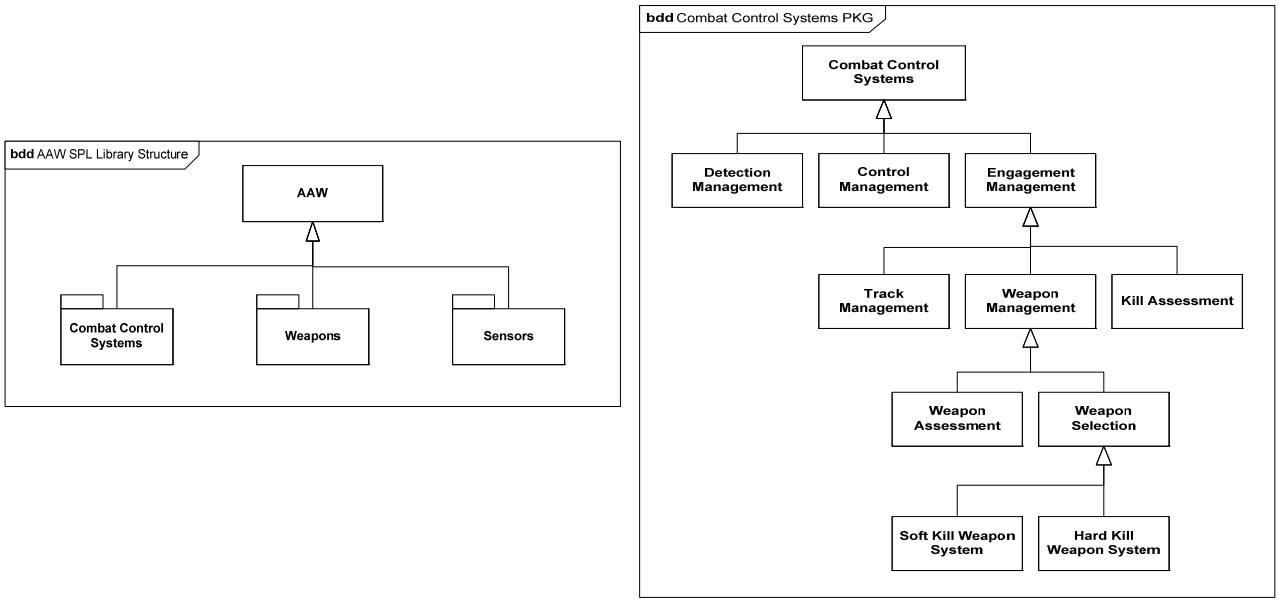


Figure 2: SPL Library Structure (Left: First Level Breakdown, Right: Example of Breakdown beyond the First Level

The SPL library structure from Figure 2 showed the inheritance relationship between AAW domain and the three packages, which composed of Combat control systems, Weapons and Sensor. The AAW library would contain all of the artifacts from the lower packages. When the library has similar structure to the system, users of the library can easily obtain information because they already know where to look in the library. The above figure also illustrated the break down of the Combat Control Systems package.

Package has a "is a" relationship in which they are all share similar attributes and method that they'd inherited from the AAW domain. This is extremely useful because it allows the library to use an existing class to help define new classes, making it easier to reuse software. In addition, the inheritance of the package showed that the information that are placed or stored in the individual package has the same construct and can be decomposed into the parent child relationship.

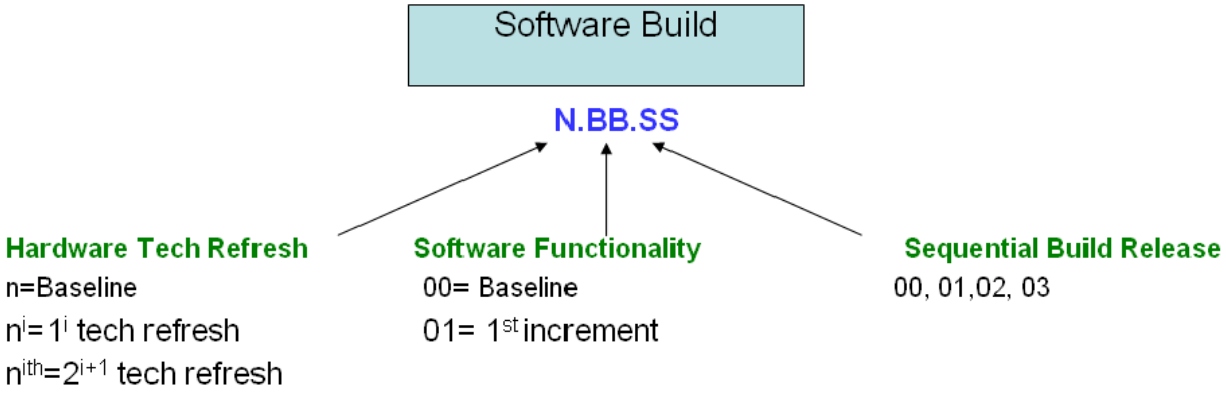


Figure 3: Software Suite Configurations

For example, the software musts be able to integrate products with existing combat systems using new software codes on existing hardware or vice versa. Figure 3 illustrates the ability of the software library component capability to employ the software build based upon the hardware tech refresh with the software functionality and the sequential build release. All of these three components will then package as a software build that will show the same common capabilities that will be able to work across all various ship platforms. Therefore, the result of the continual software update with hardware tech refresh will provide better performance, higher reliability and reduced maintenance costs.

4.2 Predictive reuse concepts through the use of SPL

Software product line development refers to software engineering methods, and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production. The characteristic that distinguishes software product lines is predictive versus opportunistic software reuse. Rather than put general software components into a repository in hopes that opportunities for reuse will arise, software product lines only call for software artifacts to be created when reuse is predicted in one or more products. (Krueger) In each phase of the development cycle, artifacts created follow approaches to systematically implement the architecture of an SPL. SPLs make a predictive analysis of such artifacts that may be reused and integrated for the development of products.

The approach to software predictive reuse artifacts can be summarized into these three abstraction techniques: selection; specialization; and integration. Selection refers to the ability of users to located, understand, and select the appropriate artifacts from a repository with concise abstractions. Specialization refers to a generalized reusable artifact corresponding to an abstraction specification. And lastly, integration refers to an artifact interface in which internal details of the artifacts are suppressed. (Kruegar)

4.3 Implementation of SPL in an OA environment

The first step of implementing SPL successfully in the system is to have the right software architecture for the SPL. The proposed software architecture for the AAW system is layered software architecture. Advantages of Layered Software Architecture include modularization meaning that a new layer or modified layer can be inserted without threatening the entire software system as long as the new or modified layer supports adjacent layers. This well-defined interface requirement of each layer gives the commonality to all software products that belong to the layer. The proposed AAW software architecture shows that all software products will be assigned to a layer by each software product's data handling and function as it is shown in figure 1. For example, in the core layers, there are four different AAW software products with different functions; and the commonality of those four AAW software products is to interface with the user interface layer and middleware layer. The interface requirement between layers is the common set of core asset, which is what the SPL is based upon.

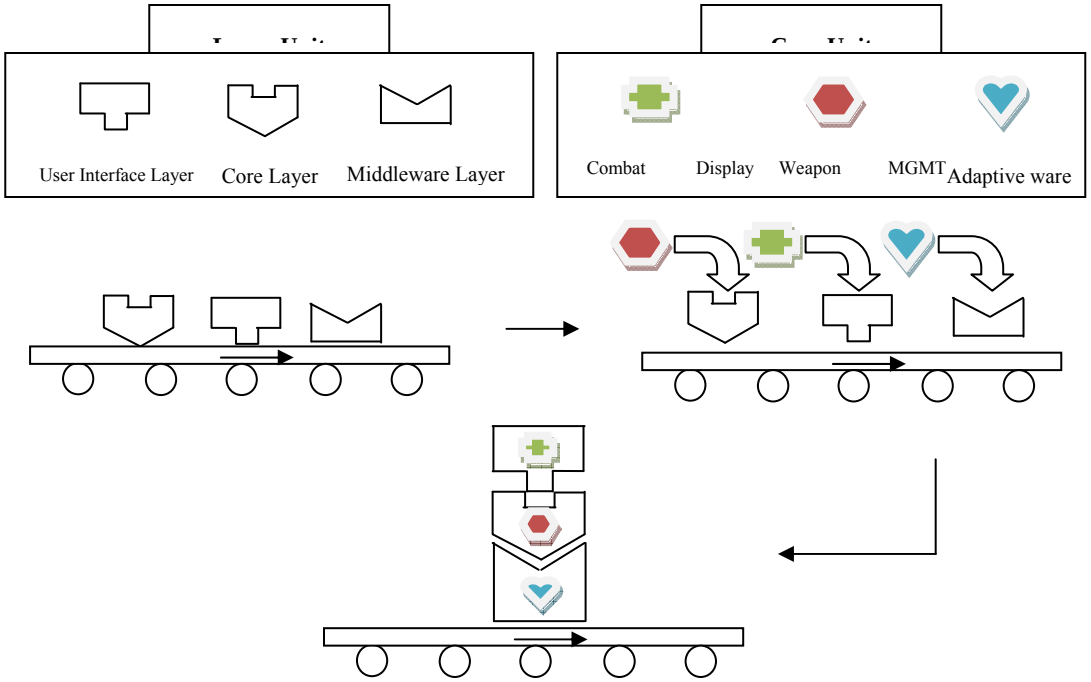


Figure 4: Simplified Production Plan of Assembling Proposed AAW Software System using Layered

Figure 4 shows a simplified production plan of assembling the proposed AAW software system using layered software architecture to help readers to understand. To assemble a software system, there are two important basic units. One is a layer unit, and the other is a core unit. The layer unit is a group of lines of codes that perform functions to meet interface requirements between layers. Each layer has different layer unit because each layer has different set of adjacent layers. The layer unit is a common lines of codes among all software products in the same layer, and it is reusable once it developed because any change in the system will not affect the interface requirement between layers. The core unit is a group of lines of codes that perform desired functions of the system. That is, the core unit is a software product that software developers create to meet system functional requirement. For that reason, the core unit might require to be developed when change in the system occurs.

In Figure 4, at the first phase of the software product line, software developers need to choose what layer units they need to use for their software products as Figure 4 shows necessary layer units enter the production line. At the second phase, software developers insert or add their desired core unit to the selected layer unit. This creates a layer module. As a final phase, software developers need to assemble layers modules together to build a software system.

By assigning the role of meeting the interface requirement to the reusable layer unit, software developer can reduce software development time, and several different configurations of a software system can be developed and produced at the same time because software developers only need to work on the core unit.

The successful implementation of SPL in the OA environment requires not only software discipline but also several other disciplines. Organizations will need to focus their development efforts to maximize the added value of SPL. However, implementing SPL will require substantial initial investments, manpower and resources. In the long term the right investment will allows for the ease of upgrading and adapting to new technologies changes, and reduction in total life cycle costs.

5 Implementation of Supportability/Logistic in SW Architecture

Our research and experience has shown that for most systems the cost of initial software development has been greatly exceeded by the cost of supporting the software throughout a system's operational life". Therefore, integrating supportability requirements into the software architecture development process may increase the overall system development costs, but will reduce the total LCC of the system.

There are many issues that affect software supportability, such as location of support, Software Support Documentation, Configuration Management, Post Mission Recovery, System Support Package Component List that includes development platform, test equipment, facilities, hardware and software tools, documentation and source code, Licenses, design rights, replication and transfer, processes, procedures, controls, consumables, parts, training and skills and personnel requirements.

There is a range of factors that affect software supportability. These factors are generally either attributes of the software item itself, or the associated development process, or of the environment within which the software is operated or supported. The factors are not all unique to software, and in some cases will be linked to system-level considerations. The factors which are of key significance are listed below:

a) Change Traffic: A measure of the rate at which S/W modification is required
b) Safety Integrity: Determined by the safety criticality of the functions that it provides. Criticality is the anomalies in the system with varying severity.
c) Expansion Capability: The degree which S/W can be modified, physical limitations are included such as available memory, processor performance, mass storage capacity, input/output bandwidth
d) Fleet Size and Disposition: number of equipments in use, locations at which software support is conducted, large fleets will have higher equipment usage

e) Modularity: low-level structure of software design, dependent on engineering design up from
f) Size: metrics are available to quantify software size. Size influences supportability and change traffic expected.
g) Security: Classification of S/W items will be dependent on application
h) Skills: S/W modification will require personnel with S/W engineering skills,
i) Standardization: applied to the computing environment which the S/W executes
j) Technology: considered with the S/W engineering methods and tools used in development and implementation, with the H/W and S/W of the host and target platforms
k) Tools and Methods: selection of tools and methods is dependent on technologies used to develop and implement the system
l) Documentation: includes all records, electronic or hardcopy that relate to the requirements, spec analysis, design, implementation, testing and operation

Refer to the UK Ministry of Defence, Defence Standard 00-60(Part 3), DEF STAN 00-60 (PART 3)/3, for a further description of each of the attributes, explanation of their relationships to any given software architecture/system and the extent of their impact on supportability. When conducting Analysis of Alternatives (AoA) and trade offs to select software products during software architecture development these factors should be used in the overall Objectives Hierarchy to determine the optimum software architecture.

The software development process will also influence the number of residual faults in the software at the end of development and the ease with which the software can be modified. Contracts for software procurement typically define the software functionality and any required development standards. Therefore, little scope is provided for optimizing designs for supportability. As a result, LSA for software aspects must commence before such contracts are defined.

The LSA strategy for any project involving software should equally apply LSA tasks concurrently to hardware and software elements. LSA tasks undertaken in respect of software items during the early, pre-design phases of a project offer the greatest potential benefits for supportability and for the achievement of optimum LCC. However, at the detailed level of task application, there are some distinct techniques and considerations for software, which are described in detail in DEF STAN 00-60 (PART 3)/3. The software supportability will be direct results of equipment technical requirements that include the functional requirements, test, testability, system monitoring, data recording and non-functional requirements such as growth capacity. Application of LSA tasks in the early project phases should be applied to both the application software and the system software architecture. The reason for conducting early LSA tasking is to clearly identify the requirements, constraints and issues requiring further analysis. Figure 2 of DEF STAN 00-60 (PART 3)/3 identifies both requirements and task analysis that need to be developed during the early phase. Example: Prime Equipment Technical Requirements include Functional aspects such as Built-in test, diagnostics and monitoring, recovery,

reconfiguration, degradation, mode reversion, data recording/access/upload/download. Non-functional aspects: growth capacity, memory, processing, and data communications.

Software Support Analysis (SSA) is a technique to assist in the application of LSA to the software aspects of systems. The objectives of SSA are to identify potential requirements for, and constraints affecting, software support, and to identify software requirements to enhance system availability and supportability. SSA comprises 2 broad activities: requirements analysis and process modeling.

Guidance on the conduct of these activities is provided in DEF STAN 00-60 (PART 3)/3. The necessary depth of application is variable and will depend upon the project phase, the type of procurement and the role of the equipment. The normal approach to the balance between costs and time for the analysis effort and the level of detail should apply.

Other supportability analysis techniques like Failure Modes Effects and criticality Analysis (FMECA) and Reliability Centered Maintenance (RCM) should be applied to software just as they are applied to hardware. FMECA analysis is applied to the overall equipment and should identify particular failure modes associated to the functionality provided by or dependent on software. Although RCM is not applicable in the direct sense as failures will not be associated with hardware, and often referred to as latent defects, but RCM may identify the safety integrity level. Process models should be developed for each deliverable and take into account support concept constraints and requirements.

Software support tasks for the system operation and maintenance will respond to software-induced failures during the mission. The tasks fall into 2 categories: action to reload/restart software to bring the system back to operating capability and the second is to record information about the system condition/configuration at the time of the failure. Post mission tasks include data extraction, fault investigation and functions such as sanitization of classified software data. When software has been modified it suggested that a level of repair analysis (LORA) be done to ensure trade-off analysis of maintenance level for hardware and software support has been addressed.

6 Conclusion

Combat systems logical data interfaces and functions have led to complex interactions that are difficult to maintain. Central processing and I/O constraints make working with newer weapon systems and sensors extremely difficult to manage. Software development and maintenance is hampered by the use of militarized non-standard software/hardware. Currently central processing is easily saturated based upon the heavy demand of required execution time.

Combat System architectures and combat system software tailored to specific platform is no longer a viable solution. It's a costly proposition in term of development, and software logistical support throughout its life cycle. Combat system processing is time critical and must be able to handle large quantities of data from either on board or off board systems.

The solution to the above is to move towards an open architecture approach for hardware and software components to meet today and future requirements. Moving towards an open environment includes working with and establishing the right interfaces to work with COTs devices and protocols. The trick is developing the right specification that will allow for flexibility and expansion.

The solution provided in this paper, the use of a layered architecture and the use of SPLs are two methods that will allow the USN to achieve:

- reduced time to create and deploy a new product
- reduced the number of defects per product
- reduce the overall engineering cost per product
- increase total number of products deployed and managed (Krueger)
- reduce life cycle supportability cost

The layered architecture approach decouples highly complex and integrated functions and the use of SPLs takes advantage of previous combat system development where it makes sense. SPL's can be developed once and used many times. New systems can take advantage of previous developed code and integrate new code with relatively little effort (compared to new development).

References

Software Line of Code (SLOC).< http://en.wikipedia.org/wiki/Source_lines_of_code >(accessed January 2009)

Eriksson, Magnus. An Introduction to Software Product Line Development. Alvis Hägglunds, SE-891 82.

Fein, Geoff. 2007. Navy's SHARE Repository Seeing Steady Growth In First Six Months. Defense Daily, 2 February.

Garlan, David and Shaw, Mary. 1994. An Introduction to Software Architecture. Carnegie Mellon University.

Krueger, Charles W. 2004. Benefits of Software Product Lines. BigLever Software. <http://softwareproductlines.com/benefits/benefits.html>.

Northrop, Linda M. 2002. SEI's Software Product Line Tenets. IEEE Software, July/August.

Toft, Peter. 2004. The HP Owen Firmware Cooperative: A Software Product Line Success Story. HP. <http://softwareproductlines.com/successes/hp.html>.

van der Linden, Frank. 2004. Multiple Product Lines for Philips Medical Systems. Philips Medical Systems. <http://softwareproductlines.com/successes/philips_med.html>.

David E. Bakken. 2003. Middleware. Washington State University: Kluwer Academic Press

Dot21 Adopts COT Middleware From Real-Time Innovations for U.S. Military Systems Upgrade, September. 6, 2005 < http://www.rti.com/company/news/dot21.html >.

Software Product Line (SPL) < http://www.sei.cmu.edu/productlines/ >(accessed Feb, 2009)

UK Ministry of Defence, Defence Standard 00-60(Part 3), DEF STAN 00-60 (PART 3)/3

Bosch, Jan**.** 2000. Design and Use of Software Architectures: Adopting and evolving a product-line approach. Addison Wesley.

APPENDIX K: HP PROCESS AND DODAF ARTIFACT RELATIONSHIPS

The Hatley-Pirbhai (HP) process was chosen because it's showed the dynamic architecture, and emphasized of reusable components to support Software Product Line development. Although the Hatley-Pirbhai methodology and DoDAF were developed independently, they both shared strong commonalities and artifacts. Both HP models and DoDAF views have the graphical representation of the real system. The textual descriptions of HP graphical models and DoDAF views enhance the fidelity of the models and views. The Table 000 below shows the equivalent relationships between the HP process models and DoDAF artifacts.

Table 1: The Relationship between Hatley-Pirbhai Process and DoDAF Artifact

HP methodology and DoDAF share similar characteristics in terms of the way they represent the real system using graphics and texts.

Hatley-Pirbhai Process	DoDAF Views
Requirements and Enhanced Requrements Models	
Entity Models	
Class Diagrams	OV-4, OV-7
Entity-Class Specifications	AV-2
Relationship Specifications	AV-2, OV-6a, SV-10a
Requirements Dictionary	AV-2
Process Models	
Data Context Diagram	OV-1
Data Flow Diagrams	OV-2, OV-5, SV-4
Process Specifications	AV-2
Control Models	
Control Context Diagram	OV-1
Control Flow Diagrams	
Control Specifications	AV-2
State Charts	OV-3, OV-6b, SV-6, SV-10b
Decision Process / Activation Tables	OV-3, OV-6b, SV-6, SV-10b
Architecture Models	
Flow Models	
Architecture Flow Context Diagram	OV-1, SV-1
Architecture Flow Diagrams	OV-2, OV-5
Architecture Module Specifications	AV-2, SV-5, SV-7
Architecture Dictionary	
Message & Inheritance Models	
Architecture Message Context Diagram	OV-1
Architecture Message Diagrams	SV-2
Module Inheritance Diagrams	OV-4, OV-7
Architecture message Specifications	AV-2
Interconnect Models	
Architecture Interconnect Context Diagram	OV-1
Architecture Interconnect Diagrams	
Architecture Interconnect Specifications	AV-2

The operational views of DoDAF are roughly equivalent to the requirements models of HP methodology. In the same way, the system views of DoDAF are roughly equivalent to the architecture models of HP methodology.

- DoDAF's OV-1 view shows the high level description of operational concept. The equivalent HP models to DoDAF's OV-1 views are context diagrams (data, control, architecture flow, architecture message, and architecture interconnect) which represent the highest level of the entire system in their own context.

- DoDAF's AV-2 view is the architecture data repository with definitions of all terms used in all products. In other words, the AV-2 view is the textual description of all the semantics of lines and graphs depicted in the DoDAF artifacts. Entity-class specifications, relationship specifications, requirements dictionary, process specifications, control specifications, architecture module specifications, architecture message specifications, and architecture interconnect specifications of HP models do exactly that for HP models.

- OV-2 describes the operational mode connectivity and shows the information exchange between nodes. The data flow diagrams and architecture flow diagrams of HP models also show the information, energy, and material exchange between processes and modules.

- OV-3 describes the operational information exchange matrix and provides a very detailed set of information about the operational interfaces. The Control Specification of HP models describe which control flows trigger, which processes.

- The organizational relationship chart (OV-4) describes the relationship between groups. The class diagrams and module inheritance diagrams of HP models also show the relationships between classes.

- Operational Activity Model (OV-5) has two diagrams. The first shows the hierarchy of operational activities, and the other diagram shows the informational flow between operational activities. The data flow diagrams and architecture flow diagrams of HP models are the same as OV-5. The HP models show the hierarchy of the processes and modules by having multiple layers. The data flow diagrams and architecture flow diagrams also show the informational flows between the processes and modules.

- Operational Rules Model (OV-6a) gives the textual description of operational rules. The relationship specification of HP models gives similar description.

- Operational State Transition Description (OV-6b) shows how the states are changed. The equivalent to OV-6b in HP process are the finite state machines, state charts, and decision process / activation tables in control specifications of HP models show how the events (i.e. control flows) and actions (i.e. processes) change the states.

- Operational Event-trace Description (OV-6c), which describes the sequence diagram, is similar to the Timing Specification in the HP process.

- Logical data model (Ov-7) describes the data elements at a high level of abstraction. The logical data model resembles the class diagrams and module inheritance diagrams of HP models.

- System Interface Description (SV-1) describes the system interface. The architecture flow context diagram of HP models resembles SV-1.

- Systems Communications Description (SV-2) describes the communications infrastructure. The architecture message diagrams of HP models depict the messages between the modules.

- System functionality description (SV-4) comes with two diagrams, the functional hierarchy and the data flow diagram. One variant of SV-4 resembles the HP models' data flow diagram.

- HP models do have traceability matrix as part of architecture module specification to map the requirements model's stores, CSPECS, and processes to architecture model's modules. HP models' traceability matrix resembles the operational activity to system function traceability matrix (SV-5), which maps the operational activities to the system functions. Additionally, CSPECS of HP models resemble the systems data exchange matrix (SV-6), which contains several more data description attributes than the OV-3.

- Systems Performance Parameters Matrix (SV-7) lists the performance parameters; the same in HP process is the architecture module specification. The architecture module specification also lists the required constraints such as reliability, maintainability, safety, physical, design, manufacturability, cost, and schedule, etc.

- The relationship specification of HP models resembles systems functionality sequence and timing description (SV-10a), which captures constraints. The control specification of HP models resembles SV-10b, which captures sequencing of states and modes. The HP models do not have equivalence to SV-10c, which resembles sequence diagram.

The development of system specification, Hatley-Pirbhai process was applied to the AAW system architecture for its versatility in top down and bottom up approach and it strong emphasis of components reuse. Hatley-Pirbhai process artifact does not have a one to one mapping to DoDAF artifacts. Both system specifications derived from each process have similarities. This paper attempts to address the rigor of system engineer process and different technique in deriving the equivalent artifacts.

References

Dam, Steven H. 2006. DoD Architecture Framework. Marshall VA. SPEC.

Department of Defense Architecture Framework (DoDAF) v1.0, Vol II. 2003. http://www.eaframeworks.com/DoDAF/Volume_II.pdf. (accessed Feb 2009).

Hatley, Derek, Peter Hruschka, and Imtiaz Pirbhai. 2000. Process for System Architecture and Requirements Engineering. New York. Dorset House Publishing.

APPENDIX L: EXTEND MODEL RESULTS

Introduction

The model was run 1000 times for 10 iterations to produce comparable results shown in the following tables:

- 15.24 meter height 2 illuminators; P_k(weapon_1) = 0.80;
- 15.24 meter height 2 illuminators; P_k(weapon_1) = 0.98;
- 15.24 meter height 3 illuminators; P_k(weapon_1) = 0.90;
- 15.24 meter height 4 illuminators; P_k(weapon_1) = 0.85.

Each table contains information for each target's identification, time of intercept, range at intercept, its time of flight duration, and the final weapon utilized to destroy the threat. Records of the ship's weapon resources (ammo) are also maintained in each run of the model.

The sample size of ten thousand is too much information to display. The data for rows 2 through row 998 are not display. The truncated tables K1 to K4 are displayed to illustrate how the 98% confidence intervals of P_{RA} were tabulated for the AAW architecture.

K1 - Calculation of P_{RA} and Threats Destroyed for Configuration A

(Data for rows 2 through 989 not shown)

Run	Threat Killed									
	Iteration									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	8	8	8	8	8	7	7	8	8	8
1	7	8	7	8	7	8	7	8	8	8
990	8	7	8	5	6	8	8	7	8	8
991	6	8	7	8	7	7	8	8	8	8
992	8	7	7	5	8	6	7	8	7	8
993	8	8	8	7	8	7	7	8	5	8
994	8	8	6	7	8	8	5	8	8	8
995	8	7	8	8	8	8	8	6	8	8
996	8	8	8	8	8	8	7	7	8	8
997	7	8	8	4	8	8	8	8	8	8
998	8	8	8	7	8	6	8	8	8	8
999	8	8	8	7	8	7	8	8	8	8

P_{RA}	0.654	0.678	0.704	0.642	0.653	0.681	0.656	0.684	0.697	0.689
Avg Kill	7.366	7.416	7.444	7.338	7.36	7.402	7.379	7.379	7.44	7.447

10 Iteration of 1000 Runs

Avg P_{RA}	0.6738			
StdDev P_{RA}	0.0211			
98% CI	0.0155			
98% Confident that P_{RA} is between:	0.6583	and	0.6893	

Avg Killed	7.3971			
StdDev Killed	0.0386			
98% CI	0.0284			
98% Confident that the number of threat Killed is between:	7.3687	and	7.4255	

K2 - Calculation of P_{RA} and Threats Destroyed for Configuration B

(Data for rows 2 through 989 not shown)

Run	Threat Killed									
	Iteration									
	0	1	2	3	4	5	6	7	8	9
0	8	8	8	8	8	8	8	8	8	8
1	8	8	8	8	8	8	8	8	8	8
990	8	8	8	8	8	8	8	8	8	8
991	8	8	8	8	8	8	8	8	8	8
992	8	8	8	8	7	8	8	8	8	8
993	8	8	8	8	8	8	8	8	8	8
994	8	8	8	8	8	8	8	8	8	8
995	8	8	8	8	8	8	8	8	8	8
996	8	8	8	8	8	8	8	8	8	8
997	8	8	8	8	8	8	8	8	8	8
998	8	8	8	8	8	8	8	8	8	8
999	8	8	8	8	8	8	8	8	8	8

P_{RA}	0.992	0.99	0.988	0.989	0.992	0.992	0.985	0.986	0.988	0.991
Avg Kill	7.992	7.985	7.986	7.988	7.992	7.992	7.985	7.985	7.984	7.988

10 Iteration of 1000 Runs

Avg P_{RA}	0.9893			
StdDev P_{RA}	0.0025			
98% CI	0.0019			
98% Confident that P_{RA} is between:	0.9874	and	0.9912	

Avg Killed	7.9877			
StdDev Killed	0.0032			
98% CI	0.0024			
98% Confident that the number of threat Killed is between:	7.9853	and	7.9901	

K3 - Calculation of P_{RA} and Threats Destroyed for Configuration C

(Data for rows 2 through 989 not shown)

Run	Threat Killed									
	Iteration									
	0	1	2	3	4	5	6	7	8	9
0	8	8	8	8	8	8	8	8	8	8
1	8	8	8	8	8	8	8	8	8	8
990	8	8	8	8	8	8	8	8	8	8
991	8	8	8	8	8	8	8	8	8	8
992	8	8	8	8	8	8	8	8	8	8
993	8	8	8	8	8	8	8	8	8	8
994	8	8	8	8	8	8	8	8	8	8
995	8	8	8	8	8	8	8	8	8	8
996	8	8	8	8	8	8	8	8	8	8
997	8	8	8	8	8	8	8	8	8	8
998	8	8	8	8	8	8	8	8	8	8
999	8	8	8	8	8	8	8	8	8	8

P_{RA}	0.995	0.998	0.989	0.994	0.994	0.997	0.997	0.996	0.995	0.988
Avg Kill	7.995	7.998	7.984	7.992	7.994	7.997	7.996	7.996	7.995	7.983

10 Iteration of 1000 Runs

Avg P_{RA}	0.9943			
StdDev P_{RA}	0.0033			
98% CI	0.0025			
98% Confident that P_{RA} is between:	0.9918	and	0.9968	

Avg Killed	7.993			
StdDev Killed	0.0053			
98% CI	0.0039			
98% Confident that the number of threat Killed is between:	7.9891	and	7.9969	

K4 - Calculation of P_{RA} and Threats Destroyed for Configuration D

(Data for rows 2 through 989 not shown)

Run	Threat Killed									
	Iteration									
	0	1	2	3	4	5	6	7	8	9
0	8	8	8	8	8	8	8	8	8	8
1	8	8	8	8	8	8	8	8	8	8
990	8	8	8	8	8	8	8	8	8	8
991	8	8	8	8	8	8	8	8	8	8
992	8	8	8	8	8	8	8	8	8	8
993	8	8	8	8	8	8	8	8	8	8
994	8	8	8	8	8	8	8	8	8	8
995	8	8	8	8	8	8	8	8	8	8
996	8	8	8	8	8	8	8	8	8	8
997	8	8	8	8	8	8	8	8	8	8
998	8	8	8	8	8	8	8	8	8	8
999	8	8	8	8	8	8	8	8	8	8

P_{RA}	0.993	0.99	0.989	0.988	0.988	0.988	0.991	0.988	0.989	0.985
Avg Kill	7.992	7.987	7.989	7.987	7.987	7.985	7.99	7.986	7.988	7.985

10 Iteration of 1000 Runs

Avg P_{RA}	0.9889			
StdDev P_{RA}	0.0021			
98% CI	0.0016			
98% Confident that P_{RA} is between:	0.9873	and	0.9905	

Avg Killed	7.9876			
StdDev Killed	0.0022			
98% CI	0.0016			
98% Confident that the number of threat Killed is between:	7.986	and	7.9892	

Summary

Data Analysis has confirmed the following AAW architectures have the capability to keep every threat of an eight-strike raid attack more than 2 km away from itself, yielding a P_{RA} of greater than 99%:

- 15.24 meter height 2 illuminators; P_k(weapon_1) = 0.98;
- 15.24 meter height 3 illuminators; P_k(weapon_1) = 0.90;
- 15.24 meter height 4 illuminators; P_k(weapon_1) = 0.85.

THIS PAGE INTENTIONALLY LEFT BLANK

AAW ARCHITECTURE

Overview and Summary Information (AV-1)

APPENDIX M: CORE PRODUCTS

Description: Introduction

The Anti-Air Warfare (AAW) Architecture Concept of Operation (ConOps) goals are to describe and develop a system that satisfies the user's needs in regards to performance against a littoral threat. The user needs were determined to be similar to the analysis stated in problem 4 of the SE 3123 Final Exam. The combat system functionality will be described through an architectural perspective given the requirements and threat analysis stated. Based on the given information, the AAW Architecture will also address and discuss scenarios describing combat engagements for both self protection and protection of high value units (i.e. Limited Area Defense).

The AAW Combat system will meet the following required thresholds for both Self Defense and Limited Area Defense:

AAW Combat System Requirements

· System reaction time is maximum 6 seconds maximum

· Firm tracks occur 1.5 seconds after target detection

· Combat weapon system will include fire-and-forget and guided weapons

· Fire-and-Forget weapons will be utilized based on performance parameters that will be required to achieve the Pra of 0.99 against the previously stated threat

- Possible fire-and-forget weapons selection option may include:

  - RAM

- Possible guided weapons selection options may include:

  - ESSM
  - SM 2

AAW ARCHITECTURE

Overview and Summary Information (AV-1)

· The ship will support either 2 8-cell VLS launchers or 2 RAM launchers

· VLS has a launch rate of 1 missile per second

· RAM can fire the first missile after 5 seconds and has a launch rate of 1 missile per 2 seconds thereafter

· The ordnance requirements are as follows:

- RAM: Speed - Mach 2.0, Range – (1km-10km), P(k) = 0.85
- ESSM: Speed - Mach 3.0, Range – (1km-30km), P(k) = 0.8
- SM 2: Speed - Mach 2.5, Range – (3km-80km), P(k) = 0.7

- Probability of kill, P(k) data account for all the missile performance and fuzing reliability issues

· There are two x-band illuminators for the ESSM and SM 2 missiles

· The illuminator tie-up time is 6 seconds maximum

· The kill assessment time is 2 seconds

· Soft-kill countermeasures will be utilized in architecture

o Soft-kill P(k) = 0.6

· Keep-out range is 2km

Operational Concept

There are three primary scenarios which can describe the ConOps with the given requirements. The first scenario will describe the long-range surveillance capabilities for conducting a horizon and volume search continuously. The second scenario will describe a Self Defense (SD) architecture that engages and achieves a 0.99 percent Pra against a littoral threat. The SD architecture will protect the ship against enemy attacks through a layered defense strategy that incorporates both hard-kill weapon systems and soft-kill countermeasures.

Generally speaking, a layered defense strategy in AAW breaks down into defending areas or zones around ownship; each zone is defined by its range from ownship. Typically, one could use eight (8) zones (zone 1 would be at the farthest range from the ship, while zone 8 would be the closest) and assign the defense of those zones with hardkill (HK)/softkill (SK) countermeasure as follows:

AAW ARCHITECTURE

Overview and Summary Information (AV-1)

Zone 1: Electronic jamming measures

Zone 2: Electronic decoy measures

Zone 3: Medium-range missiles

Zone 4: Short-range missiles

Zone 5: Medium-range chaff

Zone 6: Close-range missiles

Zone 7: Guns

Zone 8: Close-in chaff/IR decoys

Due to time constraints, this effort will integrate only HK and SK defenses and will not address a gun system.

The third scenario will describe a situation in which the AAW architecture will achieve the desired Pra in defense of High Value Units (HVUs) or within a Limited Area Defense. This situation describes a scenario in which the AAW protects any/all external HVUs such as aircraft carriers and critical combat support ships within a certain critical combat sector/range.

Basic Operational Scenario

Ownship is operating in ideal conditions (i.e. "Blue Ocean"). The ship's radar and EO/IR sensors are operating and performing constant search sweeps around the ship searching for possible enemy attacks. The EO/IR sensor and L-band radar are designed to perform horizon and above horizon surveillance and to work in conjunction with the combat weapon systems to insure a keep-out range of 2km.

The radar and sensor systems suddenly detect and initiate tracks on multiple in-bound threats. The threats consist of eight littoral ASMs, arriving at a rate of one every four seconds. The ASMs are traveling at a speed of Mach 0.9 and possess RF seekers which turn on at 7nm.

The AAW Architecture performs search, detect, track, command/control, and engagements of all eight ASMs and destroys the raid, preventing casualty to ownship.

AAW ARCHITECTURE

Overview and Summary Information (AV-1)

Scenario 1: Surveillance (Search and Detect)

Background

Surveillance and target acquisition are vital components in the overall combat system architecture. It is the first process in the architecture for self defense and protection of high value units. It is also difficult to accomplish because radar track displays detect large quantities of observations within a small section or area often in the form of clutter. It is difficult to recognize an object of interest within the clutter or noise of the radar signal; enemy radar jammers and decoys can also produce false observations. Once an observation is determined, the radar can assign and monitor (track).

Another concept of Surveillance is the Field of Regard (FOR) and Field of View (FOV). FOR is defined as a hemisphere of solid angle that is, the target could be expected to be anywhere above the earth's surface. FOV is a smaller area of interest. It is the area in which an observer can view in one direction. As the observer moves the FOV will be the direction of area in which the observer is looking. Surveillance can also be applied to horizon search patterns or above horizon search patterns.

Scenario

AAW system will conduct horizon and volume search capabilities; long range search and detection above the horizon and short range search and detect to the horizon. The long range search (L-band) radar operates and conducts volume search capabilities searching for targets above the horizon. The short range EO/IR sensor operates and conducts short range search and detect capabilities to the horizon.

Scenario 2: Self Defense (SD)

As the ASMs approach the ship, the combat system radar will perform long-range search and detection above the horizon; tracks are subsequently initiated on the ASMs. The AAW system will then conduct a command/control sequence in which the sensors provide feedback and the control system provides weapon assignment for engagement. An initial engagement could be generated from a short range missile system such as RAM.

Scenario 3: Limited Area Defense (LAD) or Protection of High Value Units (HVUs)

As the ASM raid approach the ship, the AAW architecture's long range radar system detects and tracks the ASMs; detecting the first ASM at maximum range. The AAW system will then conduct a command/control sequence, where the sensors provide feedback and the control system provides weapon assignment for engagement. In the protection of HVUs, a longer and wider range engagement is

necessary; the first possible engagement will be to assign a soft-kill weapon system such as decoys or chaff. The next possible engagement would be to deploy a long range missile system such as ESSM or SM 2 (an x-band illuminator provides real-time kill assessment for ESSM or SM 2). The third line of defense would be to engage the incoming ASM attacks with a short range missile system such as RAM.

Scope: **Scope**

The ConOps scope will require the system to ultimately achieve a probability of raid annihilation (Pra) of 0.99 against a littoral threat that consists of eight radially inbound anti-ship missiles (ASMs). Additionally, the scope of the problem will be limited to an ideal environmental situation for the area of operation; this will include clear weather and/or "blue ocean" conditions. The given threat situation will also include the following:

Threat Capabilities

· ASMs rate of arrival will be one every four seconds

· ASMs will have an area of 1 meter squared

· ASMs velocity will be Mach 0.9

· ASMs will fly at a height of 9 meters

· ASMs are equipped with an RF seeker that turns on at 7nm

Potential targets that meet these performance criteria are as follows:

- Exocet medium range anti-ship missile – transonic speed range (1134 km/hr), sea-skimming, 1.1 meter wingspan

Time Frame: As-Is

Mission(s): **AAW**

AV-2

ARCHITECTURE DESCRIPTION DOCUMENT

FOR

AAW ARCHITECTURE

Thursday, January 22, 2009

Prepared For:

Naval Postgraduate School
Monterey, CA

Prepared By:

Cohort 6
4363 Missile Way
Port Hueneme, CA 93043

Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

Table of Contents

List of Figures

List of Tables

1 Architecture Description

AAW ARCHITECTURE

Description:

Introduction

The Anti-Air Warfare (AAW) Architecture Concept of Operation (ConOps) goals are to describe and develop a system that satisfies the user's needs in regards to performance against a littoral threat. The user needs were determined to be similar to the analysis stated in problem 4 of the SE 3123 Final Exam. The combat system functionality will be described through an architectural perspective given the requirements and threat analysis stated. Based on the given information, the AAW Architecture will also address and discuss scenarios describing combat engagements for both self protection and protection of high value units (i.e. Limited Area Defense).

The AAW Combat system will meet the following required thresholds for both Self Defense and Limited Area Defense:

AAW Combat System Requirements

· System reaction time is maximum 6 seconds maximum
· Firm tracks occur 1.5 seconds after target detection
· Combat weapon system will include fire-and-forget and guided weapons
· Fire-and-Forget weapons will be utilized based on performance parameters that will be required to achieve the Pra of 0.99 against the previously stated threat

o Possible fire-and-forget weapons selection option may include:

- RAM

o Possible guided weapons selection options may include:

- ESSM
- SM 2

· The ship will support either 2 8-cell VLS launchers or 2 RAM launchers
· VLS has a launch rate of 1 missile per second
· RAM can fire the first missile after 5 seconds and has a launch rate of 1 missile per 2 seconds thereafter
· The ordnance requirements are as follows:

o RAM: Speed - Mach 2.0, Range – (1km-10km), P(k) = 0.85
o ESSM: Speed - Mach 3.0, Range – (1km-30km), P(k) = 0.8
o SM 2: Speed - Mach 2.5, Range – (3km-80km), P(k) = 0.7

- Probability of kill, P(k) data account for all the missile performance and fuzing reliability issues

· There are two x-band illuminators for the ESSM and SM 2 missiles

1 Architecture Description

· The illuminator tie-up time is 6 seconds maximum
· The kill assessment time is 2 seconds
· Soft-kill countermeasures will be utilized in architecture

o Soft-kill P(k) = 0.6

· Keep-out range is 2km

Operational Concept

There are three primary scenarios which can describe the ConOps with the given requirements. The first scenario will describe the long-range surveillance capabilities for conducting a horizon and volume search continuously. The second scenario will describe a Self Defense (SD) architecture that engages and achieves a 0.99 percent Pra against a littoral threat. The SD architecture will protect the ship against enemy attacks through a layered defense strategy that incorporates both hard-kill weapon systems and soft-kill countermeasures.

Generally speaking, a layered defense strategy in AAW breaks down into defending areas or zones around ownship; each zone is defined by it's range from ownship. Typically, one could use eight (8) zones (zone 1 would be at the farthest range from the ship, while zone 8 would be the closest) and assign the defense of those zones with hardkill (HK)/softkill (SK) countermeasure as follows:

Zone 1: Electronic jamming measures
Zone 2: Electronic decoy measures
Zone 3: Medium-range missiles
Zone 4: Short-range missiles
Zone 5: Medium-range chaff
Zone 6: Close-range missiles
Zone 7: Guns
Zone 8: Close-in chaff/IR decoys

Due to time constraints, this effort will integrate only HK and SK defenses and will not address a gun system.

The third scenario will describe a situation in which the AAW architecture will achieve the desired Pra in defense of High Value Units (HVUs) or within a Limited Area Defense. This situation describes a scenario in which the AAW protects any/all external HVUs such as aircraft carriers and critical combat support ships within a certain critical combat sector/range.

Basic Operational Scenario

Ownship is operating in ideal conditions (i.e. "Blue Ocean"). The ship's radar and EO/IR sensors are operating and performing constant search sweeps around the ship searching for possible enemy attacks. The EO/IR sensor and L-band radar are designed to perform horizon and above horizon surveillance and to work in conjunction with the combat weapon systems to insure a keep-out range of 2km.

1 Architecture Description

The radar and sensor systems suddenly detect and initiate tracks on multiple in-bound threats. The threats consist of eight littoral ASMs, arriving at a rate of one every four seconds. The ASMs are traveling at a speed of Mach 0.9 and possess RF seekers which turn on at 7nm.
The AAW Architecture performs search, detect, track, command/control, and engagements of all eight ASMs and destroys the raid, preventing casualty to ownship.

Scenario 1: Surveillance (Search and Detect)

Background

Surveillance and target acquisition are vital components in the overall combat system architecture. It is the first process in the architecture for self defense and protection of high value units. It is also difficult to accomplish because radar track displays detect large quantities of observations within a small section or area often in the form of clutter. It is difficult to recognize an object of interest within the clutter or noise of the radar signal; enemy radar jammers and decoys can also produce false observations. Once an observation is determined, the radar can assign and monitor (track).

Another concept of Surveillance is the Field of Regard (FOR) and Field of View (FOV). FOR is defined as a hemisphere of solid angle that is, the target could be expected to be anywhere above the earth's surface. FOV is a smaller area of interest. It is the area in which an observer can view in one direction. As the observer moves the FOV will be the direction of area in which the observer is looking. Surveillance can also be applied to horizon search patterns or above horizon search patterns.

Scenario

AAW system will conduct horizon and volume search capabilities; long range search and detection above the horizon and short range search and detect to the horizon. The long range search (L-band) radar operates and conducts volume search capabilities searching for targets above the horizon. The short range EO/IR sensor operates and conducts short range search and detect capabilities to the horizon.

Scenario 2: Self Defense (SD)

As the ASMs approach the ship, the combat system radar will perform long-range search and detection above the horizon; tracks are subsequently initiated on the ASMs. The AAW system will then conduct a command/control sequence in which the sensors provide feedback and the control system provides weapon assignment for engagement. An initial engagement could be generated from a short range missile system such as RAM.

Scenario 3: Limited Area Defense (LAD) or Protection of High Value Units (HVUs)

1 Architecture Description

As the ASM raid approach the ship, the AAW architecture's long range radar system detects and tracks the ASMs; detecting the first ASM at maximum range. The AAW system will then conduct a command/control sequence, where the sensors provide feedback and the control system provides weapon assignment for engagement. In the protection of HVUs, a longer and wider range engagement is necessary; the first possible engagement will be to assign a soft-kill weapon system such as decoys or chaff. The next possible engagement would be to deploy a long range missile system such as ESSM or SM 2 (an x-band illuminator provides real-time kill assessment for ESSM or SM 2). The third line of defense would be to engage the incoming ASM attacks with a short range missile system such as RAM.

Architecture Scope:

Scope

The ConOps scope will require the system to ultimately achieve a probability of raid annihilation (Pra) of 0.99 against a littoral threat that consists of eight radially inbound anti-ship missiles (ASMs). Additionally, the scope of the problem will be limited to an ideal environmental situation for the area of operation; this will include clear weather and/or "blue ocean" conditions. The given threat situation will also include the following:

Threat Capabilities

· ASMs rate of arrival will be one every four seconds
· ASMs will have an area of 1 meter squared
· ASMs velocity will be Mach 0.9
· ASMs will fly at a height of 9 meters
· ASMs are equipped with an RF seeker that turns on at 7nm

Potential targets that meet these performance criteria are as follows:

o Exocet medium range anti-ship missile – transonic speed range (1134 km/hr), sea-skimming, 1.1 meter wingspan

Architecture Time Frame:
As-Is

Mission(s):
AAW

Composed Of:
1.0 Tactical System [WS 21200]
4.0 Sustainment [USD ATL]
ON.1 Sustainment [UNTL]
OP.1 AAW Mission Tactical
OP.1.1 Search and Detect

Documentation:
Findings

Assigned To:
Cohort 6

2 Guidance

Core 101

Guides:

Operational Information Horizon Search (CAP)
Operational Node: ON.1 Sustainment [UNTL]
Operational Node: OP.1 AAW Mission Tactical

3 Operational Overview

ON.1 Sustainment [UNTL]

Description:
Provides communication for mission readiness reporting, personnel records, and training.

Allocated Activities:
Training

External Interfacing Elements:
ON.1 Sustainment [UNTL]

Guidance:
Core 101

Assigned To:
Cohort 6

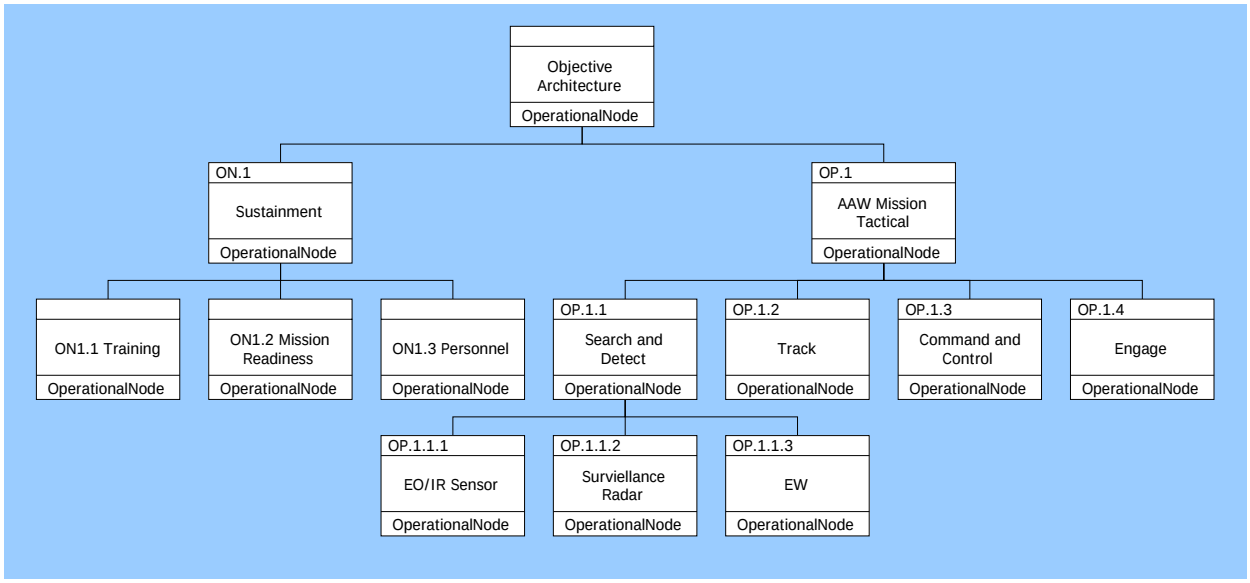


Figure 1 Operational Context Hierarchy Diagram

4 Operational Nodes

Part I - Hierarchical Operational Nodes List

ON.1 Sustainment [UNTL]
- ON1.1 Training
- ON1.2 Mission Readiness
- ON1.3 Personnel

Part II - Operational Node Definitions

ON.1 Sustainment [UNTL]

Description:
Provides communication for mission readiness reporting, personnel records, and training.

Type: Service Functionality Provider

Built From Lower-Level Operational Nodes:
ON1.1 Training
ON1.2 Mission Readiness
ON1.3 Personnel

Composes:
AAW ARCHITECTURE

Assigned To:
Cohort 6

Connected to/thru Needline(s):
Mission Readiness
Personnel
Training

4 Operational Nodes

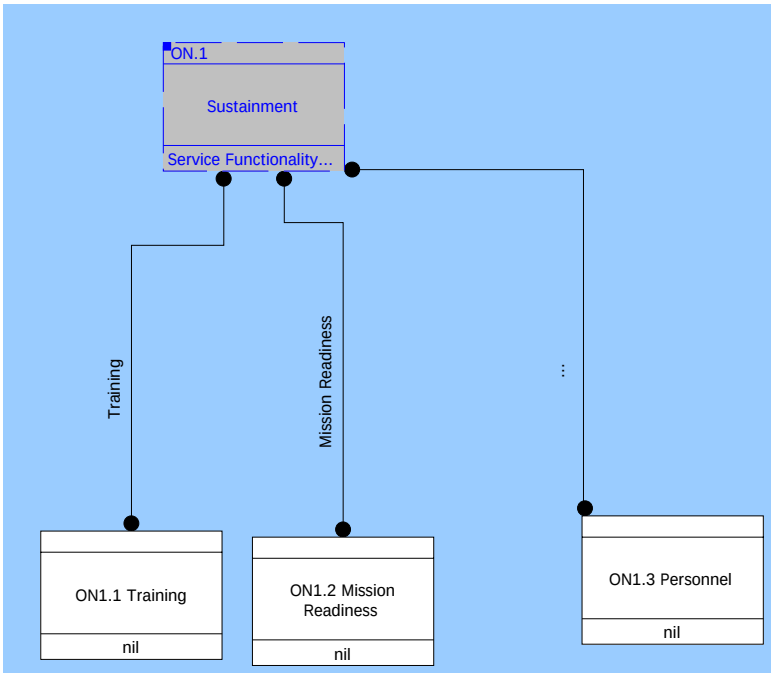


Figure 2 Sustainment Operational Node (OV-2)

Performs Operational Activities:
Training

Implemented By:
4.0 Sustainment [USD ATL]

ON1.1 Training

Connected to/thru Needline(s):
Training

Implemented By:
4.0 Sustainment [USD ATL]
4.1 Training System

ON1.2 Mission Readiness

Connected to/thru Needline(s):
Mission Readiness

Implemented By:
4.0 Sustainment [USD ATL]
4.2 Maintenance System [CNET]

ON1.3 Personnel

Connected to/thru Needline(s):
Personnel

4 Operational Nodes

Implemented By:
- 4.0 Sustainment [USD ATL]
- 4.3 User Interface [OJ-394]

5 Operational Connectivity Needs

Part I - Derived Operational Activity Exchanges

Part II - Needlines

Mission Readiness

Transferred Information:
- AAW (CAP)
- Horizon Search (CAP)

Interfacing Nodes/Externals:
- ON.1 Sustainment [UNTL]
- ON1.2 Mission Readiness

Personnel

Interfacing Nodes/Externals:
- ON.1 Sustainment [UNTL]
- ON1.3 Personnel

Training

Interfacing Nodes/Externals:
- ON.1 Sustainment [UNTL]
- ON1.1 Training

Part III - Exchange Characteristics

6 Activity Model

Part I - Hierarchical Operational Activity List

Training

Individual Training

Watch Team Training

Part II - Activity Model

Individual Training

Implemented By:

5.0 Sustainment

Training

Performed By:

ON.1 Sustainment [UNTL]

Implemented By:

5.0 Sustainment

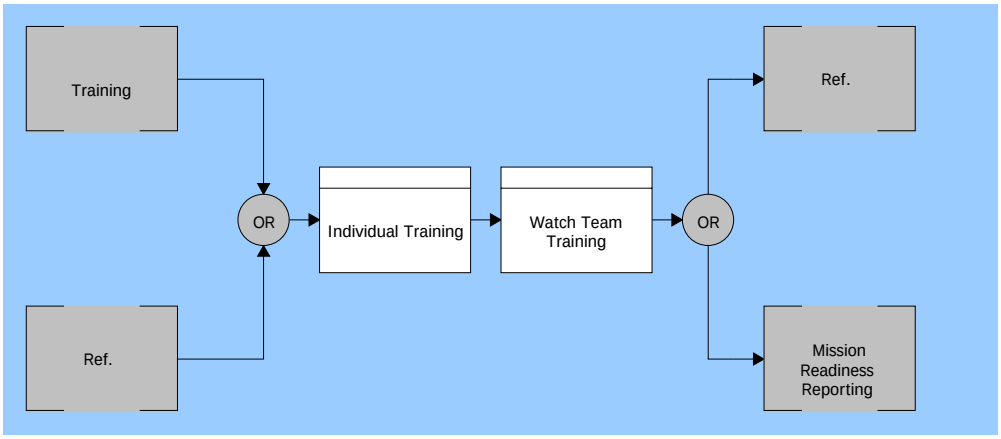


Figure 3 Training Enhanced FFBD

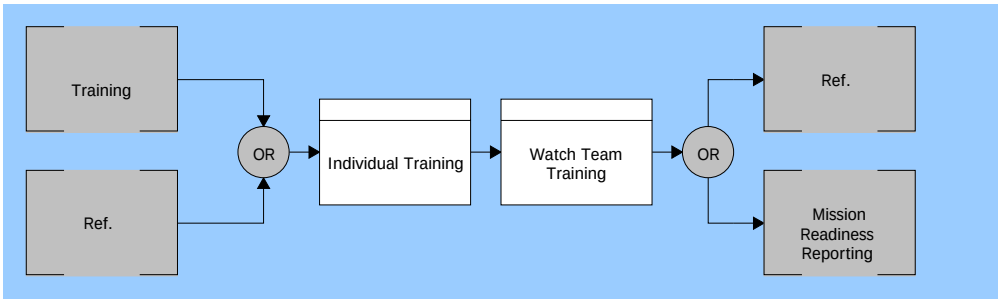


Figure 4 Training FFBD

Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

6 Activity Model

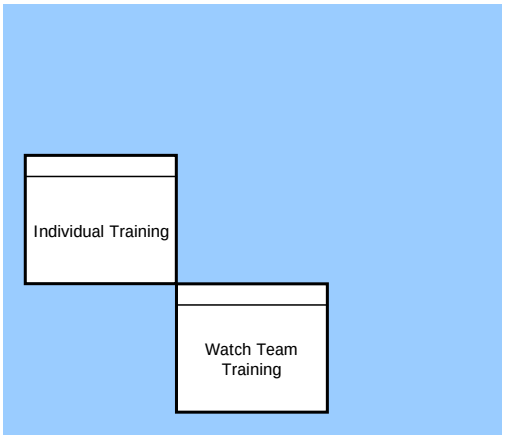


Figure 5 Training N2 Diagram

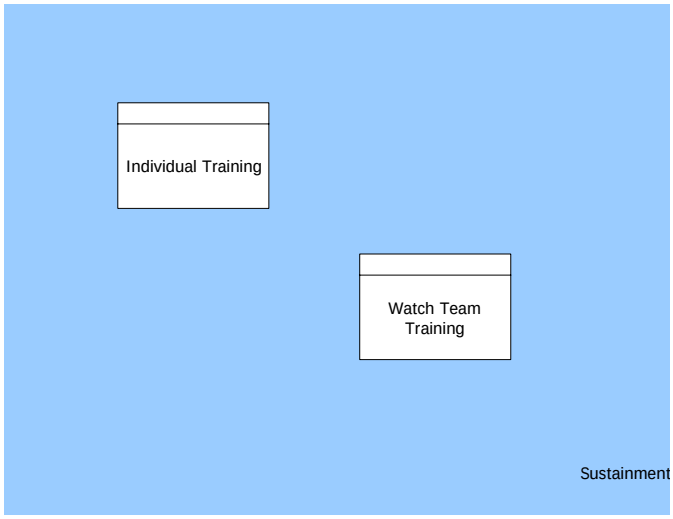


Figure 6 Training IDEF0 Diagram

Watch Team Training

Implemented By:

5.0 Sustainment

7 Operational Information

N/A

8 Activity Model Resources

N/A

9 Systems Overview

1.0 Tactical System [WS 21200]

Description:
Composes System Components required to host and execute mission functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.

System Mission:
AAW, LAD, Surveillance

Allocated Functions:
2.0 Tracking
3.0 Command and Control
4.0 Engage

Source Document(s):
AAW Concept of Operations (ConOps)
Document Number: DW-112
Document Date: Sunday, August 03, 2008
Description: The Anti-Air Warfare (AAW) Architecture Concept of Operation (ConOps) goals are to describe and develop a system that satisfies the user's needs in regards to performance against a littoral threat. The user needs were determined to be similar to the analysis stated in problem 4 of the SE 3123 Final Exam. The combat system functionality will be described through an architectural perspective given the requirements and threat analysis stated. Based on the given information, the AAW Architecture will also address and discuss scenarios describing combat engagements for both self protection and protection of high value units (i.e. Limited Area Defense).
Q3 Status Report
Document Number: WD-145
Document Date: Sunday, November 10, 2002
Description: Provides updated status report information for Q3
SSDD
Document Number: MS B-2
Description: Provides System Design functions and attributes

External Interfacing System(s):
4.0 Sustainment [USD ATL]

Assigned Design Constraints:
D.1.3 Standardization

Inputs from External Source(s):
Composite ID Message (Item)Source of Input(s):
1.1 Horizon Search
2.0 Tracking
Engage Order (Item)Source of Input(s):
3.0 Command and Control
Engage Ready (Item)Source of Input(s):
4.0 Engage
Kill Assessment Request (Item)Source of Input(s):
4.0 Engage

9 Systems Overview

Kill Confirm (Item)Source of Input(s):
4.0 Engage

Triggers from External Source(s):
Composite ID Message (Item)
Source of Trigger(s):
1.1 Horizon Search
2.0 Tracking
Engage Order (Item)
Source of Trigger(s):
3.0 Command and Control
Kill Assessment Request (Item)
Source of Trigger(s):
4.0 Engage
Kill Confirm (Item)
Source of Trigger(s):
4.0 Engage
Raw Data Items
Source of Trigger(s):
1.0 Search and Detect
1.1.1 Provide Radar Surveillance
1.1.2 Perform EO/IR Search
1.1.3 Perform EW Search

Outputs To External Destination(s):
Composite ID Message (Item)
Destination of Output(s):
3.0 Command and Control
3.0 Command and Control
Control Degraded (Item)
Destination of Output(s):
5.0 Sustainment
5.2 Mission Readiness
5.3 Provide Maintenance
Control Ready (Item)
Destination of Output(s):
5.0 Sustainment
5.2 Mission Readiness
Engage Degraded (Item)
Destination of Output(s):
5.0 Sustainment
5.2 Mission Readiness
5.3 Provide Maintenance
Engage Order (Item)
Destination of Output(s):
4.0 Engage
4.0 Engage
Engage Ready (Item)

9 Systems Overview

Destination of Output(s):
3.0 Command and Control
5.0 Sustainment
5.2 Mission Readiness

Kill Assessment Request (Item)
Destination of Output(s):
3.0 Command and Control
3.0 Command and Control

Kill Confirm (Item)
Destination of Output(s):
3.0 Command and Control
3.0 Command and Control

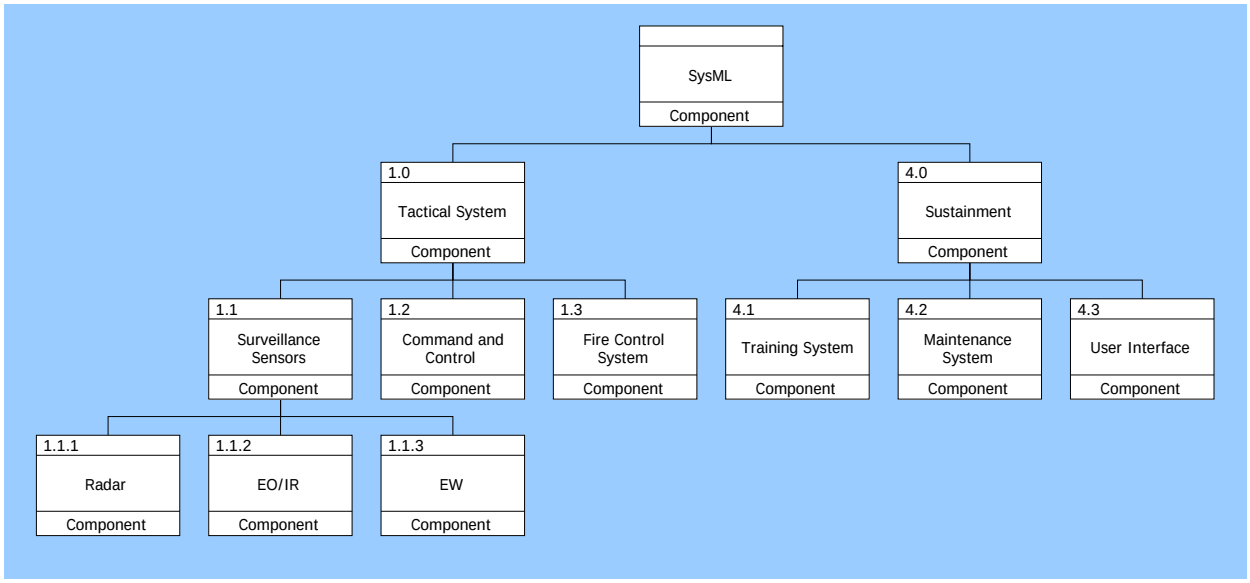


Figure 7 Tactical System Physical Context

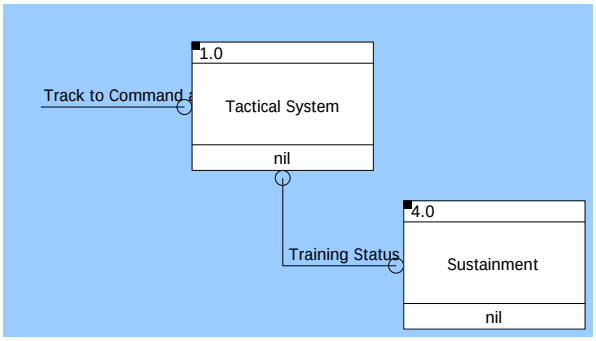


Figure 8 Tactical System Physical Interface Context

10 Systems/Components

Part I - Hierarchical Component List

1.0 Tactical System [WS 21200]
- 1.1 Surveillance Sensors [WS 21200]
  - 1.1.1 Radar
  - 1.1.2 EO/IR [WS21TYR]
  - 1.1.3 EW
- 1.2 Command and Control [WS 21340]
- 1.3 Fire Control System [WS-31333]

Part II - Component Definitions

1.0 Tactical System [WS 21200]

Description:
Composes System Components required to host and execute mission functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
SysML [AP 233]

Built From Lower-Level Component(s):
1.1 Surveillance Sensors [WS 21200]
1.2 Command and Control [WS 21340]
1.3 Fire Control System [WS-31333]

Composes:
AAW ARCHITECTURE

Implements:
OP.1 AAW Mission Tactical

Source Documents:
AAW Concept of Operations (ConOps)
SSDD

Connected thru Physical Link(s):
Track to Command and Control
Training Status

Joined To Logical Interface:
Combat Systems to C4I
Combat Systems to Ship

Joined Thru Logical Interface:
Sustainment Radar Interface

10 Systems/Components

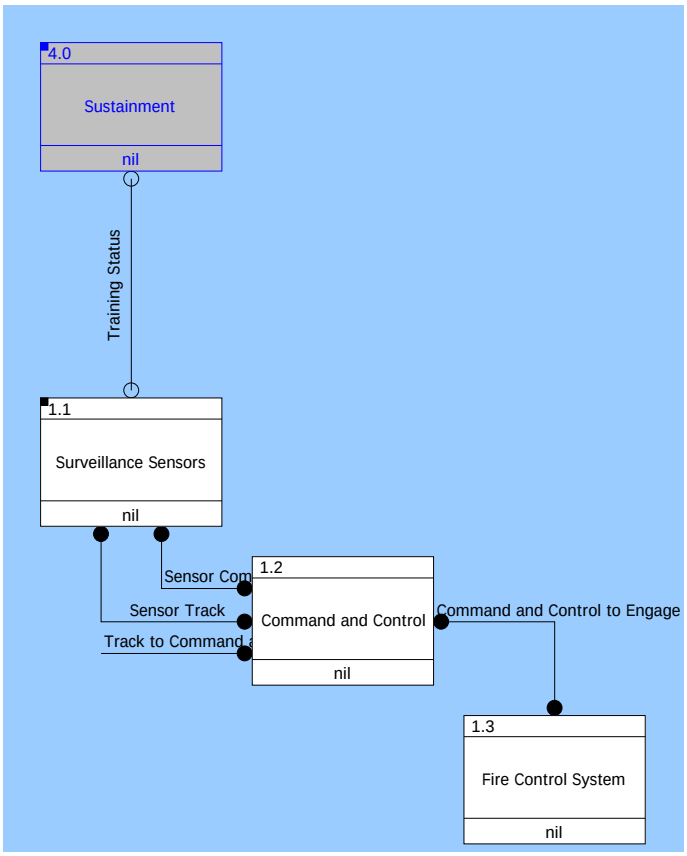


Figure 9 Tactical System Physical Block Diagram

Performs Function(s):
- 2.0 Tracking
- 3.0 Command and Control
- 4.0 Engage

Constrained By Design Constraints:
- D.1.3 Standardization

1.1 Surveillance Sensors [WS 21200]

Description:
Composes System Components required to host and execute Sensor functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
- 1.0 Tactical System [WS 21200]

Built From Lower-Level Component(s):
- 1.1.1 Radar
- 1.1.2 EO/IR [WS21TYR]
- 1.1.3 EW

Connected to Physical Link(s):

10 Systems/Components

Sensor Command and Control
Sensor Track

Connected thru Physical Link(s):
Training Status

Joined To Logical Interface:
Search C2 Interface
Search Track Interface
Training User Interface

Joined Thru Logical Interface:
Sustainment Radar Interface

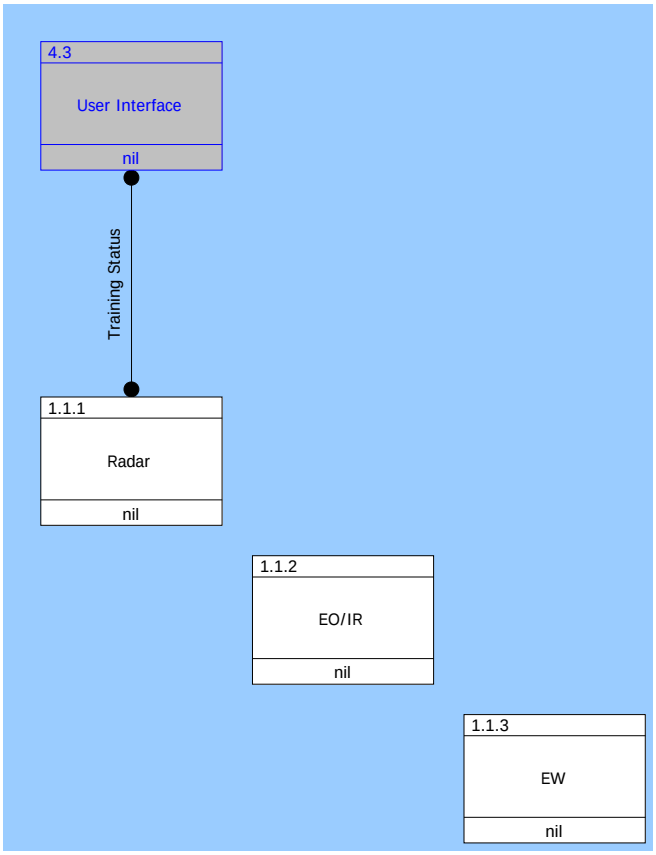


Figure 10 Surveillance Sensors Physical Block Diagram

Performs Function(s):
1.0 Search and Detect
1.1 Horizon Search

1.1.1 Radar

Description:

10 Systems/Components

Composes System Components required to host and execute Radio Frequency transmission and receive Sensor functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
1.1 Surveillance Sensors [WS 21200]

Connected to Physical Link(s):
Training Status

Joined To Logical Interface:
Sustainment Radar Interface

Performs Function(s):
1.1.1 Provide Radar Surveillance
1.3.1 Determine Position

1.1.2 EO/IR [WS21TYR]

Description:
Thermal Imaging processor for Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
1.1 Surveillance Sensors [WS 21200]

1.1.3 EW

Description:
Electronic Warfare component for Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
1.1 Surveillance Sensors [WS 21200]

1.2 Command and Control [WS 21340]

Description:
Composes System Components required to host and execute Track and Command and Control functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
1.0 Tactical System [WS 21200]

Connected to Physical Link(s):
Command and Control to Engage
Sensor Command and Control
Sensor Track
Track to Command and Control

Joined To Logical Interface:
C2 to Engage Interface
Search C2 Interface
Search Track Interface
Training User Interface

10 Systems/Components

Performs Function(s):
2.0 Tracking
3.0 Command and Control
4.0 Engage

1.3 Fire Control System [WS-31333]

Description:
Composes System Components required to host and execute Track and Command and Control functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
1.0 Tactical System [WS 21200]

Connected to Physical Link(s):
Command and Control to Engage

Joined To Logical Interface:
C2 to Engage Interface
Training User Interface

Performs Function(s):
4.0 Engage

11 Interfaces

Part I - Derived Functional Interfaces

Table 1 1.0 Tactical System External I/O

Function	Interface Item	Interfacing Entity
3.0 Command and Control	→ Control Degraded (Item)	5.0 Sustainment4.0 Sustainment [USD ATL] 5.2 Mission Readiness4.2 Maintenance System [CNET]
3.0 Command and Control	→ Control Ready (Item)	5.0 Sustainment4.0 Sustainment [USD ATL] 5.2 Mission Readiness4.2 Maintenance System [CNET]
4.0 Engage	→ Engage Degraded (Item)	5.0 Sustainment4.0 Sustainment [USD ATL] 5.2 Mission Readiness4.2 Maintenance System [CNET]
4.0 Engage	→ Engage Ready (Item)	5.0 Sustainment4.0 Sustainment [USD ATL] 5.2 Mission Readiness4.2 Maintenance System [CNET]

Table 2 1.2 Command and Control External I/O

Function	Interface Item	Interfacing Entity
2.0 Tracking	← Raw Data Items	1.0 Search and Detect1.1 Surveillance Sensors [WS 21200] 1.1.1 Provide Radar Surveillance1.1.1 Radar
3.0 Command and Control	→ Control Degraded (Item)	5.0 Sustainment4.0 Sustainment [USD ATL] 5.2 Mission Readiness4.2 Maintenance System [CNET]
3.0 Command and Control	→ Control Ready (Item)	5.0 Sustainment4.0 Sustainment [USD ATL] 5.2 Mission Readiness4.2 Maintenance System [CNET]
3.0 Command and Control	→ Engage Order (Item)	4.0 Engage1.3 Fire Control System [WS-31333]
3.0 Command and Control	← Composite ID Message (Item)	1.1 Horizon Search1.1 Surveillance Sensors [WS 21200]
3.0 Command and Control	← Engage Ready (Item)	4.0 Engage1.3 Fire Control System [WS-31333]

11 Interfaces

Table 2 1.2 Command and Control External I/O

Function	Interface Item	Interfacing Entity
3.0 Command and Control	← Kill Assessment Request (Item)	4.0 Engage1.3 Fire Control System [WS-31333]
3.0 Command and Control	← Kill Confirm (Item)	4.0 Engage1.3 Fire Control System [WS-31333]
4.0 Engage	→ Engage Degraded (Item)	5.0 Sustainment4.0 Sustainment [USD ATL] 5.2 Mission Readiness4.2 Maintenance System [CNET]
4.0 Engage	→ Engage Ready (Item)	5.0 Sustainment4.0 Sustainment [USD ATL] 5.2 Mission Readiness4.2 Maintenance System [CNET]

Table 3 1.3 Fire Control System External I/O

Function	Interface Item	Interfacing Entity
4.0 Engage	→ Engage Degraded (Item)	5.0 Sustainment4.0 Sustainment [USD ATL] 5.2 Mission Readiness4.2 Maintenance System [CNET]
4.0 Engage	→ Engage Ready (Item)	3.0 Command and Control1.2 Command and Control [WS 21340] 5.0 Sustainment4.0 Sustainment [USD ATL] 5.2 Mission Readiness4.2 Maintenance System [CNET]
4.0 Engage	→ Kill Assessment Request (Item)	3.0 Command and Control1.2 Command and Control [WS 21340]
4.0 Engage	→ Kill Confirm (Item)	3.0 Command and Control1.2 Command and Control [WS 21340]
4.0 Engage	← Engage Order (Item)	3.0 Command and Control1.2 Command and Control [WS 21340]

11 Interfaces

Part II - Interfaces

C2 to Engage Interface

Comprised Of Links:
Command and Control to Engage

Connecting Systems/Components:
1.2 Command and Control [WS 21340]
1.3 Fire Control System [WS-31333]

Combat Systems to C4I

Connecting Systems/Components:
1.0 Tactical System [WS 21200]
SysML [AP 233]

Combat Systems to Ship

Connecting Systems/Components:
1.0 Tactical System [WS 21200]
SysML [AP 233]

Search C2 Interface

Comprised Of Links:
Sensor Command and Control

Connecting Systems/Components:
1.1 Surveillance Sensors [WS 21200]
1.2 Command and Control [WS 21340]

Search Track Interface

Comprised Of Links:
Sensor Track

Connecting Systems/Components:
1.1 Surveillance Sensors [WS 21200]
1.2 Command and Control [WS 21340]

Sustainment Radar Interface

Connecting Systems/Components:
1.1.1 Radar
1.1 Surveillance Sensors [WS 21200]
1.0 Tactical System [WS 21200]
4.0 Sustainment [USD ATL]

11 Interfaces

Training User Interface

Comprised Of Links:
- Personnel Status
- Training Status

Connecting Systems/Components:
- 1.1 Surveillance Sensors [WS 21200]
- 1.2 Command and Control [WS 21340]
- 1.3 Fire Control System [WS-31333]
- 4.1 Training System

Part III - Links

Command and Control to Engage

Transmitted Data:
- Composite ID Message (Item)
- Engage Order (Item)
- Engage Ready (Item)
- Kill Assessment Request (Item)
- No Engage (Item)
- Weapon Status
- Weapons Ready (Item)

Connecting Systems/Components:
- 1.2 Command and Control [WS 21340]
- 1.3 Fire Control System [WS-31333]

Personnel Status

Transmitted Data:
- Data Entry Screen
- Personnel Log On

Connecting Systems/Components:
- 4.1 Training System
- 4.2 Maintenance System [CNET]

Sensor Command and Control

Transmitted Data:
- Altitude (Item)
- Bearing (Item)
- EW Data (Item)
- IR Signature (Item)
- Range (Item)
- Raw Data Items
- Search Degraded (Item)
- Search Ready (Item)

11 Interfaces

Sensor Status (Item)

Connecting Systems/Components:
1.1 Surveillance Sensors [WS 21200]
1.2 Command and Control [WS 21340]

Sensor Track

Transmitted Data:
Altitude (Item)
Bearing (Item)
EW Data (Item)
IR Signature (Item)
Range (Item)
Raw Data Items
Search Sector Message (Item)
Sensor Status (Item)

Connecting Systems/Components:
1.1 Surveillance Sensors [WS 21200]
1.2 Command and Control [WS 21340]

Track to Command and Control

Transmitted Data:
Composite ID Message (Item)
Search Sector Message (Item)
Sensor Status (Item)

Connecting Systems/Components:
1.2 Command and Control [WS 21340]
1.0 Tactical System [WS 21200]
SysML [AP 233]

Training Status

Transmitted Data:
Safe to Train
Scenario Data
Training Enable
Training Status

Connecting Systems/Components:
1.1.1 Radar
1.1 Surveillance Sensors [WS 21200]
1.0 Tactical System [WS 21200]
4.3 User Interface [OJ-394]
4.0 Sustainment [USD ATL]

11 Interfaces

Part IV - Exchange Characteristics

Data Latency

Description:
Provides a measure for Signal Delay

Triggering Event: Queue

Interoperability Level: 1 - Connected

Criticality: 3 - Mission Critical (Core Functions)

Exhibited By:
Horizon Search (CAP)

Operational Latency (EXCHNGE CHARA)

Exhibited By:
AAW (CAP)

12 Functional Model

Part I - Hierarchical Function List

2.0 Tracking
- 2.1 Perform Local Tracking
  - 2.1.1 Perform Single Sensor Tracking
  - 2.1.2 Perform Multiple Sensor Tracking

3.0 Command and Control
- 3.1 Manage Tracks
  - 3.1.1 Assign New Tracks
  - 3.1.2 Monitor Air Tracks
  - 3.1.3 Correlate Air Tracks
  - 3.1.4 Update Air Tracks
  - 3.1.5 Drop Air Tracks
- 3.2 Manage Tactical Control
  - 3.2.1 Manage Information
  - 3.2.2 Provide Display Support for Tactical Operation
  - 3.2.3 Provide Combatant Commanders Decision Aide
  - 3.2.4 Provide Alerts and Prompts
  - 3.2.5 Manage Sub mode
  - 3.2.6 Perform Tactical Control
  - 3.2.7 Perform Initialization
  - 3.2.8 Reconfigure
  - 3.2.9 Perform Readiness Assessment
  - 3.2.10 Manage Ship Sensors in Search, Detection and Track
- 3.3 Plan
- 3.4 Direct

4.0 Engage
- 4.1 Select/Manage Weapons
  - 4.1.1 Designate Electronic Warfare
  - 4.1.2 Designate Fire-and-Forget Weapon
  - 4.1.3 Designate Semi-Active Homing Weapon

Part II - Behavioral Model

2.0 Tracking

Duration: 6.0

Allocated To:
- 1.0 Tactical System [WS 21200]
- 1.2 Command and Control [WS 21340]

12 Functional Model

Implements:
OA.0 Gain and Maintain Air Superiority
Evaluate Integrate Interpret Operational Information

Based On:
A.1.1.1 LAD Track

Table 4 2.0 Tracking Interfacing Items

Interfacing Item	Source / Destination
Composite ID Message (Item)	Input To: 3.0 Command and Control Triggers: 3.0 Command and Control Output From: 1.1 Horizon Search 2.0 Tracking
Raw Data Items	Input To: 1.1 Horizon Search 1.3.1 Determine Position Triggers: 2.0 Tracking Output From: 1.0 Search and Detect 1.1.1 Provide Radar Surveillance 1.1.2 Perform EO/IR Search 1.1.3 Perform EW Search

Consumes Resource(s):
LAN Bandwidth
Acquire Available: true
Amount: 5.0

Operator
Acquire Available: true
Amount: 5.0

RAM Requirements
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 6.0, stream: 1)

12 Functional Model

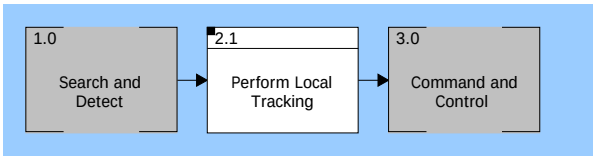


Figure 11 Tracking Enhanced FFBD

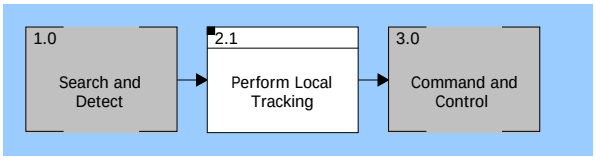


Figure 12 Tracking FFBD

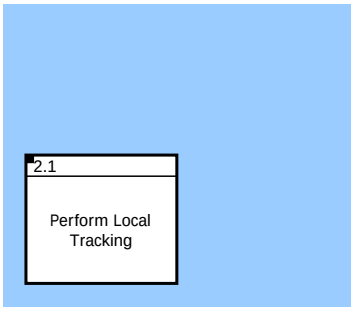


Figure 13 Tracking N2 Diagram

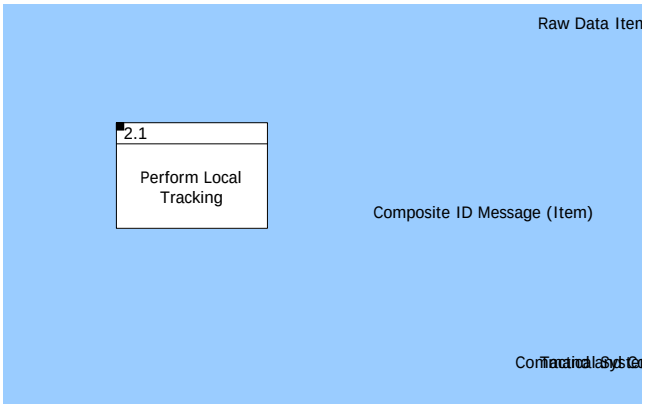


Figure 14 Tracking IDEF0 Diagram

12 Functional Model

2.1 Perform Local Tracking

Captures Resource(s):
Tracks
Acquire Available: true

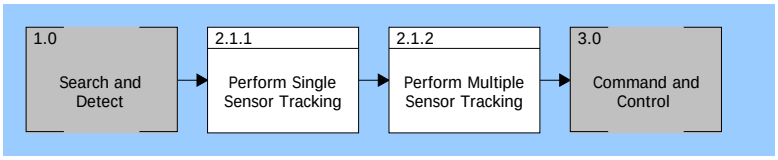


Figure 15 Perform Local Tracking Enhanced FFBD

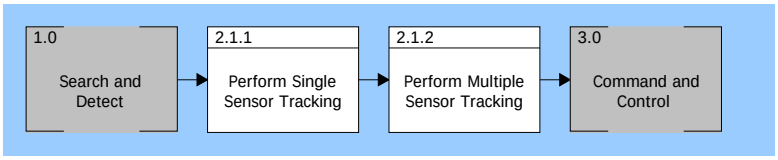


Figure 16 Perform Local Tracking FFBD

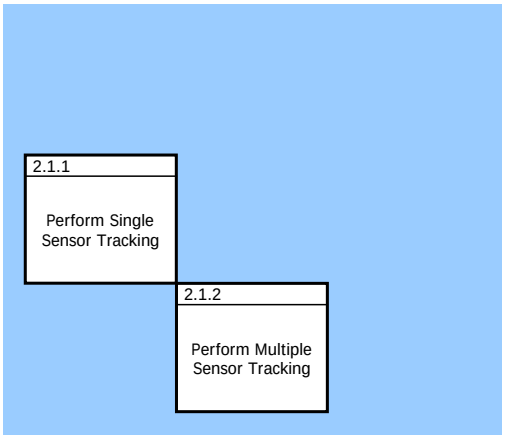


Figure 17 Perform Local Tracking N2 Diagram

12 Functional Model

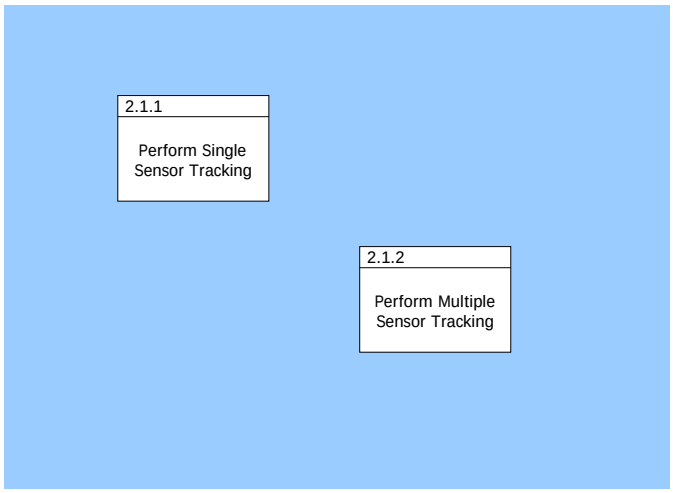


Figure 18 Perform Local Tracking IDEF0 Diagram

2.1.1 Perform Single Sensor Tracking

Duration: Normal (μ: 1.0, stdDev: 1.0, stream: 1)

Consumes Resource(s):

KSLOC
Acquire Available: true
Amount: 1.0

LAN Bandwidth
Acquire Available: true
Amount: 2.0

Operator
Acquire Available: true
Amount: 3.0

RAM Requirements
Acquire Available: true
Amount: 4.0

2.1.2 Perform Multiple Sensor Tracking

Duration: Normal (μ: 2.0, stdDev: 1.0, stream: 1)

Consumes Resource(s):

KSLOC
Acquire Available: true
Amount: 1.0

LAN Bandwidth
Acquire Available: true

12 Functional Model

Amount: 2.0

Operator
Acquire Available: true
Amount: 3.0

RAM Requirements
Acquire Available: true
Amount: 4.0

3.0 Command and Control

Duration: 15.0

Exits:
ROE Met

Allocated To:
1.0 Tactical System [WS 21200]
1.2 Command and Control [WS 21340]

Implements:
OA.0 Gain and Maintain Air Superiority
Engage Tactical Targets
Evaluate Integrate Interpret Operational Information

Based On:
A.1.1.2 LAD Command and Control

Table 5 3.0 Command and Control Interfacing Items

Interfacing Item	Source / Destination
Composite ID Message (Item)	Input To: 3.0 Command and Control Triggers: 3.0 Command and Control Output From: 1.1 Horizon Search 2.0 Tracking
Control Degraded (Item)	Input To: 5.0 Sustainment 5.2 Mission Readiness 5.3 Provide Maintenance Output From: 3.0 Command and Control
Control Ready (Item)	Input To: 5.0 Sustainment 5.2 Mission Readiness

12 Functional Model

Table 5 3.0 Command and Control Interfacing Items

Interfacing Item	Source / Destination
	Output From: 3.0 Command and Control
Engage Order (Item)	Input To: 4.0 Engage Triggers: 4.0 Engage Output From: 3.0 Command and Control
Engage Ready (Item)	Input To: 3.0 Command and Control 5.0 Sustainment 5.2 Mission Readiness Output From: 4.0 Engage
Kill Assessment Request (Item)	Input To: 3.0 Command and Control Triggers: 3.0 Command and Control Output From: 4.0 Engage
Kill Confirm (Item)	Input To: 3.0 Command and Control Triggers: 3.0 Command and Control Output From: 4.0 Engage

Consumes Resource(s):
LAN Bandwidth
Acquire Available: true
Amount: 2.0

Operator
Acquire Available: true
Amount: 7.0

RAM Requirements
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 9.0, stream: 1)

12 Functional Model

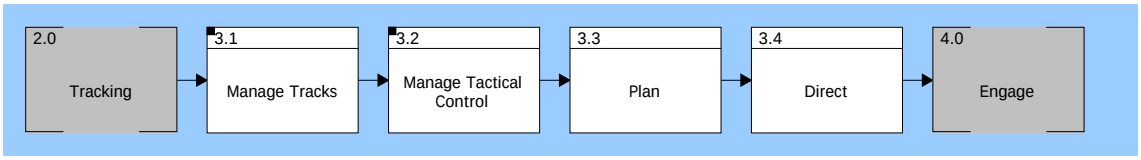


Figure 19 Command and Control Enhanced FFBD

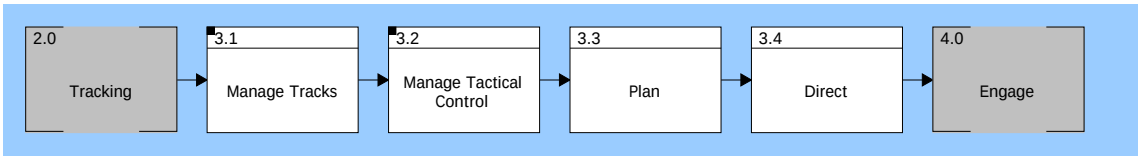


Figure 20 Command and Control FFBD

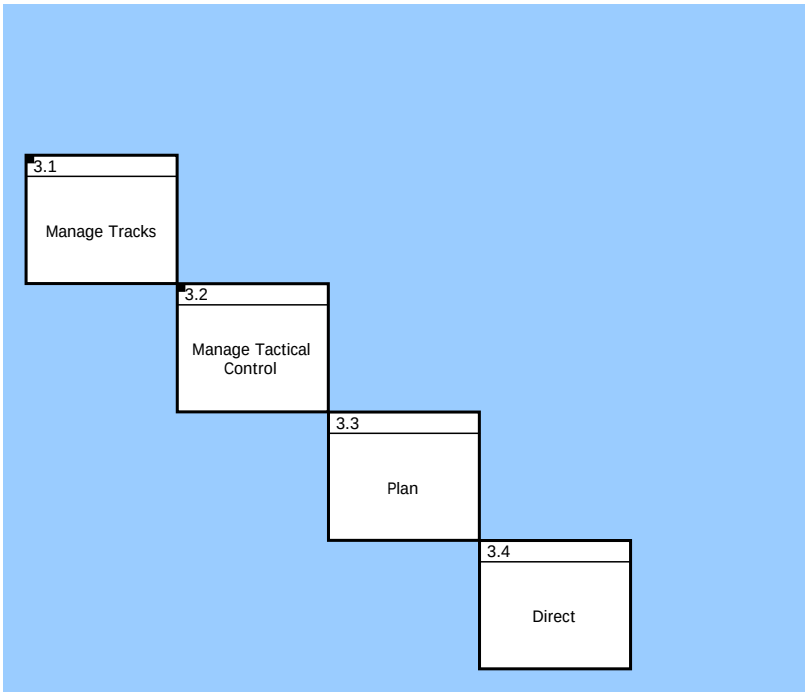


Figure 21 Command and Control N2 Diagram

12 Functional Model

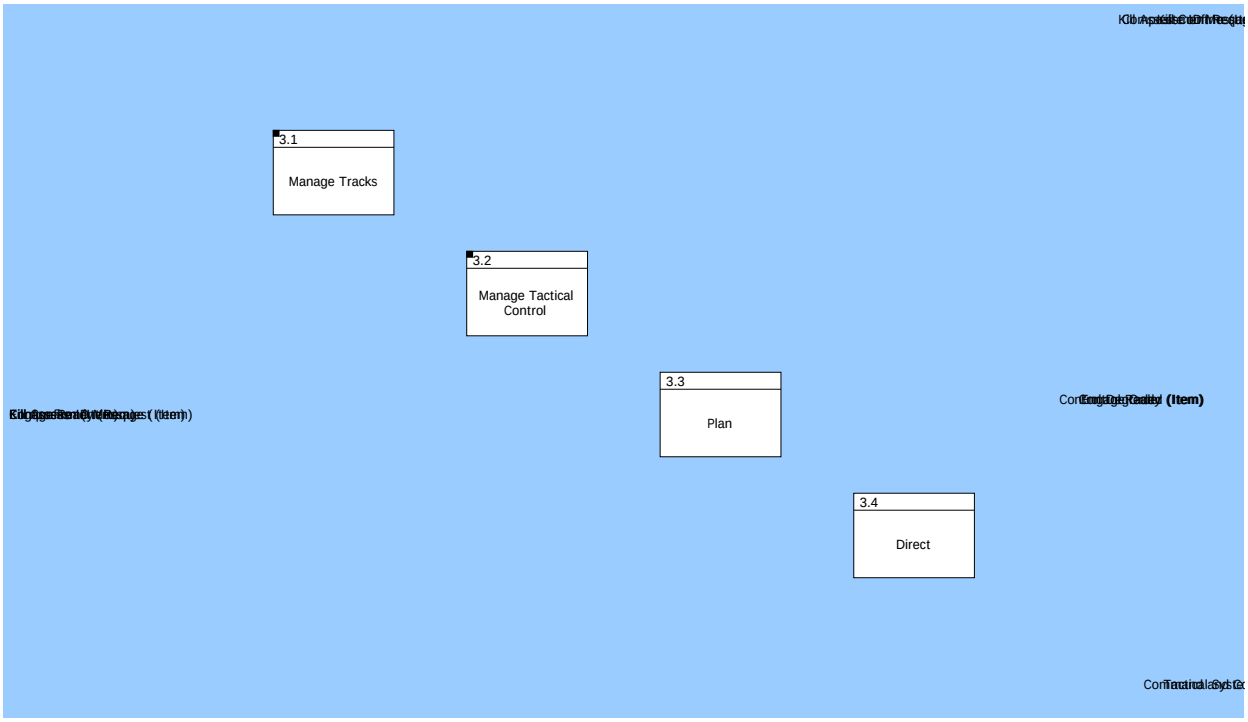


Figure 22 Command and Control IDEF0 Diagram

3.1 Manage Tracks

Implements:

OA.0 Gain and Maintain Air Superiority

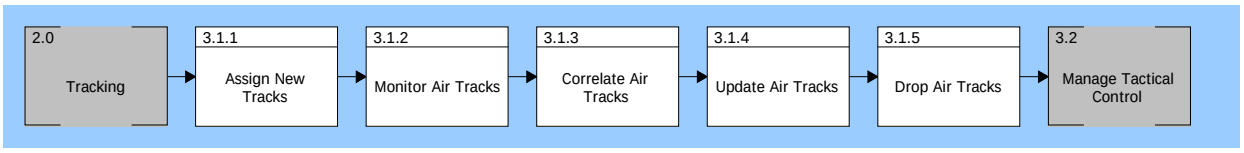


Figure 23 Manage Tracks Enhanced FFBD

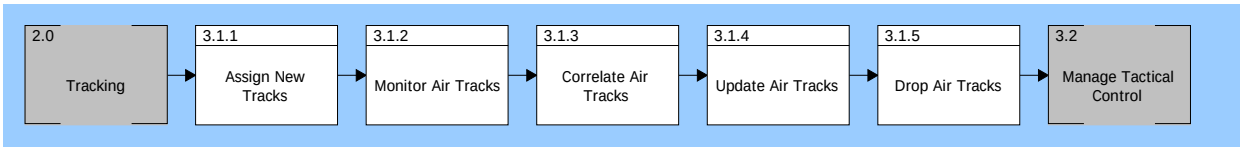


Figure 24 Manage Tracks FFBD

12 Functional Model

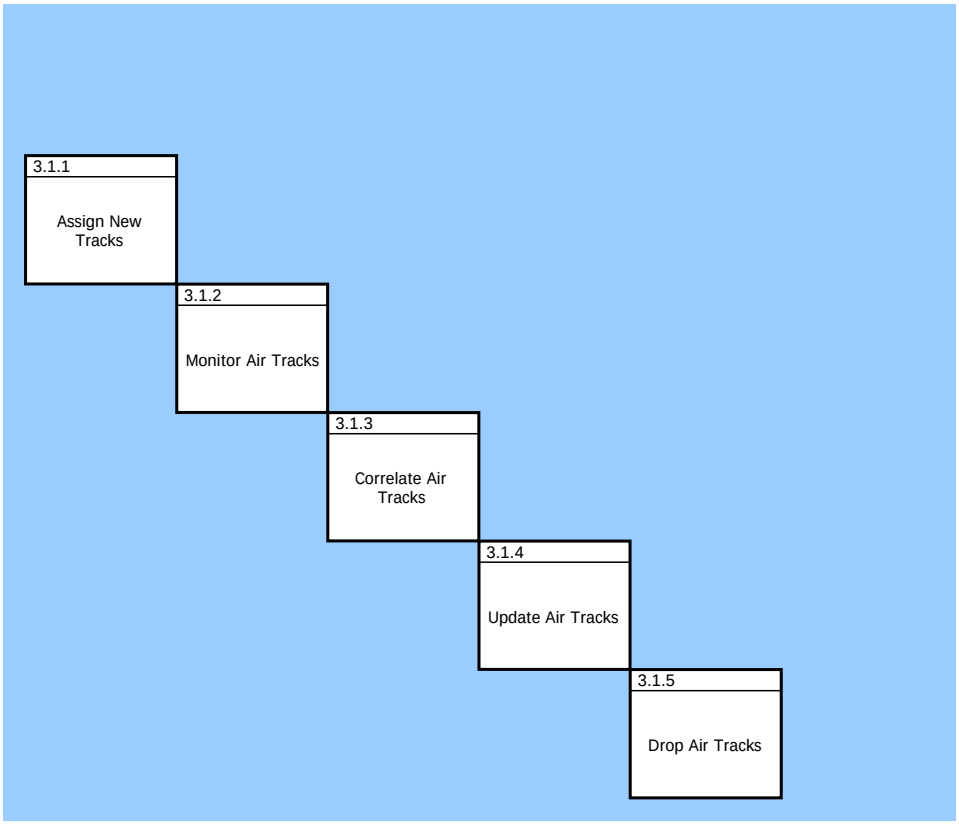


Figure 25 Manage Tracks N2 Diagram

12 Functional Model

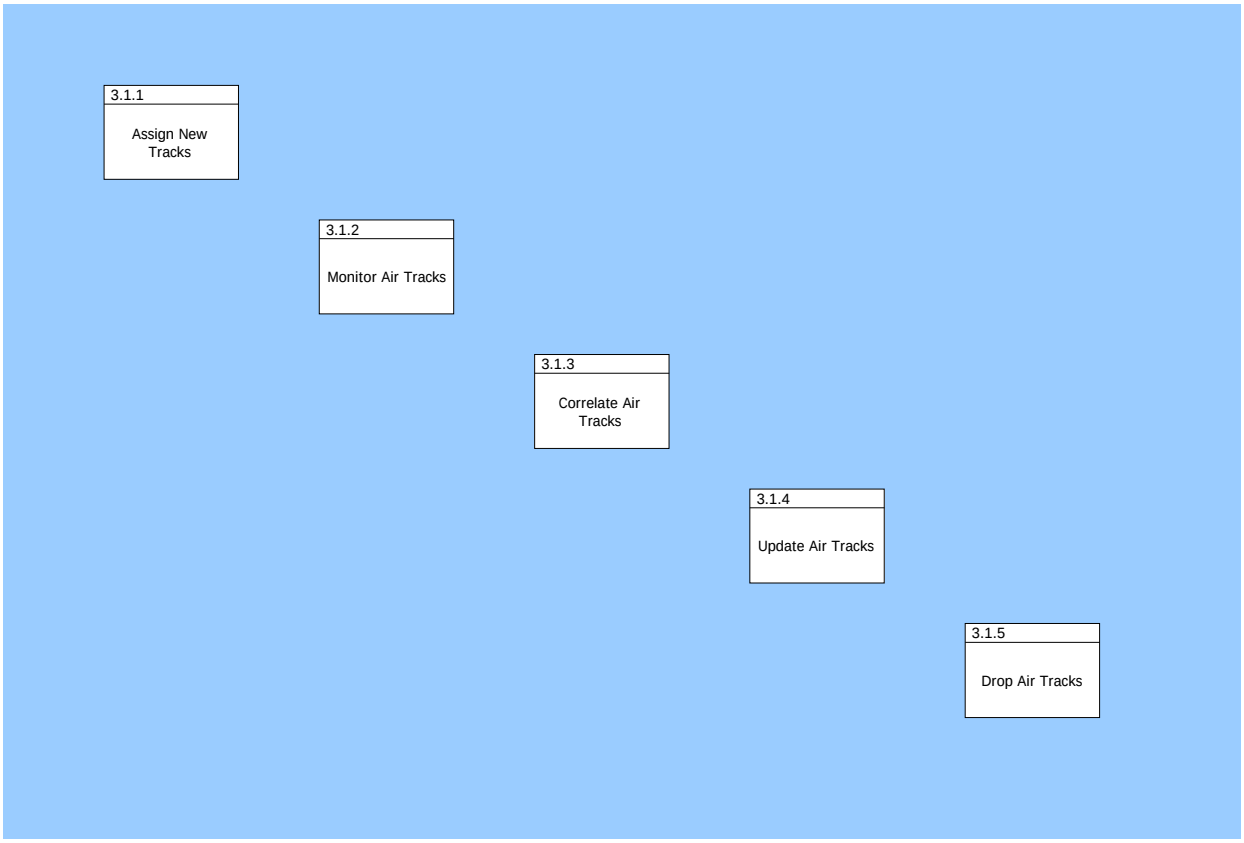


Figure 26 Manage Tracks IDEF0 Diagram

3.1.1 Assign New Tracks

Implements:

OA.0 Gain and Maintain Air Superiority

3.1.2 Monitor Air Tracks

Implements:

OA.0 Gain and Maintain Air Superiority

3.1.3 Correlate Air Tracks

Implements:

OA.0 Gain and Maintain Air Superiority

3.1.4 Update Air Tracks

Implements:

OA.0 Gain and Maintain Air Superiority

12 Functional Model

3.1.5 Drop Air Tracks

Implements:
OA.0 Gain and Maintain Air Superiority

Consumes Resource(s):
Tracks
Acquire Available: true

3.2 Manage Tactical Control

Implements:
OA.0 Gain and Maintain Air Superiority

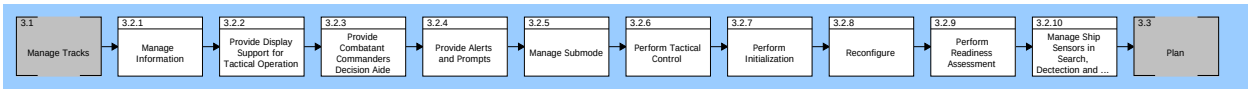


Figure 27 Manage Tactical Control Enhanced FFBD

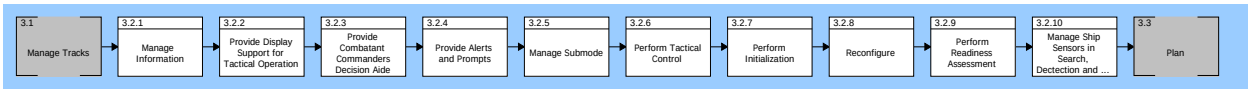


Figure 28 Manage Tactical Control FFBD

12 Functional Model

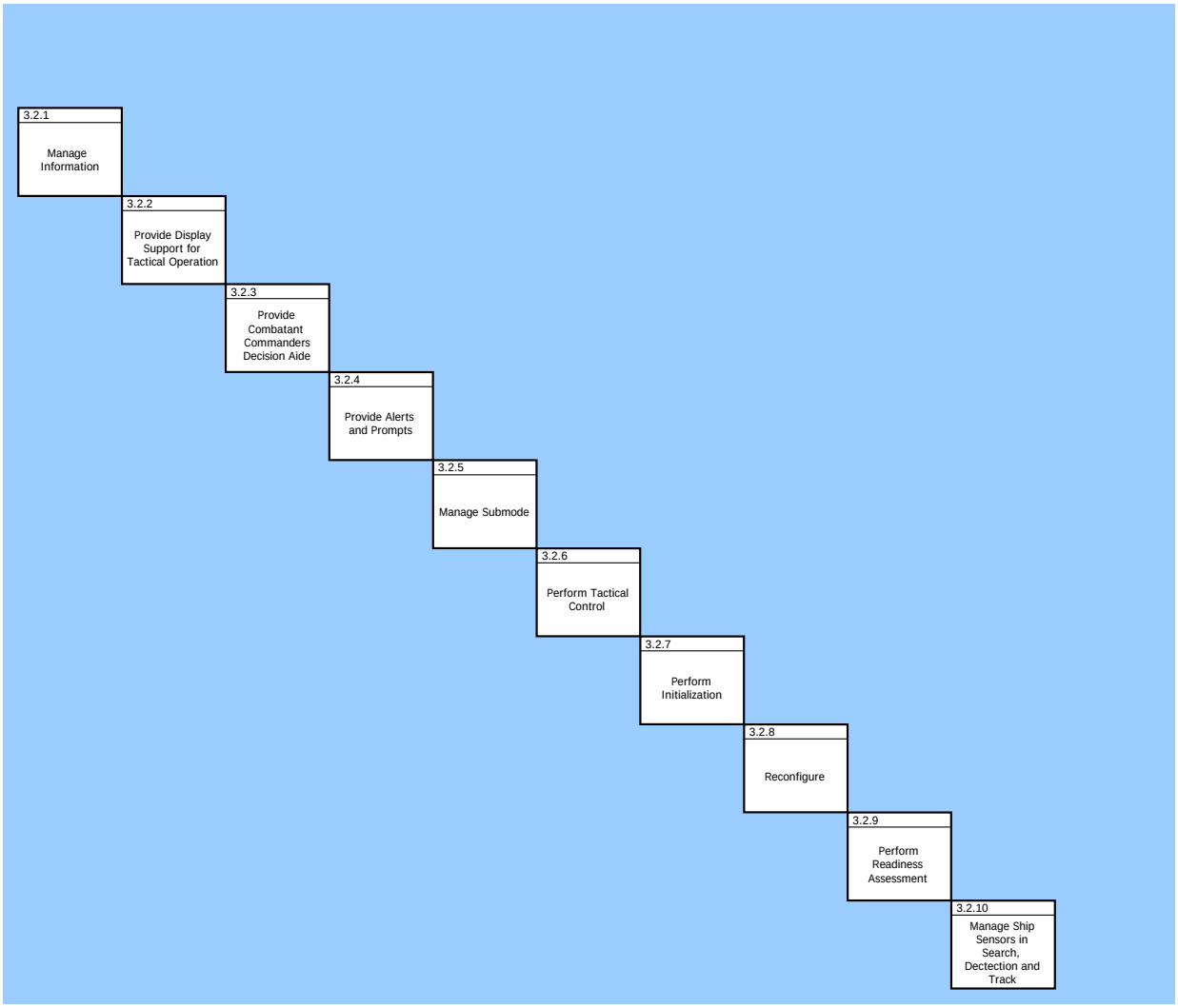


Figure 29 Manage Tactical Control N2 Diagram

12 Functional Model

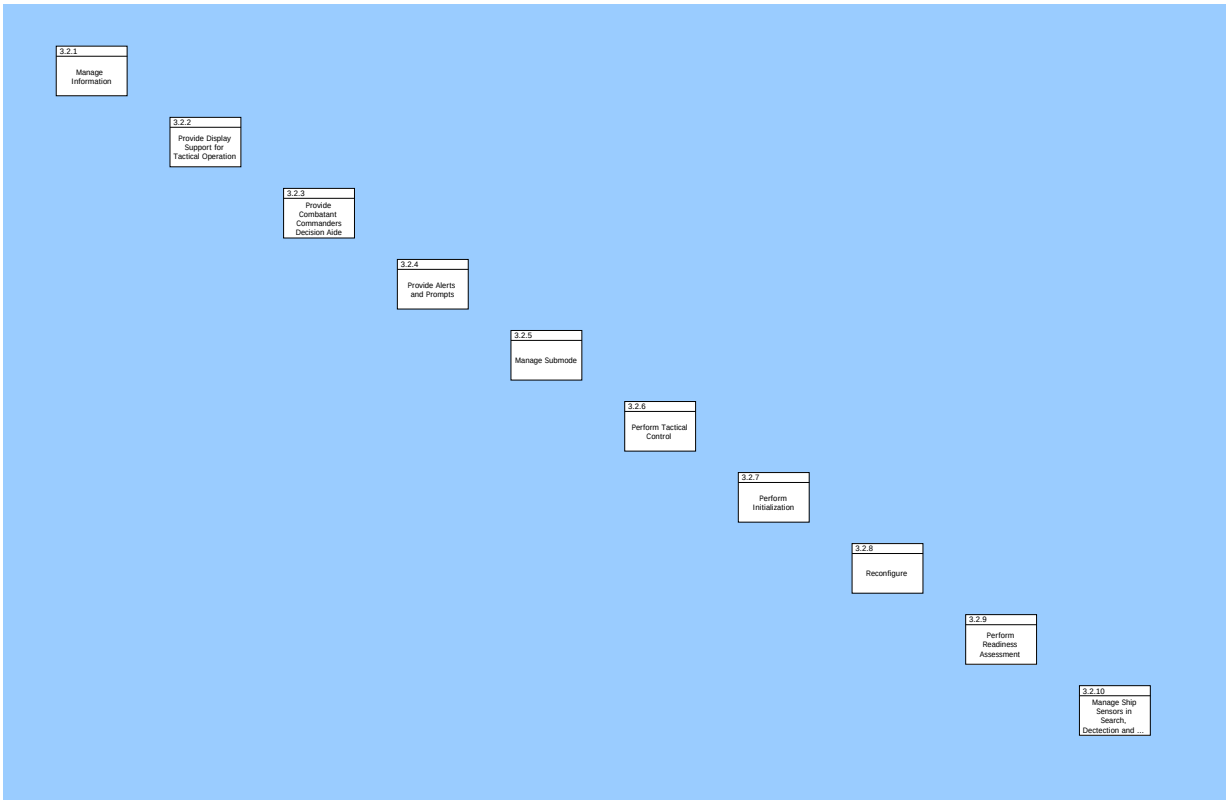


Figure 30 Manage Tactical Control IDEF0 Diagram

3.2.1 Manage Information

Implements:

OA.0 Gain and Maintain Air Superiority

3.2.2 Provide Display Support for Tactical Operation

Implements:

OA.0 Gain and Maintain Air Superiority

3.2.3 Provide Combatant Commanders Decision Aide

Implements:

OA.0 Gain and Maintain Air Superiority

3.2.4 Provide Alerts and Prompts

Implements:

OA.0 Gain and Maintain Air Superiority

12 Functional Model

3.2.5 Manage Sub mode

Implements:
OA.0 Gain and Maintain Air Superiority

3.2.6 Perform Tactical Control

Implements:
OA.0 Gain and Maintain Air Superiority

3.2.7 Perform Initialization

Implements:
OA.0 Gain and Maintain Air Superiority

3.2.8 Reconfigure

Implements:
OA.0 Gain and Maintain Air Superiority

3.2.9 Perform Readiness Assessment

Implements:
OA.0 Gain and Maintain Air Superiority

3.2.10 Manage Ship Sensors in Search, Detection and Track

Implements:
OA.0 Gain and Maintain Air Superiority

3.3 Plan

Implements:
OA.0 Gain and Maintain Air Superiority

3.4 Direct

Implements:
OA.0 Gain and Maintain Air Superiority

4.0 Engage

Duration: 1.0

Exits:
Threat Killed

Allocated To:
1.0 Tactical System [WS 21200]
1.2 Command and Control [WS 21340]
1.3 Fire Control System [WS-31333]

12 Functional Model

Implements:
OA.0 Gain and Maintain Air Superiority
Engage Tactical Targets

Based On:
A.1.1.3 LAD Engage

Table 6 4.0 Engage Interfacing Items

Interfacing Item	Source / Destination
Engage Degraded (Item)	Input To: 5.0 Sustainment 5.2 Mission Readiness 5.3 Provide Maintenance Output From: 4.0 Engage
Engage Order (Item)	Input To: 4.0 Engage Triggers: 4.0 Engage Output From: 3.0 Command and Control
Engage Ready (Item)	Input To: 3.0 Command and Control 5.0 Sustainment 5.2 Mission Readiness Output From: 4.0 Engage
Kill Assessment Request (Item)	Input To: 3.0 Command and Control Triggers: 3.0 Command and Control Output From: 4.0 Engage
Kill Confirm (Item)	Input To: 3.0 Command and Control Triggers: 3.0 Command and Control Output From: 4.0 Engage

12 Functional Model

Consumes Resource(s):
LAN Bandwidth
Acquire Available: true
Amount: 5.0

Operator
Acquire Available: true
Amount: 15.0

RAM Requirements
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 5.0, stream: 1)

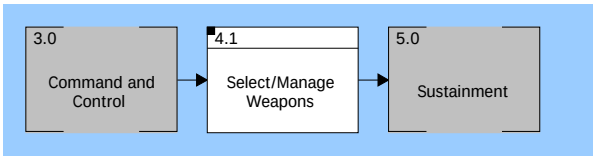


Figure 31 Engage Enhanced FFBD

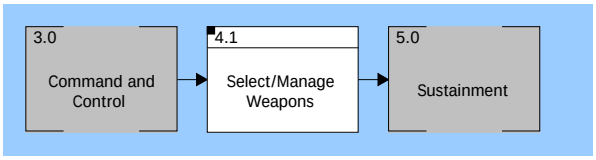


Figure 32 Engage FFBD

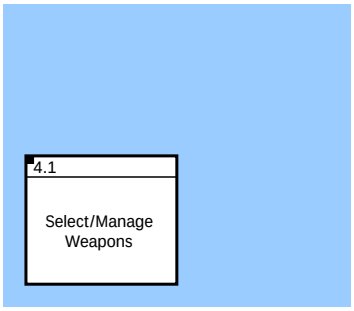


Figure 33 Engage N2 Diagram

12 Functional Model

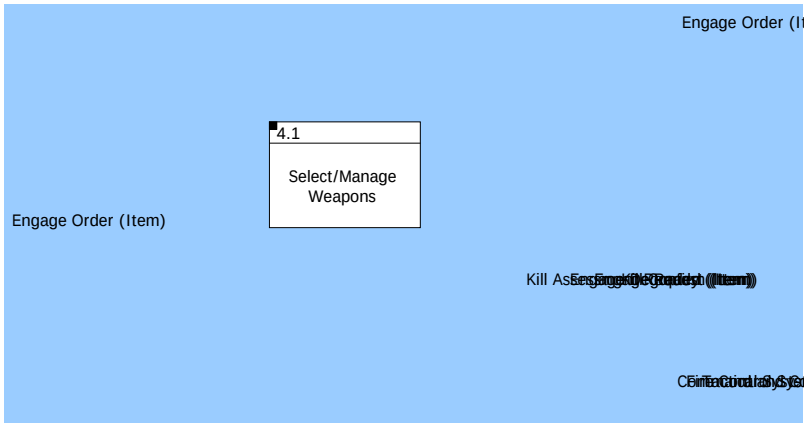


Figure 34 Engage IDEF0 Diagram

4.1 Select/Manage Weapons

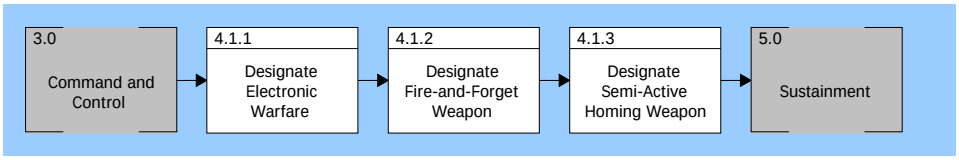


Figure 35 Select/Manage Weapons Enhanced FFBD

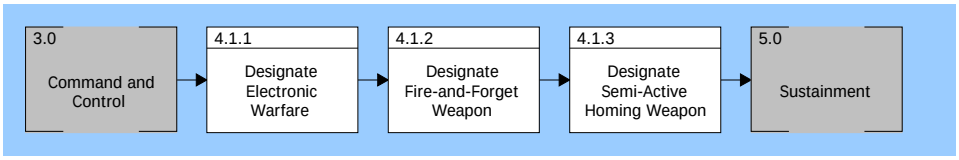


Figure 36 Select/Manage Weapons FFBD

12 Functional Model

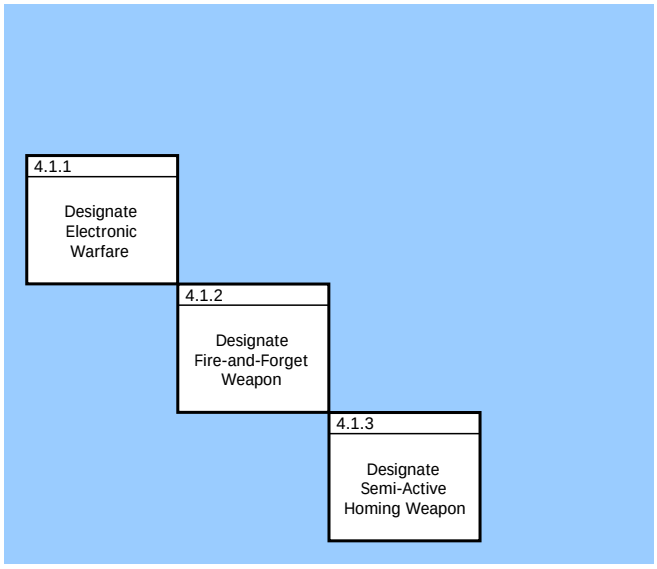


Figure 37 Select/Manage Weapons N2 Diagram

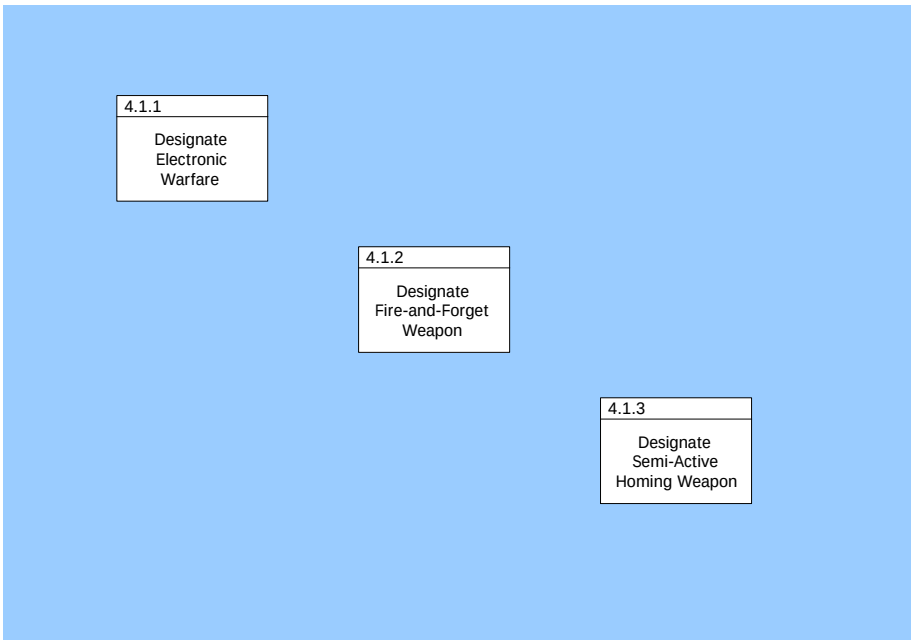


Figure 38 Select/Manage Weapons IDEF0 Diagram

12 Functional Model

4.1.1 Designate Electronic Warfare

Duration: 10.0

4.1.2 Designate Fire-and-Forget Weapon

Duration: 5.0

4.1.3 Designate Semi-Active Homing Weapon

13 Item Dictionary

Altitude (Item)

Transferred By Interface Link:
- Sensor Command and Control
- Sensor Track

Bearing (Item)

Transferred By Interface Link:
- Sensor Command and Control
- Sensor Track

Composite ID Message (Item)

Input To:
- 3.0 Command and Control

Triggers Function(s):
- 3.0 Command and Control

Output From:
- 1.1 Horizon Search
- 2.0 Tracking

Transferred By Interface Link:
- Command and Control to Engage
- Track to Command and Control

Implements Information Exchange:
- Track Correlation (CAP)

Control Degraded (Item)

Input To:
- 5.0 Sustainment
- 5.2 Mission Readiness
- 5.3 Provide Maintenance

Output From:
- 3.0 Command and Control

Transferred By Interface Link:
- Mission Readiness Status

Control Ready (Item)

Input To:
- 5.0 Sustainment
- 5.2 Mission Readiness

Output From:
- 3.0 Command and Control

13 Item Dictionary

Data Entry Screen

Transferred By Interface Link:
Personnel Status

Engage Degraded (Item)

Input To:
5.0 Sustainment
5.2 Mission Readiness
5.3 Provide Maintenance

Output From:
4.0 Engage

Transferred By Interface Link:
Mission Readiness Status

Engage Order (Item)

Input To:
4.0 Engage

Triggers Function(s):
4.0 Engage

Output From:
3.0 Command and Control

Transferred By Interface Link:
Command and Control to Engage

Engage Ready (Item)

Input To:
3.0 Command and Control
5.0 Sustainment
5.2 Mission Readiness

Output From:
4.0 Engage

Transferred By Interface Link:
Command and Control to Engage

EW Data (Item)

Transferred By Interface Link:
Sensor Command and Control
Sensor Track

13 Item Dictionary

IR Signature (Item)

Transferred By Interface Link:
Sensor Command and Control
Sensor Track

Kill Assessment Request (Item)

Input To:
3.0 Command and Control

Triggers Function(s):
3.0 Command and Control

Output From:
4.0 Engage

Transferred By Interface Link:
Command and Control to Engage

Kill Confirm (Item)

Input To:
3.0 Command and Control

Triggers Function(s):
3.0 Command and Control

Output From:
4.0 Engage

No Engage (Item)

Transferred By Interface Link:
Command and Control to Engage

Personnel Log On

Transferred By Interface Link:
Personnel Status

Range (Item)

Output From:
1.3.1 Determine Position

Transferred By Interface Link:
Sensor Command and Control
Sensor Track

Raw Data Items

Input To:

13 Item Dictionary

1.1 Horizon Search
1.3.1 Determine Position

Triggers Function(s):
2.0 Tracking

Output From:
1.0 Search and Detect
1.1.1 Provide Radar Surveillance
1.1.2 Perform EO/IR Search
1.1.3 Perform EW Search

Transferred By Interface Link:
Sensor Command and Control
Sensor Track

Safe to Train

Input To:
5.0 Sustainment

Transferred By Interface Link:
Training Status

Scenario Data

Transferred By Interface Link:
Training Status

Search Degraded (Item)

Input To:
5.0 Sustainment
5.2 Mission Readiness
5.3 Provide Maintenance

Output From:
1.0 Search and Detect

Transferred By Interface Link:
Sensor Command and Control

Search Ready (Item)

Input To:
5.0 Sustainment
5.2 Mission Readiness

Output From:
1.0 Search and Detect

Transferred By Interface Link:
Sensor Command and Control

13 Item Dictionary

Implements Information Exchange:
Horizon Search (CAP)
Limited Area (CAP)

Search Sector Message (Item)

Transferred By Interface Link:
Sensor Track
Track to Command and Control

Implements Information Exchange:
Limited Area (CAP)

Sensor Status (Item)

Transferred By Interface Link:
Sensor Command and Control
Sensor Track
Track to Command and Control

Training Enable

Output From:
5.0 Sustainment

Transferred By Interface Link:
Training Status

Training Status

Output From:
5.0 Sustainment

Transferred By Interface Link:
Training Status

Weapon Status

Output From:
5.2 Mission Readiness

Transferred By Interface Link:
Command and Control to Engage
Mission Readiness Status

Weapons Ready (Item)

Transferred By Interface Link:
Command and Control to Engage

Implements Information Exchange:
AAW (CAP)

13 Item Dictionary

Kill (CAP)
Limited Area (CAP)

14 Functional Model Resources

KSLOC

Amount Type: Float

Initial Amount: Normal (μ: 200.0, stdDev: 1.0, stream: 1)

Maximum Amount: Normal (μ: 200.0, stdDev: 1.0, stream: 1)

Amount Units: Lines of code in 1000s

Consumed By:

1.1.1 Provide Radar Surveillance
Acquire Available: true
Amount: 1.0

1.1.2 Perform EO/IR Search
Acquire Available: true
Amount: 3.0

1.1.3 Perform EW Search
Acquire Available: true
Amount: 5.0

1.2.2 Perform Manual Search
Acquire Available: true
Amount: 10.0

1.3.1 Determine Position
Acquire Available: true
Amount: 18.0

1.3.2 Provide Track Data
Acquire Available: true
Amount: 10.0

2.1.1 Perform Single Sensor Tracking
Acquire Available: true
Amount: 1.0

2.1.2 Perform Multiple Sensor Tracking
Acquire Available: true
Amount: 1.0

LAN Bandwidth

Amount Type: Integer

Initial Amount: Normal (μ: 20.0, stdDev: 1.0, stream: 1)

Maximum Amount: Normal (μ: 20.0, stdDev: 1.0, stream: 1)

14 Functional Model Resources

Amount Units: Seconds

Consumed By:

1.0 Search and Detect
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 7.0, stream: 1)

2.0 Tracking
Acquire Available: true
Amount: 5.0

2.1.1 Perform Single Sensor Tracking
Acquire Available: true
Amount: 2.0

2.1.2 Perform Multiple Sensor Tracking
Acquire Available: true
Amount: 2.0

3.0 Command and Control
Acquire Available: true
Amount: 2.0

4.0 Engage
Acquire Available: true
Amount: 5.0

5.0 Sustainment
Acquire Available: true
Amount: 8.0

Assess Operational Situation
Acquire Available: true
Amount: Normal (μ: 5.0, stdDev: 1.0, stream: 1)

Attack Operational Targets
Acquire Available: true
Amount: Normal (μ: 4.0, stdDev: 1.0, stream: 1)

Collect Information on Operational Situation
Acquire Available: true
Amount: Normal (μ: 2.0, stdDev: 1.0, stream: 1)

Engage Tactical Targets
Acquire Available: true
Amount: Normal (μ: 2.0, stdDev: 1.0, stream: 1)

Evaluate Integrate Interpret Operational Information

14 Functional Model Resources

Acquire Available: true
Amount: Normal (μ: 5.0, stdDev: 1.0, stream: 1)

Provide for Combat Identification
Acquire Available: true
Amount: Normal (μ: 3.0, stdDev: 1.0, stream: 1)

Operator

Amount Type: Float

Initial Amount: Normal (μ: 120.0, stdDev: 1.0, stream: 1)

Maximum Amount: Normal (μ: 120.0, stdDev: 1.0, stream: 1)

Amount Units: Operator interface time requirements in seconds

Consumed By:
1.0 Search and Detect
Acquire Available: true
Amount: 10.0

1.1.1 Provide Radar Surveillance
Acquire Available: true
Amount: Normal (μ: 2.0, stdDev: 1.0, stream: 1)

1.1.2 Perform EO/IR Search
Acquire Available: true
Amount: Normal (μ: 5.0, stdDev: 1.0, stream: 1)

1.1.3 Perform EW Search
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 1.0, stream: 1)

1.2.2 Perform Manual Search
Acquire Available: true
Amount: Normal (μ: 12.0, stdDev: 1.0, stream: 1)

1.3.1 Determine Position
Acquire Available: true
Amount: Normal (μ: 12.0, stdDev: 1.0, stream: 1)

1.3.2 Provide Track Data
Acquire Available: true
Amount: Normal (μ: 5.0, stdDev: 1.0, stream: 1)

2.0 Tracking
Acquire Available: true
Amount: 5.0

14 Functional Model Resources

2.1.1 Perform Single Sensor Tracking
Acquire Available: true
Amount: 3.0

2.1.2 Perform Multiple Sensor Tracking
Acquire Available: true
Amount: 3.0

3.0 Command and Control
Acquire Available: true
Amount: 7.0

4.0 Engage
Acquire Available: true
Amount: 15.0

5.0 Sustainment
Acquire Available: true
Amount: 1.0

RAM Requirements

Amount Type: Float

Initial Amount: Normal (μ: 64.0, stdDev: 1.0, stream: 1)

Maximum Amount: Normal (μ: 64.0, stdDev: 1.0, stream: 1)

Amount Units: Mb

Consumed By:
1.0 Search and Detect
Acquire Available: true
Amount: 5.0

1.1.1 Provide Radar Surveillance
Acquire Available: true
Amount: Normal (μ: 3.0, stdDev: 1.0, stream: 1)

1.1.2 Perform EO/IR Search
Acquire Available: true
Amount: Normal (μ: 1.0, stdDev: 1.0, stream: 1)

1.1.3 Perform EW Search
Acquire Available: true
Amount: Normal (μ: 3.0, stdDev: 1.0, stream: 1)

1.2.2 Perform Manual Search
Acquire Available: true
Amount: Normal (μ: 4.0, stdDev: 1.0, stream: 1)

14 Functional Model Resources

1.3.1 Determine Position
Acquire Available: true
Amount: Normal (μ: 6.0, stdDev: 1.0, stream: 1)

1.3.2 Provide Track Data
Acquire Available: true
Amount: Normal (μ: 4.0, stdDev: 1.0, stream: 1)

2.0 Tracking
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 6.0, stream: 1)

2.1.1 Perform Single Sensor Tracking
Acquire Available: true
Amount: 4.0

2.1.2 Perform Multiple Sensor Tracking
Acquire Available: true
Amount: 4.0

3.0 Command and Control
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 9.0, stream: 1)

4.0 Engage
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 5.0, stream: 1)

5.0 Sustainment
Acquire Available: true
Amount: 4.0

Tracks

Amount Type: Float

Initial Amount: 0.0

Maximum Amount: 50.0

Amount Units: Number of Tracks in System

Captured By:
2.1 Perform Local Tracking
Acquire Available: true

Assess Operational Situation
Acquire Available: true

14 Functional Model Resources

Consumed By:

3.1.5 Drop Air Tracks

Acquire Available: true

Produced By:

1.1.1 Provide Radar Surveillance

15 Issues & Decisions

Part I - Open Issues

Command and Control Issue

Originator: Administrator

Originating Date: Tuesday, January 13, 2009 at 03:45:59 PM

Severity: Critical

Status: Open

Generated By:
VerificationRequirement: Contractor Demonstration

Maintenance Issue

Originator: Administrator

Originating Date: Tuesday, January 13, 2009 at 03:46:19 PM

Severity: Critical

Status: Open

Track Issue

Originator: Administrator

Originating Date: Tuesday, January 13, 2009 at 03:46:09 PM

Severity: Critical

Status: Open

Part II - Closed Issues

None

Part III - Rejected Issues

None

16 Risks

AAW Mission Risk

Caused By:
Verification Requirement: Contractor Demonstration

Sustainability Risk

17 Acronyms

Acronym	Definition
AAW	Anti-Air Warfare
AoA	Analysis of Alternatives
BIT	Built-In-Test
C4I	Command, Control, Communications, Computers, & Intelligence
CBA	Capabilities Based Document
CM	Configuration Management
Context sensitive modeling (Clymer)	Context Sensitive Modeling
COTS	Commercial-Off-The-Shelf
DDG (1000)	Destroyer, Guided Missile, Next Generation
DoD	Department of Defense
DODAF	Department of Defense Architecture Framework
Domain analysis	Domain analysis
EFFBD	Enhanced Functional Flow Block Diagram
FAA	Functional Area Analysis
FFBD	Functional Flow Block Diagram
FNA	Functional Needs Analysis
FSA	Functional Solutions Analysis
GPR	Government Purpose Rights
ICD	Initial Capabilities Document
IDEF0	Integration Definition for Function Modeling
IDFM	Integrated Definition for Function Modeling
INCOSE	International Council on Systems Engineering
IPPD	Integrated Product and Process Development
IPR	In Process Review
IPT	Integrated Product Team
IWS	Integrated Warfare Systems
JCIDS	Joint Capabilities Integration Development System
KPP	Key Performance Parameter
LCC	Life Cycle Cost
LCC	Life Cycle Cost Estimate
LCCE	Life Cycle Support Plan
M&S	Modeling and Simulation
MIL-STD	Military Standard
Model-based systems engineering	Model-based systems engineering

17 Acronyms

MOE	Measure Of Effectiveness
MOP	Measure Of Performance
MORS	The Modular Command Evaluation Structure (MORS)
MSSE	Masters of Science in Systems Engineering
MSSEM	Masters of Science in Systems Engineering Management
NAVSEA	Naval Sea Systems Command
Net-centric architectures	Net-Centric Architectures
NOA	Naval Open Architecture
NPS	Naval Postgraduate School
NSWC	Naval Surface Warfare Center
OA	Open Architecture
OPNAV	Office of the Chief of Naval Operations
PBL	Performance Based Logistics
PEO	Program Executive Office; Program Executive Officer
PESHE	Programmatic Environmental, Safety, and Health Evaluation
PHD	Port Hueneme Division
PIA	Post Independent Analysis
PMP	Project Management Plan
POC	Point of Contact
Reliability Theory	Reliability Theory
SE	Systems Engineering
SOA	Service Oriented Architecture
SPAWAR	Space and Naval Warfare
SPL	Software Product Line
SSDS	Ship Self Defense System
Supportability	Supportability
Systems Architecture and Requirements Engineering (Hatley et al)	Systems Architecture and Requirements Engineering (Hatley et al)
The Systems Engineering "VEE" model	VEE Model
UML	Unified Modeling Language

18 Glossary

Term	Definition
Analysis of Alternatives	The evaluation of the performance, operational effectiveness, operational suitability, and estimated costs of alternative systems to meet a mission capability. The analysis assesses the advantages and disadvantages of alternatives being considered to satisfy capabilities, including the sensitivity of each alternative to possible changes in key assumptions or variables. The AoA is normally conducted during the Concept Refinement phase of the Defense Acquisition Framework to refine the system concept contained in the Initial Capabilities Document (ICD) approved at the Concept Decision. (DoDI 5000.2 and CJCSI 3170.01F)
Anti-Air Warfare	A US Navy/US Marine Corps term used to indicate that action required to destroy or reduce to an acceptable level the enemy air and missile threat. It includes such measures as the use of interceptors, bombers, antiaircraft guns, surface-to-air and air-to-air missiles, electronic attack and destruction of the air or missile threat both before and after it is launched.
Built-In-Test	An integral capability of the mission system or system which provides an automated test capability to detect, diagnose, or isolate failures.
Capabilities Based Document	Document that is based on capabilities of the system (could not find a good definition)
Command, Control, Communications, Computers, & Intelligence	*C4I Systems* provides the latest information on the systems that support the command and control of operations, from strategic command to tactical battle management, of countries all around the world. Each entry provides detailed and accurate descriptions of the system or equipment, development and operational status, technical specifications for appraisal and comparison, and manufacturer details to aid procurement.
Commercial-Off-The-Shelf	Commercial items that require no unique government modifications or maintenance over the life cycle of the product to meet the needs of the procuring agency.
Configuration Management	The technical and administrative direction and surveillance actions taken to identify and document the functional and physical characteristics of a Configuration Item (CI), to control changes to a CI and its characteristics, and to record and report change processing and implementation status. It provides a complete audit trail of decisions and design modifications.
Department of Defense	DoD is the federal department charged with coordinating and supervising all agencies and functions of the government relating directly to national security and the military. The organization and functions of the DOD are set forth in Title 10 of the United States Code.
Department of Defense Architecture Framework	The Department of Defense Architecture Framework (DoDAF) defines how to organize the specification of enterprise architectures for U.S. Department of Defense (DoD) applications. All major DoD weapons and information technology system procurements are required to document their enterprise architectures using the view products prescribed by the DoDAF. DoDAF is well suited to large systems and systems-of-systems (SoSs) with complex integration and interoperability issues.

18 Glossary

Destroyer, Guided Missile, Next Generation	Developed under the DD(X) destroyer program, DDG-1000 Zumwalt is the lead ship in a class of next-generation, multi-mission surface combatants tailored for land attack and littoral dominance, with capabilities designed to defeat current and projected threats as well as improve battle force defense.
Domain analysis	Domain analysis is "the process of identifying, collecting, organizing, and representing the relevant information in a domain, based upon the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory, and emerging technology within a domain. Def from CMU/SEI-90-TR-21
Enhanced Functional Flow Block Diagram	Functional Flow Block Diagrams provide a hierarchical decomposition of the system's functions and show a control structure that dictates the order in which the functions can be executed at each level of decomposition. The enhanced FFBD enables you to see how the inputs and output affect the functional sequencing. The Engineering Design of Systems, Dennis Buede, DoDAF,Steven Dam
Functional Area Analysis	Identifies the mission area or mission problem to be assessed, the concepts to be examined, the timeframe in which the problem is being assessed, and the scope of the assessment, and describes the relevant objectives and concept of operations (ConOps) or concepts and the relevant effects to be generated. (CJCSI 3170.01F)
Functional Flow Block Diagram	Shows functional flow including control logic. FFBD is a multi-tier, time-sequenced, step-by-step flow diagram of a system's functional flow
Functional Needs Analysis	Assesses the ability of the current and programmed warfighting systems to deliver the capabilities the Functional Area Analysis (FAA) identified under the full range of operating conditions and to the designated measures of effectiveness. The FNA produces a list of capability gaps that require solutions and indicates the time frame in which those solutions are needed. It may also identify redundancies in capabilities that reflect inefficiencies. (CJCSI 3170.01F)
Functional Solutions Analysis	Operationally based assessment of all potential Doctrine, Organization, Training, Materiel, Leadership, Personnel, and Facilities (DOTMLPF) approaches to solving (or mitigating) one or more of the capability gaps (needs) previously identified.
Government Purpose Rights	In Defense Department acquisitions, the resulting contract can permit delivery of technical data and computer software using a "middle way," known as Government Purpose Rights, which is an Intellectual Property licensing system that is available to DOD acquisitions. Government Purpose Rights ("GPR") lie somewhere between the broad Unlimited Rights license rights allowing unrestricted Government release of information and the more restrictive Limited or Restricted Rights licensing rights that forbid most releases outside the Government.
In Process Review	Interim Program or Progress Review

18 Glossary

Initial Capabilities Document	Documents the need for a materiel approach, or an approach that is a combination of materiel and non-materiel, to satisfy specific capability gap(s). The ***ICD*** defines the gap in terms of the functional area; the relevant range of military operations; desired effects; time and Doctrine, Organization, Training, Materiel, Leadership and education, Personnel, and Facilities (DOTMLPF); and policy implications and constraints. The outcome of an ***ICD*** could be one or more DOTMLPF Change Recommendations (DCRs) or Capability Development Documents. (CJCSI 3170.01F)
Integrated Definition for Function Modeling	IDEFO models the decisions, actions and activities of a system in order to communicate the functional perspective of the system.
Integrated Product and Process Development	IPPD is the DoD management technique that simultaneously integrates all essential acquisition activities through the use of multidisciplinary teams to optimize design, manufacturing, and supportability processes. One of the key IPPD tenets is multidisciplinary teamwork through Integrated Product Teams.
Integrated Product Team	IPTs are an integral part of the Defense acquisition oversight and review process. For Acquisition Category ID and IAM programs, there are generally two levels of IPT: the Overarching Integrated Product Team and the Working-level Integrated Product Team(s). Each program should have an OIPT and at least one WIPT. WIPTs should focus on a particular topic such as cost/performance, test, or contracting. An Integrating Integrated Product Team (IIPT), which is itself a WIPT, should coordinate WIPT efforts and cover all topics not otherwise assigned to another IPT. IPT participation is the primary way for any organization to participate in the acquisition program. DAU
Integrated Warfare Systems	Systems that delivers Enterprise solutions for Naval warfare systems that operate seamlessly and effectively within the Fleet and Joint Force.
Integration Definition for Function Modeling	Integration Definition for Function Modeling
International Council on Systems Engineering	INCOSE is a not-for-profit membership organization founded in 1990. The mission is to advance the state of the art and practice of systems engineering in industry, academia, and government by promoting interdisciplinary, scalable approaches to produce technologically appropriate solutions that meet societal needs.
Joint Capabilities Integration Development System	The **Joint Capabilities Integration Development System**, or **JCIDS**, is the formal United States Department of Defense (DoD) procedure which defines acquisition requirements and evaluation criteria for future defense programs. JCIDS was created to replace the previous service-specific requirements generation system, which allegedly created redundancies in capabilities and failed to meet the combined needs of all US military services. In order to correct these problems, JCIDS is intended to guide the development of requirements for future acquisition systems to reflect the needs of all four services (Army, Navy, Marines, and Air Force) by focusing the requirements generation process on needed *capabilities* as requested or defined by one of the US combatant commanders. In the JCIDS process, regional and functional combatant commanders give feedback early in the development process to ensure that their requirements are met. (Wikipedia)

18 Glossary

Key Performance Parameter	Those attributes or characteristics of a system that are considered critical or essential to the development of an effective military capability and those attributes that make a significant contribution to the key characteristics as defined in the Joint Operations Concept. KPPs are validated by the Joint Requirements Oversight Council (JROC) for JROC Interest documents, and by the DoD Component for Joint Integration or Independent documents. The Capability Development Document (CDD) and the Capability Production Document (CPD) KPPs are included verbatim in the Acquisition Program Baseline (APB). (CJCSI 3170.01E)
Life Cycle Cost	The total cost to the government of acquisition and ownership of that system over its useful life. It includes the cost of development, acquisition, operations, and support (to include manpower), and where applicable, disposal. For defense systems, LCC is also called Total Ownership Cost (TOC).
Life Cycle Cost Estimate	Estimate of the cost of a program from cradle to grave
Life Cycle Support Plan	The total phases through which an item passes from the time it is initially developed until the time it is either consumed in use or disposed of as being excess to all known materiel requirements. The plan covers the entire period.
Masters of Science in Systems Engineering	A rigorous study and an Advanced degree for Systems Engineering offered by the Naval Post Graduate School
Masters of Science in Systems Engineering Management	A rigorous study and an Advanced degree for Systems Engineering Management that incorporates the Systems Engineering and Supportability aspects of a System offered by the Naval Post Graduate School
Measure Of Effectiveness	Measure designed to correspond to accomplishment of mission objectives and achievement of desired results. (CJCSI 3170.01E) MOEs may be further decomposed into Measures of Performance and Measures of Suitability. See operational effectiveness, Measure of Performance, operational suitability, and Measure of Suitability.
Measure Of Performance	Measure of a system s performance expressed as speed, payload, range, time on station, frequency, or other distinctly quantifiable performance features. Several MOPs and/or Measures of Suitability may be related to the achievement of a particular Measure of Effectiveness (MOE). See Measure of Suitability, operational suitability, and Measure of Effectiveness.
Military Standard	A United States **Defense Standard**, often called a **military standard**, "**MIL-STD**", "**MIL-SPEC**", or (informally) "**MilSpecs**", is used to help achieve standardization objectives by the U.S. Department of Defense. Standardization is beneficial in achieving interoperability, ensuring products meet certain requirements, commonality, reliability, total cost of ownership, compatibility with logistics systems, and similar defense-related objectives.
Model-based systems engineering	"Model-based systems engineering is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing through out development and later life cycle phases". INCOSE, Systems Engineering Vision 2020, Version 2.03, TP-2004-004-02 Sept 2007
Modeling and Simulation	Modeling and Simulation is a discipline for developing a level of understanding of the interaction of the parts of a system, and of the system as a whole.

18 Glossary

Naval Open Architecture	The Navy OA is a systems design approach supported by verifiable governmental testing platforms, such as the OACE, that seeks to implement open specifications for interfaces, services and supporting formats. It enables software components to work across a range of systems and interoperate with other software components on local and remote systems
Naval Postgraduate School	Advanced studies focused on DoD
Naval Sea Systems Command	Surface Command of Fleet for DoD
Naval Surface Warfare Center	Field activity that is part of Naval Sea Systems Command
Net-Centric Architectures	Enables information sharing by connecting people and systems who have information with people/systems who need information. It includes situational awareness, self-synchronizing ops, information pull, collaboration, shared data, bandwidth on demand, diverse routing, Enterprise services
Office of the Chief of Naval Operations	The Chief of Naval Operations (CNO) is the senior military officer in the Navy. The CNO is a four-star admiral and is responsible to the Secretary of the Navy for the command, utilization of resources and operating efficiency of the operating forces of the Navy and of the Navy shore activities assigned by the Secretary.
Open Architecture	The confluence of business and technical practices yielding modular, interoperable systems that adhere to open standards with published interfaces
Performance Based Logistics	The preferred sustainment strategy for weapon system product support that employs the purchase of support as an integrated, affordable performance package designed to optimize system readiness. PBL meets performance goals for a weapon system through a support structure based on long-term performance agreements with clear lines of authority and responsibility.
Point of Contact	Person serving as coordinator, action officer, or focal point for an activity.
Port Hueneme Division	Division of the NSWC under the Naval Sea Systems Command
Post Independent Analysis	Post Independent Analysis. The final step in the JCIDS analysis process is the PIA. In this step, the sponsor will assess the compiled information and analysis results of the FSA (non-materiel and materiel approaches) to ensure the list of approaches with the potential to deliver the capability identified in the FAA and FNA is complete. The sponsor team performing the PIA shall be made up of individuals who were not involved in the FSA. This information will be compiled into an appropriate recommendation and documented in an ICD or joint DCR
Program Executive Office; Program Executive Officer	The military or civilian official who has responsibility for directing several Major Defense Acquisition Programs (MDAPs) and for assigned major system and non-major system acquisition programs. A PEO has no other command or staff responsibilities within the Component, and only reports to and receives guidance and direction from the DoD Component Acquisition Executive (CAE).
Programmatic Environmental, Safety, and Health Evaluation	PESHE incorporates all the environmental, safety and health regulations into a document that is required by Milestone B and is updated through each acquisition phase.

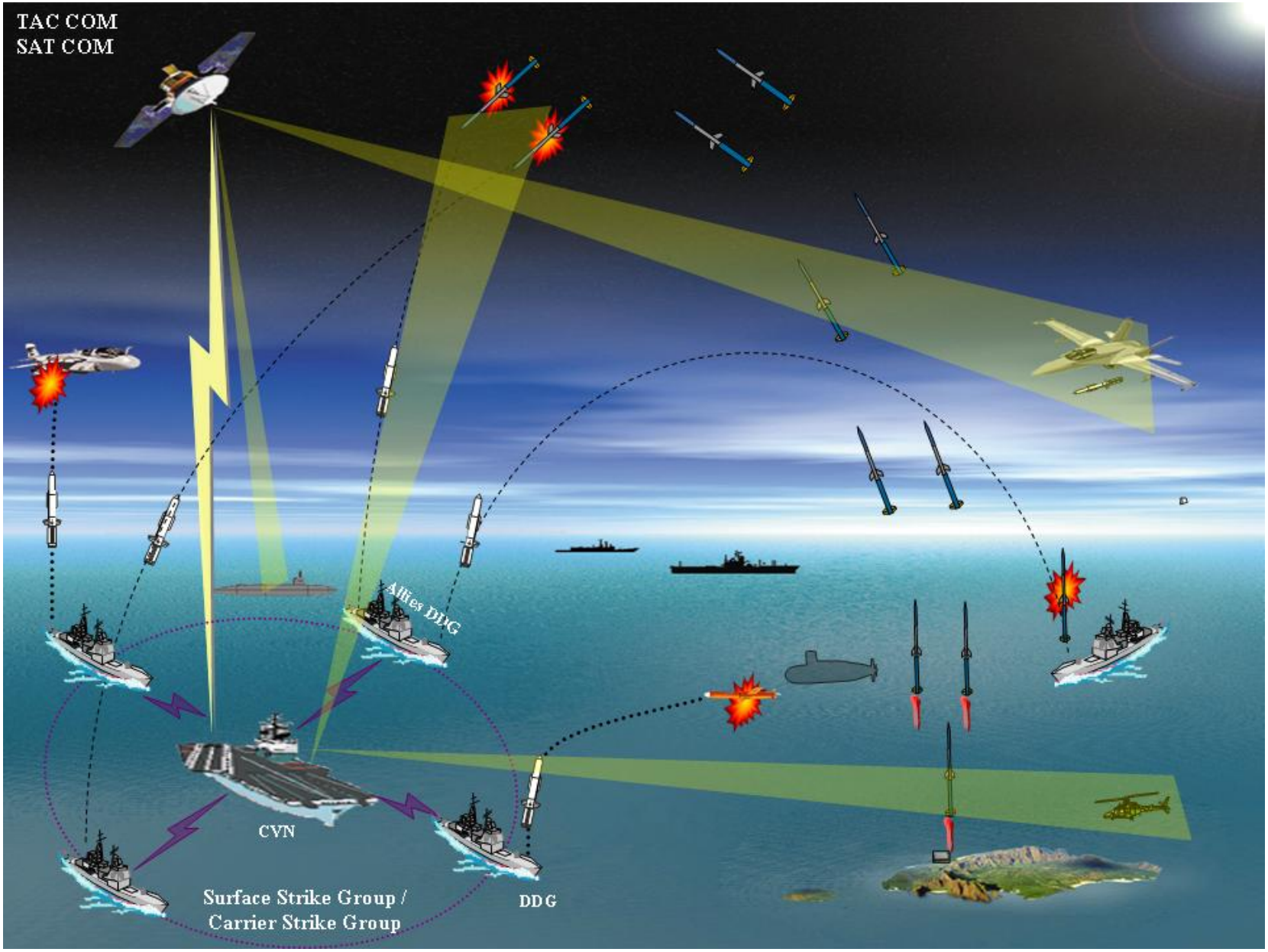
18 Glossary

Project Management Plan	Project management is a carefully planned and organized effort to accomplish a specific (and usually) one-time effort, for example, construct a building or implement a new computer system. Project management includes developing a project plan, which includes defining project goals and objectives, specifying tasks or how goals will be achieved, what resources are need, and associating budgets and timelines for completion. It also includes implementing the project plan, along with careful controls to stay on the "critical path", that is, to ensure the plan is being managed according to plan. Project management usually follows major phases (with various titles for these phases), including feasibility study, project planning, implementation, evaluation and support/maintenance
Reliability Theory	Reliability theory is a general theory about systems failure. It allows researchers to predict the age-related failure kinetics for a system of given architecture (reliability structure) and given reliability of its components. Reliability theory predicts that even those systems that are entirely composed of non-aging elements (with a constant failure rate) will nevertheless deteriorate (fail more often) with age, if these systems are *redundant* in irreplaceable elements. Aging, therefore, is a direct consequence of systems redundancy. Reliability theory also predicts the late-life mortality deceleration with subsequent leveling-off, as well as the late-life mortality plateaus, as an inevitable consequence of *redundancy exhaustion* at extreme old ages.
Service Oriented Architecture	In computing, **service-oriented architecture** (**SOA**) provides methods for systems development and integration where systems group functionality around business processes and package these as *interoperable services*. SOA also describes IT infrastructure which allows different applications to exchange data with one another as they participate in business processes. Service-orientation aims at a *loose coupling* of services with operating systems, programming languages and other technologies which underlie applications
Ship Self Defense System	**Ship's Self Defense System** (**SSDS**) is an integrated weapons system used aboard large U.S. Navy ships, such as Nimitz class super carriers and various amphibious assault ships, such as LHDs and LSDs. SSDS has similar attributes to the combat system used aboard DDGs and CGs, in that it is an integrated Combat Direction System (CDS). The combat direction systems aboard DDGs and CGs, Aegis, are purpose built integrated designs from the outset. SSDS follows a different approach and uses existing shipboard radars and weapons systems, integrated under a COTS framework, to provide a cohesive CDS. The first SSDS designs were back fit onto existing U.S. Navy ships.
Software Product Line	A software product line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. Def from SW Eng Institute, CMU
Space and Naval Warfare	Space and Naval Warfare Systems Center Pacific (SSC Pacific) is responsible for development of the technology to collect, transmit, process, display and, most critically, manage information essential to successful military operations. The Center develops the capabilities that allow decision-makers of the Navy, and increasingly of the joint services, to carry out their operational missions and protect their forces.

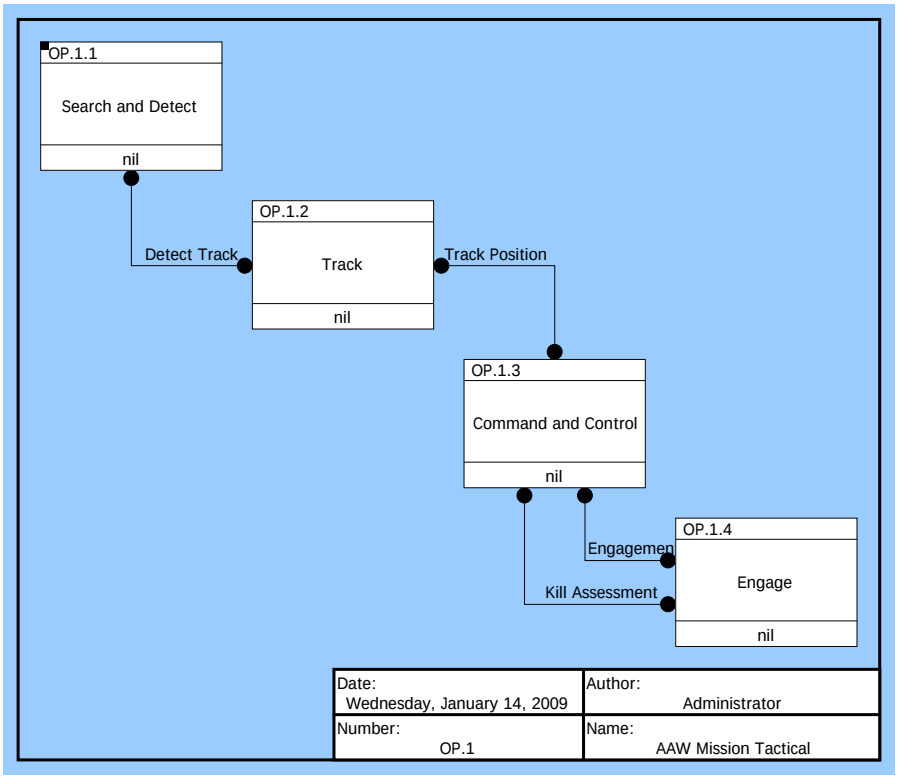
18 Glossary

Supportability	A key component of availability. It includes design, technical support data, and maintenance procedures to facilitate detection, isolation, and timely repair and/or replacement of system anomalies. This includes factors such as diagnostics, prognostics, real time maintenance data collection, and human system integration considerations. (CJCSI 3170.01E)
Systems Architecture and Requirements Engineering (Hatley et al)	Broad approach to the effective development of systems.
Systems Engineering	The overarching process that a program team applies to transition from a stated capability to an operationally effective and suitable system. SE encompasses the application of SE processes across the acquisition life cycle (adapted to each and every phase) and is intended to be the integrating mechanism for balanced solutions addressing capability needs, design considerations and constraints, as well as limitations imposed by technology, budget, and schedule. The SE processes are applied early in concept definition, and then continuously throughout the total life cycle. (Defense Acquisition Guidebook)
Unified Modeling Language	UML is a standardized general-purpose modeling language in the field of software engineering. UML includes a set of graphical notation techniques to create abstract models of specific systems
VEE Model	The Vee model is rooted in the project cycle, which is displayed from left to right to represent project time and maturity. Coupled with this depiction is the recognition of levels of decomposition, which are illustrated, in the vertical dimension from top to bottom. The User is at the highest level and parts and lines of code the lowest. The plane orthogonal to the plane of the Vee illustrates the number of entities at each level of decomposition, which relates to the complexity of the system. INCOSE definition

OV-1
Operational Concept (OV-1)



AAW Mission Tactical
Operational Node Connectivity (OV-2)

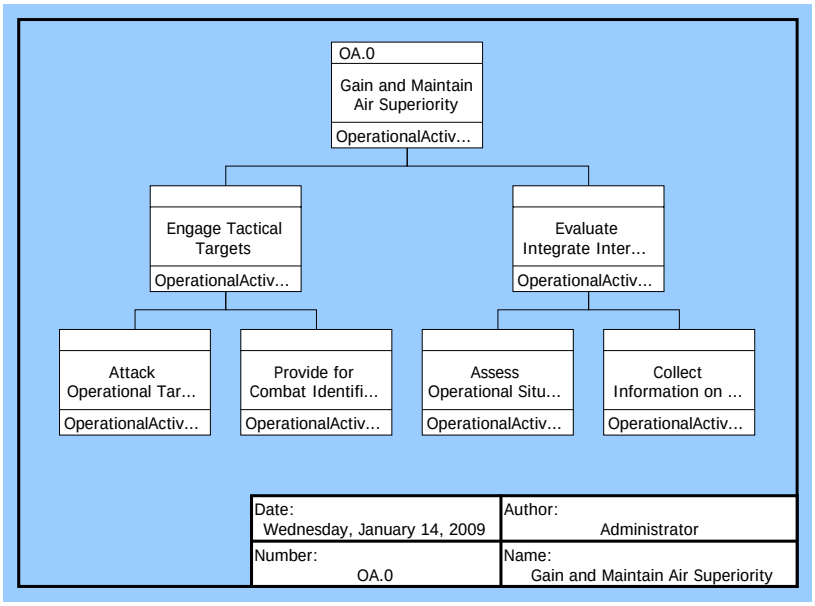


AAW Mission Tactical
Operational Node Connectivity (OV-2)

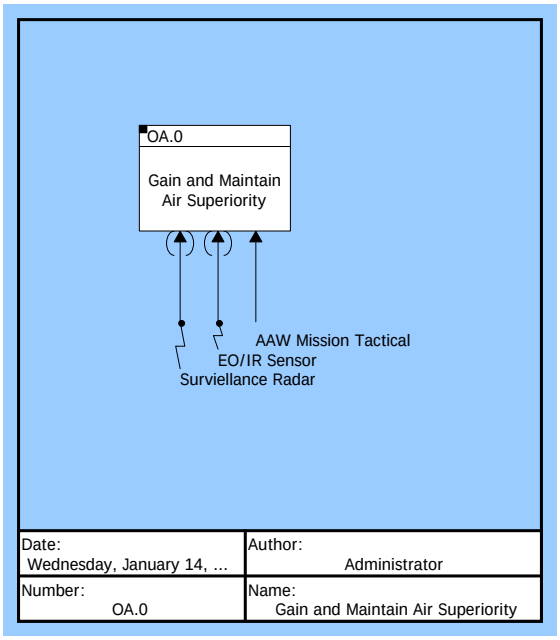
Associated Element Definitions

Element	Definition
Needline	
Detect Track	Primary need from Search to Track.
Engagement Data	Primary need from Control to Engage
Kill Assessment	Primary need from Engage to Control
Track Position	Primary need from Track to Control
Operational Node	
OP.1 AAW Mission Tactical	Supports ST1.6.2 Gain and Maintain Air Superiority through Search and Detect, Tracking, Command and Control, and Engage.
OP.1.1 Search and Detect	
OP.1.2 Track	
OP.1.3 Command and Control	
OP.1.4 Engage	

Gain and Maintain Air Superiority
Operational Activity Model (OV-5)

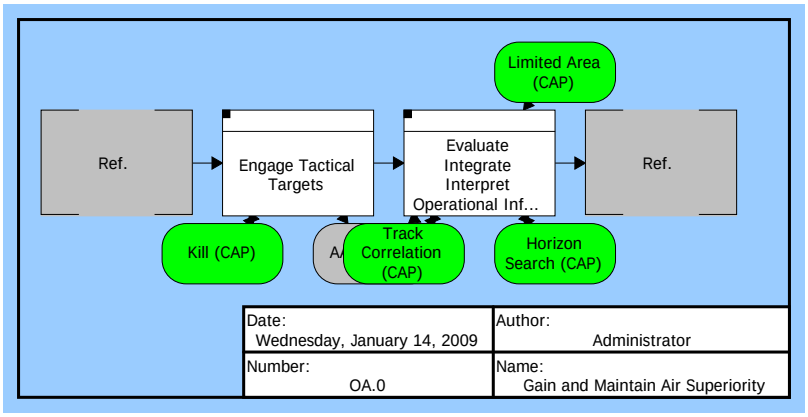


OA.0 Gain and Maintain Air Superiority Hierarchy Diagram

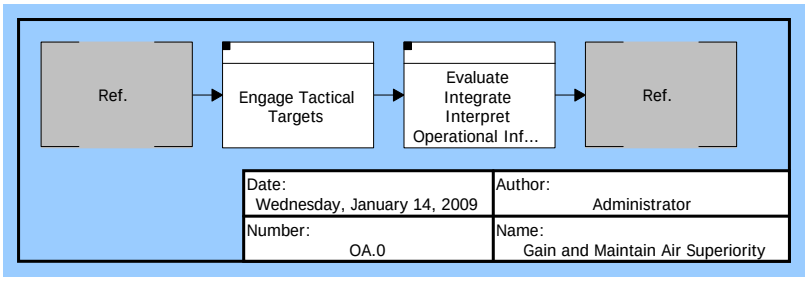


OA.0 Gain and Maintain Air Superiority IDEF0 A-0 Context Diagram

Gain and Maintain Air Superiority
Operational Activity Model (OV-5)

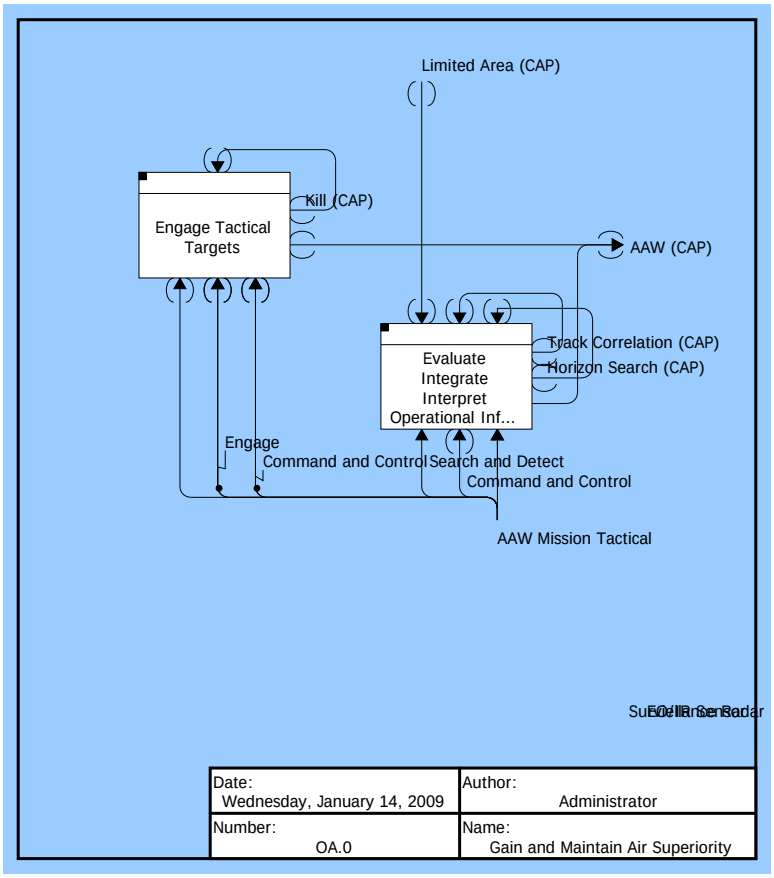


OA.0 Gain and Maintain Air Superiority Enhanced FFBD

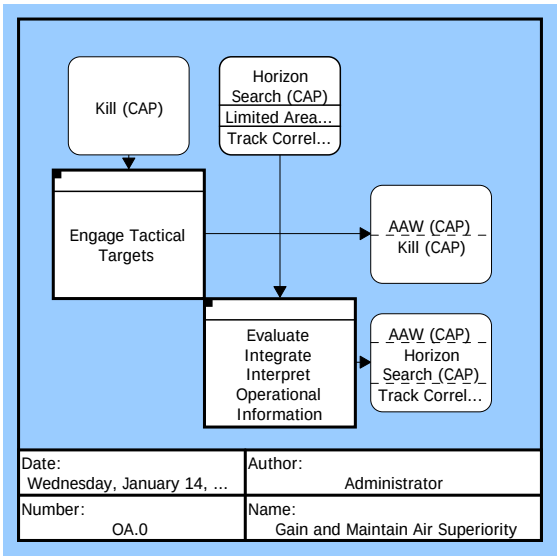


OA.0 Gain and Maintain Air Superiority FFBD

Gain and Maintain Air Superiority
Operational Activity Model (OV-5)



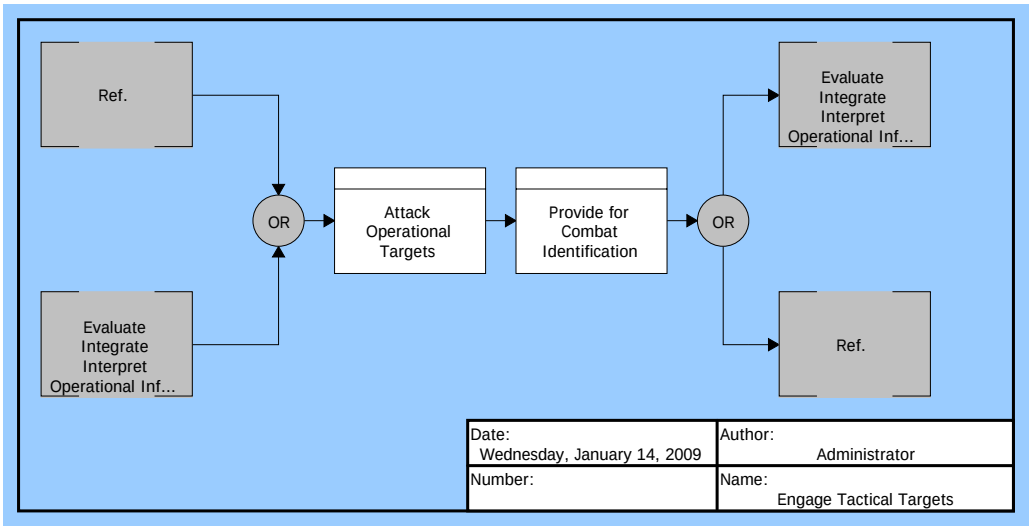
OA.0 Gain and Maintain Air Superiority IDEF0 Diagram



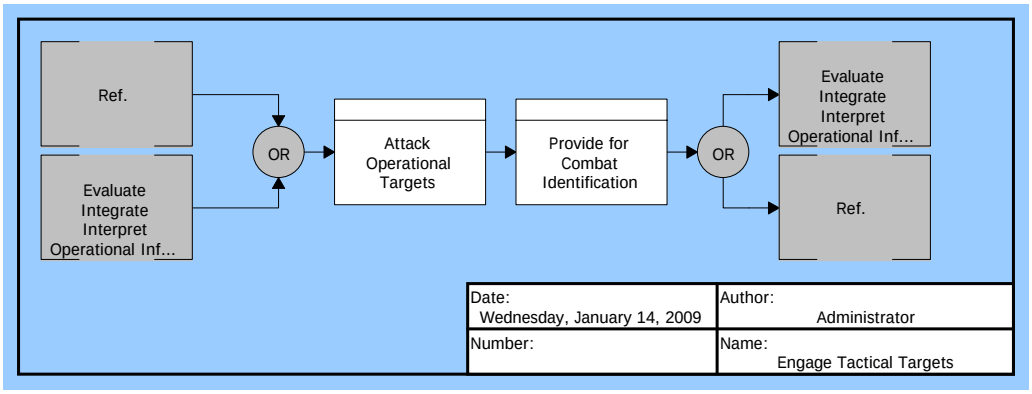
OA.0 Gain and Maintain Air Superiority N2 Diagram

Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

Gain and Maintain Air Superiority
Operational Activity Model (OV-5)

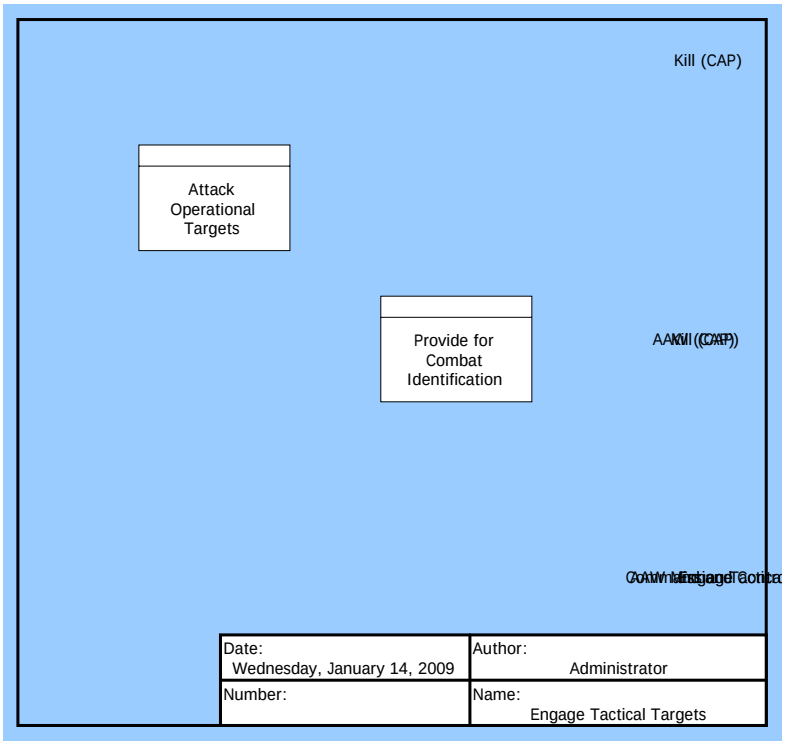


Engage Tactical Targets Enhanced FFBD

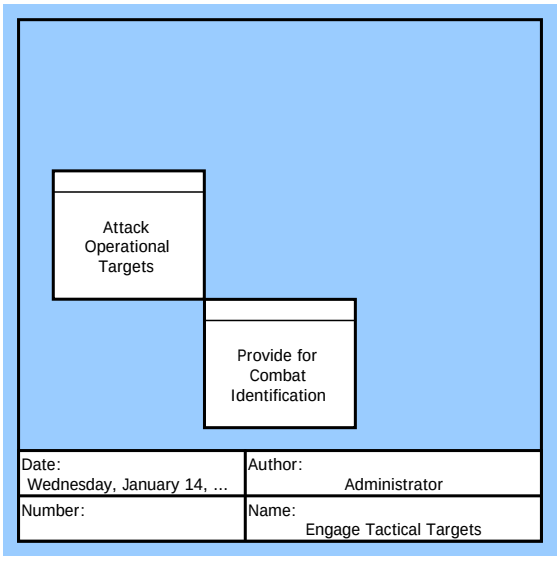


Engage Tactical Targets FFBD

Gain and Maintain Air Superiority
Operational Activity Model (OV-5)



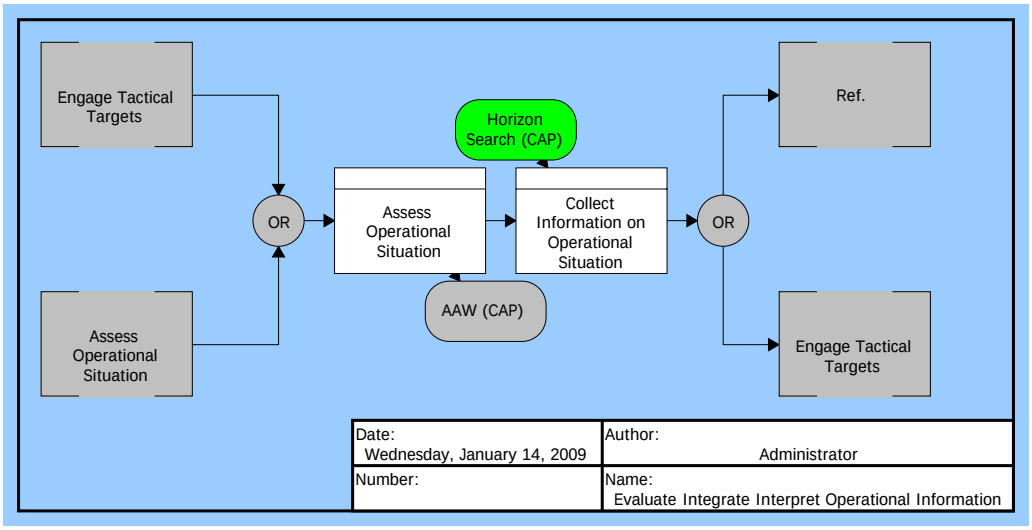
Engage Tactical Targets IDEF0 Diagram



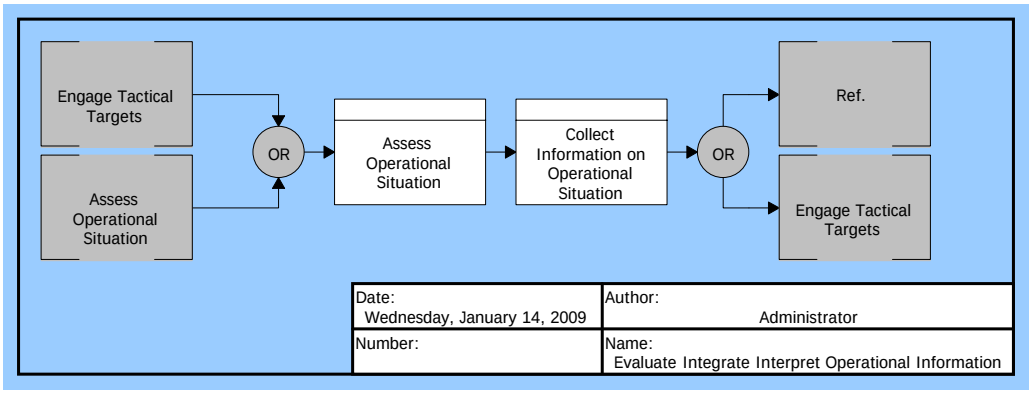
Engage Tactical Targets N2 Diagram

Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

Gain and Maintain Air Superiority
Operational Activity Model (OV-5)

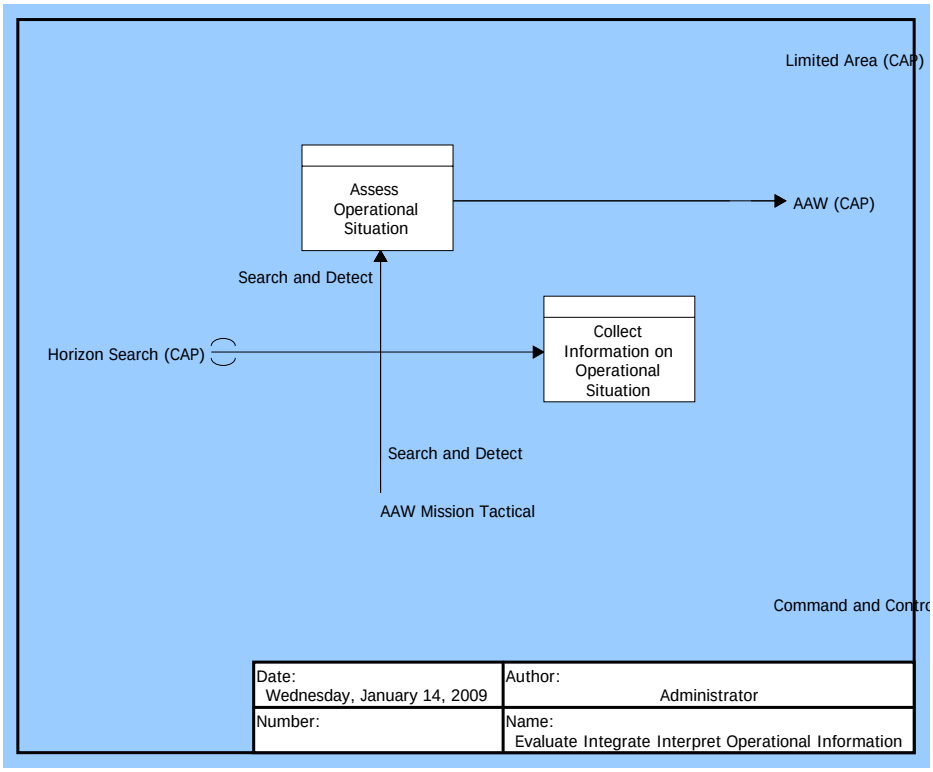


Evaluate Integrate Interpret Operational Information Enhanced FFBD

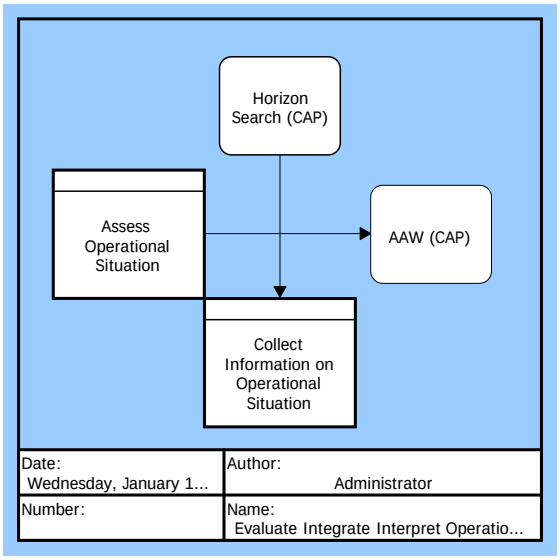


Evaluate Integrate Interpret Operational Information FFBD

Gain and Maintain Air Superiority
Operational Activity Model (OV-5)



Evaluate Integrate Interpret Operational Information IDEF0 Diagram



Evaluate Integrate Interpret Operational Information N2 Diagram

Gain and Maintain Air Superiority
Operational Activity Model (OV-5)

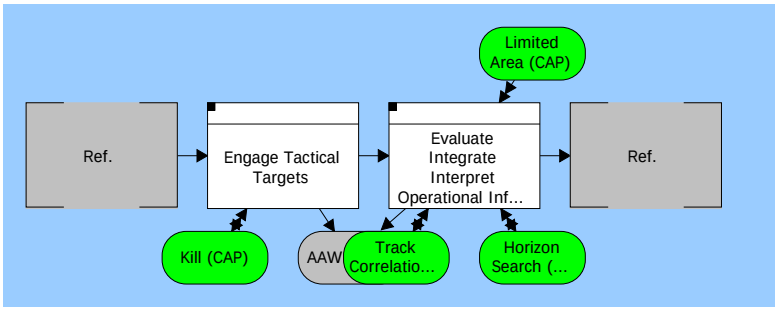
Associated Element Definitions

Element	Definition
Operational Activity	
OA.0 Gain and Maintain Air Superiority	Ownship is operating in ideal conditions (i.e. "Blue Ocean"). The ship's radar and EO/IR sensors are operating and performing constant search sweeps around the ship searching for possible enemy attacks. The EO/IR sensor and L-band radar are designed to perform horizon and above horizon surveillance and to work in conjunction with the combat weapon systems to insure a keep-out range of 2km. The radar and sensor systems suddenly detect and initiate tracks on multiple in-bound threats. The threats consist of eight littoral ASMs, arriving at a rate of one every four seconds. The ASMs are traveling at a speed of Mach 0.9 and possess RF seekers which turn on at 7nm. The AAW Architecture performs search, detect, track, command/control, and engagements of all eight ASMs and destroys the raid, preventing casualty to ownship.
Assess Operational Situation	OP5.2
Attack Operational Targets	OP3.2
Collect Information on Operational Situation	
Engage Tactical Targets	ST3.2
Evaluate Integrate Interpret Operational Information	OP2.4.1
Provide for Combat Identification	TA6.5
Operational Information	
AAW (CAP)	
Horizon Search (CAP)	
Kill (CAP)	
Limited Area (CAP)	

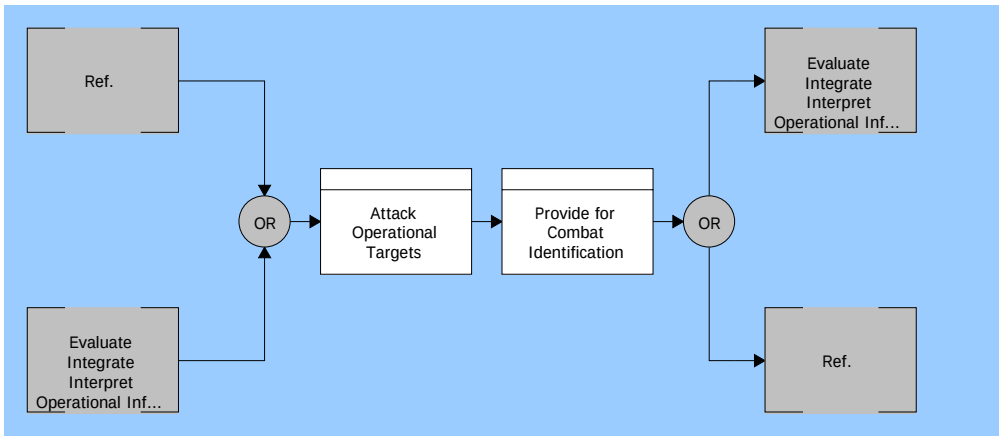
Gain and Maintain Air Superiority
Operational Activity Model (OV-5)

Element	Definition
Track Correlation (CAP)	
Operational Node	
AAW Mission Tactical	Supports ST1.6.2 Gain and Maintain Air Superiority through Search and Detect, Tracking, Command and Control, and Engage.
Command and Control	
Engage	
EO/IR Sensor	
Search and Detect	
Surviellance Radar	

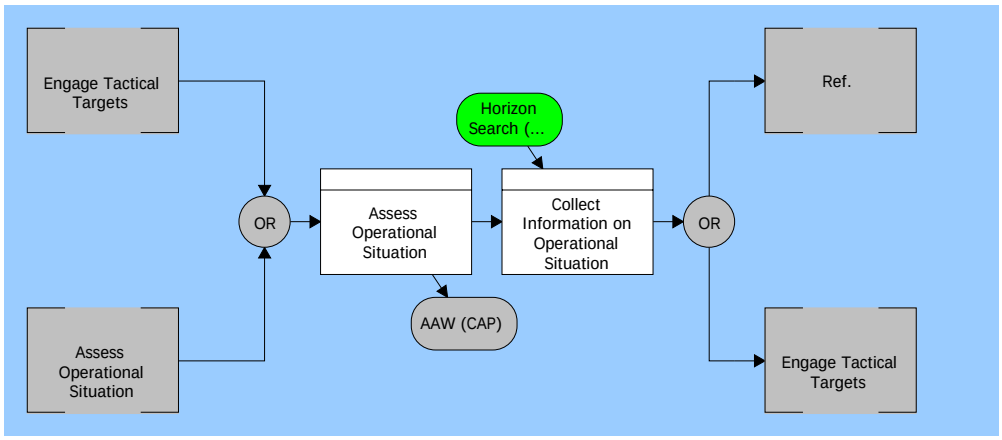
Gain and Maintain Air Superiority
Operational Activity Sequence Model (OV-6)



OA.0 Gain and Maintain Air Superiority Enhanced FFBD



Engage Tactical Targets Enhanced FFBD

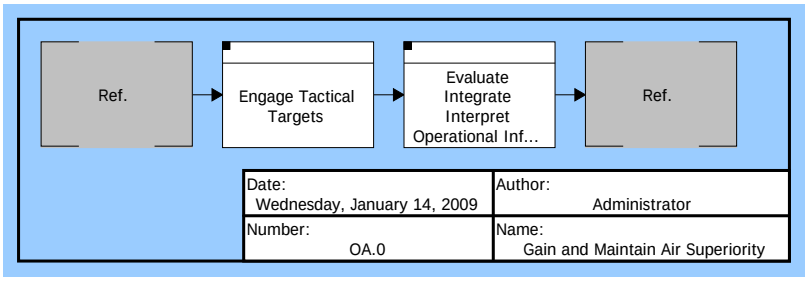


Evaluate Integrate Interpret Operational Information Enhanced FFBD

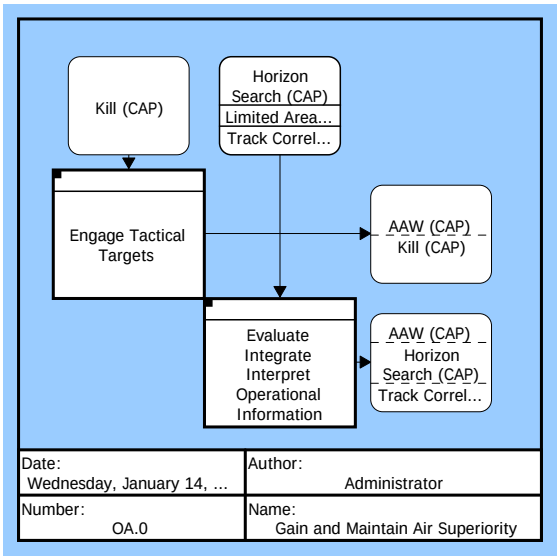
Gain and Maintain Air Superiority
Operational Activity Sequence Model (OV-6)
Associated Element Definitions

Element	Definition
Operational Activity	
OA.0 Gain and Maintain Air Superiority	Ownship is operating in ideal conditions (i.e. "Blue Ocean"). The ship's radar and EO/IR sensors are operating and performing constant search sweeps around the ship searching for possible enemy attacks. The EO/IR sensor and L-band radar are designed to perform horizon and above horizon surveillance and to work in conjunction with the combat weapon systems to insure a keep-out range of 2km. The radar and sensor systems suddenly detect and initiate tracks on multiple in-bound threats. The threats consist of eight littoral ASMs, arriving at a rate of one every four seconds. The ASMs are traveling at a speed of Mach 0.9 and possess RF seekers which turn on at 7nm. The AAW Architecture performs search, detect, track, command/control, and engagements of all eight ASMs and destroys the raid, preventing casualty to ownship.
Assess Operational Situation	OP5.2
Attack Operational Targets	OP3.2
Collect Information on Operational Situation	
Engage Tactical Targets	ST3.2
Evaluate Integrate Interpret Operational Information	OP2.4.1
Provide for Combat Identification	TA6.5
OperationalInformation	
AAW (CAP)	
Horizon Search (CAP)	
Kill (CAP)	
Limited Area (CAP)	
Track Correlation (CAP)	

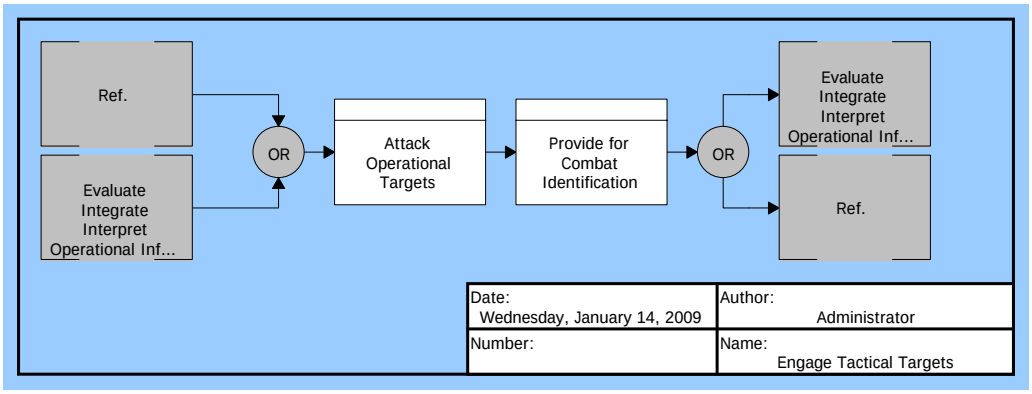
Gain and Maintain Air Superiority
Operational Activity Sequence Model (OV-6)



OA.0 Gain and Maintain Air Superiority FFBD



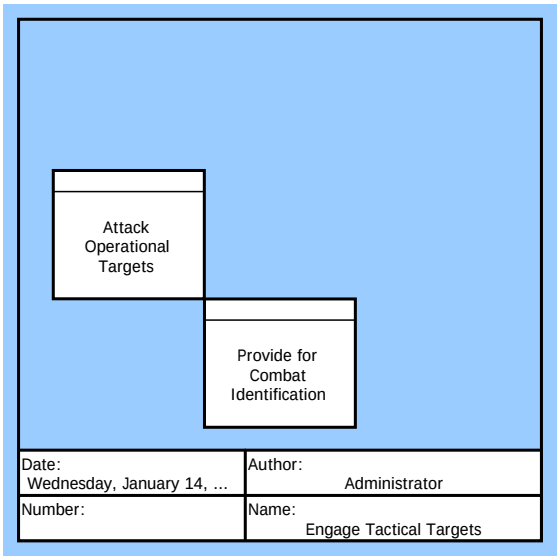
OA.0 Gain and Maintain Air Superiority N2 Diagram



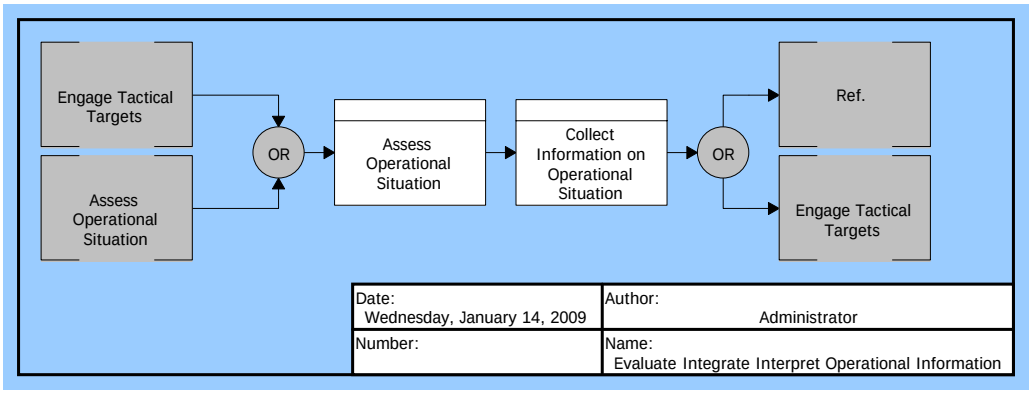
Engage Tactical Targets FFBD

Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

Gain and Maintain Air Superiority
Operational Activity Sequence Model (OV-6)

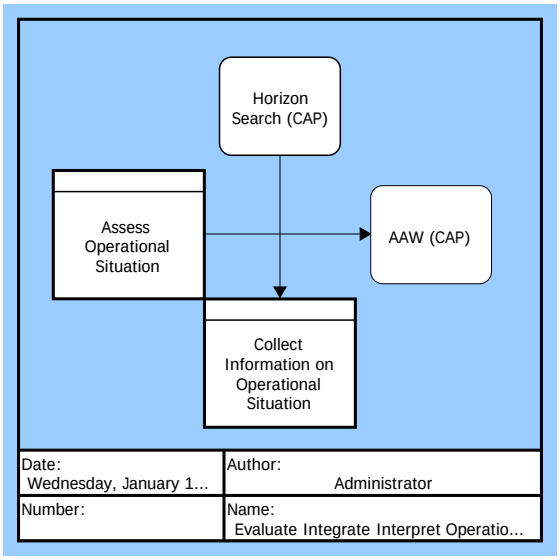


Engage Tactical Targets N2 Diagram



Evaluate Integrate Interpret Operational Information FFBD

Gain and Maintain Air Superiority
Operational Activity Sequence Model (OV-6)



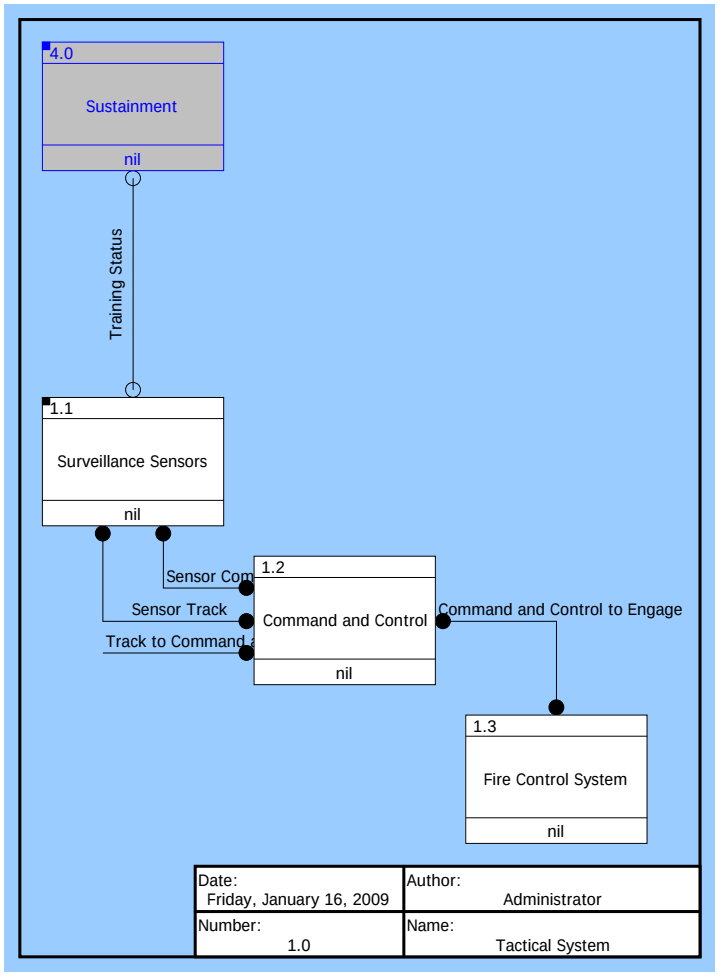
Evaluate Integrate Interpret Operational Information N2 Diagram

Gain and Maintain Air Superiority
Operational Activity Sequence Model (OV-6)

Associated Element Definitions

Element	Definition
Operational Activity	
OA.0 Gain and Maintain Air Superiority	Ownship is operating in ideal conditions (i.e. "Blue Ocean"). The ship's radar and EO/IR sensors are operating and performing constant search sweeps around the ship searching for possible enemy attacks. The EO/IR sensor and L-band radar are designed to perform horizon and above horizon surveillance and to work in conjunction with the combat weapon systems to insure a keep-out range of 2km. The radar and sensor systems suddenly detect and initiate tracks on multiple in-bound threats. The threats consist of eight littoral ASMs, arriving at a rate of one every four seconds. The ASMs are traveling at a speed of Mach 0.9 and possess RF seekers which turn on at 7nm. The AAW Architecture performs search, detect, track, command/control, and engagements of all eight ASMs and destroys the raid, preventing casualty to ownship.
Assess Operational Situation	OP5.2
Attack Operational Targets	OP3.2
Collect Information on Operational Situation	
Engage Tactical Targets	ST3.2
Evaluate Integrate Interpret Operational Information	OP2.4.1
Provide for Combat Identification	TA6.5
Operational Information	
AAW (CAP)	
Horizon Search (CAP)	
Kill (CAP)	
Limited Area (CAP)	
Track Correlation (CAP)	

Tactical System
Systems/Services Interface Diagram (SV-1)

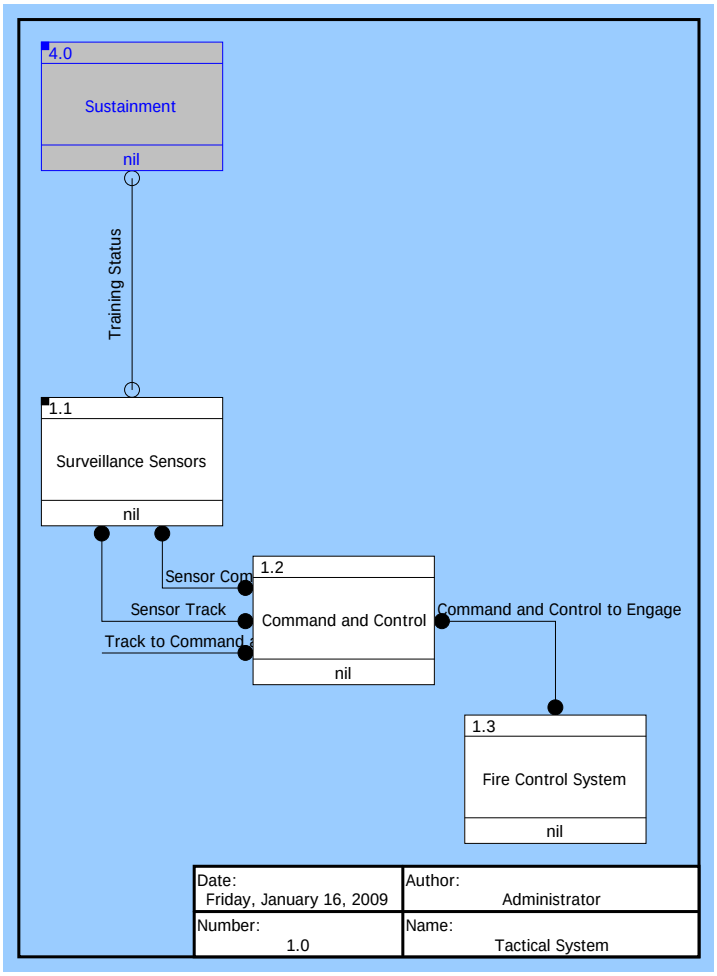


Tactical System
Systems/Services Interface Diagram (SV-1)

Associated Element Definitions

Element	Definition
Component	
1.0 Tactical System	Composes System Components required to host and execute mission functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.
1.1 Surveillance Sensors	Composes System Components required to host and execute Sensor functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.
1.2 Command and Control	Composes System Components required to host and execute Track and Command and Control functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.
1.3 Fire Control System	Composes System Components required to host and execute Track and Command and Control functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.
4.0 Sustainment	Composes System Components required to host and execute sustainment support functionality associated with meeting Mra based on AAW, LAD, and Surveillance Missions.
Link	
Command and Control to Engage	
Sensor Command and Control	
Sensor Track	
Track to Command and Control	
Training Status	

Tactical System
Systems/Services Communications Diagram (SV-2)



Tactical System
Systems/Services Communications Diagram (SV-2)

Associated Element Definitions

Element	Definition
Component	
1.0 Tactical System	Composes System Components required to host and execute mission functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.
1.1 Surveillance Sensors	Composes System Components required to host and execute Sensor functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.
1.2 Command and Control	Composes System Components required to host and execute Track and Command and Control functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.
1.3 Fire Control System	Composes System Components required to host and execute Track and Command and Control functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.
4.0 Sustainment	Composes System Components required to host and execute sustainment support functionality associated with meeting Mra based on AAW, LAD, and Surveillance Missions.
Link	
Command and Control to Engage	
Sensor Command and Control	
Sensor Track	
Track to Command and Control	
Training Status	

Gain and Maintain Air Superiority to Tactical System Operational Activity to Systems Function Traceability Matrix (SV-5)

Component	Function	Operational Activity	
		Engage Tactical Targets	Evaluate Integrate Interpret Operational Information
Command and Control [WS 21340]	Command and Control	**X**	**X**
	Engage	**X**	
	Tracking		**X**
Fire Control System [WS-31333]	Engage	**X**	
Surveillance Sensors [WS 21200]	Horizon Search		**X**
	Search and Detect		**X**

Gain and Maintain Air Superiority to Tactical System Operational Activity to Systems Function Traceability Matrix (SV-5)

Associated Element Definitions

Element	Definition
Component	
Command and Control	Composes System Components required to host and execute Track and Command and Control functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.
Fire Control System	Composes System Components required to host and execute Track and Command and Control functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.
Surveillance Sensors	Composes System Components required to host and execute Sensor functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.
Tactical System	Composes System Components required to host and execute mission functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.
Function	
Command and Control	
Engage	
Horizon Search	No known in-bound threat
Search and Detect	
Tracking	
Operational Activity	
Engage Tactical Targets	ST3.2
Evaluate Integrate Interpret Operational Information	OP2.4.1
Gain and Maintain Air Superiority	Ownship is operating in ideal conditions (i.e. "Blue Ocean"). The ship's radar and EO/IR sensors are operating and performing constant search sweeps around the ship searching for possible enemy attacks. The EO/IR sensor and L-band radar are designed to perform horizon and above horizon surveillance and to work in conjunction with the combat weapon systems to insure a keep-out range of 2km.

Gain and Maintain Air Superiority to Tactical System Operational Activity to Systems Function Traceability Matrix (SV-5)

Element	Definition
	The radar and sensor systems suddenly detect and initiate tracks on multiple in-bound threats. The threats consist of eight littoral ASMs, arriving at a rate of one every four seconds. The ASMs are traveling at a speed of Mach 0.9 and possess RF seekers which turn on at 7nm. The AAW Architecture performs search, detect, track, command/control, and engagements of all eight ASMs and destroys the raid, preventing casualty to ownship.

SYSTEM DESCRIPTION DOCUMENT

FOR

Tactical System

Friday, February 06, 2009

Prepared For:

Naval Postgraduate School
Monterey, CA

Prepared By:

Cohort 6
4363 Missile Way
Port Hueneme, CA 93043

Table of Contents

List of Figures

List of Figures

List of Figures

List of Tables

1 Component Overview

Tactical System [WS 21200]

Description:
Composes System Components required to host and execute mission functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.

System Mission:
AAW, LAD, Surveillance

Allocated Functions:
2.0 Tracking
3.0 Command and Control
4.0 Engage

Source Document(s):
AAW Concept of Operations (ConOps)
Document Number: DW-112
Document Date: Sunday, August 03, 2008
Description: The Anti-Air Warfare (AAW) Architecture Concept of Operation (ConOps) goals are to describe and develop a system that satisfies the user's needs in regards to performance against a littoral threat. The user needs were determined to be similar to the analysis stated in problem 4 of the SE 3123 Final Exam. The combat system functionality will be described through an architectural perspective given the requirements and threat analysis stated. Based on the given information, the AAW Architecture will also address and discuss scenarios describing combat engagements for both self protection and protection of high value units (i.e. Limited Area Defense).
Q3 Status Report
Document Number: WD-145
Document Date: Sunday, November 10, 2002
Description: Provides updated status report information for Q3
SSDD
Document Number: MS B-2
Description: Provides System Design functions and attributes

External Interfacing System(s):
4.0 Sustainment [USD ATL]

Assigned Design Constraints:
D.1.3 Standardization

Inputs from External Source(s):
Composite ID Message (Item)
Source of Input(s):
1.1 Horizon Search
2.0 Tracking
Engage Order (Item)
Source of Input(s):
3.0 Command and Control
Engage Ready (Item)
Source of Input(s):

1 Component Overview

4.0 Engage
Kill Assessment Request (Item)
Source of Input(s):
4.0 Engage
Kill Confirm (Item)
Source of Input(s):
4.0 Engage

Triggers from External Source(s):
Composite ID Message (Item)
Source of Trigger(s):
1.1 Horizon Search
2.0 Tracking
Engage Order (Item)
Source of Trigger(s):
3.0 Command and Control
Kill Assessment Request (Item)
Source of Trigger(s):
4.0 Engage
Kill Confirm (Item)
Source of Trigger(s):
4.0 Engage
Raw Data Items
Source of Trigger(s):
1.0 Search and Detect
1.1.1 Provide Radar Surveillance
1.1.2 Perform EO/IR Search
1.1.3 Perform EW Search

Outputs To External Destination(s):
Composite ID Message (Item)
Destination of Output(s):
3.0 Command and Control
3.0 Command and Control
Control Degraded (Item)
Destination of Output(s):
5.0 Sustainment
5.2 Mission Readiness
5.3 Provide Maintenance
Control Ready (Item)
Destination of Output(s):
5.0 Sustainment
5.2 Mission Readiness
Engage Degraded (Item)
Destination of Output(s):
5.0 Sustainment
5.2 Mission Readiness
5.3 Provide Maintenance

1 Component Overview

Engage Order (Item)
- Destination of Output(s):
  - 4.0 Engage
  - 4.0 Engage

Engage Ready (Item)
- Destination of Output(s):
  - 3.0 Command and Control
  - 5.0 Sustainment
  - 5.2 Mission Readiness

Kill Assessment Request (Item)
- Destination of Output(s):
  - 3.0 Command and Control
  - 3.0 Command and Control

Kill Confirm (Item)
- Destination of Output(s):
  - 3.0 Command and Control
  - 3.0 Command and Control

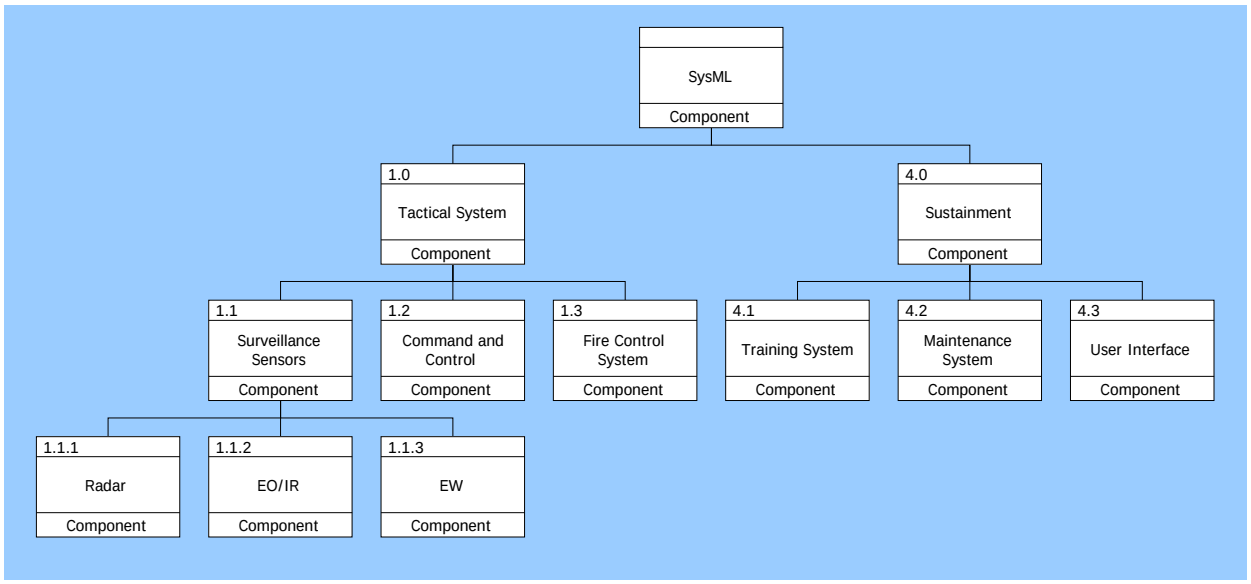


Figure 1 Tactical System Physical Context

1 Component Overview

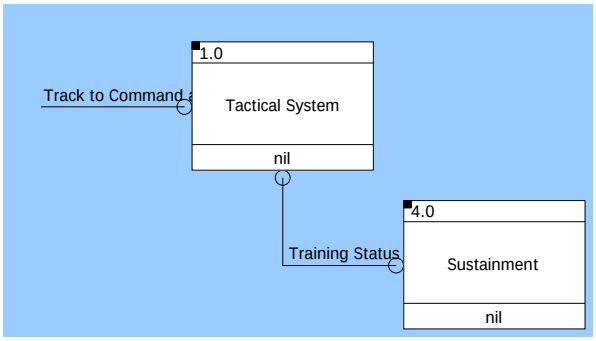


Figure 2 Tactical System Physical Interface Context

2 Originating Requirements

A Anti-Air Warfare

Requirement Statement:
System will defend itself and other units from aerial threats.

Refined By Subordinate Requirements:
A.1 Limited Area Defense
A.2 Self Defense
A.3 Surveillance
D Supportability Performance Range Threshold and Objectives

Specifies:
Component: 1.0 Tactical System [WS 21200]

A.1.1.1 LAD Track

Requirement Statement:
System shall create track based on valid detect.

Refines Higher-Level Requirement:
A.1.1 LAD Pra

Basis Of:
Function: 1.0 Search and Detect
Function: 2.0 Tracking

A.1.1.2 LAD Command and Control

Requirement Statement:
System shall command and control tracks.

Refines Higher-Level Requirement:
A.1.1 LAD Pra

Refined By Subordinate Requirements:
A.1.1.2.1 LAD Manage Tracks

Basis Of:
Function: 3.0 Command and Control

A.1.1.3 LAD Engage

Requirement Statement:
System shall engage tracks.

Refines Higher-Level Requirement:
A.1.1 LAD Pra

Refined By Subordinate Requirements:
A.1.1.3.1 LAD Select Weapon
A.1.1.3.2 LAD Manage Weapon

Basis Of:
Function: 4.0 Engage

2 Originating Requirements

A.2.1 SD Pra

Requirement Statement:
System shall defend ownship with Probability of Raid Annihilation = 0.99 against 8 evenly spaced threats within 32 seconds.

Refines Higher-Level Requirement:
A.2 Self Defense

Refined By Subordinate Requirements:
A.2.1.1 SD Track
A.2.1.2 SD Command and Control
A.2.1.3 SD Engage
B SD Threat
C SD Threat Environment
SD Search and Detect

D Supportability Performance Range Threshold and Objectives

Requirement Statement:
System level Ao and associated supportability attributes.

Refines Higher-Level Requirement:
A Anti-Air Warfare

Refined By Subordinate Requirements:
D.1 Readiness

Specifies:
Component: 4.0 Sustainment [USD ATL]

D.1 Readiness

Requirement Statement:
Maintenance concept and system availability

Refines Higher-Level Requirement:
D Supportability Performance Range Threshold and Objectives

Refined By Subordinate Requirements:
D.1.1 Testability
D.1.2 Ao
D.1.3 Standardization
D.1.4 Human Systems Engineering Factors

Specifies:
Interface: Sustainment Radar Interface

D.1.4 Human Systems Engineering Factors

Requirement Statement:
Man-in-the-Loop

Refines Higher-Level Requirement:

2 Originating Requirements

D.1 Readiness

Refined By Subordinate Requirements:
- D.1.4.1 Training
- D.1.4.2 Manning
- D.1.4.3 Human Systems Integration

3 Design Constraints

C SD Threat Environment

Design Constraint Statement:
The threats are expected to be encountered in a variety of environments.

Refines Higher-Level Requirement:
A.2.1 SD Pra

Refined By Lower-Level Requirements:
C.1 SD Clear Environment
C.2 SD Clutter Environment
Threat CPA

Constrains:
Interface: Search C2 Interface

D.1 Readiness

Design Constraint Statement:
Maintenance concept and system availability

Refines Higher-Level Requirement:
D Supportability Performance Range Threshold and Objectives

Refined By Lower-Level Requirements:
D.1.1 Testability
D.1.2 Ao
D.1.3 Standardization
D.1.4 Human Systems Engineering Factors

Constrains:
Interface: Sustainment Radar Interface

D.1.3 Standardization

Design Constraint Statement:
Modular Open System Architecture and Interoperability.

Refines Higher-Level Requirement:
D.1 Readiness

Constrains:
Component: 1.0 Tactical System [WS 21200]

D.1.4.1 Training

Design Constraint Statement:
Operate, Maintain, and Sustain System

Refines Higher-Level Requirement:
D.1.4 Human Systems Engineering Factors

Constrains:

3 Design Constraints

Interface: Mission Readiness Training Interface
Interface: Training User Interface
State/Mode: Training

4 Performance Requirements

N/A

5 Issues & Decisions

Part IV - Open Issues

Command and Control Issue

Originator: Administrator

Originating Date: Tuesday, January 13, 2009 at 03:45:59 PM

Severity: Critical

Status: Open

Generated By:
Verification Requirement: Contractor Demonstration

Maintenance Issue

Originator: Administrator

Originating Date: Tuesday, January 13, 2009 at 03:46:19 PM

Severity: Critical

Status: Open

Track Issue

Originator: Administrator

Originating Date: Tuesday, January 13, 2009 at 03:46:09 PM

Severity: Critical

Status: Open

Part V - Closed Issues

None

Part VI - Rejected Issues

None

6 Risks

AAW Mission Risk

Caused By:
Verification Requirement: Contractor Demonstration

Sustainability Risk

7 Functional Behavior Model

Part I - Hierarchical Function List

2.0 Tracking
- 2.1 Perform Local Tracking
  - 2.1.1 Perform Single Sensor Tracking
  - 2.1.2 Perform Multiple Sensor Tracking

3.0 Command and Control
- 3.1 Manage Tracks
  - 3.1.1 Assign New Tracks
  - 3.1.2 Monitor Air Tracks
  - 3.1.3 Correlate Air Tracks
  - 3.1.4 Update Air Tracks
  - 3.1.5 Drop Air Tracks
- 3.2 Manage Tactical Control
  - 3.2.1 Manage Information
  - 3.2.2 Provide Display Support for Tactical Operation
  - 3.2.3 Provide Combatant Commanders Decision Aide
  - 3.2.4 Provide Alerts and Prompts
  - 3.2.5 Manage Sub mode
  - 3.2.6 Perform Tactical Control
  - 3.2.7 Perform Initialization
  - 3.2.8 Reconfigure
  - 3.2.9 Perform Readiness Assessment
  - 3.2.10 Manage Ship Sensors in Search, Detection and Track
- 3.3 Plan
- 3.4 Direct

4.0 Engage
- 4.1 Select/Manage Weapons
  - 4.1.1 Designate Electronic Warfare
  - 4.1.2 Designate Fire-and-Forget Weapon
  - 4.1.3 Designate Semi-Active Homing Weapon

Part II - Behavior Model

2.0 Tracking

Duration: 6.0

Allocated To:
- 1.0 Tactical System [WS 21200]
- 1.2 Command and Control [WS 21340]

7 Functional Behavior Model

Based On:
A.1.1.1 LAD Track

Table 1 2.0 Tracking Interfacing Items

Interfacing Items	Source / Destination
Composite ID Message (Item)	Input To: 3.0 Command and Control Triggers Function(s): 3.0 Command and Control Output From: 1.1 Horizon Search 2.0 Tracking
Raw Data Items	Input To: 1.1 Horizon Search 1.3.1 Determine Position Triggers Function(s): 2.0 Tracking Output From: 1.0 Search and Detect 1.1.1 Provide Radar Surveillance 1.1.2 Perform EO/IR Search 1.1.3 Perform EW Search

Consumes Resource(s):
LAN Bandwidth
Acquire Available: true
Amount: 5.0

Operator
Acquire Available: true
Amount: 5.0

RAM Requirements
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 6.0, stream: 1)

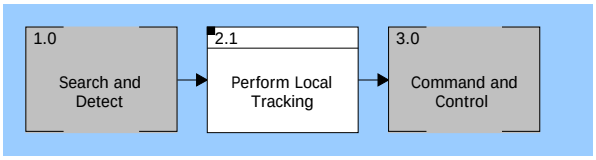


Figure 3 Tracking Enhanced FFBD

7 Functional Behavior Model

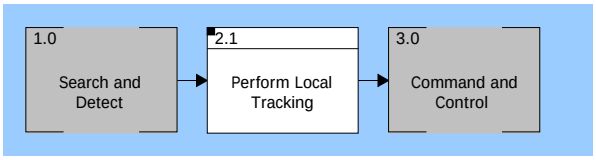


Figure 4 Tracking FFBD

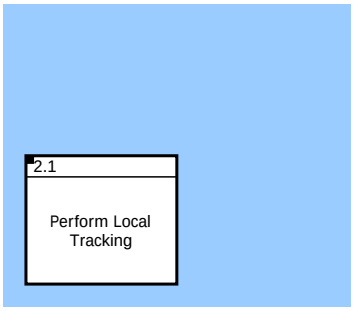


Figure 5 Tracking N2 Diagram

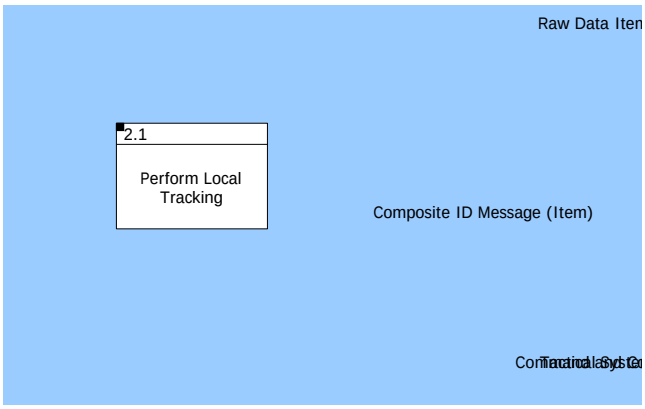


Figure 6 Tracking IDEF0 Diagram

2.1 Perform Local Tracking

Captures Resource(s):

Tracks

Acquire Available: true

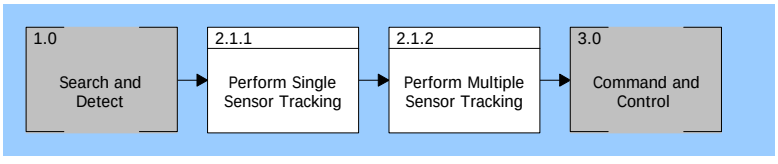


Figure 7 Perform Local Tracking Enhanced FFBD

Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

7 Functional Behavior Model

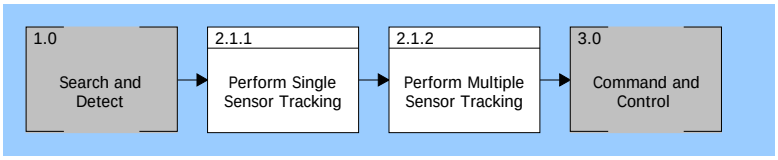


Figure 8 Perform Local Tracking FFBD

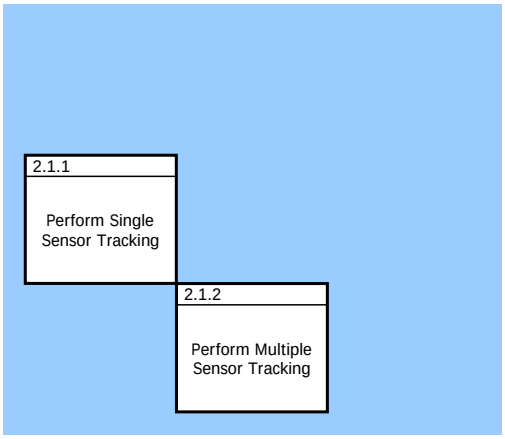


Figure 9 Perform Local Tracking N2 Diagram

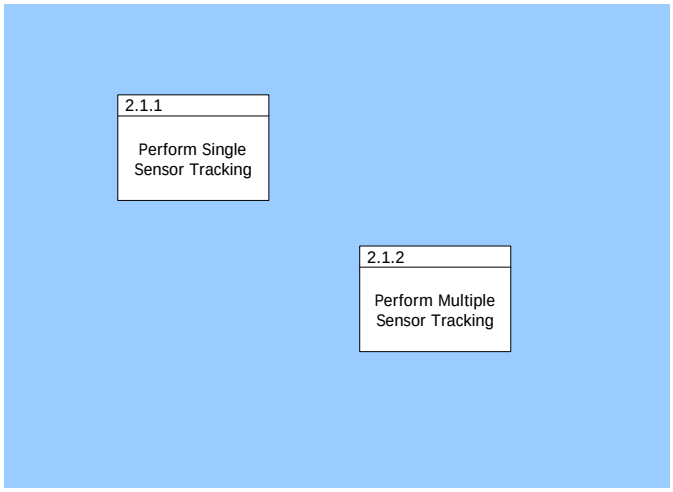


Figure 10 Perform Local Tracking IDEF0 Diagram

2.1.1 Perform Single Sensor Tracking

Duration: Normal (μ: 1.0, stdDev: 1.0, stream: 1)

Consumes Resource(s):

7 Functional Behavior Model

KSLOC
Acquire Available: true
Amount: 1.0

LAN Bandwidth
Acquire Available: true
Amount: 2.0

Operator
Acquire Available: true
Amount: 3.0

RAM Requirements
Acquire Available: true
Amount: 4.0

2.1.2 Perform Multiple Sensor Tracking

Duration: Normal (μ: 2.0, stdDev: 1.0, stream: 1)

Consumes Resource(s):
KSLOC
Acquire Available: true
Amount: 1.0

LAN Bandwidth
Acquire Available: true
Amount: 2.0

Operator
Acquire Available: true
Amount: 3.0

RAM Requirements
Acquire Available: true
Amount: 4.0

3.0 Command and Control

Duration: 15.0

Exits:
ROE Met

Allocated To:
1.0 Tactical System [WS 21200]
1.2 Command and Control [WS 21340]

Based On:
A.1.1.2 LAD Command and Control

7 Functional Behavior Model

Table 2 3.0 Command and Control Interfacing Items

Interfacing Items	Source / Destination
Composite ID Message (Item)	Input To: 3.0 Command and Control Triggers Function(s): 3.0 Command and Control Output From: 1.1 Horizon Search 2.0 Tracking
Control Degraded (Item)	Input To: 5.0 Sustainment 5.2 Mission Readiness 5.3 Provide Maintenance Output From: 3.0 Command and Control
Control Ready (Item)	Input To: 5.0 Sustainment 5.2 Mission Readiness Output From: 3.0 Command and Control
Engage Order (Item)	Input To: 4.0 Engage Triggers Function(s): 4.0 Engage Output From: 3.0 Command and Control
Engage Ready (Item)	Input To: 3.0 Command and Control 5.0 Sustainment 5.2 Mission Readiness Output From: 4.0 Engage
Kill Assessment Request (Item)	Input To: 3.0 Command and Control Triggers Function(s): 3.0 Command and Control Output From: 4.0 Engage
Kill Confirm (Item)	Input To: 3.0 Command and Control Triggers Function(s):

7 Functional Behavior Model

Table 2 3.0 Command and Control Interfacing Items

Interfacing Items	Source / Destination
	3.0 Command and Control Output From: 4.0 Engage

Consumes Resource(s):
LAN Bandwidth
Acquire Available: true
Amount: 2.0

Operator
Acquire Available: true
Amount: 7.0

RAM Requirements
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 9.0, stream: 1)

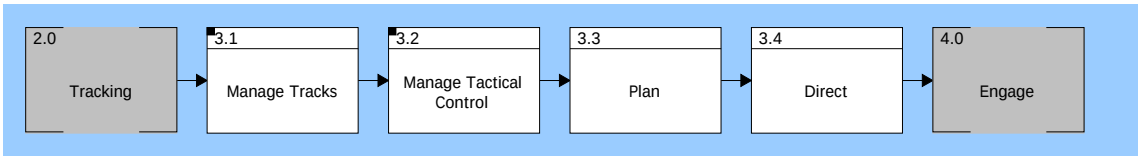


Figure 11 Command and Control Enhanced FFBD

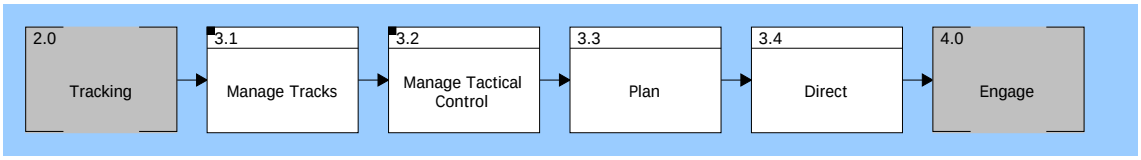


Figure 12 Command and Control FFBD

7 Functional Behavior Model

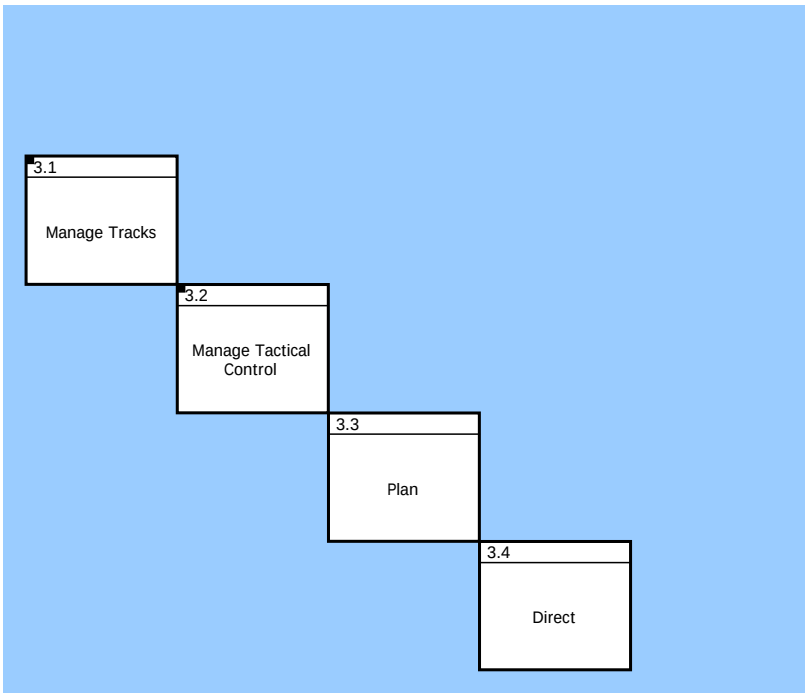


Figure 13 Command and Control N2 Diagram

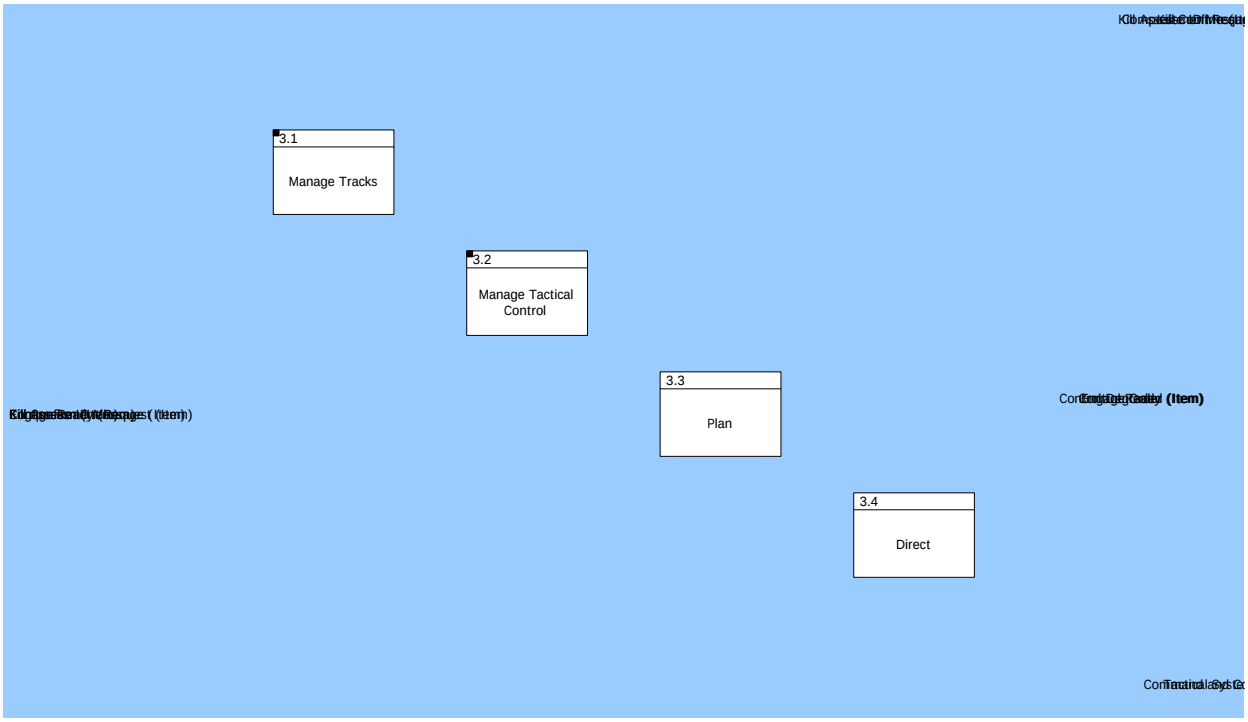


Figure 14 Command and Control IDEF0 Diagram

Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

7 Functional Behavior Model

3.1 Manage Tracks

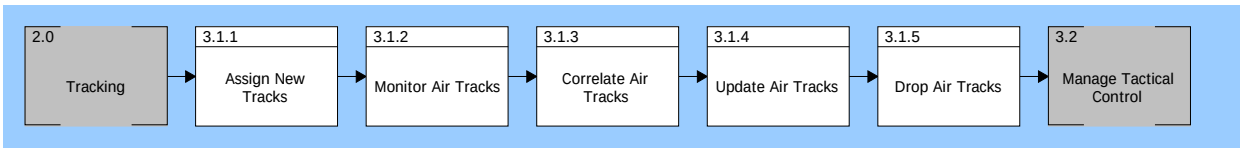


Figure 15 Manage Tracks Enhanced FFBD

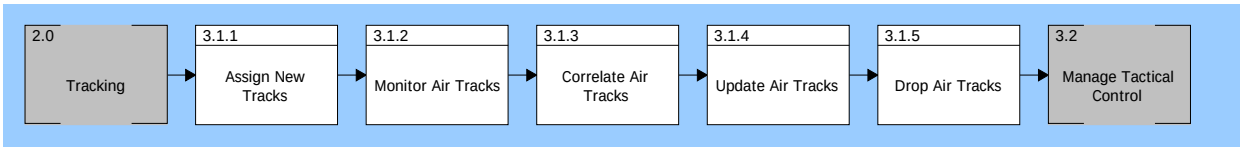


Figure 16 Manage Tracks FFBD

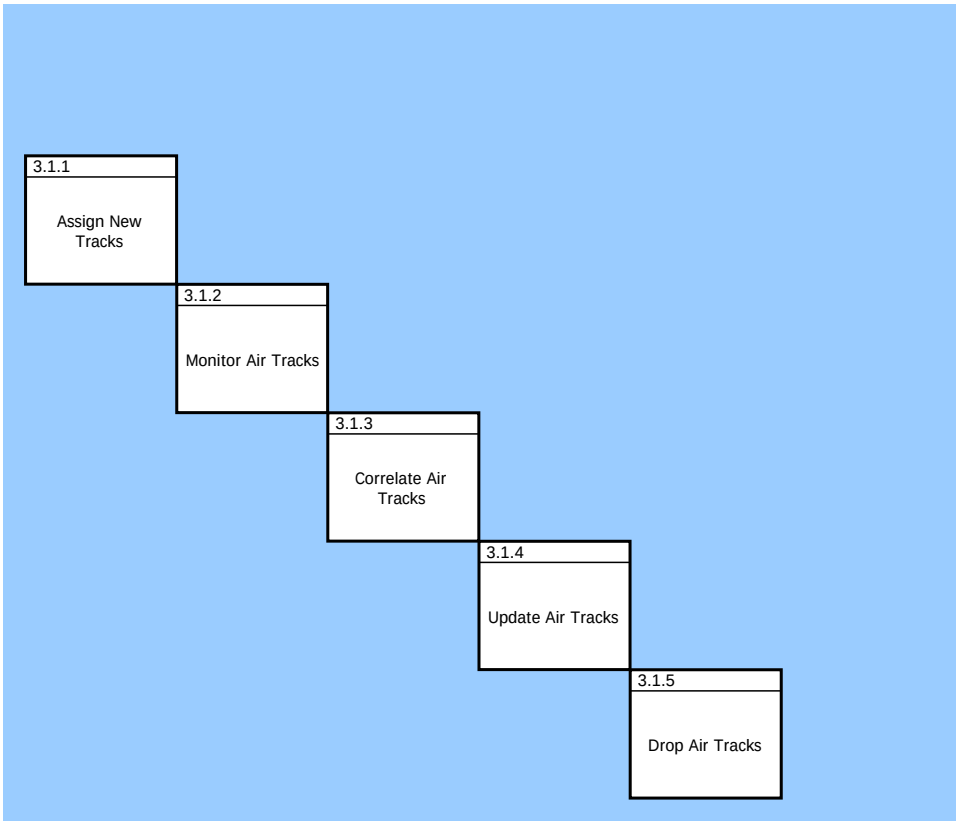


Figure 17 Manage Tracks N2 Diagram

7 Functional Behavior Model

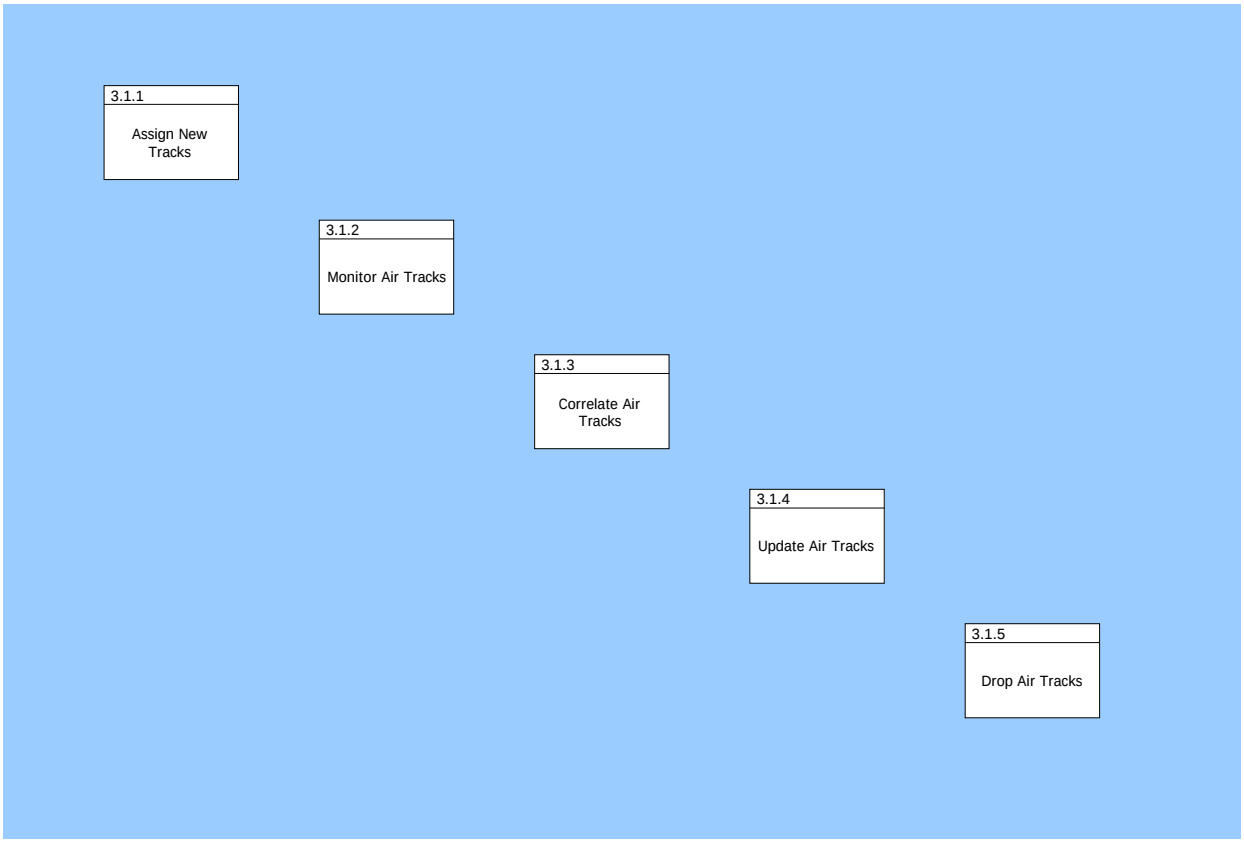


Figure 18 Manage Tracks IDEF0 Diagram

3.1.1 Assign New Tracks

3.1.2 Monitor Air Tracks

3.1.3 Correlate Air Tracks

3.1.4 Update Air Tracks

3.1.5 Drop Air Tracks

Consumes Resource(s):
Tracks
Acquire Available: true

3.2 Manage Tactical Control

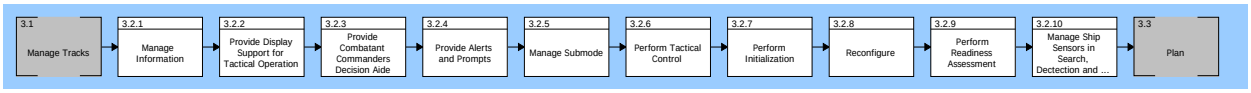


Figure 19 Manage Tactical Control Enhanced FFBD

Note: All information contained herein is for academic purposes only and was solely obtained from open source materials.

7 Functional Behavior Model

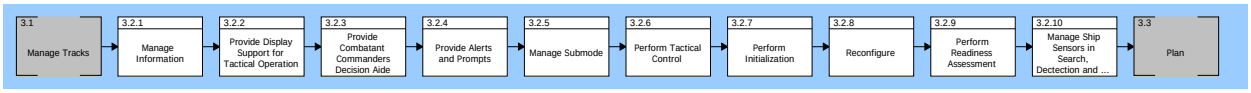


Figure 20 Manage Tactical Control FFBD

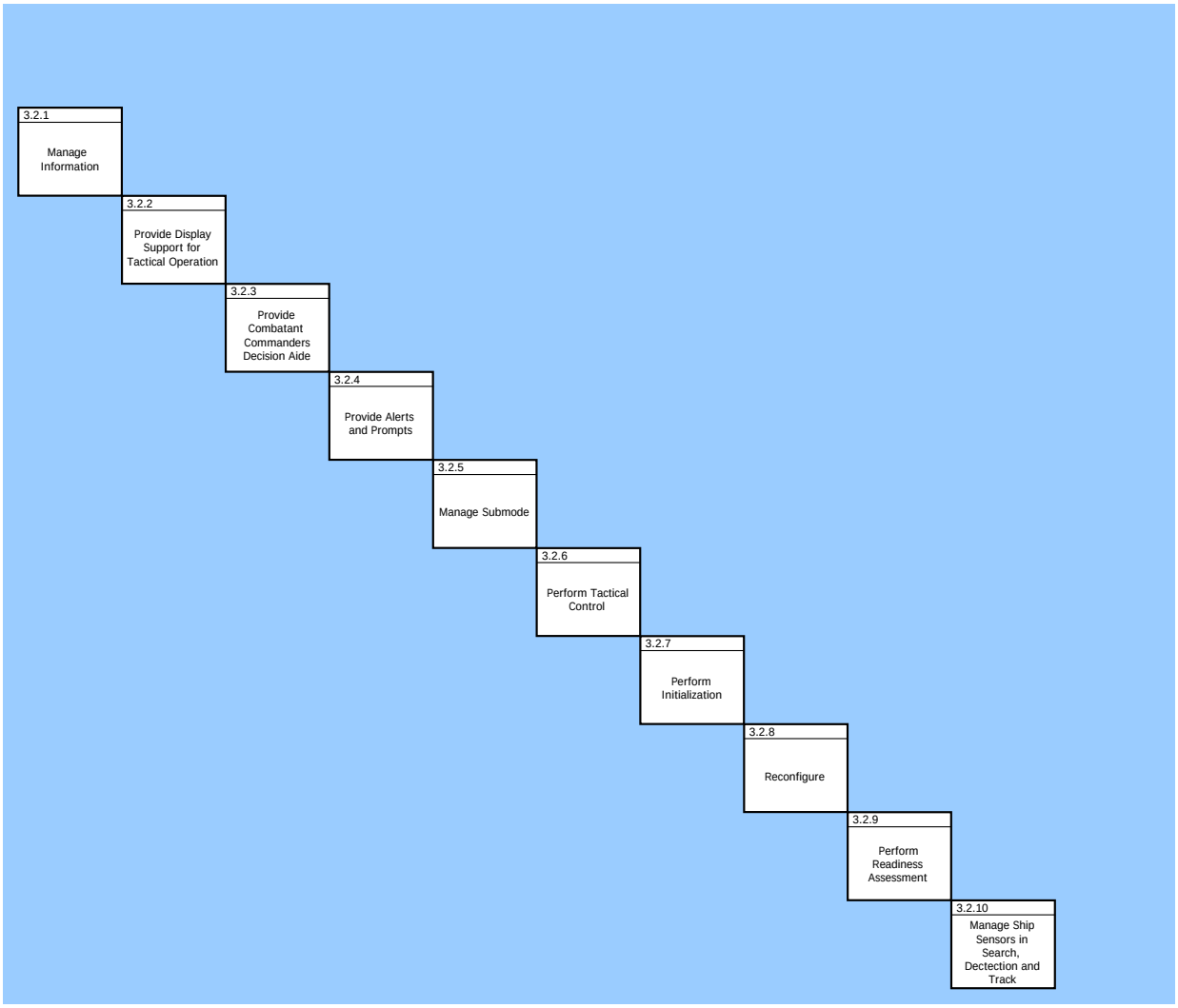


Figure 21 Manage Tactical Control N2 Diagram

7 Functional Behavior Model

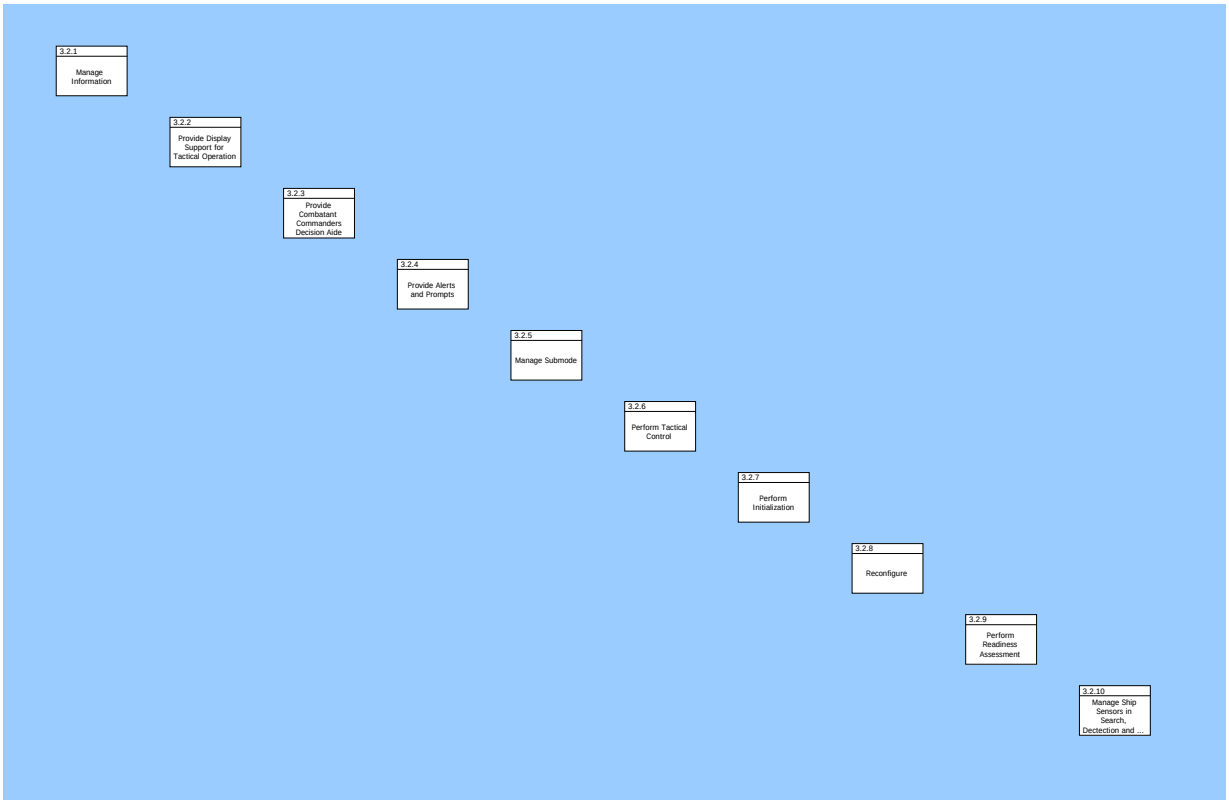


Figure 22 Manage Tactical Control IDEF0 Diagram

7 Functional Behavior Model

3.2.1 Manage Information

3.2.2 Provide Display Support for Tactical Operation

3.2.3 Provide Combatant Commanders Decision Aide

3.2.4 Provide Alerts and Prompts

3.2.5 Manage Sub mode

3.2.6 Perform Tactical Control

3.2.7 Perform Initialization

3.2.8 Reconfigure

3.2.9 Perform Readiness Assessment

3.2.10 Manage Ship Sensors in Search, Detection and Track

3.3 Plan

3.4 Direct

4.0 Engage

Duration: 1.0

Exits:
Threat Killed

Allocated To:
1.0 Tactical System [WS 21200]
1.2 Command and Control [WS 21340]
1.3 Fire Control System [WS-31333]

Based On:
A.1.1.3 LAD Engage

Table 3 4.0 Engage Interfacing Items

Interfacing Items	Source / Destination
Engage Degraded (Item)	Input To: 5.0 Sustainment 5.2 Mission Readiness 5.3 Provide Maintenance Output From:

7 Functional Behavior Model

Table 3 4.0 Engage Interfacing Items

Interfacing Items	Source / Destination
	4.0 Engage
Engage Order (Item)	Input To: 4.0 Engage Triggers Function(s): 4.0 Engage Output From: 3.0 Command and Control
Engage Ready (Item)	Input To: 3.0 Command and Control 5.0 Sustainment 5.2 Mission Readiness Output From: 4.0 Engage
Kill Assessment Request (Item)	Input To: 3.0 Command and Control Triggers Function(s): 3.0 Command and Control Output From: 4.0 Engage
Kill Confirm (Item)	Input To: 3.0 Command and Control Triggers Function(s): 3.0 Command and Control Output From: 4.0 Engage

Consumes Resource(s):
LAN Bandwidth
Acquire Available: true
Amount: 5.0

Operator
Acquire Available: true
Amount: 15.0

RAM Requirements
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 5.0, stream: 1)

7 Functional Behavior Model

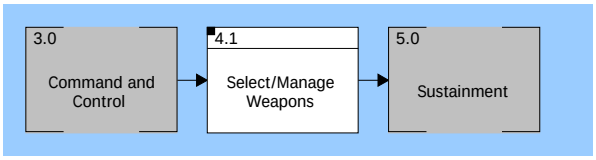


Figure 23 Engage Enhanced FFBD

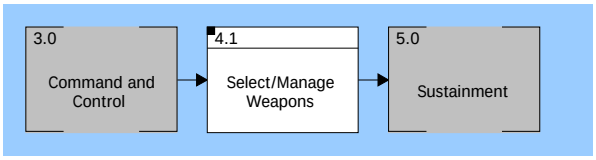


Figure 24 Engage FFBD

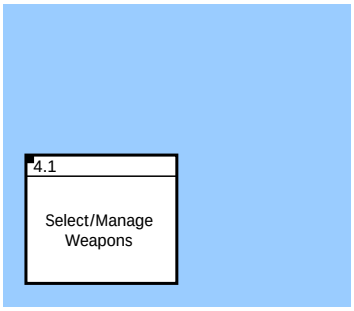


Figure 25 Engage N2 Diagram

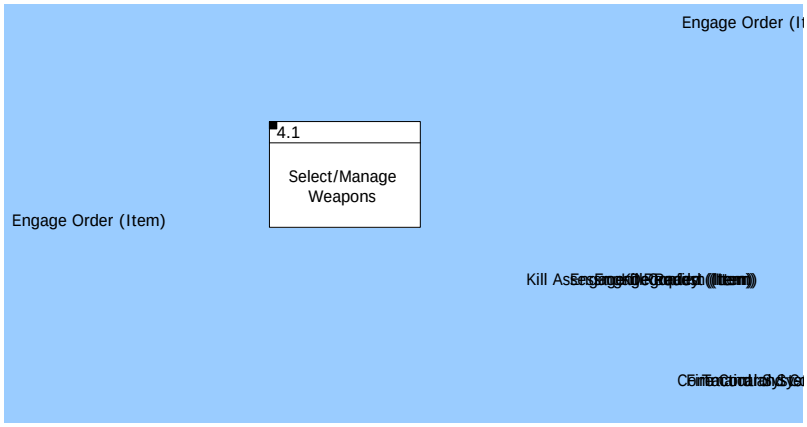


Figure 26 Engage IDEF0 Diagram

7 Functional Behavior Model

4.1 Select/Manage Weapons

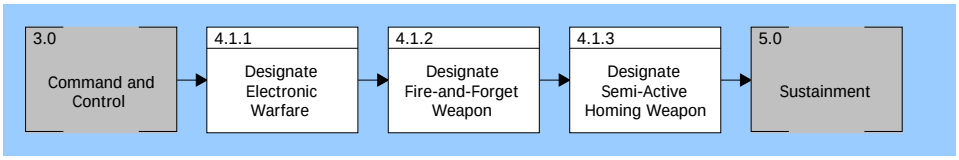


Figure 27 Select/Manage Weapons Enhanced FFBD

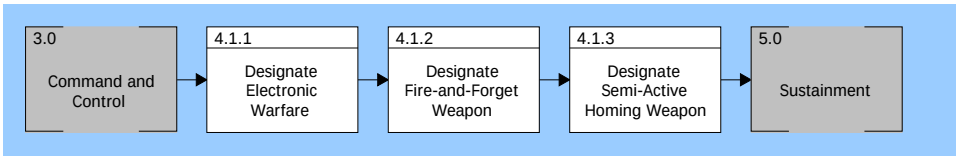


Figure 28 Select/Manage Weapons FFBD

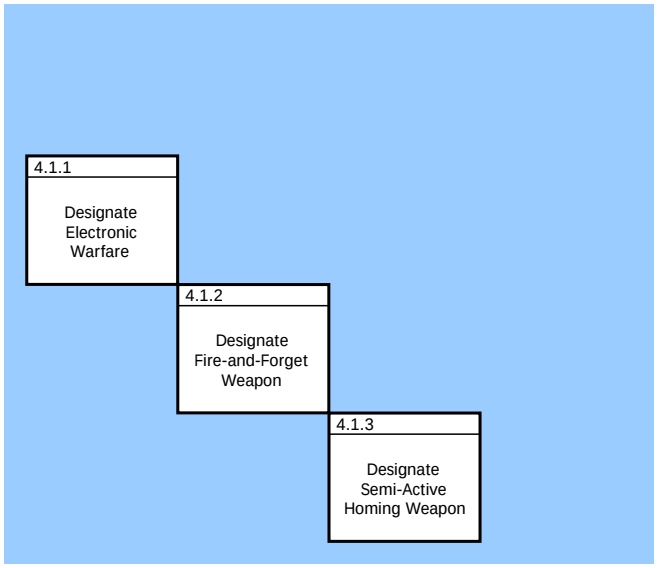


Figure 29 Select/Manage Weapons N2 Diagram

7 Functional Behavior Model

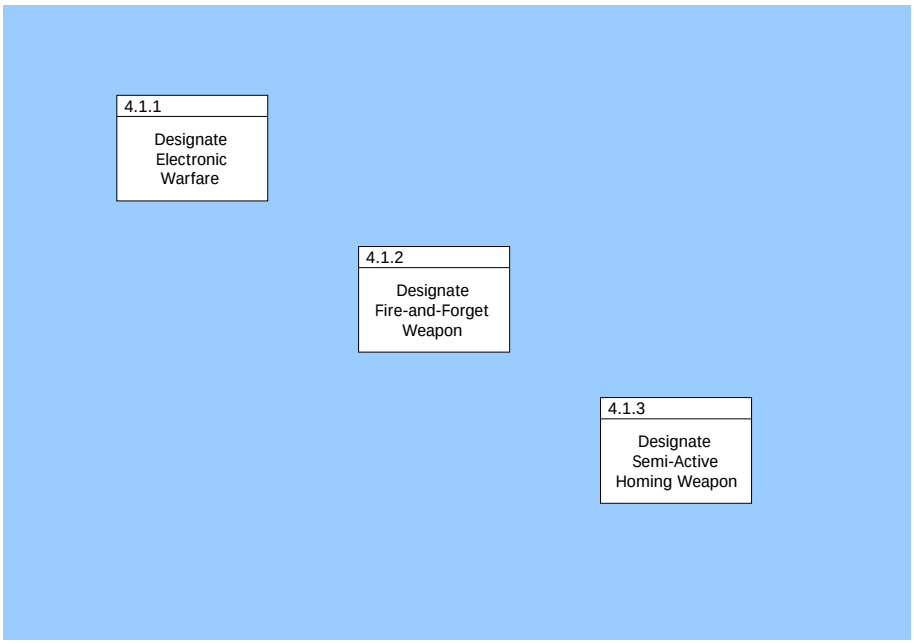


Figure 30 Select/Manage Weapons IDEF0 Diagram

4.1.1 Designate Electronic Warfare

Duration: 10.0

4.1.2 Designate Fire-and-Forget Weapon

Duration: 5.0

4.1.3 Designate Semi-Active Homing Weapon

8 Item Dictionary

Altitude (Item)

Transferred By Interface Link:
- Sensor Command and Control
- Sensor Track

Bearing (Item)

Transferred By Interface Link:
- Sensor Command and Control
- Sensor Track

Composite ID Message (Item)

Input To:
- 3.0 Command and Control

Triggers:
- 3.0 Command and Control

Output From:
- 1.1 Horizon Search
- 2.0 Tracking

Transferred By Interface Link:
- Command and Control to Engage
- Track to Command and Control

Control Degraded (Item)

Input To:
- 5.0 Sustainment
- 5.2 Mission Readiness
- 5.3 Provide Maintenance

Output From:
- 3.0 Command and Control

Transferred By Interface Link:
- Mission Readiness Status

Control Ready (Item)

Input To:
- 5.0 Sustainment
- 5.2 Mission Readiness

Output From:
- 3.0 Command and Control

8 Item Dictionary

Data Entry Screen

Transferred By Interface Link:
Personnel Status

Engage Degraded (Item)

Input To:
5.0 Sustainment
5.2 Mission Readiness
5.3 Provide Maintenance

Output From:
4.0 Engage

Transferred By Interface Link:
Mission Readiness Status

Engage Order (Item)

Input To:
4.0 Engage

Triggers:
4.0 Engage

Output From:
3.0 Command and Control

Transferred By Interface Link:
Command and Control to Engage

Engage Ready (Item)

Input To:
3.0 Command and Control
5.0 Sustainment
5.2 Mission Readiness

Output From:
4.0 Engage

Transferred By Interface Link:
Command and Control to Engage

EW Data (Item)

Transferred By Interface Link:
Sensor Command and Control
Sensor Track

8 Item Dictionary

IR Signature (Item)

Transferred By Interface Link:
Sensor Command and Control
Sensor Track

Kill Assessment Request (Item)

Input To:
3.0 Command and Control

Triggers:
3.0 Command and Control

Output From:
4.0 Engage

Transferred By Interface Link:
Command and Control to Engage

Kill Confirm (Item)

Input To:
3.0 Command and Control

Triggers:
3.0 Command and Control

Output From:
4.0 Engage

No Engage (Item)

Transferred By Interface Link:
Command and Control to Engage

Personnel Log On

Transferred By Interface Link:
Personnel Status

Range (Item)

Output From:
1.3.1 Determine Position

Transferred By Interface Link:
Sensor Command and Control
Sensor Track

Raw Data Items

Input To:

8 Item Dictionary

1.1 Horizon Search
1.3.1 Determine Position

Triggers:
2.0 Tracking

Output From:
1.0 Search and Detect
1.1.1 Provide Radar Surveillance
1.1.2 Perform EO/IR Search
1.1.3 Perform EW Search

Transferred By Interface Link:
Sensor Command and Control
Sensor Track

Safe to Train

Input To:
5.0 Sustainment

Transferred By Interface Link:
Training Status

Scenario Data

Transferred By Interface Link:
Training Status

Search Degraded (Item)

Input To:
5.0 Sustainment
5.2 Mission Readiness
5.3 Provide Maintenance

Output From:
1.0 Search and Detect

Transferred By Interface Link:
Sensor Command and Control

Search Ready (Item)

Input To:
5.0 Sustainment
5.2 Mission Readiness

Output From:
1.0 Search and Detect

Transferred By Interface Link:
Sensor Command and Control

8 Item Dictionary

Search Sector Message (Item)

Transferred By Interface Link:
- Sensor Track
- Track to Command and Control

Sensor Status (Item)

Transferred By Interface Link:
- Sensor Command and Control
- Sensor Track
- Track to Command and Control

Training Enable

Output From:
- 5.0 Sustainment

Transferred By Interface Link:
- Training Status

Training Status

Output From:
- 5.0 Sustainment

Transferred By Interface Link:
- Training Status

Weapon Status

Output From:
- 5.2 Mission Readiness

Transferred By Interface Link:
- Command and Control to Engage
- Mission Readiness Status

Weapons Ready (Item)

Transferred By Interface Link:
- Command and Control to Engage

9 Resources

KSLOC

Amount Type: Float

Initial Amount: Normal (μ: 200.0, stdDev: 1.0, stream: 1)

Maximum Amount: Normal (μ: 200.0, stdDev: 1.0, stream: 1)

Amount Units: Lines of code in 1000s

Consumed By:

1.1.1 Provide Radar Surveillance
Acquire Available: true
Amount: 1.0

1.1.2 Perform EO/IR Search
Acquire Available: true
Amount: 3.0

1.1.3 Perform EW Search
Acquire Available: true
Amount: 5.0

1.2.2 Perform Manual Search
Acquire Available: true
Amount: 10.0

1.3.1 Determine Position
Acquire Available: true
Amount: 18.0

1.3.2 Provide Track Data
Acquire Available: true
Amount: 10.0

2.1.1 Perform Single Sensor Tracking
Acquire Available: true
Amount: 1.0

2.1.2 Perform Multiple Sensor Tracking
Acquire Available: true
Amount: 1.0

LAN Bandwidth

Amount Type: Integer

Initial Amount: Normal (μ: 20.0, stdDev: 1.0, stream: 1)

Maximum Amount: Normal (μ: 20.0, stdDev: 1.0, stream: 1)

9 Resources

Amount Units: Seconds

Consumed By:

1.0 Search and Detect
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 7.0, stream: 1)

2.0 Tracking
Acquire Available: true
Amount: 5.0

2.1.1 Perform Single Sensor Tracking
Acquire Available: true
Amount: 2.0

2.1.2 Perform Multiple Sensor Tracking
Acquire Available: true
Amount: 2.0

3.0 Command and Control
Acquire Available: true
Amount: 2.0

4.0 Engage
Acquire Available: true
Amount: 5.0

5.0 Sustainment
Acquire Available: true
Amount: 8.0

Assess Operational Situation
Acquire Available: true
Amount: Normal (μ: 5.0, stdDev: 1.0, stream: 1)

Attack Operational Targets
Acquire Available: true
Amount: Normal (μ: 4.0, stdDev: 1.0, stream: 1)

Collect Information on Operational Situation
Acquire Available: true
Amount: Normal (μ: 2.0, stdDev: 1.0, stream: 1)

Engage Tactical Targets
Acquire Available: true
Amount: Normal (μ: 2.0, stdDev: 1.0, stream: 1)

Evaluate Integrate Interpret Operational Information

9 Resources

Acquire Available: true
Amount: Normal (μ: 5.0, stdDev: 1.0, stream: 1)

Provide for Combat Identification
Acquire Available: true
Amount: Normal (μ: 3.0, stdDev: 1.0, stream: 1)

Operator

Amount Type: Float

Initial Amount: Normal (μ: 120.0, stdDev: 1.0, stream: 1)

Maximum Amount: Normal (μ: 120.0, stdDev: 1.0, stream: 1)

Amount Units: Operator interface time requirements in seconds

Consumed By:
1.0 Search and Detect
Acquire Available: true
Amount: 10.0

1.1.1 Provide Radar Surveillance
Acquire Available: true
Amount: Normal (μ: 2.0, stdDev: 1.0, stream: 1)

1.1.2 Perform EO/IR Search
Acquire Available: true
Amount: Normal (μ: 5.0, stdDev: 1.0, stream: 1)

1.1.3 Perform EW Search
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 1.0, stream: 1)

1.2.2 Perform Manual Search
Acquire Available: true
Amount: Normal (μ: 12.0, stdDev: 1.0, stream: 1)

1.3.1 Determine Position
Acquire Available: true
Amount: Normal (μ: 12.0, stdDev: 1.0, stream: 1)

1.3.2 Provide Track Data
Acquire Available: true
Amount: Normal (μ: 5.0, stdDev: 1.0, stream: 1)

2.0 Tracking
Acquire Available: true
Amount: 5.0

9 Resources

2.1.1 Perform Single Sensor Tracking
Acquire Available: true
Amount: 3.0

2.1.2 Perform Multiple Sensor Tracking
Acquire Available: true
Amount: 3.0

3.0 Command and Control
Acquire Available: true
Amount: 7.0

4.0 Engage
Acquire Available: true
Amount: 15.0

5.0 Sustainment
Acquire Available: true
Amount: 1.0

RAM Requirements

Amount Type: Float

Initial Amount: Normal (μ: 64.0, stdDev: 1.0, stream: 1)

Maximum Amount: Normal (μ: 64.0, stdDev: 1.0, stream: 1)

Amount Units: Mb

Consumed By:
1.0 Search and Detect
Acquire Available: true
Amount: 5.0

1.1.1 Provide Radar Surveillance
Acquire Available: true
Amount: Normal (μ: 3.0, stdDev: 1.0, stream: 1)

1.1.2 Perform EO/IR Search
Acquire Available: true
Amount: Normal (μ: 1.0, stdDev: 1.0, stream: 1)

1.1.3 Perform EW Search
Acquire Available: true
Amount: Normal (μ: 3.0, stdDev: 1.0, stream: 1)

1.2.2 Perform Manual Search
Acquire Available: true
Amount: Normal (μ: 4.0, stdDev: 1.0, stream: 1)

9 Resources

1.3.1 Determine Position
Acquire Available: true
Amount: Normal (μ: 6.0, stdDev: 1.0, stream: 1)

1.3.2 Provide Track Data
Acquire Available: true
Amount: Normal (μ: 4.0, stdDev: 1.0, stream: 1)

2.0 Tracking
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 6.0, stream: 1)

2.1.1 Perform Single Sensor Tracking
Acquire Available: true
Amount: 4.0

2.1.2 Perform Multiple Sensor Tracking
Acquire Available: true
Amount: 4.0

3.0 Command and Control
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 9.0, stream: 1)

4.0 Engage
Acquire Available: true
Amount: Normal (μ: 10.0, stdDev: 5.0, stream: 1)

5.0 Sustainment
Acquire Available: true
Amount: 4.0

Tracks

Amount Type: Float

Initial Amount: 0.0

Maximum Amount: 50.0

Amount Units: Number of Tracks in System

Captured By:
2.1 Perform Local Tracking
Acquire Available: true

Assess Operational Situation
Acquire Available: true

9 Resources

Consumed By:

3.1.5 Drop Air Tracks

Acquire Available: true

Produced By:

1.1.1 Provide Radar Surveillance

10 Components

Part I - Hierarchical Component List

1.0 Tactical System [WS 21200]
- 1.1 Surveillance Sensors [WS 21200]
  - 1.1.1 Radar
  - 1.1.2 EO/IR [WS21TYR]
  - 1.1.3 EW
- 1.2 Command and Control [WS 21340]
- 1.3 Fire Control System [WS-31333]

Part II - Component Definitions

1.0 Tactical System [WS 21200]

Description:
Composes System Components required to host and execute mission functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
SysML [AP 233]

Built From Lower-Level Component(s):
1.1 Surveillance Sensors [WS 21200]
1.2 Command and Control [WS 21340]
1.3 Fire Control System [WS-31333]

Joined To Logical Interface:
Combat Systems to C4I
Combat Systems to Ship

Joined Through Logical Interface:
Sustainment Radar Interface

Connected through Physical Link(s):
Track to Command and Control
Training Status

10 Components

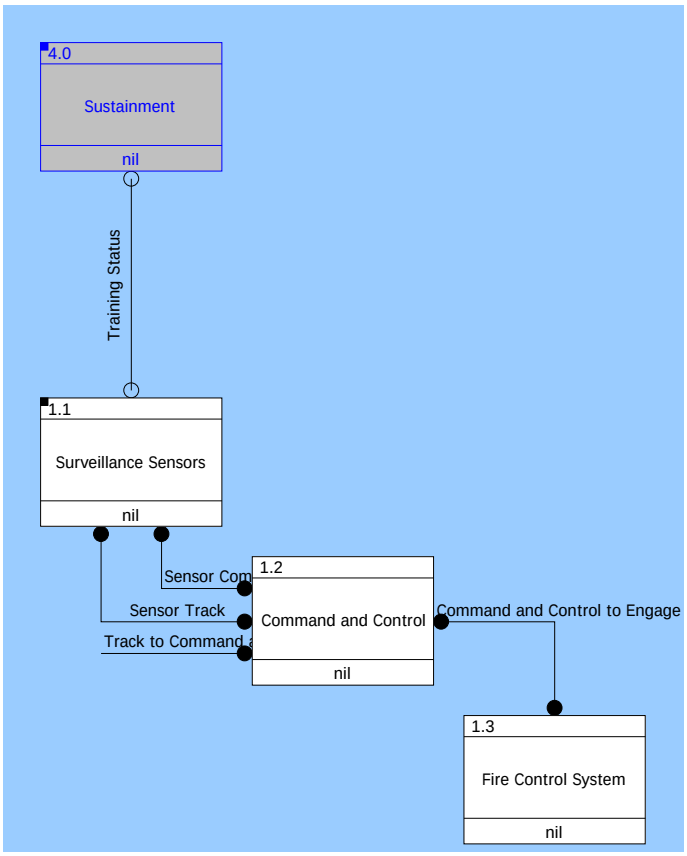


Figure 31 Tactical System Subcomponent Connectivity

Performs Function(s):
- 2.0 Tracking
- 3.0 Command and Control
- 4.0 Engage

Specified By:
- A Anti-Air Warfare
- D.1.3 Standardization

Source Documents:
- AAW Concept of Operations (ConOps)
- SSDD

1.1 Surveillance Sensors [WS 21200]

Description:
Composes System Components required to host and execute Sensor functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
- 1.0 Tactical System [WS 21200]

Built From Lower-Level Component(s):

10 Components

1.1.1 Radar
1.1.2 EO/IR [WS21TYR]
1.1.3 EW

Joined To Logical Interface:
Search C2 Interface
Search Track Interface
Training User Interface

Joined Through Logical Interface:
Sustainment Radar Interface

Connected to Physical Link(s):
Sensor Command and Control
Sensor Track

Connected through Physical Link(s):
Training Status

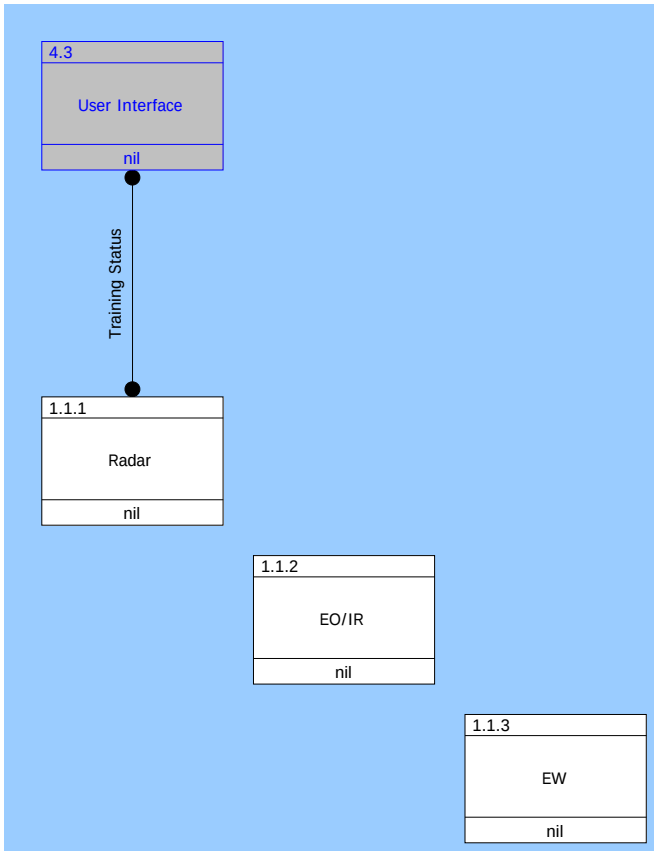


Figure 32 Surveillance Sensors Subcomponent Connectivity

Performs Function(s):
1.0 Search and Detect
1.1 Horizon Search

10 Components

1.1.1 Radar

Description:
Composes System Components required to host and execute Radio Frequency transmission and receive Sensor functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
1.1 Surveillance Sensors [WS 21200]

Joined To Logical Interface:
Sustainment Radar Interface

Connected to Physical Link(s):
Training Status

Performs Function(s):
1.1.1 Provide Radar Surveillance
1.3.1 Determine Position

Specified By:
A.3.1.3 Search Volume
A.3.1.3.1 Radar Horizon Range

1.1.2 EO/IR [WS21TYR]

Description:
Thermal Imaging processor for Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
1.1 Surveillance Sensors [WS 21200]

1.1.3 EW

Description:
Electronic Warfare component for Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
1.1 Surveillance Sensors [WS 21200]

1.2 Command and Control [WS 21340]

Description:
Composes System Components required to host and execute Track and Command and Control functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
1.0 Tactical System [WS 21200]

Joined To Logical Interface:
C2 to Engage Interface
Search C2 Interface
Search Track Interface
Training User Interface

10 Components

Connected to Physical Link(s):
Command and Control to Engage
Sensor Command and Control
Sensor Track
Track to Command and Control

Performs Function(s):
2.0 Tracking
3.0 Command and Control
4.0 Engage

1.3 Fire Control System [WS-31333]

Description:
Composes System Components required to host and execute Track and Command and Control functionality associated with meeting Pra for AAW, LAD, and Surveillance Missions.

Built In Higher-Level Component(s):
1.0 Tactical System [WS 21200]

Joined To Logical Interface:
C2 to Engage Interface
Training User Interface

Connected to Physical Link(s):
Command and Control to Engage

Performs Function(s):
4.0 Engage

11 Interfaces

Part I - Derived Functional Interfaces

Table 4 1.0 Tactical System External I/O

Functions	Interface Items	Interfacing Elements
3.0 Command and Control	→ Control Degraded (Item)	5.0 Sustainment 4.0 Sustainment [USD ATL] 5.2 Mission Readiness 4.2 Maintenance System [CNET]
	→ Control Ready (Item)	5.0 Sustainment 4.0 Sustainment [USD ATL] 5.2 Mission Readiness 4.2 Maintenance System [CNET]
4.0 Engage	→ Engage Degraded (Item)	5.0 Sustainment 4.0 Sustainment [USD ATL] 5.2 Mission Readiness 4.2 Maintenance System [CNET]
	→ Engage Ready (Item)	5.0 Sustainment 4.0 Sustainment [USD ATL] 5.2 Mission Readiness 4.2 Maintenance System [CNET]

Table 5 1.2 Command and Control External I/O

Functions	Interface Items	Interfacing Elements
2.0 Tracking	← Raw Data Items	1.0 Search and Detect 1.1 Surveillance Sensors [WS 21200] 1.1.1 Provide Radar Surveillance 1.1.1 Radar
3.0 Command and Control	→ Control Degraded (Item)	5.0 Sustainment 4.0 Sustainment [USD ATL] 5.2 Mission Readiness 4.2 Maintenance System [CNET]
	→ Control Ready (Item)	5.0 Sustainment 4.0 Sustainment [USD ATL] 5.2 Mission Readiness 4.2 Maintenance System [CNET]
	→ Engage Order (Item)	4.0 Engage 1.3 Fire Control System [WS-31333]

11 Interfaces

Table 5 1.2 Command and Control External I/O

Functions	Interface Items	Interfacing Elements
	← Composite ID Message (Item)	1.1 Horizon Search 1.1 Surveillance Sensors [WS 21200]
	← Engage Ready (Item)	4.0 Engage 1.3 Fire Control System [WS-31333]
	← Kill Assessment Request (Item)	4.0 Engage 1.3 Fire Control System [WS-31333]
	← Kill Confirm (Item)	4.0 Engage 1.3 Fire Control System [WS-31333]
4.0 Engage	→ Engage Degraded (Item)	5.0 Sustainment 4.0 Sustainment [USD ATL] 5.2 Mission Readiness 4.2 Maintenance System [CNET]
	→ Engage Ready (Item)	5.0 Sustainment 4.0 Sustainment [USD ATL] 5.2 Mission Readiness 4.2 Maintenance System [CNET]

Table 6 1.3 Fire Control System External I/O

Functions	Interface Items	Interfacing Elements
4.0 Engage	→ Engage Degraded (Item)	5.0 Sustainment 4.0 Sustainment [USD ATL] 5.2 Mission Readiness 4.2 Maintenance System [CNET]
	→ Engage Ready (Item)	3.0 Command and Control 1.2 Command and Control [WS 21340] 5.0 Sustainment 4.0 Sustainment [USD ATL] 5.2 Mission Readiness 4.2 Maintenance System [CNET]
	→ Kill Assessment Request (Item)	3.0 Command and Control 1.2 Command and Control [WS 21340]
	→ Kill Confirm (Item)	3.0 Command and Control 1.2 Command and Control [WS

11 Interfaces

Table 6 1.3 Fire Control System External I/O

Functions	Interface Items	Interfacing Elements
		21340]
	← Engage Order (Item)	3.0 Command and Control 1.2 Command and Control [WS 21340]

Part II - Logical Interfaces

C2 to Engage Interface

Physical Links:
Command and Control to Engage

Connecting Elements:
1.2 Command and Control [WS 21340]
1.3 Fire Control System [WS-31333]

Specified By:
A.2.1.3.1 SD Select Weapon
A.2.1.3.2 SD Manage Weapon

Combat Systems to C4I

Connecting Elements:
1.0 Tactical System [WS 21200]
SysML [AP 233]

Specified By:
A.2.1.2.1 SD Manage Tracks

Combat Systems to Ship

Connecting Elements:
1.0 Tactical System [WS 21200]
SysML [AP 233]

Search C2 Interface

Physical Links:
Sensor Command and Control

Connecting Elements:
1.1 Surveillance Sensors [WS 21200]
1.2 Command and Control [WS 21340]

Specified By:
C SD Threat Environment

11 Interfaces

Search Track Interface

Physical Links:
Sensor Track

Connecting Elements:
1.1 Surveillance Sensors [WS 21200]
1.2 Command and Control [WS 21340]

Sustainment Radar Interface

Connecting Elements:
1.1.1 Radar
1.1 Surveillance Sensors [WS 21200]
1.0 Tactical System [WS 21200]
4.0 Sustainment [USD ATL]

Specified By:
D.1 Readiness

Training User Interface

Physical Links:
Personnel Status
Training Status

Connecting Elements:
1.1 Surveillance Sensors [WS 21200]
1.2 Command and Control [WS 21340]
1.3 Fire Control System [WS-31333]
4.1 Training System

Specified By:
D.1.4.1 Training

Part III - Physical Interfaces

Command and Control to Engage

Transmitted Data:
Composite ID Message (Item)
Engage Order (Item)
Engage Ready (Item)
Kill Assessment Request (Item)
No Engage (Item)
Weapon Status
Weapons Ready (Item)

Connecting Elements:
1.2 Command and Control [WS 21340]
1.3 Fire Control System [WS-31333]

11 Interfaces

Personnel Status

Transmitted Data:
- Data Entry Screen
- Personnel Log On

Connecting Elements:
- 4.1 Training System
- 4.2 Maintenance System [CNET]

Sensor Command and Control

Transmitted Data:
- Altitude (Item)
- Bearing (Item)
- EW Data (Item)
- IR Signature (Item)
- Range (Item)
- Raw Data Items
- Search Degraded (Item)
- Search Ready (Item)
- Sensor Status (Item)

Connecting Elements:
- 1.1 Surveillance Sensors [WS 21200]
- 1.2 Command and Control [WS 21340]

Sensor Track

Transmitted Data:
- Altitude (Item)
- Bearing (Item)
- EW Data (Item)
- IR Signature (Item)
- Range (Item)
- Raw Data Items
- Search Sector Message (Item)
- Sensor Status (Item)

Connecting Elements:
- 1.1 Surveillance Sensors [WS 21200]
- 1.2 Command and Control [WS 21340]

Track to Command and Control

Transmitted Data:
- Composite ID Message (Item)
- Search Sector Message (Item)
- Sensor Status (Item)

Connecting Elements:

11 Interfaces

1.2 Command and Control [WS 21340]
1.0 Tactical System [WS 21200]
SysML [AP 233]

Training Status

Transmitted Data:
Safe to Train
Scenario Data
Training Enable
Training Status

Connecting Elements:
1.1.1 Radar
1.1 Surveillance Sensors [WS 21200]
1.0 Tactical System [WS 21200]
4.3 User Interface [OJ-394]
4.0 Sustainment [USD ATL]

12 Requirements Traceability Matrix (RTM)

Allocated Capabilities/Requirements	Traced From Higher-Level Elements
1.0 Tactical System (Component)	
2.0 Tracking (Function)	A.1.1.1 LAD Track (Requirement)
3.0 Command and Control (Function)	A.1.1.2 LAD Command and Control (Requirement)
4.0 Engage (Function)	A.1.1.3 LAD Engage (Requirement)
A Anti-Air Warfare (Requirement)	
A.2.1.2.1 SD Manage Tracks (Requirement)	A.2 Self Defense (Requirement) A Anti-Air Warfare (Requirement)
D.1.3 Standardization (Requirement)	D Supportability Performance Range Threshold and Objectives (Requirement) A Anti-Air Warfare (Requirement)
1.1 Surveillance Sensors (Component)	
1.0 Search and Detect (Function)	A.1.1.1 LAD Track (Requirement)
1.1 Horizon Search (Function)	A.3.1.1 Search Horizon (Requirement)
A.3.1.1 Search Horizon (Requirement)	A.3 Surveillance (Requirement) A Anti-Air Warfare (Requirement)
C SD Threat Environment (Requirement)	A.2 Self Defense (Requirement) A Anti-Air Warfare (Requirement)
D.1.4.1 Training (Requirement)	D Supportability Performance Range Threshold and Objectives (Requirement) A Anti-Air Warfare (Requirement)
1.1.1 Radar (Component)	
1.1.1 Provide Radar Surveillance (Function)	1.1 Horizon Search (Function) A.3.1.1 Search Horizon (Requirement)
1.3.1 Determine Position (Function)	1.3 Perform Target Detection (Function) A.3.1.1 Search Horizon (Requirement)
A.3.1.3 Search Volume (Requirement)	A.3 Surveillance (Requirement) A Anti-Air Warfare (Requirement)
A.3.1.3.1 Radar Horizon Range (Requirement)	A.3 Surveillance (Requirement) A Anti-Air Warfare (Requirement)
D.1 Readiness (Requirement)	D Supportability Performance Range Threshold and Objectives (Requirement) A Anti-Air Warfare (Requirement)
1.1.2 EO/IR (Component)	
1.1.3 EW (Component)	
1.2 Command and Control (Component)	
2.0 Tracking (Function)	A.1.1.1 LAD Track (Requirement)
3.0 Command and Control (Function)	A.1.1.2 LAD Command and Control (Requirement)

12 Requirements Traceability Matrix (RTM)

Allocated Capabilities/Requirements	Traced From Higher-Level Elements
4.0 Engage (Function)	A.1.1.3 LAD Engage (Requirement)
A.2.1.3.1 SD Select Weapon (Requirement)	A.2 Self Defense (Requirement) A Anti-Air Warfare (Requirement)
A.2.1.3.2 SD Manage Weapon (Requirement)	A.2 Self Defense (Requirement) A Anti-Air Warfare (Requirement)
C SD Threat Environment (Requirement)	A.2 Self Defense (Requirement) A Anti-Air Warfare (Requirement)
D.1.4.1 Training (Requirement)	D Supportability Performance Range Threshold and Objectives (Requirement) A Anti-Air Warfare (Requirement)
1.3 Fire Control System (Component)	
4.0 Engage (Function)	A.1.1.3 LAD Engage (Requirement)
A.2.1.3.1 SD Select Weapon (Requirement)	A.2 Self Defense (Requirement) A Anti-Air Warfare (Requirement)
A.2.1.3.2 SD Manage Weapon (Requirement)	A.2 Self Defense (Requirement) A Anti-Air Warfare (Requirement)
D.1.4.1 Training (Requirement)	D Supportability Performance Range Threshold and Objectives (Requirement) A Anti-Air Warfare (Requirement)

Acronym	**Definition**
AAW	Anti-Air Warfare
AoA	Analysis of Alternatives
BIT	Built-In-Test
C4I	Command, Control, Communications, Computers, & Intelligence
CBA	Capabilities Based Document
CM	Configuration Management
Context sensitive modeling (Clymer)	Context Sensitive Modeling
COTS	Commercial-Off-The-Shelf
DDG (1000)	Destroyer, Guided Missile, Next Generation
DoD	Department of Defense
DODAF	Department of Defense Architecture Framework
Domain analysis	Domain analysis
EFFBD	Enhanced Functional Flow Block Diagram
FAA	Functional Area Analysis
FFBD	Functional Flow Block Diagram
FNA	Functional Needs Analysis
FSA	Functional Solutions Analysis
GPR	Government Purpose Rights
ICD	Initial Capabilities Document
IDEF0	Integration Definition for Function Modeling
IDFM	Integrated Definition for Function Modeling
INCOSE	International Council on Systems Engineering
IPPD	Integrated Product and Process Development
IPR	In Process Review
IPT	Integrated Product Team
IWS	Integrated Warfare Systems
JCIDS	Joint Capabilities Integration Development System
KPP	Key Performance Parameter
LCC	Life Cycle Cost
LCC	Life Cycle Cost Estimate
LCCE	Life Cycle Support Plan
M&S	Modeling and Simulation
MIL-STD	Military Standard
Model-based systems engineering	Model-based systems engineering
MOE	Measure Of Effectiveness
MOP	Measure Of Performance
MORS	The Modular Command Evaluation Structure (MORS)

MSSE	Masters of Science in Systems Engineering
MSSEM	Masters of Science in Systems Engineering Management
NAVSEA	Naval Sea Systems Command
Net-centric architectures	Net-Centric Architectures
NOA	Naval Open Architecture
NPS	Naval Postgraduate School
NSWC	Naval Surface Warfare Center
OA	Open Architecture
OPNAV	Office of the Chief of Naval Operations
PBL	Performance Based Logistics
PEO	Program Executive Office; Program Executive Officer
PESHE	Programmatic Environmental, Safety, and Health Evaluation
PHD	Port Hueneme Division
PIA	Post Independent Analysis
PMP	Project Management Plan
POC	Point of Contact
Reliability Theory	Reliability Theory
SE	Systems Engineering
SOA	Service Oriented Architecture
SPAWAR	Space and Naval Warfare
SPL	Software Product Line
SSDS	Ship Self Defense System
Supportability	Supportability
Systems Architecture and Requirements Engineering (Hatley et al)	Systems Architecture and Requirements Engineering (Hatley et al)
The Systems Engineering "VEE" model	VEE Model
UML	Unified Modeling Language

Term	**Definition**
Analysis of Alternatives	The evaluation of the performance, operational effectiveness, operational suitability, and estimated costs of alternative systems to meet a mission capability. The analysis assesses the advantages and disadvantages of alternatives being considered to satisfy capabilities, including the sensitivity of each alternative to possible changes in key assumptions or variables. The AoA is normally conducted during the Concept Refinement phase of the Defense Acquisition Framework to refine the system concept contained in the Initial Capabilities Document (ICD) approved at the Concept Decision. (DoDI 5000.2 and CJCSI 3170.01F)
Anti-Air Warfare	A US Navy/US Marine Corps term used to indicate that action required to destroy or reduce to an acceptable level the enemy air and missile threat. It includes such measures as the use of interceptors, bombers, antiaircraft guns, surface-to-air and air-to-air missiles, electronic attack and destruction of the air or missile threat both before and after it is launched.
Built-In-Test	An integral capability of the mission system or system which provides an automated test capability to detect, diagnose, or isolate failures.
Capabilities Based Document	Document that is based on capabilities of the system (could not find a good definition)
Command, Control, Communications, Computers, & Intelligence	*C4I Systems* provides the latest information on the systems that support the command and control of operations, from strategic command to tactical battle management, of countries all around the world. Each entry provides detailed and accurate descriptions of the system or equipment, development and operational status, technical specifications for appraisal and comparison, and manufacturer details to aid procurement.
Commercial-Off-The-Shelf	Commercial items that require no unique government modifications or maintenance over the life cycle of the product to meet the needs of the procuring agency.
Configuration Management	The technical and administrative direction and surveillance actions taken to identify and document the functional and physical characteristics of a Configuration Item (CI), to control changes to a CI and its characteristics, and to record and report change processing and implementation status. It provides a complete audit trail of decisions and design modifications.
Department of Defense	DoD is the federal department charged with coordinating and supervising all agencies and functions of the government relating directly to national security and the military. The organization and functions of the DOD are set forth in Title 10 of the United States Code.
Department of Defense Architecture Framework	The Department of Defense Architecture Framework (DoDAF) defines how to organize the specification of enterprise architectures for U.S. Department of Defense (DoD) applications. All major DoD weapons and information technology system procurements are required to document their enterprise architectures using the view products prescribed by the DoDAF. DoDAF is well suited to large systems and systems-of-systems (SoSs) with complex integration and interoperability issues.
Destroyer, Guided Missile, Next Generation	Developed under the DD(X) destroyer program, DDG-1000 Zumwalt is the lead ship in a class of next-generation, multi-mission surface combatants tailored for land attack and littoral dominance, with capabilities designed to defeat current and projected threats as well as improve battle force defense.

Domain analysis	Domain analysis is "the process of identifying, collecting, organizing, and representing the relevant information in a domain, based upon the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory, and emerging technology within a domain. Def from CMU/SEI-90-TR-21
Enhanced Functional Flow Block Diagram	Functional Flow Block Diagrams provide a hierarchical decomposition of the system's functions and show a control structure that dictates the order in which the functions can be executed at each level of decomposition. The enhanced FFBD enables you to see how the inputs and output affect the functional sequencing. The Engineering Design of Systems, Dennis Buede, DoDAF, Steven Dam
Functional Area Analysis	Identifies the mission area or mission problem to be assessed, the concepts to be examined, the timeframe in which the problem is being assessed, and the scope of the assessment, and describes the relevant objectives and concept of operations (ConOps) or concepts and the relevant effects to be generated. (CJCSI 3170.01F)
Functional Flow Block Diagram	Shows functional flow including control logic. FFBD is a multi-tier, time-sequenced, step-by-step flow diagram of a system's functional flow
Functional Needs Analysis	Assesses the ability of the current and programmed warfighting systems to deliver the capabilities the Functional Area Analysis (FAA) identified under the full range of operating conditions and to the designated measures of effectiveness. The FNA produces a list of capability gaps that require solutions and indicates the time frame in which those solutions are needed. It may also identify redundancies in capabilities that reflect inefficiencies. (CJCSI 3170.01F)
Functional Solutions Analysis	Operationally based assessment of all potential Doctrine, Organization, Training, Materiel, Leadership, Personnel, and Facilities (DOTMLPF) approaches to solving (or mitigating) one or more of the capability gaps (needs) previously identified.
Government Purpose Rights	In Defense Department acquisitions, the resulting contract can permit delivery of technical data and computer software using a "middle way," known as Government Purpose Rights, which is an Intellectual Property licensing system that is available to DOD acquisitions. Government Purpose Rights ("GPR") lie somewhere between the broad Unlimited Rights license rights allowing unrestricted Government release of information and the more restrictive Limited or Restricted Rights licensing rights that forbid most releases outside the Government.
In Process Review	Interim Program or Progress Review
Initial Capabilities Document	Documents the need for a materiel approach, or an approach that is a combination of materiel and non-materiel, to satisfy specific capability gap(s). The ***ICD*** defines the gap in terms of the functional area; the relevant range of military operations; desired effects; time and Doctrine, Organization, Training, Materiel, Leadership and education, Personnel, and Facilities (DOTMLPF); and policy implications and constraints. The outcome of an ***ICD*** could be one or more DOTMLPF Change Recommendations (DCRs) or Capability Development Documents. (CJCSI 3170.01F)
Integrated Definition for Function Modeling	IDEFO models the decisions, actions and activities of a system in order to communicate the functional perspective of the system.

Integrated Product and Process Development	IPPD is the DoD management technique that simultaneously integrates all essential acquisition activities through the use of multidisciplinary teams to optimize design, manufacturing, and supportability processes. One of the key IPPD tenets is multidisciplinary teamwork through Integrated Product Teams.
Integrated Product Team	IPTs are an integral part of the Defense acquisition oversight and review process. For Acquisition Category ID and IAM programs, there are generally two levels of IPT: the Overarching Integrated Product Team and the Working-level Integrated Product Team(s). Each program should have an OIPT and at least one WIPT. WIPTs should focus on a particular topic such as cost/performance, test, or contracting. An Integrating Integrated Product Team (IIPT), which is itself a WIPT, should coordinate WIPT efforts and cover all topics not otherwise assigned to another IPT. IPT participation is the primary way for any organization to participate in the acquisition program. DAU
Integrated Warfare Systems	Systems that delivers Enterprise solutions for Naval warfare systems that operate seamlessly and effectively within the Fleet and Joint Force.
Integration Definition for Function Modeling	Integration Definition for Function Modeling
International Council on Systems Engineering	INCOSE is a not-for-profit membership organization founded in 1990. The mission is to advance the state of the art and practice of systems engineering in industry, academia, and government by promoting interdisciplinary, scalable approaches to produce technologically appropriate solutions that meet societal needs.
Joint Capabilities Integration Development System	The **Joint Capabilities Integration Development System**, or **JCIDS**, is the formal United States Department of Defense (DoD) procedure which defines acquisition requirements and evaluation criteria for future defense programs. JCIDS was created to replace the previous service-specific requirements generation system, which allegedly created redundancies in capabilities and failed to meet the combined needs of all US military services. In order to correct these problems, JCIDS is intended to guide the development of requirements for future acquisition systems to reflect the needs of all four services (Army, Navy, Marines, and Air Force) by focusing the requirements generation process on needed *capabilities* as requested or defined by one of the US combatant commanders. In the JCIDS process, regional and functional combatant commanders give feedback early in the development process to ensure that their requirements are met. (Wikipedia)
Key Performance Parameter	Those attributes or characteristics of a system that are considered critical or essential to the development of an effective military capability and those attributes that make a significant contribution to the key characteristics as defined in the Joint Operations Concept. KPPs are validated by the Joint Requirements Oversight Council (JROC) for JROC Interest documents, and by the DoD Component for Joint Integration or Independent documents. The Capability Development Document (CDD) and the Capability Production Document (CPD) KPPs are included verbatim in the Acquisition Program Baseline (APB). (CJCSI 3170.01E)
Life Cycle Cost	The total cost to the government of acquisition and ownership of that system over its useful life. It includes the cost of development, acquisition, operations, and support (to include manpower), and where applicable, disposal. For defense systems, LCC is also called Total Ownership Cost (TOC).
Life Cycle Cost Estimate	Estimate of the cost of a program from cradle to grave

Life Cycle Support Plan	The total phases through which an item passes from the time it is initially developed until the time it is either consumed in use or disposed of as being excess to all known materiel requirements. The plan covers the entire period.
Masters of Science in Systems Engineering	A rigorous study and an Advanced degree for Systems Engineering offered by the Naval Post Graduate School
Masters of Science in Systems Engineering Management	A rigorous study and an Advanced degree for Systems Engineering Management that incorporates the Systems Engineering and Supportability aspects of a System offered by the Naval Post Graduate School
Measure Of Effectiveness	Measure designed to correspond to accomplishment of mission objectives and achievement of desired results. (CJCSI 3170.01E) MOEs may be further decomposed into Measures of Performance and Measures of Suitability. See operational effectiveness, Measure of Performance, operational suitability, and Measure of Suitability.
Measure Of Performance	Measure of a system s performance expressed as speed, payload, range, time on station, frequency, or other distinctly quantifiable performance features. Several MOPs and/or Measures of Suitability may be related to the achievement of a particular Measure of Effectiveness (MOE). See Measure of Suitability, operational suitability, and Measure of Effectiveness.
Military Standard	A United States **Defense Standard**, often called a **military standard**, "**MIL-STD**", "**MIL-SPEC**", or (informally) "**MilSpecs**", is used to help achieve standardization objectives by the U.S. Department of Defense. Standardization is beneficial in achieving interoperability, ensuring products meet certain requirements, commonality, reliability, total cost of ownership, compatibility with logistics systems, and similar defense-related objectives.
Model-based systems engineering	"Model-based systems engineering is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing through out development and later life cycle phases". INCOSE, Systems Engineering Vision 2020, Version 2.03, TP-2004-004-02 Sept 2007
Modeling and Simulation	Modeling and Simulation is a discipline for developing a level of understanding of the interaction of the parts of a system, and of the system as a whole.
Naval Open Architecture	The Navy OA is a systems design approach supported by verifiable governmental testing platforms, such as the OACE, that seeks to implement open specifications for interfaces, services and supporting formats. It enables software components to work across a range of systems and interoperate with other software components on local and remote systems
Naval Postgraduate School	Advanced studies focused on DoD
Naval Sea Systems Command	Surface Command of Fleet for DoD
Naval Surface Warfare Center	Field activity that is part of Naval Sea Systems Command
Net-Centric Architectures	Enables information sharing by connecting people and systems who have information with people/systems who need information. It includes situational awareness, self-synchronizing ops, information pull, collaboration, shared data, bandwith on demand, diverse routing, Enterprise services
Office of the Chief of Naval Operations	The Chief of Naval Operations (CNO) is the senior military officer in the Navy. The CNO is a four-star admiral and is responsible to the Secretary of the Navy for the command, utilization of resources and operating efficiency of the operating forces of the Navy and of the Navy shore activities assigned by the Secretary.

Open Architecture	The confluence of business and technical practices yielding modular, interoperable systems that adhere to open standards with published interfaces
Performance Based Logistics	The preferred sustainment strategy for weapon system product support that employs the purchase of support as an integrated, affordable performance package designed to optimize system readiness. PBL meets performance goals for a weapon system through a support structure based on long-term performance agreements with clear lines of authority and responsibility.
Point of Contact	Person serving as coordinator, action officer, or focal point for an activity.
Port Hueneme Division	Division of the NSWC under the Naval Sea Systems Command
Post Independent Analysis	Post Independent Analysis. The final step in the JCIDS analysis process is the PIA. In this step, the sponsor will assess the compiled information and analysis results of the FSA (non-materiel and materiel approaches) to ensure the list of approaches with the potential to deliver the capability identified in the FAA and FNA is complete. The sponsor team performing the PIA shall be made up of individuals who were not involved in the FSA. This information will be compiled into an appropriate recommendation and documented in an ICD or joint DCR
Program Executive Office; Program Executive Officer	The military or civilian official who has responsibility for directing several Major Defense Acquisition Programs (MDAPs) and for assigned major system and non-major system acquisition programs. A PEO has no other command or staff responsibilities within the Component, and only reports to and receives guidance and direction from the DoD Component Acquisition Executive (CAE).
Programmatic Environmental, Safety, and Health Evaluation	PESHE incorporates all the environmental, safety and health regulations into a document that is required by Milestone B and is updated through each acquisition phase.
Project Management Plan	Project management is a carefully planned and organized effort to accomplish a specific (and usually) one-time effort, for example, construct a building or implement a new computer system. Project management includes developing a project plan, which includes defining project goals and objectives, specifying tasks or how goals will be achieved, what resources are need, and associating budgets and timelines for completion. It also includes implementing the project plan, along with careful controls to stay on the "critical path", that is, to ensure the plan is being managed according to plan. Project management usually follows major phases (with various titles for these phases), including feasibility study, project planning, implementation, evaluation and support/maintenance
Reliability Theory	Reliability theory is a general theory about systems failure. It allows researchers to predict the age-related failure kinetics for a system of given architecture (reliability structure) and given reliability of its components. Reliability theory predicts that even those systems that are entirely composed of non-aging elements (with a constant failure rate) will nevertheless deteriorate (fail more often) with age, if these systems are *redundant* in irreplaceable elements. Aging, therefore, is a direct consequence of systems redundancy. Reliability theory also predicts the late-life mortality deceleration with subsequent leveling-off, as well as the late-life mortality plateaus, as an inevitable consequence of *redundancy exhaustion* at extreme old ages.

Service Oriented Architecture	In computing, **service-oriented architecture** (**SOA**) provides methods for systems development and integration where systems group functionality around business processes and package these as *interoperable services*. SOA also describes IT infrastructure which allows different applications to exchange data with one another as they participate in business processes. Service-orientation aims at a *loose coupling* of services with operating systems, programming languages and other technologies which underlie applications
Ship Self Defense System	**Ship's Self Defense System** (**SSDS**) is an integrated weapons system used aboard large U.S. Navy ships, such as Nimitz class super carriers and various amphibious assault ships, such as LHDs and LSDs. SSDS has similar attributes to the combat system used aboard DDGs and CGs, in that it is an integrated Combat Direction System (CDS). The combat direction systems aboard DDGs and CGs, Aegis, are purpose built integrated designs from the outset. SSDS follows a different approach and uses existing shipboard radars and weapons systems, integrated under a COTS framework, to provide a cohesive CDS. The first SSDS designs were back fit onto existing U.S. Navy ships.
Software Product Line	A software product line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. Def from SW Eng Institute, CMU
Space and Naval Warfare	Space and Naval Warfare Systems Center Pacific (SSC Pacific) is responsible for development of the technology to collect, transmit, process, display and, most critically, manage information essential to successful military operations. The Center develops the capabilities that allow decision-makers of the Navy, and increasingly of the joint services, to carry out their operational missions and protect their forces.
Supportability	A key component of availability. It includes design, technical support data, and maintenance procedures to facilitate detection, isolation, and timely repair and/or replacement of system anomalies. This includes factors such as diagnostics, prognostics, real time maintenance data collection, and human system integration considerations. (CJCSI 3170.01E)
Systems Architecture and Requirements Engineering (Hatley et al)	Broad approach to the effective development of systems.
Systems Engineering	The overarching process that a program team applies to transition from a stated capability to an operationally effective and suitable system. SE encompasses the application of SE processes across the acquisition life cycle (adapted to each and every phase) and is intended to be the integrating mechanism for balanced solutions addressing capability needs, design considerations and constraints, as well as limitations imposed by technology, budget, and schedule. The SE processes are applied early in concept definition, and then continuously throughout the total life cycle. (Defense Acquisition Guidebook)
Unified Modeling Language	is a standardized general-purpose modeling language in the field of software engineering. UML includes a set of graphical notation techniques to create abstract models of specific systems.

VEE Model	The Vee model is rooted in the project cycle, which is displayed from left to right to represent project time and maturity. Coupled with this depiction is the recognition of levels of decomposition, which are illustrated, in the vertical dimension from top to bottom. The User is at the highest level and parts and lines of code the lowest. The plane orthogonal to the plane of the Vee illustrates the number of entities at each level of decomposition, which relates to the complexity of the system. INCOSE definition

ACRONYMS LIST

Acronym	Definition
AAW	Anti-Air Warfare
ABD	Architecture Block Diagram
AC	Air Contacts
ACAT	Acquisition Category
ACD	Architecture Context Diagram
AFCD	Architecture Flow Context Diagram
AFD	Architecture Flow Diagram
AFD0	Architecture Flow Diagram level 0
AID	Architecture Interconnection Diagram
A_O	Operational Availability
AoA	Analysis of Alternatives
APL	Applied Physics Laboratory
AP233	A STEP-based data exchange standard targeted to support the needs of the systems engineering community.
A-RCI	Acoustic – Rapid Commercial Off The Shelf Insertion
ASCM	Anti-Ship Cruise Missile
ASM	Anti-Ship Missile
ASMD	Anti-Ship Missile Defense
ASN	Assistant Secretary of the Navy
ASR	Alternative System Review
ASW	Anti-Submarine Warfare
AT&L	Acquisition, Technology, & Logistics
AV	All View
BDD	Block Definition Diagram
BIT	Built-In-Test
C2	Command and Control
C4I	Command, Control, Communications, Computers, & Intelligence
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
C5ISR	Command, Control, Communications, Computers, Combat Systems, Intelligence, Surveillance and Reconnaissance
CAP	Combat Air Patrol

Acronym	Definition
CASE	Computer Aided Systems Engineering
CBA	Capabilities Based Acquisition
CBT	Computer Based Training
CDD	Capabilities Design Document
CFD	Control Flow Diagram
CG	Guided Missile Cruiser
CM	Configuration Management
CMS2	Compiler Monitor Systems - 2
CNO	Chief Of Naval Operations
ConOps	Concept of Operations
COTS	Commercial-Off-The-Shelf
CPA	Closest Point of Approach
CSPECS	Control Specifications
CVN	Carrier Vessel Nuclear; also nuclear powered aircraft carrier
DAU	Defense Acquisition University
DAWIA	Defense Acquisition Workforce Improvement Act
DCD	Data Context Diagram
DCE	Detect Control Engage
DDG	Destroyer, Guided Missile
DDG (1000)	Destroyer, Guided Missile, Next Generation
DDS	Digital Data Storage
DFD	Data Flow Diagram
DFD0	Data Flow Diagram Level 0
DFD2	Data Flow Diagram Level 2
DoD	Department of Defense
DoDAF	Department of Defense Architecture Framework
DODD	Department of Defense Directive
DODI	Department Of Defense Instruction
DoN	Department of Navy
DOORS	Dynamic Object Oriented Requirements System
DOT&E	Director, Operational Test and Evaluation
DSB	Defense Science board
DT	Developmental Test

Acronym	Definition
DTE	Detect to Engage
DT&E	Developmental Test and Evaluation
EDFD	Enhanced Data Flow Diagram
EDFD0	Enhanced Data Flow Diagram Level 0
EFFBD	Enhanced Functional Flow Block Diagram
EO	Electro Optic
ER	Element Relationship
ESSM	Evolved Sea Sparrow Missile
EXCOMM	External Communication
EW	Electronic Warfare
FAA	Functional Area Analysis
FEMA	Federal Emergency Management Agency
FFBD	Functional Flow Block Diagram
FMEA	Failure Modes and Effects Analysis
FMECA	Failure Mode Effects and Criticality Analysis
FNA	Functional Needs Analysis
FoS	Family of Systems
FSA	Functional Solutions Analysis
FSM	Finite State Machine
GAO	Government Accountability Office
GFE	Government Furnished Equipment
GFI	Government Furnished Information
GOTS	Government-Off-The-Shelf
GPR	Government Purpose Rights
GUI	Graphical User Interface
H-H-P	Hatley-Hruschka-Pirbhai
HK	Hard Kill
HP	Hatley-Pirbhai
Hr	Radar Height
HSI	Human Systems Integration
Ht	Target Maximum Height of target Height
HTML	HyperText Markup Language
HVU	High Value Unit

Acronym	Definition
HW	Hardware
HWIL	Hardware In the Loop
IBD	Internal Block Diagram
ID	Identification
ICD	Initial Capabilities Document
IDEF0	Integration Definition for Function Modeling
IETM	Interactive Electronic Technical Manuals
IFF	Identification Friend/Foe
ILS	Integrated Logistics Support
INCOSE	International Council on Systems Engineering
IOT&E	Initial Operational Test And Evaluation
IPPD	Integrated Product and Process Development
IPR	In Process Review
IPT	Integrated Product Team
IR	Infrared
ISEA	In Service Engineering Agent
ISO	International Organization for Standardization
IT	Information Technology
IWS	Integrated Warfare Systems
JCIDS	Joint Capabilities Integration Development System
KA	Kill Assessment
KPP	Key Performance Parameter
KSA	Key System Attribute
KSLOC	Source Lines of Code in Thousands
L	Launch Rate
LAD	Limited Area Defense
LAN	Local Area Network
LCC	Life Cycle Cost
LCCE	Life Cycle Cost Estimate
LCS	Littoral Combat Ship
LFT&E	Live Fire Test and Evaluation
LORA	Level Of Repair Analysis
LPD	Landing Platform Dock

Acronym	Definition
M&S	Modeling and Simulation
MathML	Mathematic Modeling Language
MBSE	Model-Based Systems Engineering
MIL-STD	Military Standard
MLDT	Mean Logistics Delay Time
MOE	Measure Of Effectiveness
MOP	Measure Of Performance
MOSA	Modular Open Systems Architecture
M_{RA}	Measure of Raid Annihilation
MSSE	Masters of Science in Systems Engineering
MSSEM	Masters of Science in Systems Engineering Management
MTBF	Mean Time Between Failure
MTTR	Mean Time To Repair
N2	N-squared
Na	Number of attack
NAV	Navigation
NAVSEA	Naval Sea Systems Command
Nc	Number of cells
NCA	Net-Centric Architecture
NETWAR	Navy Network Warfare Command
NMCI	Navy Marine Corps Intranet
nm or NM or nmi	Nautical Mile
NMSO	Navy Modeling and Simulation Office
NOA	Naval Open Architecture
NPS	Naval Postgraduate School
NSWC PHD	Naval Surface Warfare Center Port Hueneme Division
OA	Open Architecture
OACE	Open Architecture Computing Environment
OAET	Open Architecture Enterprise Team
OCL	Object Constraint Language
OIPT	Overarching Integrated Product Team
OMB	Office of Management and Budget
OMG SysML	Object Management Group Systems Modeling Language

Acronym	Definition
OO	Object Oriented
OP Area	Operational Area
OPNAV	Office of the Chief of Naval Operations
OSD	Office of the Secretary of Defense
OSJTF	Open Systems Joint Task Force
OT&E	Operational Test and Evaluation
OV	Operational View
PBL	Performance Based Logistics
P_d	Probability of detection
PEO	Program Executive Office; Program Executive Officer
PESHE	Programmatic Environmental, Safety, and Health Evaluation
PIA	Post Independent Analysis
P_K or $P_{(K)}$	Probability of Kill
PMP	Project Management Plan
PMS	Planned Maintenance System
POC	Point of Contact
P_{RA}	Probability of Raid Annihilation
PSPEC	Process Specifications
PW	People Ware
RAM	Reliability, Availability, and Maintainability / Random Access Memory
r_{CPA}	Range to Closest Point of Approach
RCS	Radar Cross Section
RDA	Research, Development & Acquisition
Rd(km)	Radar Horizon
RDT&E	Research, Development, Test and Evaluation
Ref	Reference Nodes
RF	Radio Frequency
RFP	Request for Proposal
r_{MAX} or Rmax	Maximum Range
Rmin	Minimum Range
Rout	Keep out Range
RTF	Rich Text Format
SAM	Surface To Air Missile

Acronym	Definition
SD	Self Defense
SE	Systems Engineering
SECDEF	Secretary Of Defense
SEI	Software Engineering Institute
SEP	Systems Engineering Plan
SK	Soft Kill
SM-2	Standard Missile – 2
SME	Subject Matter Expert
SOA	Service Oriented Architecture
SoS	System of Systems
SOW	Statement Of Work
SPAWAR	Space and Naval Warfare
SPL	Software Product Line
SSDS	Ship Self Defense System
STT	State Transition Table
SUW	Surface Warfare
SV	Systems View
SW	Software
SysML	Systems Modeling Language
T&E	Test and Evaluation
Tar	Target arrival rate
TCP/IP	Transmission Control Protocol/Internet Protocol
Te	Kill evaluation time or Threat Engagement
TEMP	Test and Evaluation Master Plan
TEP	Test and Evaluation Plan
TES	Test and Evaluation Strategy
TEWA	Threat Evaluation & Weapons Assignment
Tft	Firm track time
Tgt	Target
TL	Initial Launch
TOF	Time Of Flight
T/R	Transmit Receive
Tr	Reaction Time

Acronym	Definition
Ts	Rescan Time
t_t	Threat track
Tup	Tie Up Time
TV	Technical Standards View
UML	Unified Modeling Language
US	United States
USD	Under Secretary Of Defense
USN	United States Navy
V&V	Verification and Validation
VLS	Vertical Launching System
Vm	Velocity missile
VPN	Virtual Private Network
V target	Velocity of target
WBS	Work Breakdown Structure
WEM	Weapon Engagement Manager
WMF	Windows Metafile
XMI	Extended Memory Interconnect
XML	eXtensible Markup Language

REFERENCES

Berk, J.V., A.H. Muntz, and P.R. Stevens. 1989. Software Architecture Concepts for Avionics. Paper presented at the Aerospace and Electronics Conference, May 22, in Dayton, OH.

Blanchard, Benjamin S. and Wolter J. Fabrycky. 2006. *Systems Engineering and Analysis*. New Jersey: Pearson Prentice Hall.

Bosch, Jan. 2000. *Design and Use of Software Architectures: Adopting and evolving a product-line approach*. United Kingdom: ACM Press Books.

Buede, Dennis M. 2000. *The Engineering Design of Systems, Models and Methods*. New York: John Wiley & Sons, Inc.

Crisp, Harry E. 2006. Systems Engineering Vision 2020. Presentation to the Washington Metropolitan Area Chapter, May 9, in Washington, D.C.

Dam, Steven H. 2006. *DoD Architecture Framework: A Guide to Applying System Engineering to Develop Integrated, Executable Architectures*. Marshall, VA: System and Proposal Engineering Company (SPEC).

Defense Acquisition Guidebook (DAG). 2004. *AT&L Knowledge Sharing System*. Defense Acquisition University. https://akss.dau.mil/dag/ (accessed February 2009).

Defense Acquisition Workforce Improvement Act (DAWIA). 1990. Procurement Training Information: DAWIA Information and Certification Application. http://www.dscp.dla.mil/contract/dawia.htm (accessed February 2009).

Defense Science Board (DSB). 2008. Report of the Defense Science Board Task Force on Developmental Test & Evaluation. Report presented to the Office of the Undersecretary of Defense for Acquisition, Technology, and Logistics, May 27, in Washington D.C.

Department of Defense (DoD). 2003. Department of Defense Architecture Framework (DoDAF) v1.0, Vol II. http://www.eaframeworks.com/DoDAF/Volume_II.pdf. (accessed February 2009).

DoD. 2007. Department of Defense Architecture Framework (DoDAF) v 1.5. http://jitc.fhu.disa.mil/jitc_dri/pdfs/dodaf_v1v1.pdf. (accessed February 2009).

DoD. 2004. Department of Defense Directive (DoDD) 4630.5: Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS). http://jitc.fhu.disa.mil/jitc_dri/pdfs/dd46305p.pdf. (accessed February 2009).

DoD. 2008. Department of Defense Instruction (DoDI) 5000.02. AT&L Knowledge Sharing System. Defense Acquisition University. http://www.dtic.mil/whs/directives/corres/pdf/500002p.pdf (accessed February 2009).

Dorset House. 2008. PSARE, H/H/P, and Object-Orientation. Dorset House Publishing. http://www.dorsethouse.com/books/psare/archive/UML_index.html (accessed February 2009)

Eriksson, Magnus. 2003. An Introduction to Software Product Line Development. Proceedings of Ume's Seventh Student Conference in Computing Science, Örnsköldsvik, Sweden. http://www.cs.umu.se/~magnuse/papers/productLines2003.pdf (accessed October 2008).

Estefan, Jeff A. 2008. Survey of Model-Based Systems Engineering (MBSE) Methodologies. Jet Propulsion Laboratory, California Institute of Technology. http://www.omgsysml.org/MBSE_Methodology_Survey_RevB.pdf (accessed February 2009).

Fein, Geoff. 2008. Navy Developing Path Forward For Open Architecture Implementation. (Interview of RDML Terry Benedict). *Defense Daily*. (August 21) http://findarticles.com/p/articles/mi_6712/is_37_239/ai_n29462370 (accessed February 2009).

Fisher, Jerry. 1998. Model-Based Systems Engineering: A New Paradigm. *International Council On Systems Engineering (INCOSE) Insight,* Vol 1 Issue 3.

Friedenthal, Sanford, Alan Moore and Rick Steiner. 2008. *A Practical Guide to SysML: The Systems Modeling Language*. Burlington, MA: Morgan Kaufmann OMG Press.

Gomaa, H. and G.A. Farrukh. 1999. Methods and Tools for the Automated Configuration of Distributed Applications from Reusable Software Architectures and Components. (December) *IEE Proceedings-Software*, Vol.146, No. 6.

Government Accounting Office (GAO). 2007. Report Defense Acquisitions, Assessment of Selected Weapon Programs, March, in Washington D.C.

Green, John M. 2008a. Applying Open Architecture Concepts to Mission and Ship Systems. Paper presented at American Society of Naval Engineers (ASNE) Day, June 23-24, in Arlington, VA.

Green, John M. 2008b. SE3122/SE3123 Naval Weapon Systems Technology I/II Course Materials. Monterey: Naval Postgraduate School.

Guertin, Nick. 2008. Q&A with Mr. Nick Guertin, deputy director of Open Architecture, Navy program executive office for integrated warfare systems—PEO IWS 7B. BNET Business Network, Technology Industry. http://findarticles.com/p/articles/mi_m0OBA/is_4_25/ai_n21094788. (accessed February 2009).

Haggerty, Kip and Leslie. 2000. *Introduction to Structured Methods*. El Segundo, CA: H&A Systems Engineering.

Hardy, Dwayne. 2006. Model Based Systems Engineering and How it Aids DoD Acquisition & Systems Engineering. Paper presented at the 9th Annual Systems

Engineering Conference "Focusing on Improving Performance of Defense Systems Programs", October 23-26, in San Diego, CA.

Hatley, Derek, Peter Hruschka, and Imtiaz Pirbhai. 2000. *Process for System Architecture and Requirements Engineering*. New York: Dorset House Publishing.

Hause, Matthew. 2008. Editorial: An overview of UPDM – A Unified Profile for DoDAF/MODAF. *Military Embedded Systems*, (October). http://www.mil-embedded.com/articles/id/?3653 (accessed February 17, 2009).

Hoeper, Paul J, Honorable Army Acquisition Executive. 1999. Statement Before The Subcommittee On Readiness and Management Support, Committee On Armed Services, United States Senate, First Session, 106th Congress, On Acquisition Reform. (March 17) http://armed-services.senate.gov/statemnt/1999/990317ph.pdf (accessed February 2009).

Hoffmann, Hans-Peter. 2007. SysML-Based Systems Engineering Using a Model-Driven Development Approach. Paper presented at INCOSE-LA: SysML-Based SE Speaker Meeting, February 20, in El Segundo, CA.

Informatica Corporation. 2005. *Data Integration in a Service-Oriented Architecture: A Strategic Foundation for Maximizing the Value of Enterprise Data*. Redwood City, CA: Informatica Corporation.

Iyer, Raj G. 2007. A Program Managers Guide to Technical Data Rights. Report presented to the United States Army TACOM Lifecycle Management Command conference, April 16, in Warren, MI.

Johns Hopkins Applied Physics Laboratory (APL). 2001. Air Defense Part II – Combat Systems. *Johns Hopkins APL Technical Digest*. October-December, Volume 22.

Jones, James V. 2006. *Integrated Logistics Support Handbook*, 3rd Edition. New York: SOLE Logistics Press, McGraw-Hill.

Kang, Kyo C., Sholom G. Cohen, James A. Hess, William E. Novak, A. Spencer Peterson. 2007. *Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21*. Pittsburgh: Carnegie-Melon University, Software Engineering Institute.

King, Dr. Timothy. 2007. SIMBASE PD09 Executive Summary – Issue 1.0. Paper presented to the European Defense Agency, November 27, in Brussels.

Krueger, Charles W. 2008. Introduction to Software Product Lines. http://www.softwareproductlines.com (accessed February 2009).

Kunstar, Jan, Iveta Adamuscinova, and Zdenek Havlice. 2009. The Use of Development Models for Improvement of Software Maintenance. *Acta Univ. Sapientiae, Informatica*, 1, 1, 45-52. http://acta.sapientia.ro/acta-info/C1-1/info1-4.pdf (accessed February 2009)

Larish, Bryan. 2008. Interview by author: DoD Architecture Framework Lessons Learned. Space and Naval Warfare Systems Center (SPAWAR) San Diego, CA, November 13.

Levis, Alexander H. January 2008. Implementing SOA in DoDAF 1.5. Paper presented at the IBM 'DoD use of SOA' Workshop, January 9, in Fairfax, VA George Mason University.

Long, Jim. 2002. Relationships between Common Graphical Representations in Systems Engineering. *Proceedings for the 1995 INCOSE Symposium*. July. Vitech Corporation.

Maier, Mark W., and Eberhardt Rechtin. 2002. *The Art of Systems Architecting*. Boca Raton, FL: CRC Press.

Martin, James N. 1997. A Process for Requirements and Architecture Definition. Paper presented at INCOSE Symposium, April 2, in Los Angeles.

Melancon, Paul W. 2008. Categorization and Representation of Functional Decomposition by Experts. M.S. diss., Naval Postgraduate School.

Miles, Daniel W. 2008. Program Life Cycle Cost Driver Model (LCCDM). http://government.gpworldwide.com/common/pdf/govt/cdProgramLifeCycle.pdf (accessed Feb 2009)

Naval Surface Warfare Center Dahlgren Division (NSWCDD). 2004. Open Architecture Computing Environment (OACE) Design Guidance Version 1.0. Report presented to the Program Executive Office, Integrated Warfare Systems, August 23, in Washington D.C.

Nelson, Eric M. 2007. Open Architecture for the Enterprise, Part 1: Architectural Principles of Open Architecture. http://www.ibm.com/developerworks/library/ar-openarch1/index.html <x-excid://AC5E0000/pas:http://www.ibm.com/developerworks/library/ar-openarch1/index.html> (accessed March 2009)

Nilson, Roslyn, Paul Kogut, and George Jackelen. 1994. Component Provider's and Tool Developer's Handbook Central Archive for Reusable Defense Software (CARDS). *STARS Informal Technical Report STARS-VC-B017/001/00. Unisys Corporation*.

Nuffort, Matthew. 2001. Managing Subsystem Commonality. M.S. diss., Massachusetts Institute of Technology.

Oliver, David W. 1998. The Benefits of Model Based Engineering. *International Council On Systems Engineering (INCOSE) Insight, Vol 1 Issue 3.*

Oliver, David W., Timothy P. Kelliher, and James G. Keegan, Jr. 1997. *Engineering Complex Systems with Models and Objects*. New York: McGraw-Hill Companies.

Object Management Group (OMG). 2008. *The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification*. Needham, MA: Object Management Group.

Object Management Group (OMG). 2009. OMG Systems Modeling Language: The Official OMG SysML site. http://www.omgsysml.org/ (accessed March 2009).

Ortiz, Vincent M. 2006. LPD 17 PRA Testbed VV&A Database: A Disciplined Approach for VV&A. AVW Technologies. http://www.dtic.mil/ndia/2006test/ortiz3.pdf (accessed February 2009)

Posadas, Serg. 2007. OS3301 Simulation Modeling and Analysis Course Material (Lesson 1.0 Overview and 1.1 Math Models). Monterey: Naval Postgraduate School.

Rickman, Dale M. 2000. A Process for Combining Object Oriented and Structured Analysis and Design. Reston, VA: Raytheon Systems Company.

Roedler, Gary J. and Cheryl Jones. December 2005. Technical Measurement: A Collaborative Project of PSM, INCOSE, and Industry. *Practical Software and Systems Measurement (PSM) and International Council on Systems Engineering (INCOSE)*. Section 3.1.

Schmidt, Douglas C. 2006. Model Driven Engineering. *IEEE Computer Society*.

Shannon, James, Capt. USN. 2006. Naval Open Architecture – Overview on OA. Presented at the second annual Open Architecture Industry Day, February 24, in Washington D.C.

Sharmal, Vibhu Saujanya, Pankaj Jalote, and Kishor S. Trivedi. 2005. Evaluating Performance Attributes of Layered Software Architecture. *CBSE.*

Soares, M.S., and J.L.M Vrancken. 2007. Requirements Specification and Modeling through SysML. Presented at IEEE International Conference on Systems, Man, and Cybernetics - SMC 2007, 7-10 October, in Montreal, Quebec, Canada.

Stevens, Jim, Capt. USN. 2008. The How and Why of Open Architecture. *Undersea Warfare*.

Strei, Thomas J., Capt. USN. 2003. Open Architecture in Naval Combat System Computing of the 21st Century: Network-Centric Applications. Paper presented to the Navy Open Architecture Program Executive Office, Integrated Warfare Systems, April 1, in Washington D.C.

SysML Partners. 2008. SysML – Open Source Specification Project. http://www.sysml.org (accessed February 2009)

Van Bennekom, Dr. Frederick C. and Goffin, Dr. Keith. 2002. *Problem Prevention Through Design for Supportability: Gaining Competitive Advantage from Customer Support*. Bolton, MA: Customer Service Press.

Vitech Corporation. 2007a. *CORE® 5 Architecture Definition Guide (DoDAF v1.5).* Vienna, VA: Vitech Corporation.

Vitech Corporation. 2007b. *CORE® 5 Systems Definition Guide.* Vienna, VA: Vitech Corporation.

Vitech Corporation. 2007c. *CORE® 5 Systems Engineering Guided Tour*. Vienna, VA: Vitech Corporation.

Wagner, Daniel H., W. Charles Mylander, and Thomas J. Sanders. 1999. Naval Operations Analysis. Annapolis, MD: Naval Institute Press.

Wood, Brando C. 2001. An Analysis Method for Conceptual Design of Complexity and Autonomy in Complex Space System Architectures. M.S. diss., Massachusetts Institute of Technology.

Young, John. 2009. Memorandum to the U.S. Secretary of Defense (SECDEF) Robert Gates, January 26, in Washington D.C.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. John Mike Green
   Naval Postgraduate School
   Monterey, CA

4. Raymond Madachy, Ph.D.
   Naval Postgraduate School
   Monterey, CA

5. Kristin Baldwin
   Deputy Director, Strategic Initiatives
   ODUSD (A&T) SSE/SI
   3090 Defense Pentagon Room 3B648A
   Washington, DC 20301-3090

6. Brian Gannon, Capt USN (PEO IWS 7)
   1333 ISAAC HULL AVE SE, Washington Navy Yard
   Washington, DC 20376

7. Office of the Chief of Naval Operations
   ATTN: N76F
   2000 Navy Pentagon
   Washington, DC 20350-2000

8. Deborah Anderson (PMS 502L)
   1333 ISAAC HULL AVE SE, Washington Navy Yard
   Washington, DC 20376

9. Andrew Li (PEO IWS 1)
   2 ISAAC HULL AVE SE, Floor 2, Washington Navy Yard
   Washington, DC 20376

10. Timothy Morrow
    Software Engineering Institute
    Carnegie Mellon University
    Pittsburgh, PA 15213

11. Terence Sheehan
    19008 Way Side Drive, Suite 2073
    Dahlgren, VA 22448-5162

12. Julie Streets, Code 105
    NAVSURFWARCENDIV
    4363 Missile Way Bldg. 444
    Port Hueneme, CA 93043

THIS PAGE INTENTIONALLY LEFT BLANK