NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

TRANSLATION OF USER NEEDS TO SYSTEM REQUIREMENTS

by

Patrick R. Hoff

March 2009

Thesis Advisor: John M. Green
Second Reader: Paul V. Shebalin

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2009	Master's Thesis

4. TITLE AND SUBTITLE Translation of User Needs to System Requirements	**5. FUNDING NUMBERS**
6. AUTHOR(S) Hoff, Patrick R.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
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 thesis 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	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited	

13. ABSTRACT (maximum 200 words)

Department of the Navy system acquisition begins with a statement of user need. Delivery of required capability depends heavily on the effective translation of user need to system requirements. Failure typically results in program cost overruns, schedule slippage, and sometimes partial or complete failure to deliver needed capability.

Architectures as part of systems engineering were created to cope with the growing complexity of modern systems. The Navy develops and operates some of the most complex systems in the world. Yet, architecture development, while mandated, remains largely ancillary to the systems engineering process. As a result, much of the engineering advantage of architectures remains untapped.

This study examined U.S. Navy policy, process, and current engineering and architectures standards and identified recommendations to improve the process of translating user needs to system requirements while facilitating the use of architectures.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Architecture, architecture framework, DoDAF, JCIDS, requirements, systems engineering	143
	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

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

TRANSLATION OF USER NEEDS TO SYSTEM REQUIREMENTS

Patrick R. Hoff
Civilian, Naval Sea Systems Command
B.S., University of Houston, 1978
M.S., University of Maryland, 1992

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2009

Author: Patrick R. Hoff

Approved by: John M. Green
Thesis Advisor

Paul V. Shebalin
Second Reader

David Olwell
Chairman, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Department of the Navy system acquisition begins with a statement of user need. Delivery of required capability depends heavily on the effective translation of user need to system requirements. Failure typically results in program cost overruns, schedule slippage, and sometimes partial or complete failure to deliver needed capability.

Architectures as part of systems engineering were created to cope with the growing complexity of modern systems. The Navy develops and operates some of the most complex systems in the world. Yet, architecture development, while mandated, remains largely ancillary to the systems engineering process. As a result, much of the engineering advantage of architectures remains untapped.

This study examined U.S. Navy policy, process, and current engineering and architectures standards and identified recommendations to improve the process of translating user needs to system requirements while facilitating the use of architectures.

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

Successful translation of user needs to system requirements is foundational to successful product development. This translation occurs during the concept phase of the systems engineering process, “pre-Milestone B” per (DoD Instruction 5000.2, 2008).

Requirements definition for DoN warfighting systems predominantly involves upgrades or modifications of existing systems. Even unprecedented systems have to integrate and interoperate with existing or “legacy” systems. Legacy requirements may be incomplete, ambiguous, out-of-date, or in conflict with other requirements. Similarly, legacy architectures may not be sufficiently developed to support requirements definition.

Systems engineering principles and processes and system architecting methods were established in large measure to facilitate successful development of complex systems (INCOSE Systems Engineering Handbook v. 3.1, 2007), (Rhodes, 2007). DoN requirements for planning and implementation of systems engineering are based on Government and industry standards and best practices. DoN also requires for major systems acquisitions, development of architecture products in accordance with the DoD Architecture Framework (DoD Architecture Framework, version 1.5, 2007). However, DoN acquisition policy (SECNAVINST 5000.2D, 2008) does not describe a relationship or dependency between architecture requirements and systems engineering requirements. Programs frequently develop architecture products through a process that is essentially

separate from the systems engineering process. This practice marginalizes the utility of architectures (Osvalds, 2006).

The purpose of this research is to review and correlate systems engineering standards, the Joint Capabilities Integration and Development System (CJCSI 3170.01F, 2007), (CJCSM 3170.01C, 207), and system architecting policy and standards, then, to formulate and present recommendations for improving the integration of systems engineering with system architecting, and the contribution of architectures to requirements definition.

At the time of this writing, the JCIDS process, as well as DoN acquisition policy, require programs to develop DoDAF-compliant architecture products. For a typical, complex warfare system, substantial time and resources are required to develop those products. In spite of requirements to develop architecture products, there is substantial evidence that the architecture products are poorly executed. Architecture products form part of the foundation for requirements definition. Poor execution of system architecting and requirements definition leads to program delays, cost overruns, and products whose performance does not provide required capability. Therefore, better alignment and implementation of architecture development and systems engineering processes holds the potential to deliver greater value from architecture products and for those products to improve the output of the requirements definition process. Achievement of these objectives will facilitate more effective program management and will contribute to improved system

performance, system supportability, system interoperability, and system-of-systems integration.

The systems engineering process spans the entirety of a product's life cycle, from identification of a user need to system retirement and disposal. Descriptions of the systems engineering process life cycle phases in different standards are generally well correlated at a high level, with some semantic differences. The standards provide a framework of tasks, but do not provide the process detail, examples, or accommodation of differences among product types necessary to be sufficient as stand-alone source references from which a detailed, systems engineering plan (SEP) can be developed and implemented. System engineering standards neither preclude nor prescribe the use of architectures to support requirements analysis. System engineering guides and handbooks produced by the Services and by industry complement and augment the standards. Among both standards and guides, terminology is often not well-defined and varies among standards and guides. For example, a "performance specification" in one guide may have a subtly different meaning in another guide. Variations and lack of implementation specificity among systems engineering process standards and guides can make it difficult to implement a uniform, auditable systems engineering process within an organization.

To begin the systems engineering process, a customer presents a statement of need to a system developer. Through an iterative process between user and developer, a comprehensive set of system requirements is formed,

establishing the basis for design. The DoD has its own system for requirements generation.

In 2003, the DoD implemented the Joint Capabilities Integration and Development System (JCIDS), a top-down, joint capabilities (i.e., requirements) generation, validation, and prioritization system intended to reduce functionally overlapping, Service-specific systems as well as inadequate intra-Service and inter-Service interoperability among systems. The JCIDS process interjects additional steps between the statement of user need and the development of a system specification in comparison with a typical commercial systems engineering process as described in Fabrycky (2006).

The JCIDS process, as incorporated into systems engineering processes described in the Naval Systems Engineering Guide (2004) offers the potential for a cross-service requirements analysis and prioritization capability necessary to achieve the DoD's mandate for capability-based acquisition. It could also use architectures as an analytical and management framework. However, research for this study indicates the use of architectures in conjunction with JCIDS is neither explicitly described nor explained in policy, and to date, the Services' approaches to warfighting requirements continue to align to Service perspective rather than a Joint perspective.

Understanding the role of system architecting as part of the system engineering process first requires definition of terms. To define "architecture framework" the terms "architecture" and "framework" must each be defined. The definition of architecture cited in the DoD Architecture

Framework Version 1.5 (2007) is: “the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time.” Every existing system embodies an architecture. Webster’s II Dictionary (1984) defines framework as: “a basic arrangement, form, or system.” Rolf Siegers (2005) defined architecture framework simply as “a resource that aids in the development or description of an architecture.” Maier (2004) asserts that an architecture framework exhibits one or more of the following five characteristics:

1. A definition of the word “architecture”
2. A conceptual framework explaining key concepts or terms
3. An approach to describing architectures
4. Architectural methods (e.g., creating, analyzing, interpreting, realizing)
5. A theory of how architectures and architectural descriptions fit into a larger context (e.g., systems engineering, design, etc.)

Motivation to establish and utilize architecture frameworks can derive from business considerations, technical considerations, or both. An architecture framework used for enterprise product development can for example facilitate shorter product development cycles, lower manufacturing, support and upgrade costs, and a reduced number of functionally duplicative products. This accrual of benefits can be explained by a framework’s ability to make individual product architectures comparable and integratable, facilitating cross-product insights that would be otherwise difficult. From a technical perspective, and for the same reasons cited for the business

perspective, products developed in conformance with an architecture that is in turn developed in conformance with an architecture framework potentially exhibit a greater degree of interoperability, a simpler logistics support infrastructure, and in systems of systems, a greater ability to be reconfigured to adapt to changes in mission needs. The DoD architecture framework (2007) is cited in DoD requirements, acquisition, and systems engineering policy and guidance for both business and technical reasons.

DoDAF use, specified in DoN policy and guidance, is the framework standard for architectures in DoN. However, considerable variation in DoDAF architecture implementation is evident within the boundaries of DoDAF compliance, such that architectures for different but related systems may not be effectively comparable or integratable. The current framework is necessary but not sufficient to ensure the effective use of architectures in acquisition and systems engineering. The DoDAF Version 2.0, currently in development, is intended to address some long-recognized limitations of the DoDAF, but the framework is only one part of what should be a multi-faceted approach to realizing the potential of architectures in the systems engineering process.

The systems engineering process, the DoD requirements process, i.e., JCIDS, the DoD architecture framework, and to a limited extent, the DoD acquisition process constitute a complex system-of-systems in and of themselves. Process components critical to successful requirements translation in a systems engineering context and the necessary elements

of a good system specification can be identified in this system-of-systems. In turn, an approach for how best to integrate necessary process components can be developed.

DoD and DoN requirements, acquisition, and systems engineering processes and associated guidance and standards comprise a substantial amount of interrelated data, even when limited to discussion of requirements development as in this study. Research performed for this study has revealed semantic inconsistencies and ambiguities as well as a multiplicity of system engineering processes, guidance and standards. This makes it difficult to trace data and process relationships in a systems engineering context. As a consequence it is difficult to rigorously and unambiguously trace a top-level operational requirement to a comprehensive and complete set of system requirements.

This study asserts that for Naval warfare systems, the process of translation of user needs to system requirements is not sufficiently rigorous and repeatable to ensure consistently complete and valid system requirements. The result is significant increases to program cost, schedule and risk when requirements issues are resolved later in development. Government Accounting Office Report (GAO-08-782T, 2008) states:

> At the strategic level, DOD does not prioritize weapon system investments and the department's processes for matching warfighter needs with resources are fragmented and broken. Furthermore, the requirements and acquisition processes are not agile enough to support programs that can meet current operational requirements. At the program level, programs are started without knowing what resources will truly be needed and are managed with lower levels of product

> knowledge at critical junctures than expected under best practices standards. In the absence of such knowledge, managers rely heavily on assumptions about system requirements, technology, and design maturity, which are consistently too optimistic. This exposes programs to significant and unnecessary technology, design, and production risks, and ultimately damaging cost growth and schedule delays.

Typical U.S. Navy surface combatants systems involve a number of constraints and characteristics that lead to significant complexity and difficulty in the requirements analysis process. However, this study illustrates a “blended” use case and Process for System Architecture and Requirements Engineering (PSARE) process, from which requirements and architectures models can be constructed in an integrated, structured, repeatable manner that when complete, provide the basis for a complete set of system requirements for complex systems. Furthermore, the requirements model is in fact an operational view of the system’s architecture and the architecture model represents the system view of the architecture, enabling generation of required DoDAF products. Importantly, the products are created by and for the systems engineering process. It is done in a way that provides a full accounting of requirements in a design. This type of process is not provided as part of the DoD Architecture Framework (DoDAF, 2007) or the Naval Systems Engineering Guide (2004).

In summary, a substantial body of work among Government and industry exists regarding system engineering standards and processes. Standards and practices have been evolved, refined, interpreted, and exercised over a period

of approximately 50 years. The value of systems engineering has been shown to lie in its ability to manage complexity in system development such that systems produced will consistently satisfy user needs. Yet, numerous examples exist of warfare systems that exceed schedule and cost requirements and do not meet operational requirements. Shortcomings appear to exist in the application and management of systems engineering principles.

The JCIDS process, ostensibly dependent on architectures as an analytical basis, has not resulted in desired levels of improvement in terms of ensuring Joint solutions, supporting capability acquisition, and reducing redundant or excess capability, while reliability identifying capability gaps. This may be partly a result of insufficiently developed architectural bases for the necessary analysis.

In spite of mandated systems engineering and architecture standards, the relationship between system architecting and system engineering is poorly identified in DoD policy and instructions, and processes to develop architecture models as part of a systems engineering process are not prescribed. Until a better connection between system architecting and systems engineering is made, architecture development efforts will be significantly challenged to demonstrate a clear return on investment.

Translation of operational requirements to a set of system requirements is a vitally important part of the systems engineering process. However, ambiguity exists in definition of engineering and analytical roles, in terms

and definitions, and in the dependencies of timing of events in this process. As a result, demonstration of system requirements completeness and linkage of system requirements to operational requirements are not consistently established.

Rigorous methods exist to model system requirements and architectures in a manner that supports mandated DoD Architecture Framework products while maintaining close-coupling with the systems engineering process. These methods can capture the behavior of complex, Naval warfare systems, maintain traceability to higher level requirements, and incorporate plain language views of requirements. However, the community of skilled practitioners may not be large enough to increase the use of these methods to the point of becoming standard practices.

LIST OF ACRONYMS AND ABBREVIATIONS

AITS	Adopted Information Technology Standards
ASD(C3I)	Assistant Secretary of Defense, Command, Control, Communication, and Intelligence
ASW	Anti-submarine Warfare
C&D	Command and Decision
C4I	Command, Control, Communications, Computers, Intelligence
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
CADM	Core Architecture Data Model
CBA	Capability-Based Assessment
CDD	Capabilities Definition Document
CDD	Capability Development Document
CFD	Control Flow Diagram
CIO	Chief Information Officer
CJCSI	Chairman of the Joint Chiefs of Staff Instruction
CJCSM	Chairman of the Joint Chiefs of Staff Manual
CNO	Chief of Naval Operations
CONOP	Concept of Operation
CPD	Capability Production Document
CSPEC	Control Specification
DAR	Defense Architecture Repository
DAU	Defense Acquisition University

DCR	DOTMLPF Change Recommendation
DFD	Data Flow Diagram
DISA	Defense Information Systems Agency
DISR	DoD IT Standards Registry
DoD	Department of Defense
DoDAF	Department of Defense Architecture Framework
DoN	Department of Navy
DOTMLPF	Doctrine, Organization, Training, Material, Leadership and education, Personnel, and Facilities
EIA	Electronic Industries Alliance
FAA	Functional Area Analysis
FBI	Federal Bureau of Investigation
FCCC	FORCEnet Consolidated Compliance Checklist
FNA	Functional Needs Analysis
FSA	Functional Solutions Analysis
GAO	Government Accounting Office
IAMD	Integrated Air and Missile Defense
ICD	Initial Capabilities Document
IDA	Institute for Defense Analysis
IEEE	Institute of Electrical and Electronics Engineers
IFF	Identify, Friend or Foe
INCOSE	International Council on System Engineering
IR	Infrared

ISO/IEC	International Organization for Standardization/International Electrotechnical Commission
IT	Information Technology
JCIDS	Joint Capabilities Integration and Development System
JIC	Joint Integrating Concept
JOC	Joint Operational Concept
JROC	Joint Requirements Oversight Council
JTA	Joint Technical Architecture
KPP	Key Performance Parameter
MARCORSYSCOM	Marine Corps Systems Command
MOA	Memorandum Of Agreement
NASA	National Aeronautics and Space Administration
NAVAIR	Naval Air Systems Command
NAVSEA	Naval Sea Systems Command
NAVSUP	Naval Supply Systems Command
NDIA	National Defense Industries Association
NIST	National Institute for Standards and Technology
NSS	National Security System
OODA	Observe, Orient, Decide, Act
OSD	Office of the Secretary of Defense
OV	Operational View
PSARE	Process for System Architecture and Requirements Engineering
PSPEC	Performance Specification

RDAV	Requirements Development and Validation
SDD	System Development and Demonstration
SEP	Systems Engineering Plan
SMC	Space and Missile Systems Center
SoS	System of Systems
SPAWAR	Space Systems Command
SRD	System Requirements Document
SRS	System Requirements Specification
SV	Systems View
TAFIM	Technical Architecture Framework for Information Management
TV	Technical View

ACKNOWLEDGMENTS

To Dr. James Meng and Mr. Carl Siel, for your endorsement, which provided me this learning opportunity, and for your extraordinary patience and support while I completed this “last piece of the puzzle.”

To Professors Mike Green and Paul Shebalin, for your encouragement and guidance in performing the research, writing about it, and maintaining the proper perspective.

To Mr. Michael Collins, for sharing your ideas on architecting and systems engineering which epitomize Albert Einstein’s observation: “Everything should be made as simple as possible, but not simpler.”

Finally, to my family, for not questioning this quest, and for helping me remember what is most important.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND:

Department of Navy (DoN) warfare systems are becoming more complex. At the same time, the DoN is also moving toward implementation of network-centric warfare and increasingly complex systems-of-systems. One effect of these trends on systems acquisition is significantly greater requirements complexity. Proper identification of requirements is a tenet of systems engineering. Based on statistical analysis of Department of Defense (DoD) programs performed by the Defense Acquisition University (DAU), 80% of a system's life cycle cost is determined by the time 20% of a system's actual cost is accrued. Within that first 20% of cost lies the concept phase of a program, accounting for, on average, 8% of the cost. It is during that concept phase, "pre-Milestone B" per (DoD Instruction 5000.2, 2008) that user requirements are translated to system requirements (INCOSE Systems Engineering Handbook v. 3.1, 2007). Some well-known examples of failures resulting from poorly defined, verified, or validated requirements sets are discussed by Bahill (2004).

Specifically, the complexity of requirements is dependent upon required functionality, the number and nature of nodes with which the system will interoperate, the extent to which the system, or system-of-systems can be reconfigured, and the range of environments and conditions in which the system is expected to operate. Note that operational flexibility, high interconnectivity and high

reconfigurability are all considered valuable attributes by the DoN and are typically required design attributes of new systems or system upgrades.

For DoN warfighting systems the requirements definition challenge is compounded by the fact that development programs predominantly involve upgrades of existing systems. Even truly new systems have to interoperate with existing or “legacy” systems. Legacy requirements may be incomplete, ambiguous, out-of-date, in conflict with other requirements, or un-testable. Similarly, legacy architectures may not be sufficiently developed to support requirements or interface analysis.

Systems engineering principles and processes and architecture frameworks were established in large measure to facilitate successful development of complex systems (INCOSE Systems Engineering Handbook v. 3.1, 2007), (Rhodes, 2007). DoN requires the planning for and use of systems engineering based on Government and industry standards and best practices. For major acquisition programs, DoN also requires the development of architecture products in accordance with DoD Architecture Framework (DoD Architecture Framework, version 1.5, 2007). DoN acquisition policy (SECNAVINST 5000.2D, 2008) does not however recognize a relationship or dependency between required architectures and systems engineering requirements. DoN guidance (Naval systems engineering guide, 2004), (Naval “systems of systems” systems engineering guidebook, 2006) *does* recognize and describe the contribution of architectures in the requirements definition process. However guidance, by definition, cannot mandate use of

architectures for the requirements definition part of the system engineering process. Programs frequently develop architectures through a process that is essentially separate from the systems engineering process. This practice marginalizes the utility of architectures (Osvalds, 2006).

B. PURPOSE:

The purpose of this research is to:

- Correlate systems engineering industry standards, Department of the Navy standards, the Joint Capabilities Integration and Development System (CJCSI 3170.01F, 2007), (CJCSM 3170.01C, 207), and recent systems engineering research as evidenced in published technical papers; and similarly, architecture and architecture framework industry and defense policy, standards, guidance, and technical papers. The research focuses on the portion of the systems engineering process leading up to and including system requirements definition. This correlation should reveal the extent to which Navy system engineering and architecture policy and guidance reflects best industry standards and practices and current research efforts. It should also provide insight on the potential versus realized utility of architectures in the requirements definition process, and the extent to which DoN policy and guidance support the use of architectures for requirements definition.
- Present recommendations for improving the contribution of architectures to requirements definition.

C. RESEARCH QUESTIONS:

Question: Within DoN, how can system architecting, e.g., architectures described in compliance with DoD Architecture Framework version 1.5 (2007), be better integrated with the systems engineering process to improve

requirements analysis and system requirements generation. In order to answer that question, the following questions and others concerning DoD policy, industry standards, system engineering and architecture semantics, and the engineering experience of others are also considered:

- What is an architecture framework and what is the intended purpose(s)?
- What are the advantages and objectives of system architecting for DoN systems during and as part of the requirements definition process?
- How do system architecting processes relate to the systems engineering process?
- What strategies, methodologies, or tools exist for integration of architecture development and requirements generation processes?

D. BENEFITS OF STUDY

At the time of this writing, the JCIDS process, as well as DoN acquisition policy, require programs to develop DoDAF-compliant architecture products. For a typical, complex warfare system, substantial time and resources are required to develop those products. In spite of requirements to develop architecture products there is substantial evidence that the architecture products are poorly executed. The architecture products should form the analytical framework for the requirements definition process. Poor execution of system architecting and requirements definition leads to program delays, cost overruns, and products whose performance does not provide the required capability. Therefore, better alignment and implementation of architecture development and systems engineering processes holds the potential for the DoN to derive greater value from architecture products and for

those products to improve the output of the requirements definition process. Achievement of these objectives will facilitate more effective program management and will contribute to improved system performance, system supportability, system interoperability, and system-of-systems integration.

E. SCOPE AND METHODOLOGY

1. Scope

This study focuses on Department of Navy warfare system requirements development. As such, it emphasizes software-intensive and complex systems, systems-of-systems, and integration and interoperability of new systems and newly modified legacy systems with legacy systems. Incorporation of Joint and coalition architectures and requirements is included to the extent required by discussion and analysis of the JCIDS process and DoN policy.

Although the systems engineering process and architecture framework requirements derived from DoD requirements and imposed by DoN are a primary topic, other established, in-development, and conceptualized processes and frameworks are included to better understand and illustrate where improvements or changes to DoN standards may merit consideration. As one example, the DoD Architecture Framework (DoDAF) version 2.0 is under development and its goals and objectives are considered.

2. Methodology

This study is primarily composed of the results of researching and analyzing documents from DoD, DoN, industry, and academia. Process steps include:

1. Conduct literature review and analysis of architecture frameworks, architecture development and use, and the requirements definition portion of systems engineering processes.
2. Correlate established systems engineering and architecture standards and processes with DoN requirements and guidance.
3. Research and discuss the state of practice of the requirements definition portion of the systems engineering process and the architecture development process in DoN or DoD.
4. Develop recommendations to improve integration of architecture development with requirements development.
5. Demonstrate by example, better integration of architectures with system engineering processes.

F. CHAPTER SUMMARY

The fundamental premise of this paper is that understanding a problem is the most important step toward a solution. The structure of this paper overall, is to examine systems engineering, requirements-setting, and architecture standards, guidance and processes, observe their strengths and weaknesses, consider where process gaps or lack of process synchronization may exist and recommend improvements.

Chapter II focuses on the systems engineering process, both in the DoD, industry, and academia. Chapter III reviews the initiation of user requirements, particularly from a DoD perspective. Chapter IV provides an overview of the evolution of the DoD Architecture Framework (2007).

Chapter V considers the integration of component processes, and Chapter VI presents development of system requirements and architectures in a DoN warfare system context. The last chapter, Chapter VII, presents conclusions and recommendations for further study.

THIS PAGE INTENTIONALLY LEFT BLANK

II. SYSTEMS ENGINEERING PROCESS

A. INTRODUCTION

This section begins with an overview of the systems engineering life cycle. Subsequent sections present a literature review of industry standards and DoD and DoN policy and standards. Emphasis is placed on the beginning of the systems engineering process; the requirements definition phase.

B. SYSTEMS ENGINEERING PROCESS LIFE CYCLE

The systems engineering process spans the entirety of a product's life cycle, from identification of a user need to system retirement and disposal. In texts commonly used for university-level system engineering education, descriptions of the systems engineering process life cycle phases are generally well correlated at a high level, albeit with some semantic differences. In Blanchard and Fabrycky's *Systems Engineering and Analysis* (2006), the phases, in sequence, are identified as: *Conceptual Design*, *Preliminary Design*, *Detail Design and Development*, *Production/Construction*, and *Operational Use and System Support*. These phases are structured to follow a system's life cycle. The sequence is shown graphically in Figure 1 (Blanchard, 2006:31). By comparison, Figure 2 (INCOSE Systems Engineering Handbook, 2007) shows alternative definitions of life cycle phases.

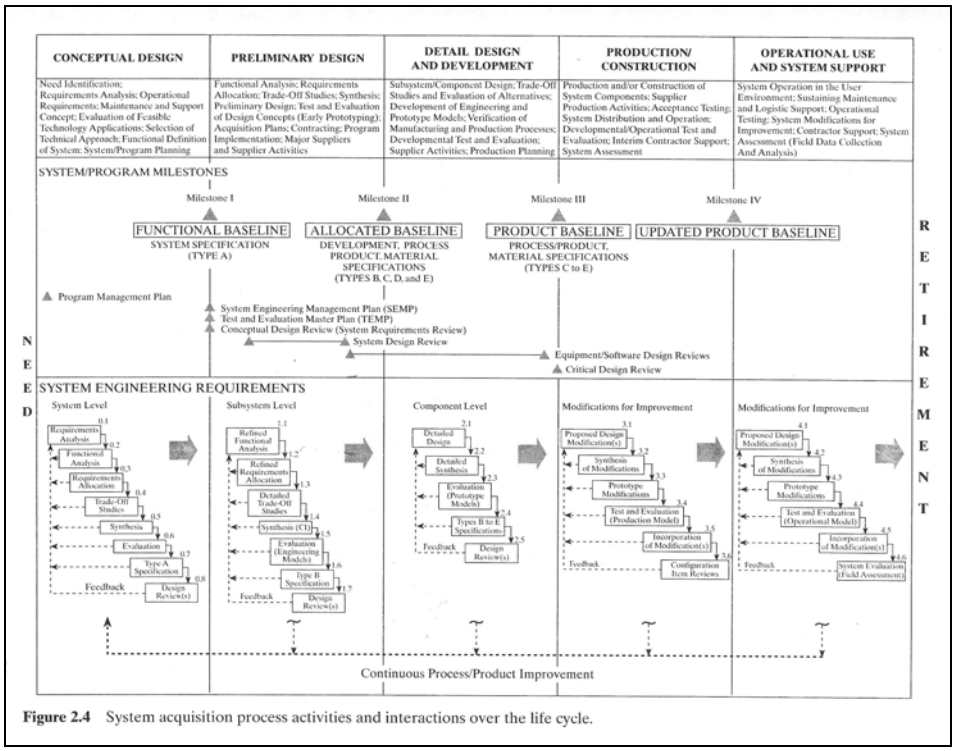

Figure 2.4 System acquisition process activities and interactions over the life cycle.

Figure 1 Systems Engineering Process, From Blanchard, 2006.

This shows systems engineering activities, milestones, and products in a product, life cycle framework.

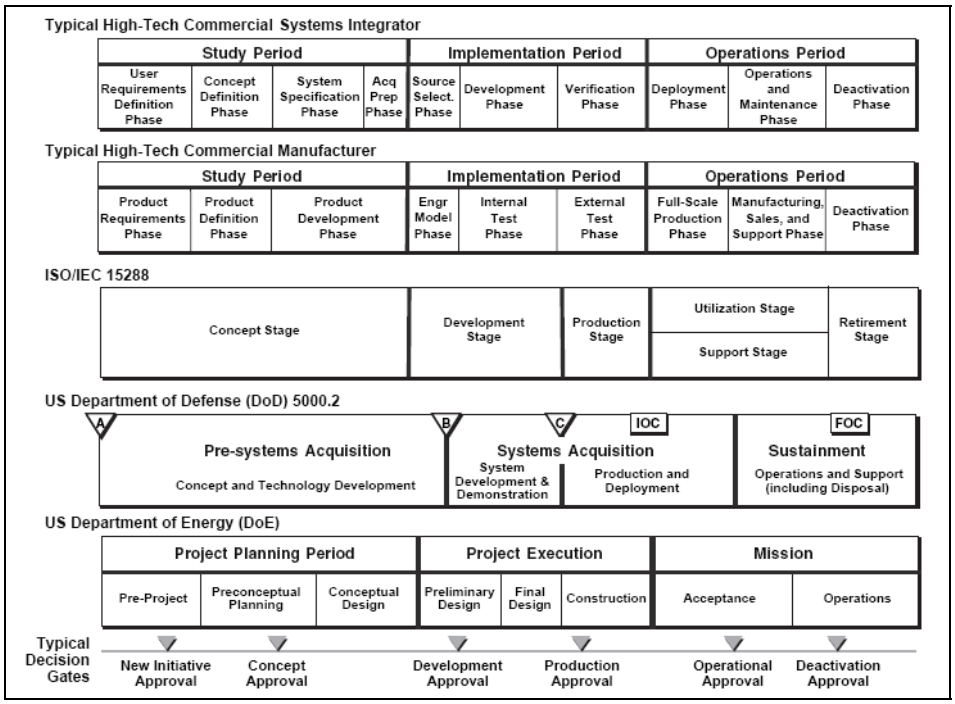

Figure 2 System Life Cycles, From INCOSE SE Handbook v3.1, 2007.

High-level similarities of various standards, and differences at a more detailed level are illustrated.

Systems engineering process models, for example the "Vee" model, the spiral model, and the waterfall model (Blanchard, 2006), provide systems engineering activity timing, logic, and sequence structure for the systems engineering process. They are applied to each of the life cycle phases to establish an agreed-to structure for product development and to facilitate understanding of development progress, status, and risk.

The objective of this section is to identify the life cycle phases, typical associated activities and engineering products, and to highlight the requirements analysis portion of the life cycle, a focus of this study. To manage the scope and complexity of this study, the systems

engineering process is predominantly described linearly and progress is discussed in terms of requirements development maturity and detail. Consequently, differences in system engineering process models and bases for selection are beyond the scope of this study.

This linear simplification makes comparison of the systems engineering process with other related processes easier, i.e., the DoN acquisition process (SECNAVINST 5000.2D), the DoN Requirements/Acquisition Two Pass, Six Gate Review Process (SECNAVINST 5000.2D, 2008), and the JCIDS process (CJCSM 3170.01C, 2007).

For this study, discussion of the systems engineering process is primarily concerned with what Blanchard (2006) describes as the *Conceptual Design* phase. As seen in Figure 1, an output of this phase is an "A spec" or system specification, or functional baseline, all taken to be equivalent in systems engineering terms. The system specification is defined as "…the top 'technical-requirements' document that provides overall guidance for system design from the beginning" (Blanchard, 2006: 86).

Placement of system specification development in the timeline of the DoN acquisition process is after the Capabilities Definition Document (CDD). The CDD, defined in CJCSM 3170.01C (2007) is an entrance criterion for Milestone B which leads into System Development & Demonstration (DoDI 5000.02, 2008) or "Preliminary Design" per Figure 1. It follows that the system specification represents the top of the specification hierarchy or "specification tree" for product development.

C. REQUIREMENTS ANALYSIS CHARACTERIZATION OF ESTABLISHED SYSTEMS ENGINEERING STANDARDS

Several key systems engineering standards exist. Some have been developed from or in coordination with others. Problems arising from multiple frameworks and standards are addressed in Sheard (1997). That paper discusses *framework trends* that include: *evolution, proliferation, integration and coordination,* and *consolidation*. Although the value of evolution, i.e., improvement, is recognized, continually changing and overlapping frameworks and standards create a burden on companies required to comply with different frameworks for different programs. Also, from the standpoints of both the invoking organization and the complying organization, a succession of changing frameworks and standards whose page counts typically number in the hundreds makes it more difficult to establish and maintain individual and organizational expertise and experience with requirements, compliant processes, and products.

In the 11 years since Sheard (1997), the cited trends have for the most part continued. More recently, there has been some increased focus on consolidation of hardware and software frameworks and standards. As is pointed out by Sheard, "…frameworks define characteristics of good processes but do not prescribe how they should be enacted." Most Services have attempted to address this "gap" by publishing more prescriptive systems engineering guides based on a small number of established frameworks and standards.

The primary standards cited by the Defense Acquisition Guidebook, Chapter 4, (DAG, 2004) are:

- ISO/IEC 15288, Systems Engineering – System Life Cycle Processes (2008)
- EIA 632, Processes for Engineering a System (1999)
- IEEE 1220, Application Management of the Systems Engineering Process (2007).

The systems engineering framework standard for the Naval Systems Engineering Guide is EIA 632.

1. ISO/IEC 15288 (2008)

This standard is written with a broad scope, identifying the following four system life cycle process groups: Agreement processes, Organizational Project-Enabling Processes, Project Processes, and Technical Processes. Within the Technical Processes life cycle process group lie the “Stakeholder Requirements Definition Process” (ISO/IEC 15288, 2008:36) and “Requirements Analysis Process” (ISO/IEC 15288, 2008:39) which are central to this study. Each process is described in terms of its purpose, outcomes, and activities and tasks. The descriptions are written at a high level of abstraction, establishing essentially a framework to which detailed process information can be appended. IEEE Std 1220 (2005) is specifically cited as a standard to be used with ISO/IEC 15288 and updates of these two documents are synchronized. There is no mention of the use of architectures in the accomplishment of stakeholder requirements definition or requirements analysis. The section that follows “Requirements Analysis Process” is “Architecture Design Process,” (ISO/IEC 15288:40) but its stated purpose is “to

synthesize a solution that satisfies system requirements," which is beyond the requirements definition phase and outside the scope of this study.

2. IEEE Std 1220 (2005), IEEE Standard for Application and Management of the Systems Engineering Process

This Standard provides more detailed process requirements than ISO/IEC 15288 and can be used in conjunction with that standard. The "System Definition Stage" defined in IEEE Std 1220 (2005:21) aligns with the aforementioned "Stakeholder Requirements Definition Process" and "Requirements Analysis Process" from ISO/IEC 15288 (2008), however that stage goes beyond development of a system requirements document and includes the product specification (i.e. allocated baseline) and preliminary subsystem specifications. The IEEE Std 1220 addresses the process of system definition and associated specifications, configuration baselines, and technical reviews. *Verified functional and design architectures* (IEEE Std 1220, 2005:21) are identified as products of this process, but the latter pertains to a product specification and the former is not presented as a means of facilitating requirements analysis.

3. EIA 632 (1999), Processes for Engineering a System

Among the five processes identified by this standard as comprising the engineering process are thirteen sub-processes. A total of 33 requirements address those sub-processes. The standard is organized around those 33 requirements. Under the sub-process *Requirements Definition*

Process Requirements are three requirements: *Acquirer Requirements*, *Other Stakeholder Requirements*, and *System Technical Requirements*. The descriptions of and outputs for those three requirements align with the subject of this study. In particular, the output of *System Technical Requirements,* the *System Requirements Document*, aligns with this study. Each requirement section includes tasks to consider, and related outcomes are defined in Annex C. Annex G defines requirements relationships, e.g., hierarchy, dependency, etc. The "how-to" of tasks is noted as being beyond the scope of the standard. Although the data, analysis, configuration management, and product elements involved in completion of the tasks supports system architecture development, architectures as part of the *Requirements Definition Process Requirements* are only mentioned in the context of "open systems architecture."

D. INDUSTRY, DOD, AND NASA SYSTEMS ENGINEERING GUIDEBOOKS

In addition to systems engineering standards, the Defense Acquisition Guidebook (2004) cites systems engineering handbooks and guides including: Naval Systems Engineering Guide (2004), INCOSE S.E. Handbook (2007), NASA S.E. Handbook (2007), DAU Systems Engineering Fundamentals (2001), ISO/IEC TR 19760, Systems Engineering – A Guide for the Application of ISO/IEC 15288 (System Life Cycle Processes), First Edition, 2003-11-15 (2008), and SMC Systems Engineering Primer and Handbook (2005). All of these guides and handbooks are specifically or generally based on the previously-mentioned ISO/IEC-IEEE and EIA standards, but they add Service or Agency specific process

information and in some cases provide implementation information that is beyond the scope of the ISO/IEC-IEEE and EIA standards.

The Naval Systems Engineering Guide specifically cites EIA-632 as its standards basis and presents the processes from that standard, adapted for DoN use. The intent of its creation and signature by all major Navy acquisition commands (MARCORSYSCOM, NAVAIR, NAVSEA, NAVSUP, and SPAWAR,) was to establish a single, documented, systems engineering process for DoN. Starting with EIA-632, the systems commands added command-specific content. The result was addition of consideration for DoN policies and procedures, and specific implementation guidance for EIA-632 processes. Descriptions of engineering artifacts provided apply to NAVAIR systems command but not other systems commands. Notably, the guide does not try to index the systems engineering process to an acquisition process timeline context. Also, references are made to the use of architectures during the requirements definition phase.

The INCOSE Systems Engineering Handbook (2007) is intended to provide descriptions of key systems engineering process activities. It is written to be consistent with ISO/IEC 15288-2002 (Note: This standard was superseded by ISO/IEC 15288-2008). As such it is not tailored to the engineering of DoD or DoN systems. The handbook is structured around context diagrams to augment ISO/IEC 15288, showing inputs, outputs, controls and enablers for each ISO/IEC 15288 process. Portions of the standard pertinent to this study are: Section 4.2, *Stakeholder Requirements Definition Process*, Section 4.3, *Requirements*

Analysis Process, Section 4.4, *Architectural Design Process*, and Appendices I and K, “Requirements Definition Process” and “System Architecture Synthesis” respectively. These appendices add considerably more detail regarding how to perform these processes. The handbook asserts that “Architectural design begins from the baseline functional and performance requirements, architectural constraints, and traceability matrix.” (INCOSE Handbook, 2007), i.e., well into the requirements development process. However, Appendix I, *Requirements Definition Process*, discusses description of system behavior, system interfaces, flow-down of requirements and, creating models, i.e., activities associated with architecture development. Therefore, while development of architectures to support requirements definition and analysis is not specifically called out in the standard, their use in this manner is consistent with the guidance in the standard.

The NASA Systems Engineering Handbook (2007) is based on high-level NASA systems engineering policy, systems engineering best practices collected across the NASA organization, and Government, industry and academic sources. The bibliography is extensive and includes EIA-632 (1999), ISO/IEC 15288 (2008), the DoD Architecture Framework (2007) and many other sources which also form the basis for DoN systems engineering policy and processes. The two sections of the handbook most pertinent to this study are sections 4.2, *Technical Requirements Definition*, and 4.3, *Logical Decomposition*. The former describes a process of interactively and recursively translating stakeholder expectations into a set of validated technical requirements with measures of performance, taking into consideration

constraints and the operational concept. The recommended means of documenting these requirements is "…in acceptable 'shall' statements, which are complete sentences with a single 'shall' per statement." The latter prescribes as a first step, establishment of a system architecture model. These two NASA process steps align well with ISO/IEC 15288's "Stakeholder Requirements Definition" and "Requirements Analysis" process steps with the notable exception of the NASA handbook including architecture development.

The U.S. Air Force *SMC Systems Engineering Primer & Handbook* (2005) is sponsored by the Space and Missile Systems Center, Los Angeles Air Force Base, Los Angeles, CA. The close relationship of Air Force and NASA engineering for Air Force space systems explains the SMC S.E. Primer (2005) citation of NASA documents such as the NASA Systems Engineering Handbook. Other references include military, industrial and academic sources frequently cited by DoD and DoN guidance such as DAU's System Engineering Fundamentals (2001), EIA 632 (1999), IEEE Std 1220 (2005), The INCOSE Systems Engineering Handbook (2007), and S*ystems Engineering and Analysis* (Blanchard, 2006). However, for the portion of references most specifically pertinent to this study, those in Appendix D, *References and Bibliography*, "Mission Requirements," the references are NASA references. The USAF handbook is laid out somewhat differently than the other handbooks, with a major chapters dedicated to: systems engineering "primer," how the systems engineering process works, system life cycles phases, systems engineering management, system engineering tools, "companion disciplines to systems engineering," and

validation and verification. As a result, material addressing requirements analysis is spread across major chapters in the handbook.

The primer establishes some semantic basis for terms used throughout the text. Chapter II discusses requirements analysis and includes development of a functional architecture, "…how the functions will operate together to perform the system mission(s)." A figure on page 51 of the text shows a notional functional architecture for a space system, including depiction of system functions provided by existing space assets. This notion of incorporating existing architectures in new system development is an important point that is further discussed, later in this study. Chapter III focuses on relating the U.S.A.F-specific National Security Space (NSS) system development process to the acquisition process described in DoD Instruction 5000.2 (2008). Chapter IV is a treatise on systems engineering management. It does not provide further discussion of requirements analysis. Chapter V discusses systems engineering tools. There are references to sources that provide comparisons of specific software-based tools, i.e., software products, but this section's emphasis is on analytical techniques commonly found useful, arranged by systems engineering process phase. Such tools as they apply to architecture development are discussed as are requirements management tools. The U.S.A.F. has developed a Government-owned tool called "Requirements Development and Validation" (RDAV) for maintaining a database of requirements, specifications and constraints. Chapter VI, "What are the Companion Disciplines to Systems Engineering" discusses primarily engineering specialty disciplines such

as producibility, reliability, and safety. This chapter does not augment the discussion of requirements analysis, nor does Chapter VII, “Validation and Verification.”

Among the four appendices included is 111 page, Appendix C, “Templates and Examples,” which contains templates and examples for common systems engineering tasks. Section C5 of this appendix, “Techniques of Functional Analysis,” includes the methodologies and rationale for functional flow block diagrams and timeline analysis.

E. CHAPTER SUMMARY

Figure 3 summarizes the attributes of the major standards and handbooks used by the DoN, allowing a more direct comparison of major similarities and differences.

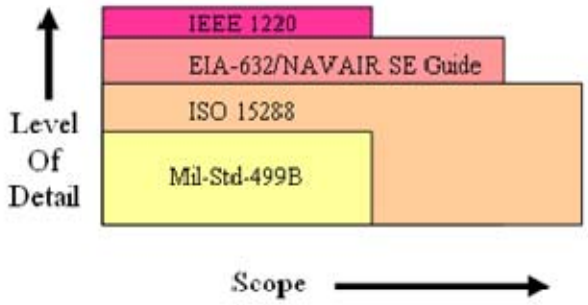

Figure 3 Comparison of SE Standards, From Langford, 2006.
The oldest standard, Mil-Std-499B has the narrowest scope, the newest, ISO 15288, has the broadest scope. IEEE 1220 is complementary to ISO 15288 and adds process detail.

The standards in general provide a framework of tasks, but not enough process detail, examples, or explanation of process variations for different product types to serve as

stand-alone source references from which a detailed, systems engineering plan (SEP) could be developed. For requirements analysis, the use of architectures is neither precluded nor prescribed. Systems engineering guides add detail, implementation considerations, and examples, and should be used as an adjunct to chosen standards. Among standards and guides, there are common threads. Semantics are often not well-defined and vary among standards and guides. A “performance specification” in one guide may have a subtly different meaning in another guide. Further, products such as performance specifications allow for variation in implementation. This allowable variation, together with the iterative and recursive nature of the requirements analysis makes it difficult to form a specific view of the sequence of process steps from user need through finalization of a performance specification that fits all product development situations.

III. ESTABLISHING USER NEED

A. INTRODUCTION

In discussing the systems engineering process, a description was presented wherein the customer would present a statement of need to the developer. Through an iterative process between user and developer, that requirement, including operational considerations and constraints is further developed to form a comprehensive set of system requirements from which product design begins. This chapter discusses the initial steps of the requirements setting process.

In 2003, DoD implemented the Joint Capabilities Integration and Development System (JCIDS), a top-down, joint capabilities (i.e., requirements) generation, validation, and prioritization system intended to reduce functional redundancy resulting from Service-specific systems as well as inadequate intra-Service and inter-Service interoperability among Services' systems. The JCIDS process interjects additional steps between the statement of user need and the development of a system specification relative to a typical systems engineering process as described in Fabrycky (2006). This chapter places the JCIDS process in the context of a systems engineering process and discusses its effectiveness.

B. THE JCIDS PROCESS

Figure 4 (Gonzales, 2007) shows the relationship of the JCIDS to the acquisition process. Tenets of the JCIDS are that it is a top-down requirements process within an

operational context and that requirements analysis is done in a joint context. The impetus for creation of the JCIDS was based on evidence that previously, when requirements analysis was performed at the Service level, Services developed Service-specific systems that were sometimes functionally redundant with other Services' systems and sometimes were not interoperable with other Services' systems. Previous to the JCIDS, requirements were also developed from the top, down, and consideration for Joint requirements is not new. However, then, as now, requirements were sometimes initiated by the Services from the bottom-up, based on operational need. The JCIDS, with a more rigorous, and Joint-led requirements analysis process, is intended to strengthen the top-down, Joint approach.

To illustrate the difference between bottom up and top down requirements generation, in a bottom up system, a numbered Fleet Commander might, in response to recognized projections for adversary, quiet, diesel-electric submarines designed to operate in shallow water, identify a need for improved, surface anti-submarine warfare (ASW) capability in the littorals. The appropriate Navy sponsor would consider that requirement, its cost to implement, and perhaps other expediencies, and decide whether to invest in a surface ship-based system or system upgrade to provide the improved ASW capability. If the Joint Requirements Oversight Council (JROC) concurred with the requirement, the Navy would proceed with the requirement. Such a process might not have taken into account airborne ASW capability, projected operational environments and operational scenarios in the timeframe the capability would be fielded,

or whether other Services' sensor capabilities (e.g. space-based) might offer part of a solution.

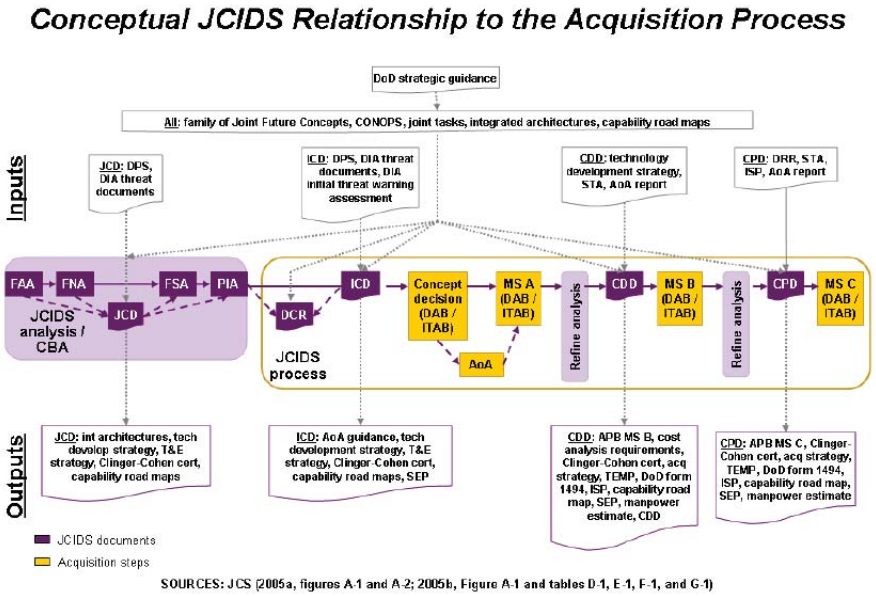

Figure 4 JCIDS; Preface to Acquisition, From Gonzales, 2007.

Top-down requirements approach, interrelationship of JCIDS and the acquisition process, and requirements artifacts are illustrated.

The JCIDS is a top-down requirements process, starting at the National Security Strategy (2006) level, and flowing down to a required capability, from which a product requirement can be derived. In concert with this is a capabilities assessment component to allow comparison of current and projected capability relative to the proposed capability improvement. In essence, the JCIDS process interjects a requirements definition and validation component between the user and the developer that assesses and aligns all Services' requirements, providing back to the Services a validated operational requirement from which product development can begin. The objectives are to find

the most efficient way to address an operational requirement and in the process, improve interoperability among the Services' systems.

C. RELATIONSHIP OF JCIDS TO THE SYSTEMS ENGINEERING PROCESS

Reviewing the JCIDS process as documented in CJCSI 3170.01F (2007) and CJCSM 3170.01C (2007), and referenced in the Defense Acquisition Guidebook (2004), indicates activities and products of the JCIDS process map to the requirements analysis portion of the systems engineering process. Since there is implementation flexibility built into the JCIDS, Defense Acquisition, and systems engineering processes, it is not possible to precisely describe a singular, linear sequence or timing of events and products that would be appropriate in all cases. However, each of the processes is defined in terms of an overall series of phases whose intent is to ensure subsequent phase does not begin until prerequisite products from a preceding phase are complete. Figure 5 is a simplified, notionalized representation of acquisition, requirements, and systems engineering process alignment. This graphically illustrates the role and positioning of JCIDS in the systems engineering and acquisition processes. It also serves to illustrate that the DoD acquisition process is not the same as a systems engineering process though there is overlap and interdependencies.

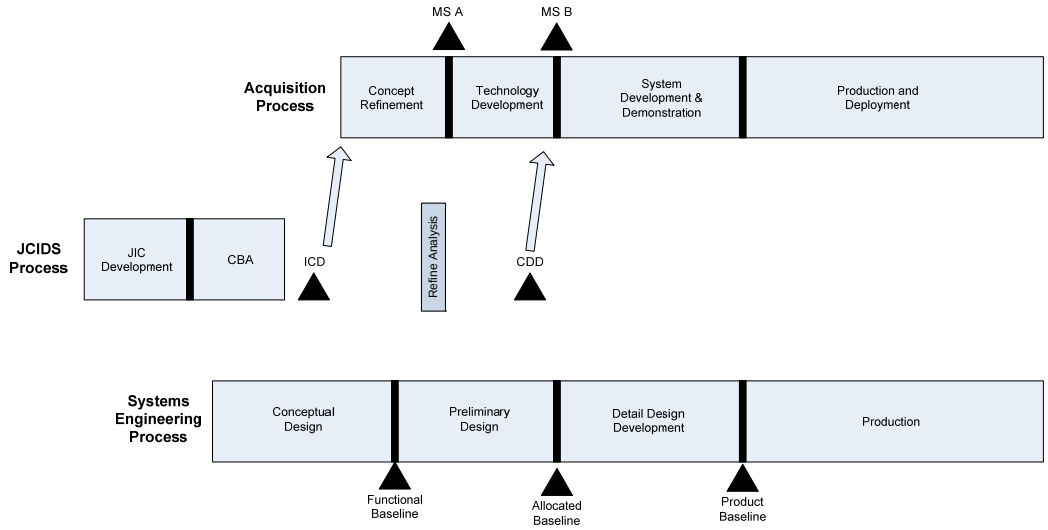

Figure 5 Alignment of Acquisition, JCIDS, and Systems Engineering Processes

These are the three, key, interrelated processes that play roles in the timing and development of system requirements.

JCIDS products are based on analysis and outputs of Capability-Based Assessment (CBA) which in turn are supported by high-level, national, military, and joint guidance, policy, and data. A CBA is composed of Functional Area Analysis (FAA), Functional Needs Analysis (FNA), and Functional Solutions Analysis (FSA). These three components can be summarized respectively as providing answers to the three questions: What is the military problem to be studied? How well does DoD address the problem with its current program? What should the DoD do to address any shortfalls? (Joint Chiefs of Staff, J-8, 2006). The answers to these questions comprise a validated user need.

Typically, an Initial Capabilities Document (ICD) is developed based on results from the CBA. Per CJCSI 3170.01F, 2007) an ICD:

> Documents the requirement for a materiel or non-materiel approach, or an approach that is a combination of materiel and non-materiel, to satisfy specific capability gap(s). It defines the capability gap(s) 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 ICD summarizes the results of the DOTMLPF and policy analysis and the DOTMLPF approaches (materiel and non-materiel) that may deliver the required capability. The outcome of an ICD could be one or more joint DCRs or capability development documents. (p. GL-9)

Subsequently, additional requirements analysis and refinement leads to development of a Capability Development Document (CDD). The CDD defines authoritative, measurable, and testable capabilities as a prerequisite for a Milestone B (DoDI5000.02, 2008) decision allowing entry into the System Development and Demonstration (SDD) program phase. It provides the basis for development of a system design (CJCSM 3170.01C, 2007).

The correlation of the CDD with the acquisition process' SDD indexes the JCIDS process to the acquisition process. For the systems engineering process, logical index points could be the functional baseline or the allocated baseline, the points of initiation of preliminary design and detailed design respectively. This study proposes that the CDD could be indexed to either baseline or some intermediate point, depending on the nature of the requirement. For instance, many major DoN programs comprise upgrades to existing systems. For ship programs, cruiser modernization, i.e., "CG Modernization," would be one

example. In this case, functional allocation, at least at the system or subsystem level, is preordained by the legacy architecture. For “new design” programs such as Zumwalt Class Destroyer, i.e., DDG-1000, a function such as survivability could be allocated during the SDD phase to (notionally) some combination of stealth technologies and the ability of the ship to maintain seaworthiness if struck by an adversary’s weapon. In the former case, the CDD would be indexed closer to the allocated baseline. In the latter case, the CDD would be indexed closer to the functional baseline.

The Naval Systems Engineering Guide (2004) incorporates JCIDS’ activities and outputs into a systems engineering context. Three of the 33 normative processes adapted from ANSI/EIA-632 (1999), are the most pertinent to the JCIDS process. They are: “Acquirer Requirements,” “Other Stakeholder Requirements,” and “System Technical Requirements.” Details of inputs, activities, and outputs from those processes can be found in the Naval Systems Engineering Guide (2004).

D. JCIDS AND INTEGRATED ARCHITECTURES

Maier (2002) defines architecture as “The structure – in terms of components, connections, and constraints – of a product, process, or element.” The JCIDS process is intended to utilize integrated architectures (CJCSI 3170.01F, 2007). The glossary in that document defines integrated architectures as “…consisting of multiple views or perspectives (operational view, systems view, and technical standards view) that facilitates integration and promotes interoperability across capabilities and among

related integrated architectures." So really, the term *integrated architectures* refers to integration of views as opposed to two or more separate architectures that have been integrated into one architecture. The meaning of the different views, simply stated, is the operational view describes what a system does, the systems view describes how a system performs, and the technical view comprises applicable technical standards that constrain the solution.

Overall, the JCIDS instruction does not suggest how integrated architectures should be used nor does it provide any guidance regarding the expected level of detail, or the information it might capture from the CBA or overarching strategic guidance that flows down to a Joint Integrating Concept (JIC), utilized in performing a CBA.

CJCSM 3170.01C (2007), in its guidelines and procedures, both prescribes the use of existing architectures to support CBA's and suggests results of the FSA can influence the future direction of integrated architectures. So, there is the implication of an evolution of integrated architectures that is interdependent with the requirements, i.e., JCIDS, process.

To illustrate, architecture products specified by CJCSM 3170.01C (2007) as required for an ICD comprise OV-1 and others if desired, while architecture products specified as required for a CDD include AV-1, OV-1, OV-2, OV-4, OV-5, OV-6C, SV-2, SV-4, SV-5, SV-6 and TV-1. These products are defined in the DoD Architecture Framework (2007). *AV* refers to *All Views*, *OV* refers to *Operational Views*, *SV* refers to *System Views*, and *TV* refers to *Technical Views*. OV's correlate to operational

requirements, SV's correlate to system requirements, and TV's correlate to standards that constrain the design solution, e.g., industry standards for data interfaces.

What is not explained in either JCIDS policy or the Naval Systems Engineering Guide (2004) is the manner in which architectures, as "a communication tool…presenting a common set of information with multiple views" (Richards, 2007) can be used as a systems engineering tool, that is, the concept of using architectures to organize, analyze, and manage the data that comprises CBA inputs and outputs. The importance of capturing and preserving the data is recognized, but not the value of capturing and preserving the interrelationships of the data, an essential element of architectures.

E. JCIDS IMPLEMENTATION INSIGHTS

GAO Report GAO-08-1060, *Defense Acquisitions; DoD's Requirements Determination Process Has not Been Effective in Prioritizing Joint Capabilities* (2008) concluded: "The JCIDS process has not yet met its objective to identify and prioritize warfighting needs from a joint capabilities perspective." The report goes on to say DoD lacks an effective analytic framework to assess and prioritize warfighting needs. The GAO report further cited poor inter-service coordination and inadequate resources applied to JCIDS analysis. The DoD partially concurred with the framework finding and concurred with the resourcing finding. Though use of architectures is not mentioned in the report, architectures can in fact constitute a framework for analysis, though the cross-domain, cross-service architectural infrastructure currently in place is

not adequate to support such analysis. A recent Rand National Defense Institute Report (Gonzales, 2007) notes "…DoD uses a bottom-up architectural development process. This bottom-up approach can result in much duplication of effort across the entire acquisition system and thus make the development of a single, integrated architecture that summarizes DoD interoperability requirements difficult to achieve."

The 2007 Rand report (Gonzales, 2007) also offers two criticisms of JCIDS. First, it states that the JCIDS processes and products are described ambiguously as are their relationships with acquisition process products. While ambiguity is apparent in some of the descriptions of processes and products (e.g., one product being listed as both an input and an output of an activity), it is noted that the authors of the Rand paper, written at the direction of the DoN, did not acknowledge or reference the Naval Systems Engineering Guide (2004) or the CJCSI White Paper (White paper on CBA, 2006). The former places the JCIDS process in a systems engineering context, albeit not a systems acquisition context, and the latter provides significant insight on practical approaches to planning and executing JCIDS analysis, based on experience. The second criticism of the report (Gonzales, 2007) cites lack of formal traceability from the source of a user need through the JCIDS process to disposition.

A report by the Institute for Defense Analysis (Hanley, 2006) was performed under contract to the Director, Force Structure and Resources (J8), the Joint Staff. The stated objective of the study documented by the

report was to produce an analytic framework for “capabilities-based planning” processes in the DoD. The stated premise was that there are multiple, capabilities-based planning processes in use, JCIDS being one, that are not synchronized with one another. The report concludes with a series of recommendations, including “next steps” for taxonomies and data that notes a dichotomy between analytical and taxonomic breadth, agility, and stability, versus depth, complexity, and rigor. The report notes that ease of implementation and holistic analysis favors the former while analytical fidelity and valid insights favors the latter. The report presents no conclusion on the possibility of establishing a standardized capability taxonomy that could serve all needs of all stakeholders.

F. CHAPTER SUMMARY

The JCIDS process, as incorporated into systems engineering processes described in the Naval Systems Engineering Guide (2004) offers the potential for a cross-service requirements analysis and prioritization capability necessary to truly implement capability-based acquisition. It could also use architectures as an analytical and management framework. However, research for this study indicates the use of architectures in conjunction with JCIDS, while prescribed, is not really explained in policy or guidance, and to date, the Services’ approaches to warfighting requirements continue to align to Service perspective rather than a Joint perspective.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. DOD ARCHITECTURE FRAMEWORK

A. INTRODUCTION

To define “architecture framework” first requires differentiation of the term “architectures” from the term “frameworks.” An architecture does not require a framework. The definition of “architecture” cited in the DoD Architecture Framework Version 1.5 (2007) is: “the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time.” Every existing system embodies an architecture yet relatively few architectures of existing systems were developed using an architecture framework. Conversely, the term “frameworks” need not be related to architectures as is evident from Webster’s II Dictionary (1984) definition of the term: “a basic arrangement, form, or system.” Rolf Siegers (2005) defined architecture framework simply as “a resource that aids in the development or description of an architecture.” Maier (2004) asserts that an architecture framework exhibits one or more of the following five characteristics:

1. A definition of the word “architecture”
2. A conceptual framework explaining key concepts or terms
3. An approach to describing architectures
4. Architectural methods (e.g., creating, analyzing, interpreting, realizing)
5. A theory of how architectures and architectural descriptions fit into a larger context (e.g., systems engineering, design, etc.)

Motivation to establish and implement architecture frameworks can be characterized as either driven by business considerations, technical considerations, or both. An architecture framework used for enterprise product development can for example facilitate shorter product development cycles, lower manufacturing, support and upgrade costs, and a reduced number of functionally duplicative products. This can be explained by a framework's ability to make individual product architectures comparable and integrable, facilitating cross-product insights that would be difficult otherwise. From a technical perspective, and for the same reasons cited in the business perspective, products developed in conformance with an architecture which is in turn developed in conformance with an architecture framework potentially exhibit a greater degree of interoperability, a simpler logistics support infrastructure, and in systems of systems, a greater ability to be reconfigured to adapt to changes in mission needs. The DoD architecture framework (2007) underlies DoD requirements, acquisition, and systems engineering policy and guidance for both business and technical reasons.

B. ORIGINS OF THE DOD ARCHITECTURE FRAMEWORK

According to a GAO report on Defense information superiority, (1998) the DoD, as a result of communication interoperability problems during the Viet Nam War, has been trying since 1967 to establish some form of Department-wide Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) architecture. In the intervening years through the early 1990's, efforts to

establish architecture frameworks for C4ISR continued, but primarily at the Service level with the expectation but not the requisite management structure that Joint interoperability would follow. Continuing interoperability issues during operations Urgent Fury, (Grenada, 1982), Just Cause, (Panama, 1989-90), and Desert Shield/Desert Storm (Kuwait, 1991) led to a concept documented in the Joint Staff paper “C4I for the Warrior” (1992) and endorsed by then-Chairman of the Joint Chiefs, Colin Powell. The paper laid out near-term, mid-term, and long-term plans and objectives to provide the battlefield commander “access to all information needed to win in war…when, where, and how he wants it.” The paper did not explicitly call for a DoD architecture framework but discussion of utilizing “common information exchange standards” and “migration from unique military standards to commercial national and international standards” showed intent to at least establish a standards-based framework.

The Defense Information Systems Agency (DISA) in October 1991 issued a set of Adopted Information Technology Standards (AITS) as the DoD Technical Architecture Framework (TAFIM) for Information Management, a technical i.e., standards-based, architecture framework. The TAFIM was in turn based on a National Institute of Standards and Technology (NIST) product called the “Application Portability Profile” (DoD Technical Architecture Framework, 1994). The Joint Technical Architecture (JTA) supplanted the TAFIM which was cancelled in January, 2000. The JTA has since been replaced by the current DoD IT Standards Registry (DISR).

The Deputy Secretary of Defense in October, 1995 directed an effort to improve processes to ensure adequate, Joint C4I for warfighters. Under the direction of ASD(C3I) the C4ISR Integrated Task Force was established, and subordinate to the task force was the Integrated Architectures Panel. That group undertook the task of establishing an architecture framework based on three architectural views: operational, systems, and technical. The group incorporated substantial content from previous and ongoing Joint and Service architecture efforts such as TAFIM. The panel's product, the C4ISR Architecture Framework, Version 1.0, was approved on 7 June 1996. Shortly thereafter, the C4ISR Architecture Working Group was established to continue and build upon the work begun by the Integrated Architecture Panel. The resulting product was Version 2.0 of the C4ISR Architecture Framework, approved on 18 December 1997.

C. EVOLUTION TO THE DOD ARCHITECTURE FRAMEWORK

The Clinger-Cohen Act of 1996 was intended to improve acquisition management of Government Information Technology (IT) systems. It established the term "National Security System" to mean essentially IT-related defense systems, and waived many of the provisions of the act for National Security Systems. However, it did require establishment of an Information Technology Architecture for National Security Systems and responsibility for that was assigned to the DoD Chief Information Officer (CIO). The term "National Security System" was significantly more inclusive than the term C4ISR. From this grew the more broadly-scoped

DoD Architecture Framework, using Version 2.0 of the DoD C4ISR Architecture Framework as a basis.

Under the auspices of the DoD Architecture Framework Working Group, DoD Architecture Framework Version 1.0 was approved on 15 August 2003. Significant changes relative to DoD C4ISR Architecture Framework Version 2.0 included:

- Guidance provided to tailor product selection based on the intended use of the architecture.
- Greater emphasis on the architecture data rather than just architecture products.
- Content (techniques, processes, and examples) was added to provide some explanation of architecture development and use.

A revised DoD Architecture Framework, Version 1.5, was approved on 23 April 2007. The Version 1.5 document notes among significant changes, more emphasis on architecture data rather than architecture products, introduction of the concept of federated architectures, and incorporation of the Core Architecture Data Model (CADM) as an integral component of the DoDAF. It is noted that Version 1.0 of the framework also asserted greater emphasis on architectural data and also presented CADM as “the DoD standard architecture data model for Framework-based architecture data elements” (DoDAF Version 1.0, 2003) though in that version, CADM compliance was not directive. This may be explained by the fact that at the time of Version 1.0 issuance, the Defense Architecture Repository (DAR) which requires CADM compliance, was still under development.

D. REQUIREMENTS TO USE THE DODAF

DoD policy, which requires use of integrated architectures, flows down to DoN policy. In DoDD 5000.1

(2003), integrated architectures are mentioned only in the context of ensuring interoperability requirements are met. There is no direct citation of the DoDAF as a reference or requirement. DoDI 5000.2 (2008) also does not directly cite the DoDAF as a reference or requirement. DoD 5000.2 (2008) does stipulate "…The capability needs and acquisition management systems shall use…*integrated architectures*…in an integrated, collaborative process to define needed capabilities to guide the development of affordable systems" It further stipulates operational, systems, and technical views, and the use of the DISR for selection of standards. The primary emphasis regarding use of architectures is on requirements validation and interoperability, not as a vehicle for translating user needs into system requirements.

SECNAVINST 5000.2D (2008) notes that the Defense Acquisition Guidebook (DAG 2004) provides "…best practices and lessons learned…" to augment DoDI 5000.2 (2008). However, Chapter 4 of the guidebook, "Systems Engineering" is silent on the use of integrated architectures and does not list the DoDAF as a resource. Chapter 7 of the guidebook "Acquiring Information Technology and National Security Systems" does address the development and use of DoDAF-compliant, integrated architectures and cites specific architecture products required by policy. However, the focus of Chapter 7 is on successful development of net-centric systems. While net-centricity is a highly valued attribute of emergent warfare systems, it is only one facet of the overall set of requirements and therefore represents only a portion of the requirements analysis process.

DoDD 4630.5 (2004), Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS) and DoDI 4630.8 (2004), Procedures for Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS) provide specific direction with regard to the development and use of integrated architectures. DoDI 4630.8 (2004) notes “Integrated architectures are the common foundation for capability-focused, effects-based IT and NSS interoperability and supportability processes…”.

SECNAV Instruction 5000.2D (2008) says about integrated architectures: “All DoN new start IT systems, including NSS, that exchange information with external systems shall comply with NR KPP (Net Ready Key Performance Parameters) and FORCEnet integrated architecture and other elements of the FORCEnet Consolidated Compliance Checklist (FCCC) guide as described by the CDD at program initiation,” i.e., normally Milestone B. The Net-ready KPP stipulation requires production of specific DoDAF-compliant architecture products. The document also stipulates use of a Systems Engineering Plan (SEP) and references OSD SEP preparation guide, Version 2.01 (2008). SECNAVINST 5000.2D (2008) states “SEP shall address development of a systems architecture using the DoDAF, the FORCEnet integrated architecture, and the Naval open architecture.”

The Naval Systems Engineering Guide (2004), based on EIA 632 (1999), does not directly specify development or use of DoDAF architecture products, though by inclusion of the JCIDS process in the systems engineering process, ICDs, CDDs and CPDs are included and so, by extension, are DoDAF

architecture products. The Naval Systems Engineering Guide (2006) takes a holistic, DOTMLPF, i.e., doctrine, organization, training, materiel, leadership and education, personnel, and facilities, perspective to systems engineering. Interoperability requirements are not singled out from the rest of systems engineering considerations as they are in other DoD and DoN policy and guidance previously discussed in this chapter.

E. DEMONSTRATED UTILITY AND VALUE OF DODAF

This section reviews and analyzes literature that explains and summarizes application of the DoDAF and the insights it provides or decisions it supports. It also provides some overall characterization of the use of the DoDAF, the reported benefits and shortcomings in content or application, and the means by which practitioners have been able to improve application of the DoDAF or the value it provides. A substantial, systems engineering literature search from five years ago to the present, and a selective search of the last 10 years, did not identify significant “case study” type literature reporting DoDAF successes. This might be construed as evidence that among systems engineering researchers and practitioners DoDAF success stories are not widespread. A success story for the purposes of this study would be defined as implementation of DoDAF that results in improvement of the systems engineering process, measureable in terms of cost or schedule improvement or risk mitigation to a degree that more than offsets the cost and time required to develop, maintain, and utilize a DoDAF-compliant architecture.

Although significant literature addressing specific DODAF use and effectiveness was not found, a Microsoft PowerPoint presentation from an OASD, NII-sponsored, Government, DoD, and industry-wide survey (OASD (NII) Architecture Survey, 2005) was found that characterized DoDAF architecture usage in general, and summarized survey comments pertaining to specific instances of DoDAF application and the consequent value accrued. There were 120 respondents. Demographics of DoD architects were presented, types of decisions enabled by DoDAF architectures were identified, and DoDAF strengths, weaknesses, and suggestions for improvement were presented. A review of salient insights from that survey offers insight on the “state of the practice” and practitioners’ beliefs of the value of DoDAF architectures.

Almost half of the architects were contractors, i.e., performing architecture work in support of Government customers or Government contracts, half had one-to-five years experience, and almost half worked on teams of one to five. The most frequently used training source was tool vendors, i.e., tools in the sense of software supporting documentation and development of architectures. The predominance of contractors as architects, the relative inexperience of architects, and the bias toward small architecture teams may be indicative of lack of commitment to architecture development and use at the program level by the Government, and a tendency for architects to learn how to develop architectures through their own experience instead of building on a knowledge base of others’ experience.

Respondents represented 53 organizations and over 80 architecture projects. Almost 75% of respondents used Telelogic's "System Architect" tool. Telelogic's web site indicates more than 1000 commercial, government and military users of their software product line world-wide. The software product line includes System Architect as well as other engineering tools. These figures lie in contrast to the scarcity of case study literature. The numbers in fact indicate architecture development is being conducted on a fairly broad scale. One explanation of this contrast is that development of DoDAF architecture products is mandated for most DoD programs, yet they are often developed in isolation of the system engineering process, undermining much of the potential value of architectures and therefore explaining why "success stories" are hard to find. The survey did not address what the effect might be if DoDAF architectures were developed solely at the discretion of a program manager or chief systems engineer.

Some of the most interesting survey data derives from answers to open-ended questions. Three questions in particular provide insight on architects' beliefs of the utility and value of architectures:

- What values, Benefits, and Impacts are attributable to their organization's Use of Architectures?
- What decisions are expected to be made based on architecture analysis?
- What are your architecture Successes; where architectures made a difference?

In the context of the survey, key among listed values, benefits, and impacts were: “Common frame of reference for all manner of discussions and decision-making” and “Supports decision-making and identification of issues.” Both of those benefits correlate to the use of architectures during the requirements definition process. Among expected decisions supported, “…negotiate MOAs (i.e., Memoranda Of Agreement) and communicated requirements with contractors and users” and “Identify capability gaps…report capabilities based on requirements…prioritize projects” are germane to this study. Some of the most significant, architectural successes listed were those asserting that architectures provided objective justification for initial or continuation of funding for a project or program. That suggests use of architectures to support the requirements definition process, consistent with the intent of the JCIDS process.

Survey respondents commented on DoDAF strengths and weaknesses. Themes expressed in “strengths” were maturity, wide acceptance as a standard, and the consequent ability to compare and analyze architectures on a common basis. Weaknesses expressed concerned both development and use of architectures. With regard to development, there was concern that in spite of the standardized structure of DoDAF, there was too much variability in lexicon, taxonomy, metadata and other attributes to allow integration or comparison of architectures, above-mentioned strengths notwithstanding. There was also concern that architectures were not capability-focused, in other words, not inclusive of the breadth of DOTMLPF. Finally, among significant

weaknesses listed was lack of clear guidance and policy for what to do with architectures that have been developed.

Russell (2005) notes:

> …the output of most architecture efforts tends to be a three ring binder that weighs five pounds or so which no one ever reads. This has given a bad name to the architecting process and has left many decision makers asking why they spent their limited money and time producing architectures.

F. NEXT STEPS IN THE EVOLUTION OF THE DODAF

Update of the DoDAF to Version 2.0 is being managed by the Architecture and Interoperability Directorate, under the Office of the DoD CIO. The organizational structure, as presented by Mr. Brian Wilcynski of the DoD CIO is shown in Figure 6 (Wilcynski, 2007).

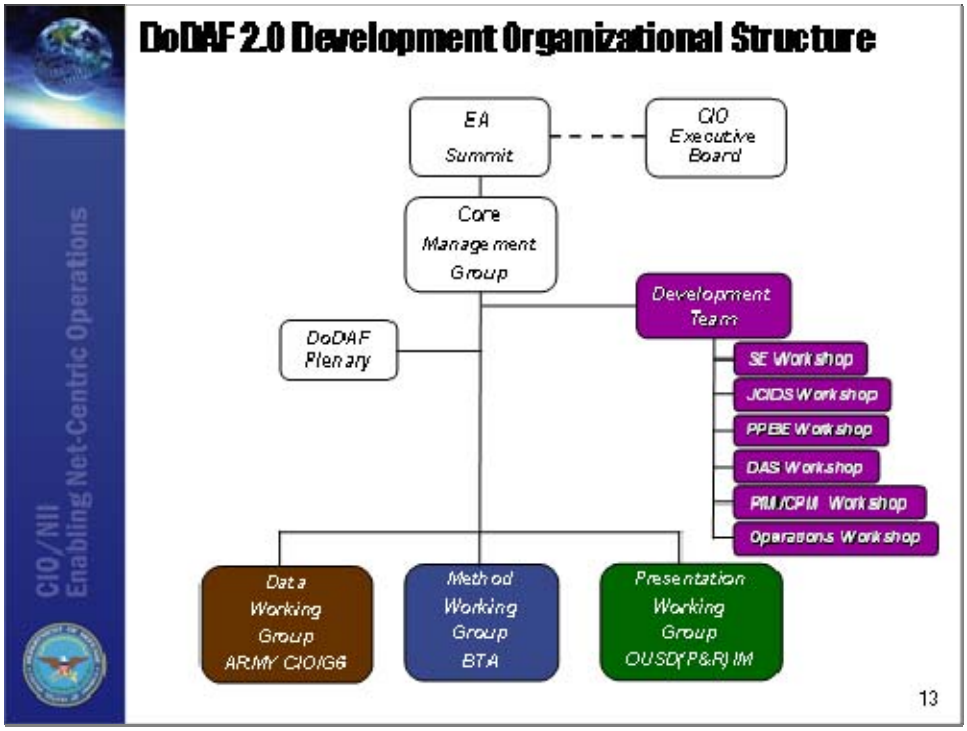

Figure 6 DoDAF Development Organizational Structure, From Wilcynski, 2007.

Development of this update to DoDAF Version 1.5 is ongoing.

The vision statement for the DoDAF 2.0 is: “To enable the development of architectures that are meaningful, useful, and relevant to the DoD Requirements, Planning, Budgeting, Systems Engineering, and Acquisition decision processes” (Wilcynski, 2007). The stated (Wilcynski, 2007), high-level areas of focus are:

- Focus on information requirements versus products as in previous versions
- Be driven by decision process requirements
- Support DoD shift to service orientation of architectures
- Support a federated enterprise architecture approach

In terms of scope, six DoD processes the DoDAF is intended to support are: JCIDS, Systems Engineering, Operations, Portfolio Management, Program, Planning, Budgeting, and Execution, and Defense Acquisition System. As of April, 2008, requirements-gathering workshops had been conducted in support of the first three of those processes.

Publication of the DoDAF Version 2.0 is expected before the end of Fiscal Year 2009. During development, a series of plenary sessions is being held to keep the architecture community in general apprized, and to solicit feedback and comments. Briefs are provided at the working group level: Data, Method, and Presentation, as shown in Figure 5.

G. CHAPTER SUMMARY

In the introduction of this chapter, reference was made to five attributes that could be used to characterize

an architecture framework: An architecture definition, key terms, an approach for describing architectures, architectural methods, and where architectures fit in a larger context (Maier, 2004). From the first standards-based framework through the in-development DoDAF Version 2.0, those attributes capture both the intent of framework development efforts and the continuing shortfalls. By virtue of the fact that DoDAF use is embedded in DoN policy and guidance, it is the standard for architectures in DoN. However, considerable variation in DoDAF architecture implementation is evident within the boundaries of DoDAF compliance, such that architectures for different but related systems in many cases cannot be effectively compared or integrated. The current framework is necessary but not sufficient to ensure the effective use of architectures in acquisition and systems engineering. The DoDAF Version 2.0 is intended to address some long-standing shortcomings, but the framework is only one part of what should be a multi-faceted approach to realizing the potential of architectures in the systems engineering process.

V. INTEGRATING KEY PROCESSES WITH THE OVERALL SYSTEMS ENGINEERING PROCESS

A. INTRODUCTION

Previous chapters discussed the systems engineering process, the DoD requirements process, i.e., JCIDS, the DoD architecture framework, and to a limited extent, the DoD acquisition process. These processes in effect constitute a complex system-of-systems. The objective of this study is to examine the portions of the processes that relate to translation of user needs to a comprehensive, unambiguous, verifiable set of system requirements. This chapter identifies the critical components for successful requirements translation in a systems engineering context, the necessary elements of a good system specification, and considers how best to integrate necessary process components that lie both inside and outside DoD systems engineering process. Weaknesses in the current processes and recommendations for improvement are identified and involve both technical and managerial components. This study is primarily focused on the technical aspect, though key, high level management issues are recognized.

B. TRANSLATING USER NEEDS TO SYSTEM REQUIREMENTS

In its simplest form, a user need can be expressed in a single requirement that embodies a set of requirements clearly understood by the person receiving the requirement. For example, if someone asks “Do you have a pencil I can borrow?” that is an easily understood user need. The user needs a pencil. Not stated, but understood with a reasonable degree of certainty are the following: The user

wants a pencil with a point on it, i.e., not broken. The user wishes to use the pencil for some task and at the completion of the task, will return the pencil. The user does not particularly care whether the pencil is a mechanical or “wooden” pencil. Does that leave any uncertainty? Yes. Maybe the receiver of the request has only a red marking pencil. In all likelihood, that person will say “I’ve only got a red pencil. Is that OK?” At that point, there has been a statement of user need and the “developer” has communicated back to develop a more precise understanding of the need. After all, the user may have wanted a pencil only to scratch a spot in the middle of his back, in which case the type of pencil and whether it had a point is inconsequential. Nonetheless, “user” and “developer” must communicate to ensure the product satisfies the need. Communication must be in clear and unambiguous terms. The developer either identifies a solution that fully satisfies the user need, or a solution is negotiated that provides a lesser, but acceptable solution, or, the developer indicates there is no feasible solution, i.e., no pencil or other implement of inscription is available.

If the object of the example is changed from a pencil to $500, immediately the requirements definition and feasibility of a solution become more complex. How long will the loan be for? Will it be interest-free? Can the “user” be depended on to pay off the loan? Maybe the user really doesn’t need $500 but is simply unaware of other means of financing that would not require a large down-payment. Maybe the intended use of the money is an important factor. Is the money to pay for medicine for a

sick child or to bet on horse-racing? Communications with the user might break down if the user and developer cannot reach a common understanding of the need. Thus, even this trivial example can easily be made complex.

The previous example, though trivial in comparison with typical DoN warfighting requirements, still embodies some important principles and concepts. The user need can usually be expressed singularly or in a very small number of statements. Chief of Naval Operations Admiral Roughead's CNO Guidance (Roughead, 2008) at the top level, is simply: "Build the Future Force, Maintain Warfighting Readiness, and Develop and Support our Sailors, Navy Civilians, and Families." That is the CNO's requirement.

The rest of the Navy's operational requirements are subordinate and should be traceable, noting that as requirements are flowed down from the CNO guidance level to the system requirement level, complexity increases by orders of magnitude. Only in trivial cases will a statement of user need be sufficiently comprehensive and complete to fully define system requirements and establish a basis for product development. As stated in *Systems Engineering and Analysis* (Blanchard, 2006):

> Accomplishing the needs analysis in a satisfactory manner can best be realized through a team approach involving the customer, the ultimate consumer or user (if different from the customer), the prime contractor or producer, and major suppliers, as appropriate. The objective is to ensure that the proper communications exist between all parties involved in the process; i.e., the 'voice of the customer' must be heard, and the system developer(s) must respond accordingly.

There is a well-known and accepted systems engineering model first documented in the U.S. by Forsberg and Mooz in 1991 (Forsberg, 1991), referred to as the "Vee" model, shown in Figure 7. The process starts at the upper left hand leg of the vee with "understanding customer requirements" and progresses downward toward the bottom of the vee through a process of decomposition and definition from which system synthesis can begin. However, there can be a step in the system engineering process that precedes the beginning of the "Vee" model and is a process of composition rather than decomposition.

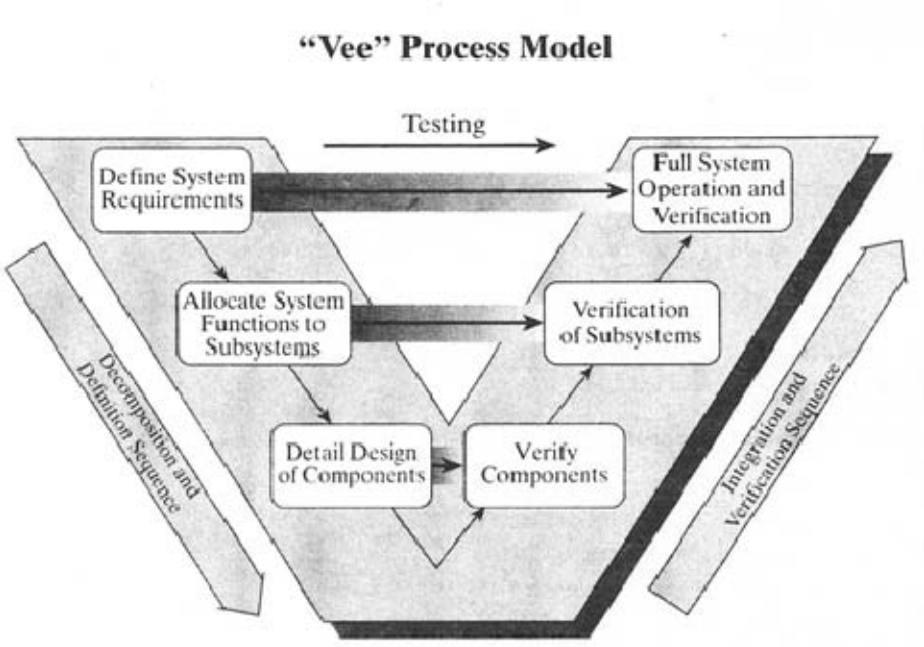

Figure 7 Engineering Vee Model, From Blanchard, 2006.

System requirement development is a process of decomposition. Synthesis of a solution is a process of composition. The "Vee" Model illustrates the progressively more detailed system requirements followed by the aggregation and integration of solution elements.

Mr. Michael Collins, president of Advantage Development, Inc., in a presentation to the NDIA (Collins, 2008), asserted that engineering requires an "initial point," defined by him as "Engineerable Requirements: the set of engineering requirements necessary and sufficient to initiate the successful engineering and production of the system." He further argues that architecting is a means of forming separate elements into a "coherent whole" that can serve as that initial point. The concept of needs analysis through a team-based series of communications described by Blanchard (2006) can be combined with the architecture process mentioned above to create a more rigorous, assessable process to derive a set of system requirements that tightly coupled to stated user needs.

The fundamental concept of translating user needs to developer's requirements, as well as where this step lies in an overall systems engineering model have been considered to this point at a high level of abstraction. In order to characterize an actionable process however, more detail is needed. For example: How are stakeholders identified? How are user needs captured in a logical, consistent manner? How is it known when this step is complete and correct? How is this most critical step performed with rigor and discipline? Recent systems engineering literature offers some answers to these questions. Bahill (1997) presents an overview of what he terms the "requirements discovery process." He asserts "…there is a uniform and identifiable process for logically discovering the system requirements regardless of system purpose, size, or complexity," and credits that belief to other authors. The steps (Bahill 97) identifies are:

Identify Customers and Stakeholders

1. Understand the Customer's Needs
2. Define and State the Problem
3. Write System Requirements
4. Review Requirements with Customer
5. Define Figures of Merit
6. Validate System Requirements
7. Verify Requirements
8. Identify Technical Performance Measures (i.e., Key Performance Parameters in DoD parlance)
9. Continue to review requirements with the customer throughout the development process.

This study is primarily concerned with steps "1" through "7." The first step, identifying customers and stakeholders, is sometimes discussed superficially in general systems engineering texts, simply saying it is very important. The terms "customer" and "stakeholder" are closely related. Bahill (1997) defines a customer as "…anyone who has a right to impose requirements on a system." Carson (2004) defines a stakeholder as "…anyone or any organization involved with or affected by the system lifecycle…". The group defined by the former, more restrictive definition can be considered a subset of the group defined by the latter definition. That is, all who have a right to impose requirements on a system presumably are affected by the system, but not all those who are affected by the system have a right to impose requirements.

To illustrate, consider in light of Bahill's (1997) definition of customer, the following example of a failed system from Bahill (2005): Management at Ford in the 1950's

overrode marketing recommendations and imposed requirements resulting in the Edsel automobile, an automobile without a market, and therefore a failed design. By Bahill's definition, management acted as a customer while marketing did not. Management withheld from marketing the right to impose requirements, leaving marketing as a stakeholder but not a customer. Management failed to recognize marketing's role as a representative for the end customer, those who would or would not buy the automobile. Next, consider Carson's (2004) definition of stakeholder. He specifically identifies and includes "nature" and *those not allowed to approve requirements* as stakeholders. Those categories would not fit Bahill's customer definition, but Carson is suggesting those members of the larger stakeholder set should be treated as customers; their requirements should be considered. In his view, someone must play the role of spokesperson for those stakeholders.

An example of failure to engage stakeholders was evident in a 2001 FBI project, a three year development contract to upgrade the Federal Bureau of Investigation's (FBI's) IT infrastructure and to design what was called the "Virtual Case File" which would allow the FBI to move from its antiquated paper-based investigation and records to computer-based investigation and records. A National Research Council letter report (McGroddy, 2004) provided a status of the project and concluded in part: "In essence, the FBI has left the task of defining and identifying its essential operational processes and its IT concept of operations to outsiders." The software-based system lacked some essential capabilities. It appeared that among other issues the FBI used contractors as FBI stakeholders, not

agency stakeholders themselves. The system developer in turn may not have exercised due diligence in validating the requirements. The program, nominally a $170M program, was cancelled and begun over with a new development contractor, eventually costing an additional two times the original program cost.

Carson (2004) suggests identifying stakeholders through a process of examining the system in its environment through all parts of its life cycle, development to disposal. For U.S. Navy warfare systems, it should be possible to establish a standard “menu” of stakeholders, though consideration should be given to any possible emergent stakeholders, particularly in the public domain. For instance, a requirement for a nuclear power plant on a ship will affect where a ship can be home-ported. The community adjacent to the proposed home port becomes a stakeholder. Or, if it was known that a ship would ultimately become an artificial reef (i.e. wildlife management or tourism stakeholder), consideration might be given in its design to ease its preparation for that final role. The associated state department of natural resources might be a stakeholder.

The developer’s understanding of customer needs is also critical and depends on effective, iterative, communication between customer and developer utilizing commonly agreed-upon semantics. This process of communication, translation, and negotiation merits further discussion. One component of the process is establishing a framework of requirements, e.g., hierarchy and categories.

Among the ways to categorize requirements, one is differentiating functional requirements from non-functional requirements. Functional requirements represent something the system must do. Non-functional requirements typically represent how well the system must do something. Among non-functional requirements, some pertain to one or a limited number of functions and others are overarching, such as satisfying operator safety standards.

There are several, well-documented methods of capturing user requirements in a manner that provides some assurance of completeness and correctness; Carson, (2004) summarizes them. They include checklist approaches, Quality Function Deployment-type approaches, use case and other functional analysis, series of reviews, and “context analysis,” the primary topic of that paper. Each has strengths and weaknesses and each is likely to be effective, depending on the application. One method stands out as being fundamentally different: Use cases. Use cases have the ability to capture the required functional behavior of a system; the other methods generally result in “shall” statements.” Defining required behavior with “shall statements” can be cumbersome and lacking in context. However, use cases alone will not capture all requirements. Daniels (2005) has suggested a hybrid approach that augments use cases to provide a comprehensive set of requirements, and uses shall-type (i.e., plain language) requirements to add detail missing from use cases.

The hybrid approach appears both sound and robust, giving consideration to both the effective communication element of translating user needs into system requirements

as well as the need for comprehensive requirements that capture the required behavior of a system, i.e., what the system has to do under conditions of use. Furthermore, this approach supports the use of architectures as a vehicle to communicate and reconcile user needs into system requirements. Daniels (2005) recognizes both the need for capturing a complex set of requirements in its entirety, as well as communicating requirements between user and developer in terms both understand. That is a strength of architectures, when applied properly. Cole (2006) asserts:

> The complexity of the SoS environment makes it difficult (if not impossible) to describe the problem with requirements alone. Architectures is a critical aspect of describing the problem, especially when user needs, technologies, organizational dynamics and external interfaces are continuously changing.

Cole (2006) focuses on the system engineering complexities that accompany systems-of-systems (SoS) development. While SoS is not the focus of this study, practically all non-trivial, Naval warfare system, systems engineering problems are SoS problems. In other words, U.S. Navy warfare systems are rarely standalone systems. They are part of or directly interact with other systems. Operational and systems architectures are implicit if not explicit.

Schindel (2005) discusses requirements statements as transfer functions and describes the necessary roles of “requirements prose” engineering models in describing complex systems, again reinforcing the idea that models, i.e., architectures, and prose, i.e., shall statements,

must be used together to describe a set of requirements and communicate those requirements between user and developer.

Referring back to Bahill's (1997) 10 steps to translate user needs into system requirements, the last six steps all contribute to ensuring completeness and correctness of the requirements. Discussion to this point has focused on requirements capture, essentially the first four steps, recognizing opportunities exist after that point to update or correct requirements. Determination of requirements completeness and correctness is the objective of validation and verification processes. Bahill (1997) described validation as ensuring requirements are consistent with one another, that a feasible solution exists, and that it can be demonstrated that the system fulfills its requirements. He describes requirements verification as determination that a requirement has been met, using testing, examination, or analytical methods. Requirements validation and verification are necessary but may not be sufficient to ensure requirements completeness. Validation and verification processes operate only on those requirements that have been recorded. Although validation and verification *may* lead to discovery of "missing" requirements, it is not assured.

Procedures exist to review requirements for completeness and correctness. Carson's (2004) objective was to "Develop and validate a methodology that can produce a complete set of requirements and that can determine the completeness of a set of requirements." Note that in this case, completeness is defined to include correctness. That is, if requirements are complete, they are correct. Carson

(2004) cites Mar (1994) when listing the following five characteristics of requirements completeness. Mar (1994) notes that among those five, numbers “1” and “3” are hardest to ensure:

1. All categories of requirements are addressed
2. All impositions of higher level requirements are accounted for
3. All scenarios and states are recognized and described
4. All assumptions are documented
5. Requirements are understandable and unambiguous

Carson (2004) also identifies two requirements cases: Those that do not inherit higher level requirements and those for which a complete set of requirements can be derived from higher level requirements. Typically, Naval warfare system requirements represent a third, hybrid case. In all cases there will be higher level requirements levied, but some higher level requirements cannot be decomposed or allocated in such a way that all lower level requirements can be unambiguously shown to support the higher level requirement. This precludes the use of allocation and traceability for determining completeness. For example, a ship’s signature, e.g., radar, acoustic, IR, etc., a ship’s electronic countermeasure capability, a ship’s ability to withstand weapon strikes, and a ship’s missile and torpedo weaponry all affect a ship’s self-defense capability. But, even at the “ship self-defense” level of abstraction, it may not be possible to quantitatively trace lower level requirements upward due to limitations in modeling. In this case, determination of

requirements completeness may have to be conducted in the manner the manner of Carson’s (2004) first case; as if higher level requirements are not inherited.

Carson (2004) focuses on the instance where higher level requirements are not inherited and suggests a set of requirements is complete if all stakeholders approve the requirements. His assertion is if all stakeholders are identified, approval of the requirements assures requirements completeness. This leads back to the issue of communication and understanding. What constitutes a stakeholder review? How can a stakeholder be assured his judgment to approve a set of requirements is well-founded? Carson offers a “formal approach to completeness,” the details of which are beyond the scope of this study. However it is important to note this approach incorporates requirements modeling and requirements prose as previously discussed. His approach also adds the stipulation that pragmatically, only a subset of the entirety of combinations of interface behavior and conditions can be analyzed. Those sets of conditions that do not affect required behavior are set aside.

Thus, an argument is made that for Naval Warfare systems, methods exist to translate user needs to system requirements utilizing models requirements and architecture models and plain language, and, that methods exist to ascertain completeness of system requirements. The scalability of this approach has not been shown and the specific models or the basis for selecting models has not been addressed. There have been instances where modeling of requirements was used to generate prose requirements,

therefore assuring consistency between the two. Next, this hybrid, model and prose type of approach will be compared with DoD and DoN policy and guidance to determine the viability, practicality, and issues that would be associated with employing this approach in a DoN context.

C. TRANSLATION OF USER NEEDS IN THE CONTEXT OF DON POLICY AND GUIDANCE

In the previous section, a high-level procedural basis for translating user needs into a complete set of system requirements was described. Requirements completeness across the DOTMLPF spectrum, analytical rigor, and effective communications between the user and the developer were emphasized. In this section, the JCIDS process and the DoN requirements definition process (Naval Systems Engineering Guide, 2004) are examined to assess the extent to which they facilitate translation of user need to system requirements in the manner described in the previous section.

A complete statement of user need should contain, or reference, the following:

1. Traceability of the user need to higher level requirements (e.g., CNO guidance) and Joint warfighting taxonomy (i.e. JOCs).
2. Definition of concept of operation for the system
3. Operational conditions for the system; environmental, threat, and other
4. Definition of the architecture within which the system must operate.

Functional Area Analysis (FAA), part of the JCIDS Capability-Based Assessment, is described in CJCSM 3170.01B (2007) in the following manner:

> … identifies the mission area or military problem to be assessed, the concepts to be examined, the timeframe in which the problem is being assessed, and the scope of the assessment. …The FAA describes the relevant objectives and CONOPs or concepts, and lists the relevant effects to be generated. Since a capability is the ability to generate an effect, the FAA connects capabilities to the defense strategy via objectives, concepts, and CONOPs.

The White Paper on CBA (2006) also notes the importance of scenario selection as part of the Capability Based Assessment (CBA) to provide a range of enemies, environments, and access challenges. Therefore, FAA addresses high-level requirements traceability, CONOPS and operational conditions, but not architecture.

Subsequently, Functional Needs Analysis (FNA) and Functional Solutions Analysis (FSA) assess current capabilities, determine whether gaps exist, and develop potential approaches to resolve identified capability gaps. The White Paper on CBA (2006) notes in reference to the entirety of JCIDS analysis:

> Architectures are useful (and probably essential) once you have decided what to do, as they provide a framework to help determine how to do it. JCIDS capability assessments, however, tend to be concerned more with what to do…

This statement is inconsistent with a statement made later in the same document: “Your statement of needs has to be tempered by rough feasibility, cost, and schedule estimates, and you have to have some idea of what the DoD

is willing to tolerate for additional investments in your areas." This statement indirectly suggests architectures are needed during the JCIDS process, as they provide constraints, conditions, interfaces and other information that are important in the development of feasible, potential approaches to address capability gaps.

Following the statement of user need, effective, iterative communication between the user and developer is needed to successfully evolve the user need into a compete set of system requirements. The next consideration then, is whether the JCIDS process facilitates this dialog, taking into account the volume and complexity of the information being exchanged and the differences between user and developer lexicons. JCIDS analyses are typically based on a number of different and changing requirements documents and analytical databases. While the prescribed structure of JCIDS-required analytical artifacts embodies some rigor in terminology, content, and underlying requirements, there is no data structure required that would maintain data relationships, data currency, or data source and credibility. Architectures based on a standard framework, e.g., DoDAF, could serve this function. DoD and DoN policy and guidance do not preclude it, but neither do they prescribe it.

The development of system requirements is part of the Navy Systems Engineering Guide (2006) "Requirements Definition Process" and its three sub-processes: "Acquirer Requirements," "Other Stakeholder Requirements," and "System Technical Requirements." The primary product of this portion of the systems engineering process is a System

Requirements Document (SRD). This process can also be indexed in a generic systems engineering sense to the “conceptual design phase” (Blanchard, 2006) whose output is a system specification or “A spec.”

Before examining the adequacy of DoD/DoN policy and guidance for development of system requirements, a brief discussion of the term “system requirements” is warranted. SECNAV Instruction 5002.D (2008) refers to a “System Design Specification” and describes it as flowing down from the Capability Development Document and providing basic functional requirements as well as major program requirements of the preferred solution alternative. SECNAVINST 5000.2D (2008) is DoN acquisition policy. The Navy Systems Engineering Guide (2006) uses the term “System Requirements Document,” noting that it evolves into a system specification. In the guide, the tasks necessary to produce a System Requirements Document and its content are described. The guide is based on EIA 632 (1999).

Other policy and guidance documents use various terms for specifications and requirements, some well defined, e.g., DoD-Std-961E, and others not. The IEEE Guide for Developing System Requirements Specifications (IEEE Std 1233, 1998) is noteworthy for several reasons. First, it provides definitions for terms which comprise the metadata for requirements and specifications. Many of those terms are linked to their own IEEE standards. Understanding and agreement between user and developer on the metadata and their definitions, is a prerequisite to users and developers being able to successfully communicate about a specific set of system requirements. Second, IEEE Std 1233

(1998) specifically recognizes the importance and challenge of communicating requirements between user and developer. Specifically:

> A System Requirements Specification (SRS) has traditionally been viewed as a document that communicates the requirements of the customer to the technical community who will specify and build the system. The collection of requirements that constitutes the specification and its representation acts as the bridge between the two groups and must be understandable by both the customer and the technical community. One of the most difficult tasks in the creation of a system is that of communicating to all of the subgroups within both groups, especially in one document. This type of communication generally requires different formalisms and languages.

Finally, while IEEE Std 1233 (1998) provides detailed guidance for developing system requirements specifications, it does not prescribe an industry-wide specification standard. It states: “This guide is written under the premise that the current state of the art of system development does not warrant or support a formal standards document.” Experience has shown and research supports the difficulty of creating specific templates for requirements and specification types. The recursive and iterative nature of the engineering development process leads to a continuum of specification and requirement types. Unfortunately this causes difficulty when trying to synchronize requirements, acquisition, and systems engineering processes.

Returning to the question of adequacy of policy and guidance for development of system requirements, (Naval Systems Engineering Guide, 2004) does provide the following details for each of the three previously discussed sub-

processes which comprise the Requirements Definition Process: Preceding Processes, Inputs, Entry Criteria, Tasks, Outputs, Exit Criteria, Next Processes, Agents, Tools, References, and Metrics and Measures. In many cases, details are found in other, cross-referenced (Naval Systems Engineering Guide, 2004) processes or appendices. However, as with JCIDS, there is no guidance for organizing or managing the metadata or data which comprise the requirements, use of tools or models, or translating between models and prose-based documents. IEEE Std 1233 (1998), while it does not address the use of architectures, provides substantial guidance on approaches to development and management of system requirements between user and developer communities. DoD and DoN guidance lack this detail.

D. CHAPTER SUMMARY

DoD and DoN requirements, acquisition, and systems engineering processes and associated guidance and standards comprise a substantial amount of interrelated data, even when limited to discussion of requirements development as in this study. Research performed for this study has revealed semantic inconsistencies and ambiguities as well as a multiplicity of system engineering processes, guidance and standards. It is difficult to trace data and process relationships in a systems engineering context. As a consequence it is difficult to rigorously and unambiguously identify a user need and translate it into a comprehensive and complete set of system requirements. Just the lack of specificity of roles and accountability for each of the

necessary activities in the requirements translation process represents a significant challenge for successful requirements development.

VI. APPLICATION TO NAVAL WARFARE SYSTEMS

A. INTRODUCTION

The premise for this study is the assertion that for Naval warfare systems, the process of translation of user needs to system requirements is not sufficiently rigorous and repeatable to ensure consistently complete and valid system requirements. This results in significant increases to program cost, schedule and risk as requirements issues are resolved later in development. Support for this assertion is found in a recent Government Accounting Office Report (GAO-08-782T, 2008):

> At the strategic level, DOD does not prioritize weapon system investments and the department's processes for matching warfighter needs with resources are fragmented and broken. Furthermore, the requirements and acquisition processes are not agile enough to support programs that can meet current operational requirements. At the program level, programs are started without knowing what resources will truly be needed and are managed with lower levels of product knowledge at critical junctures than expected under best practices standards. In the absence of such knowledge, managers rely heavily on assumptions about system requirements, technology, and design maturity, which are consistently too optimistic. This exposes programs to significant and unnecessary technology, design, and production risks, and ultimately damaging cost growth and schedule delays.

Previous chapters discussed DoN requirements and systems engineering processes, suggested system architectures are under-utilized as a means of facilitating and integrating those processes, and described conceptually

how and when architectures should be used and the benefits that would accrue from their use. This chapter provides a discussion of the use of architectures with consideration for typical constraints and characteristics of warfare systems employed by U.S. Navy surface combatants, e.g., cruisers and destroyers.

B. CHARACTERIZATION OF U.S. NAVY SURFACE WARFARE SYSTEMS

Current warfare system development efforts for surface combatants are predominantly modifications, i.e., evolutions of existing systems. Even the warfare systems being developed for the DDG 1000, Zumwalt class destroyer will be predominantly evolutionary. The following illustrates the degree to which the Navy will be using currently fielded systems for many years to come.

There are currently 62, DDG 51, Arleigh Burke class destroyers either in the Fleet, under construction, or under contract. The earliest any of those ships is planned for decommissioning is 2026. Of the 22 (of 27 constructed) Ticonderoga class cruisers remaining in commission, none is planned for decommissioning before 2026. Currently, only three Zumwalt class destroyers are planned. A follow-on cruiser requirement i.e., CGX, is underway but has not reached Milestone B, the point at which it becomes an acquisition program.

In addition to the length of time current ship classes will continue to form the backbone of the surface fleet, the time required to modernize a ship class is considerable. The time required for construction of a cruiser/destroyer and the annual production rate promotes significant configuration differences among ships from

oldest to newest. This adds further complexity to upgrade plans. Cruiser and destroyer modernization programs already underway, during which warfare systems will be upgraded, will span over 20 years from the first ship modernized to the last.

The Navy's current investment in surface combatant warfare systems, the cycle time for development and deployment, and the number of systems with which it must interoperate has created a type of system "inertia" and infrastructure that significantly constrains and adds complexity to warfare system development. Further constraints are applied by requirements for Joint and coalition-level interoperability. This stands in dramatic contrast to average, commercial product development and life cycle times.

The Apple iPod began development in February 2001, starting with a partially-developed design from another company, and was brought to market during the Christmas season the same year. Subsequently, new generations of iPods have been released almost annually.

Another study in contrast to commercial product development is cell phones. According to an Environmental Protection Agency brochure (2005), cell phones are used for only 18 months on average before being discarded. iPODs and cell phones are orders of magnitude simpler in design and integration relative to typical DoN warfare systems, but some of the commercial technologies they incorporate are representative of those sought after in DoN warfare systems.

In spite of the Navy's overall, long product development cycle and longer product life cycle, warfare systems employ significant amounts of commercially-based computer hardware and software that follow shorter, commercial product cycles. The Navy's need to take advantage of rapidly evolving computer technology within a more slowly evolving warfare system, within slower-still platform (i.e., surface combatant) development, adds further to product development complexity.

The dynamic nature of warfare capability requirements adds yet another dimension of complexity. Requirements are reviewed and changes are made at least every two years and often more frequently. The pattern of incremental improvements to capability over time together with periodic requirements changes form a metaphorical moving target.

C. TOP-DOWN REQUIREMENTS DEVELOPMENT

Integrated Air and Missile Defense (IAMD), an example of required U.S. Navy capability, exists in the Fleet today. The DoN Enterprise Architecture Hierarchy (2008) is consistent with and maps to the DoD Enterprise Architecture. The DoN IAMD capability component flows down as follows: Force Protection, to Sea Shield, to Joint Protection, to Protect Against Conventional Weapons, to Integrated Air and Missile Defense. IAMD for surface combatants is predominantly performed by radar (i.e., detect), a command and decision (C&D) system (i.e., control) and missiles (i.e., engage).

Establishment of a Fleet need for a new or improved combat C&D system can be initiated via dissemination of Fleets' prioritized operational needs and validation by the

JCIDS process. To validate the requirement, the operational need should be traced from high level documents starting with the President's National Security Strategy (2006) and flowing down to the National Defense Strategy (2008) which "informs" the National Military Strategy (2004). While these requirements documents are hierarchical, some others are complementary rather than hierarchical. The following paragraphs discuss some aspects of these key, policy documents.

Nine essential tasks comprise The National Security Strategy (2006). Some, such as "Champion Aspirations for Human Dignity" do not substantially involve military capability. Others such as "Work with Others to Defuse Regional Conflicts" clearly involve military capability. However, most should be expected to involve multi-dimensional solutions, involving military capability, diplomacy, and politics. Use of military force is typically, though not always, employed as a course of last resort when objectives cannot be achieved by non-military means. Therefore, while it is possible to map National Defense Strategy (2008) objectives (i.e., Defend the Homeland, Win the Long War, Promote Security, Deter Conflict, and Win our Nation's Wars) to the National Security Strategy, it is only abstractly possible to decompose the National Security Strategy into the National Defense Strategy. From a requirements analysis standpoint, it is more useful to begin requirements flow-down from the National Defense Strategy.

Examination of the National Defense Strategy (2008) does reveal a linkage to the National Security Strategy

(2006). But, it also illustrates that even within the Defense strategy, the types of capabilities and the levels of capabilities necessary to achieve objectives cannot be derived from those documents. In other words, they are functional statements of user need. From the National Defense Strategy: “We will work with and through like-minded states to help shrink the ungoverned areas of the world and thereby deny extremists and other hostile parties sanctuary.” Also: “…arguably the most important military component of the struggle against violent extremists is not the fighting we do ourselves, but how well we help prepare our partners to defend and govern themselves.”

The National Military Strategy (2004):

> …provides focus for military activities by defining a set of interrelated military objectives and joint operating concepts from which the Service Chiefs and combatant commanders identify desired capabilities and against which the Chairman of the Joint Chiefs of Staff assesses risk.

The National Military Strategy (2004) establishes three objectives that support the National Defense Strategy: Protect the United States, prevent conflict and surprise attack, and prevail against adversaries. Related Joint Operating Concepts (JOCs) describe how the Joint Force conducts missions and supports the Joint Functional Concepts of: Force application, protection, focused logistics, battlespace awareness, and command and control. Note that “Force Protection” is the top tier capability discussed in the requirements flow-down to IAMD.

Realization of IAMD capability requires a system of systems. It cannot deliver required warfighting capability

without its constituent systems. Furthermore, the system of systems exists operationally as parts of surface combatant platforms. When mapping between required capability and supporting systems, many-to-one and one-to-many relationships are revealed.

Consider the basis for initiation of a Fleet requirement for a new or upgraded combat C&D system. Capability shortfalls when mapped to functions can run the gamut of DOTMLPF. Obsolescence or reliability issues might exist; or, old or proprietary software may not be maintainable. Training may require high-cost, specialized equipment. Issues of this nature can be expressed in terms of either cost of ownership or system availability. If no performance improvements are warranted, the user requirement can be expressed in terms of reducing cost of ownership and increasing availability. This illustrates the point that user needs do not necessarily translate into requirements for system development, i.e., a materiel solution.

However, performance improvement may be required. For example processor speed or the computing architecture might not support the level of performance needed for the projected threat environment. Or, the current C&D, or combat system as a whole, may not be designed to counter an emergent threat, for instance small, high-speed craft, e.g., Rubber Inflatable Boats. Though improving processing speed might be one solution, improvements to detection or weapon systems as well as changes to doctrine might also

provide viable solutions. There is significant tradeoff analysis required to determine which requirements comprise the trade space.

A Fleet requirement as initially expressed may in some cases convey an implicit or explicit solution. In other cases, there may be only a generalized statement of operational need. In either case, for complex systems and systems-of-systems, use of some analytical methodology can help ensure:

- The system is properly defined (e.g., bounded)
- The trade space is properly defined
- Fleet requirements are posed in a “what” form versus a “how” form, i.e., the requirement should not contain the solution.
- All significant aspects of the system, i.e., DOTMLPF, its environment, and systems it interoperates with are considered.

A requirements model or combination of models should be used to support requirements development, and both direct and indirect requirements imposed by legacy systems must be captured in a comparable form.

Any U.S. Navy warfare system being considered for development or improvement must be integrated into existing architectures, whether or not those architectures are well documented. So, to some degree an architecture is imposed on a proposed system long before a solution, i.e., design, is conceived. Even for an unprecedented system on an unprecedented platform, the sailors who man the ship, environmental and navigations standards, the weapons, the communications networks, and other interoperating platforms

comprise an architecture into which the new system must fit. If the architecture is undocumented, it is incumbent upon the system developer to ensure accurate documentation is produced. If the architecture is documented, the adequacy must be assessed and any shortfalls addressed.

Different approaches exist to model requirements. In choosing an approach, two key questions are: Will the model(s) used result in a complete requirement's set? That is, does the model fully and accurately describe all behaviors of the system? These questions are particularly critical to DoN warfare systems considering the array of threats, operating environments, modes of operation, and the speed with which combat operations are conducted. Often, a ship's required response to a threat is measured in seconds.

Use case modeling is frequently used as the basis for architecture development, particularly for software-intensive systems, as exemplified by U.S. Navy warfare systems. On this subject, Daniels (2005) speaks to the first question above: Will the chosen approach capture all requirements? His assertion is that in spite of the popularity and utility of use cases as a basis for customer-developer communications and the resultant system requirements set, there are shortcomings including the fact that use cases do not contain all of the requirements. He notes: "To keep use cases simple, readable, and manageable, they can only tell a fraction of the complete story without becoming unwieldy and difficult to understand." He proposes combining a use case approach with a more traditional specification of "shall statements," a natural language-

based model. Shall statements can add detail to and complement use cases and are better suited for capturing non-functional requirements such as how well a system must do something versus simply what a system must do.

Hatley (1987) presents a system requirements development process that views a system from process and control aspects. The process component of the model shows what functions the system must perform. The control component describes the circumstances under which the function will be performed. The author suggests the use of “data flow diagrams” (DFDs) and “process specifications” (PSPECs) for modeling processing and “control flow diagrams” (CFDs) and “control specifications” (CSPECs) to model system control.

A simple example to illustrate the importance of modeling system control processes is the automated checkout stations in grocery and other retail stores. There are a number of such systems in use but they perform mostly the same functions. Data is taken from the customer (e.g., selection of language, telephone number or zip code, etc.), instruction is given to scan items, then the customer is prompted to select a method of payment and follow a series of steps to complete the transaction. The data flow is not particularly complex. However, the systems have subtle timing and sequencing controls. If an item is laid in the wrong area before scanning, the process will not proceed. If a scanned item sits too long without being moved to the bagging area, scanning cannot proceed. If one’s telephone

number (possibly a basis for product discounts) is not entered when prompted at the beginning of the process, that opportunity is lost.

Implicit in the design of the self-check-out system are expectations of customer behavior (e.g., how long will it take for a customer to bag an item) and constraints on customer behavior (e.g., certain timing or sequences of actions may be indicative of attempts to pilfer and are therefore not allowed). The success with which these requirements are modeled directly affects customer throughput, customer satisfaction, and likely the rate of pilferage, so it is clearly an important aspect of the system requirements. U.S. Navy warfare systems have extensive timing and sequence dependencies in order to ensure safe and effective operation.

D. REQUIREMENTS AND ARCHITECTURE DEVELOPMENT PROCESS EXAMPLE

The U.S. Navy's next generation cruiser is currently designated "CGX." Among its required capabilities is IAMD. A much-simplified decomposition of this capability is a sequence of three processes: detect, control, and engage. Incoming threats are sensed, a decision is made on how best to prosecute the target, and then weapons are utilized to defeat the threat. A similar process model is the "Observe, Orient, Detect, Act" or "OODA loop" model conceived by USAF Colonel John Boyd (Fein, 2003) to describe the air-to-air combat process. These functional models describe what a system has to do. A complete specification of user need would include what the system must do, e.g., detect-

control-engage, how well it must perform e.g., Probability of Raid Annihilation, and what constraints are imposed, e.g., open architecture design.

This example facilitates discussion of key aspects of requirements and architecture development. The objectives of the example are:

- Show how requirements can be captured and organized with a structured process that facilitates completeness as well as understanding by both user and developer
- Show how requirements and architecture models facilitate integration of a developmental system's requirements with those of legacy systems with which it interoperates.

The approach used combines aspects of a hybrid approach described in Daniels (2005) with the Process for System Architecture and Requirements Engineering (PSARE), (Hatley, 2000). It is primarily the PSARE process, but plain language use cases are employed as the means of initial requirements collection due to their ability to facilitate effective communication between user and developer. Daniel's suggestion to incorporate plain language requirements with linkages to requirements models is also adopted.

1. Process for System Architecture and Requirements Engineering (PSARE)

Hatley (1987) describes a formal process for development of system specifications that utilizes requirements and architecture models that are consistent with one another. Though Hatley only refers to one of the

models as an architecture model, the two models can be considered as two views of a system architecture. The requirements model represents the operational view and the architecture model represents the systems view. The PSARE is applicable to both hardware and software, is specifically applicable to real-time systems, and can facilitate development of DoN-mandated DoDAF products. Table 1 summarizes features and benefits.

Features	Benefits
Hierarchical model	• Specifies requirements at appropriate level • Depicts manageable amount of information at one time
Graphical and text representation of functionality	• Clearly shows interfaces (functional and physical) • Graphics depict abstract aspects of system • Text defines details
Allocation of functions to physical entities	• Greatly improved interface consistency
Rigorous method	• Promotes thorough design • Identifies gaps early

Table 1 PSARE Features and Benefits, From Haggerty, 2008. This table captures attributes of PSARE that make it widely applicable, objective, and comprehensive in a modeling sense. It is not however, necessarily easy to implement.

The methodology of PSARE is not the only viable approach for DoN warfare systems. However, Hatleys (2000) *process*-oriented approach as well as PSARE's real-time system application heritage make it more straightforward to apply and more suitable than some others.

Requirements and architecture models comprise the PSARE products. The requirements model captures functional and non-functional requirements as well as system control behavior. The architecture model describes how the system

will fulfill the requirements. A requirements-to-architecture template provides a structure for allocation of requirements to architecture modules. Though the architecture model accommodates description of a system solution which goes beyond the scope of this study, the extensive presence of legacy systems even in new, U.S. Navy system development makes it germane to the requirements development discussion. The PSARE is non-sequential, meaning it does not have to move from requirements to architectures. It accommodates pre-existing portions of an architecture and allows for derivation of requirements from architectures when expedient. Figure 8 summarizes the interaction among models. The *Enhanced Requirements Model* depicted in the lower right-hand corner accounts for technological and other constraints imposed on the system.

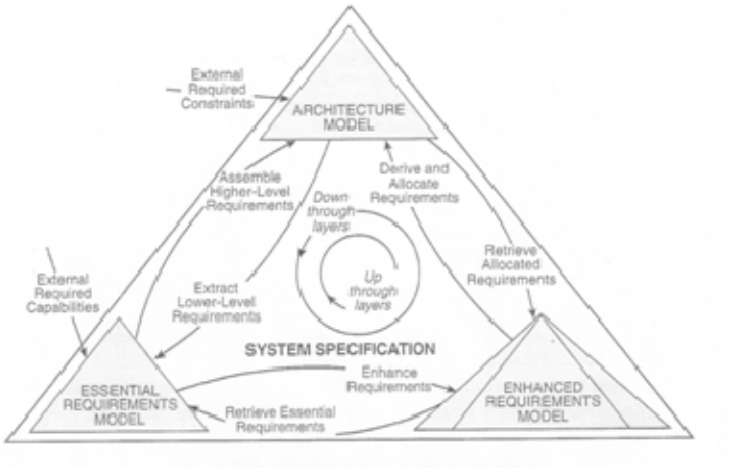

Figure 8 System Specification Models, From Hatley, 2000.

The recursive approach to development of the models which comprise PSARE are illustrated by the curves arrows to and from the models in each of the three corners. The spiral in the center indicates hierarchical layers of the models.

The requirements model, with enhancements, is developed through a series of process steps whereby:

- Customer requirements are organized into functional groups
- External systems with which the system must communicate are identified
- Information flowing between the system and external entities is identified
- A top-level flow diagram, showing the functional groups as processes, the external entities as terminator, and information groups as data flows is drawn
- A context diagram is derived from the top-level flow diagram
- A level 1 diagram is constructed from the top-level flow diagram without terminators
- Control signals are added as required
- Each process in the level 1 diagram is decomposed into a child diagram.
- Performance Specifications (PSPECS) and Control Specifications (CSPECS) are created at the last level of decomposition.
- A requirements dictionary catalogs data flows and control flows (i.e., requirements)

The architecture model is developed using the architecture template to enhance the requirements model and then allocating the requirements to physical entities (Hatley, 1987).

The example follows these steps with some changes to accommodate Daniel’s (2005) hybrid process. Use cases, if developed at the initial step in the requirements model development, help communication between user and developer and help ensure an accounting of the full range of required behavior of the system. The use cases also help in

development of subsequent requirements model diagrams and serve as a cross-checking mechanism for diagrams' correctness.

2. Problem Statement

Figure 9 shows a notional, joint, operational view of IAMD including the integration of weapons, sensors and command and control systems necessary to execute a detect-control-engage process for IAMD.

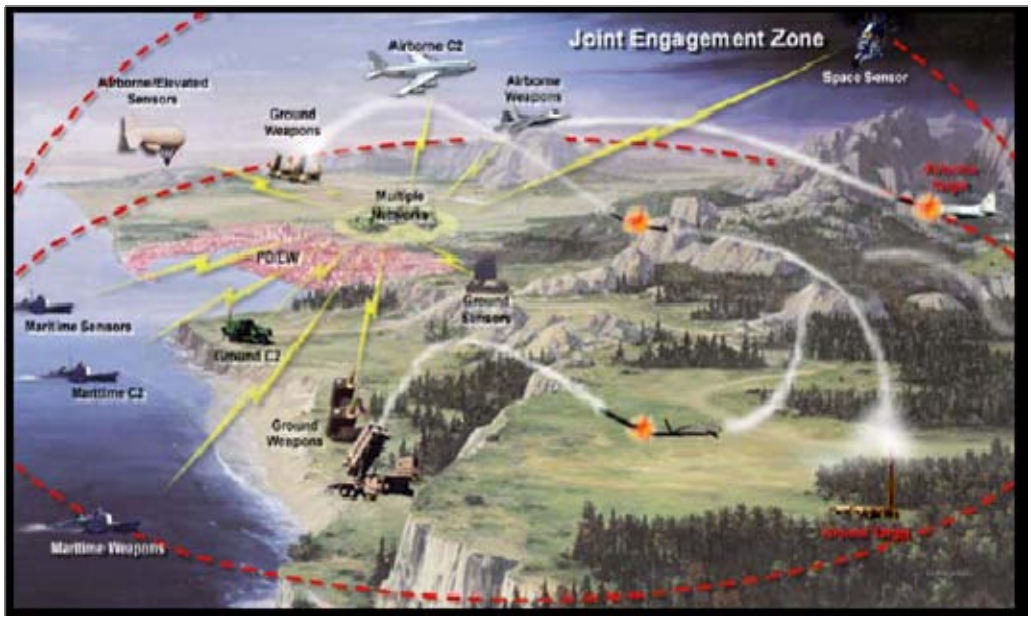

Figure 9 Joint IAMD Operational View, From Baldwin, 2006. This view, though it depicts systems, is not solution specific. The spacecraft, ships, aircraft, and ground-based systems should be viewed as operational nodes where activity occurs.

To provide IAMD, the cruiser must detect and track multiple targets, both aircraft and missiles, prioritize the threats, assign weapons or countermeasures based on

threat characteristics and kinematics, and prosecute the threats. This follows the aforementioned detect-control-engage functional construct.

3. Requirements Model

The requirements model can begin with construction of use cases. Use cases, as defined in Cockburn (2001) "…describes the system's behavior under various conditions as the system responds to a request from one of the stakeholders…". Although charts, diagrams and automated tools can be used in the construction of use cases, Cockburn suggests a fundamentally text form helps communication between user and developer, or stakeholder and developer, without imposing any special training requirements to be able to understand uses cases.

Development of use cases requires involvement or representation of most or all of the stakeholders, to the extent different stakeholders have different goals. For example, the goal of a stakeholder who is a system maintainer would be interested in system attributes that support a low mean time to repair, and probably also infrequent scheduled maintenance. The goal of a system manufacturing stakeholder might be concerned with system attributes that ensure high production rates, avoidance of specialized manufacturing labor, use of proven technologies, etc.

The process can begin with a general description of a scenario where all goals are satisfied. Cockburn (2001) defines this as the *main success scenario*. He views all other ways to succeed, or fail, as extensions of the main success scenario. The scenario is written as a series of

actions or activities required to achieve the goal. It is not the intent of this study to present a detailed description of the use case development process. Rather, the objective is to illustrate the advantage of incorporating use cases into the PSARE. Use case development can add an intuitive structure to the process of identifying the requirements model elements, their relationships, and what purpose they really serve, as well as initiating a set of plain language requirements to complement graphical and tabular models.

A Main Success Scenario for IAMD could be written as follows:

- The operator engages the ship’s sensors to scan a 3D volume for hostile aircraft and missiles.
- The search portion of the detect function provides radar and IFF data to the control function.
- Off-board target track data is received by the control function.
- The control function initiates and maintains target tracks.
- The control function determines if targets are engageable.
- The control function assigns weapons to engageable, hostile targets.
- The engage function prosecutes hostile targets with weapons.

This sequence, when reviewed by stakeholders would likely be revised, as each subject matter expert brings a different perspective and set of experiences. It is also evident that most of the steps can be decomposed. There is a great deal of complexity underlying *the control function*

initiates and maintains target tracks. The decomposition of use cases parallels decomposition as part of PSARE. To accommodate particular threat targets that cause the system to behave differently, or to accommodate different environments or states of readiness, extensions can be appended onto the main success scenario at branch points.

Having begun with development of a use case, or set of use cases, the construction of the requirements model then proceeds as described in Hatley (1987). The details of model development occupy a significant portion of that book and will not be reiterated here. Instead, examples of the graphical and tabular model elements are shown and comments are provided to explain the progression and relationship of model elements.

Building an *environment* model requires representation of both the system processes and those processes outside the system, i.e., the environment. The environment comprises those entities beyond the boundary of the IAMD system that play a role in IAMD and exchange information, energy or material. The detect function generates sensor data from both hostile targets and friendly forces. The control function receives track data from remote sources, i.e., other platforms. The control function also receives data from and contributes to onboard C4I (Command, Control, Communications, Computers, and Intelligence). The engage function passes data to and from the control function and also interacts with the target in terms of target signature and weapon homing sensors. This model is shown graphically in Figure 10.

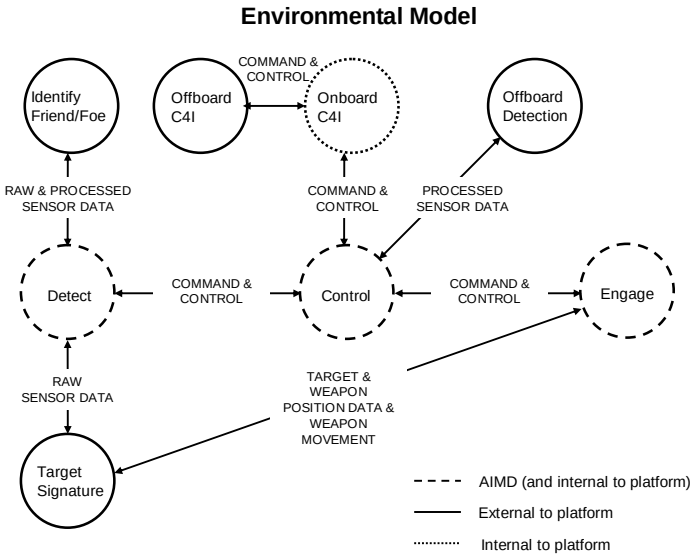

Figure 10 IAMD Environment Model.

The environment model includes all functions affecting system behavior and is the first step in identifying system boundaries.

The exchanges of information, energy or material among processes can be categorized as data or control exchanges. Continuous signals, such as target signatures, are data exchanges. Discrete signals such as commanding a weapon launch are usually control exchanges. In Figure 10, data from the control process is, logically, predominantly control-oriented. Sensor data is typically data-oriented, and engage data is a mixture of data and control exchanges. Figure 9 is a *Level 1 diagram*. Its component processes and data flows can be decomposed as necessary to achieve sufficient requirements detail.

From the environment model construction of a *context-level model* Hatley (2000) is begun by bounding those portions of the environmental model that are included in

the system. This is shown in Figure 11. The bounded portion comprises the beginning of a system specification.

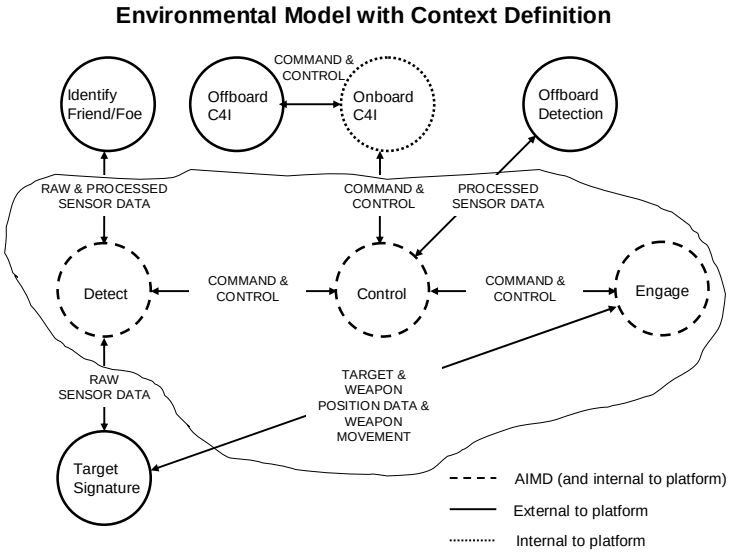

Figure 11 Environment Model with System Boundary Applied. The closed curve represents decisions regarding what to include in the system under study.

The *context process* is derived by collapsing all processes inside the bounded area into one process. Processes outside the boundary are replaced by *terminators*, the physical agents that perform the processes. Making these changes, a requirements context diagram for IAMD can be constructed. In Figure 12, the detect-control-engage processes have been collapsed into a single IAMD process and terminators have replaced external processes such as *offboard detection*.

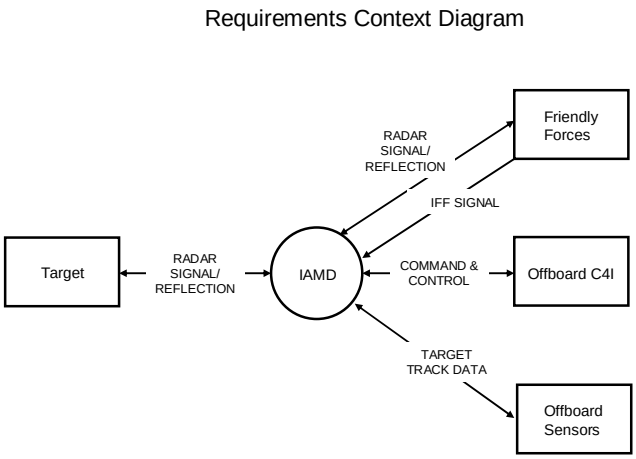

Figure 12 IAMD Requirements Context Diagram.

The context diagram aggregates all of the activities inside the system boundary into one activity, “IAMD.” Activities outside the system boundary are replaced with the actors that perform the activities.

Next, the system’s functionality is reviewed to determine if any data/material/energy flows did not get incorporated into the environment model. The resultant Data Flow Diagram (DFD) is shown in Figure 13. Earlier, control and data processes and flows were differentiated and it was stated that the IAMD process exhibited both. To simplify this example, all processes and flows are shown as data. In practice, Control Flow Diagrams (CFDs) would be constructed in addition to the DFDs. Each would depict the same processes, but one would show control flows and the other data flows.

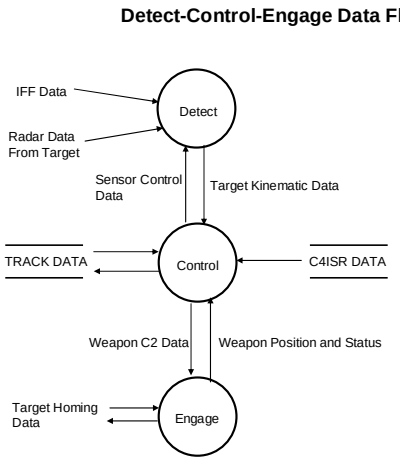

Figure 13 IAMD Data Flow Diagram.

This diagram shows processes in circles, data stores as labels with lines above and below, and data flows as arrows between processes and stores.

The last level of decomposition of DFDs is *Process Specifications* or *PSPECs*. According to Hatley (1987) "The primary role of the process specification…is *to describe how its inputs and outputs are generated from its inputs; it must do nothing more and nothing less*." PSPECs can contain textual, tabular, graphical and mathematical elements. A notionalized example is shown below, based on a decomposed element of the detect process.

PSPEC1: Detect Filter
Radar returns (i.e., signatures) of airborne objects are input to the IAMD sensor. The signal strength relative to the calculated object's range is assessed. If the signal strength correlates to aircraft or missiles, the object is reported as a detected object.

Table 2 IAMD Process Specifications.

This is a process description at the lowest level of decomposition.

If it is determined that the system requirement model must include control flow diagrams, then Control Specifications, i.e., CSPECs, also need to be generated. An example CSPEC is not presented but in one form can be visualized as a decision table or Boolean equation.

A requirements dictionary, typically instantiated in a database, contains definitions of all the elements used in the models. Per Hatley (2000), “Every data flow, control flow, and store used anywhere in the DFDs, CFDs, and CSPECs must be defined in the dictionary.” Table 3 provides some notional definitions as well as metadata, i.e., the column headings.

Name	Meaning & Composition	Type	Units	Rate
IFF Data	Friendly A/C ID and position	Data	Aircraft ID digital msg	Once per second
Radar Data From Target	Range, bearing, elevation	Data	Feet, polar coordinates	Once every 4 seconds
Sensor Control Data	Wave form commands, cuing, power level commands, etc.	Control		Once per second
Target Homing Data	Target Position, Target signature	Data	Feet, seconds, polar coordinates	10 times per second
Target Kinematic Data	Position, velocity, acceleration	Data	Feet, seconds, polar coordinates	Once every 4 seconds

Table 3 IAMD Data Requirements Dictionary Content. This is a list of data flow definitions.

4. Architecture Model

The architecture model represents a system's design, which is beyond the scope of this study. However, in most instances of DoN warfare system development, at least portions of an architecture exist at the outset of system development. They can be interfacing systems or subsystems, standards, which create constraints on the system requirement, or they can take the form of direct constraints such as size, weight, power consumption, safety, reliability, etc. System requirements can be derived from these architectural elements and incorporated into the requirements model. Therefore, construction of an architecture model supports development of a comprehensive requirements model and system specification.

Hatley (1987) describes a *Requirements-to-Architecture Template* that places the requirements model in a context of physical modules as shown in Figure 14. The requirements model resides in the portion entitled *Main Processing (Core Functions)*. The development of architecture model elements involves creation of *architecture context diagrams*, *architecture flow diagrams*, *architecture interconnect diagrams*, an *architecture dictionary*, and *module specifications*. Their forms are analogous to elements of the requirements model, but from a physical rather than operational perspective. The PSARE process maintains consistency and traceability between requirements and architecture models.

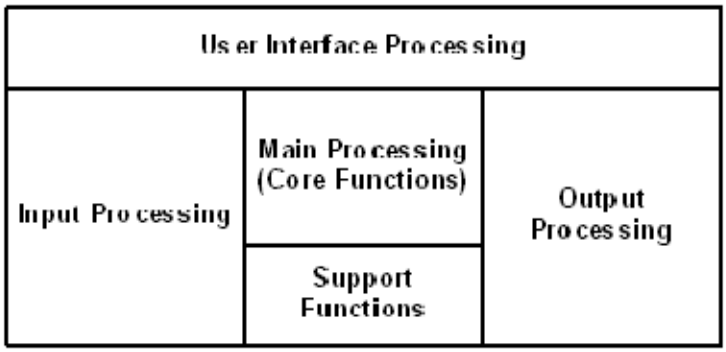

Figure 14 PSARE Architecture Template, from Hatley, 1987.

"Main processing" represents the physical system under study. The surrounding blocks represent physical modules that interact with the system under study.

Considering the IABM example, system development is certain to involve already-existing sensors, software, processors, weapons, or services infrastructure. The Navy's objectives of achieving open system design, use of Commercial, Off-The-Shelf hardware, and maximization of hardware and software re-use among systems imposes a large set of standards. Therefore, in beginning an IABM system development, major portions of the system specification can be derived from the architecture model.

E. CHAPTER SUMMARY

This chapter presents the fundamental characteristics of U.S. Navy warfare systems that affect the systems engineering and architecture processes. It further shows, utilizing a blended use case and PSARE process, that requirements and architectures models can be constructed in an integrated, structured, repeatable manner that when complete, provide the basis for a complete set of system requirements. Furthermore, the requirements model is in fact an operational view of the system's architecture and

the architecture model represents the system view of the architecture, enabling generation of required DoDAF products, but having been created by and for the systems engineering process. It is done in a way that provides a full accounting of requirements in a design.

This type of process is not provided as part of the DoD Architecture Framework (DoDAF, 2007) or the Naval Systems Engineering Guide (2004).

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS

A. KEY POINTS AND RECOMMENDATIONS

A substantial body of work among Government and industry exists regarding system engineering standards and processes. Standards and practices have been evolved, refined, interpreted, and exercised over a period of approximately 50 years. The value of systems engineering has been shown to lie in its ability to manage complexity in system development such that systems produced will predictably satisfy user needs. Yet, numerous examples exist of warfare systems that exceed schedule and cost requirements and do not meet operational requirements. Shortcomings appear to exist in the application and management of systems engineering principles.

Development of architectures to support DoD system development has a significantly shorter history than systems engineering, dating back only to the early 1990's, and the initial objective was solely to improve interoperability of C4I systems. The more broadly scoped DoD Architecture Framework Version 1.0 was approved in 2003, less than six years ago.

The JCIDS process, also less than six years old, is ostensibly dependent on architectures as an analytical basis, but has not resulted in desired levels of improvement in terms of ensuring Joint solutions, supporting capability acquisition, and reducing redundant or excess capability, while reliability identifying capability gaps.

Though efforts are underway to establish and maintain a DoN Enterprise architecture and to capture system-of-system level architectures, to date most architectures are developed independently at the system level. Governance, i.e., approval and configuration control, is weak beyond the system level. Responsibility for operational architecture and system architecture development is split, and funding for architecture development is fragmented and inconsistent from year to year. The consequence of all this is a lack of higher-level architectures to support JCIDS analysis. This problem is compounded by JCIDS' dependency on the Programming, Planning, Budgeting, and Execution (PPBE) process, which follows a biannual cycle. The JCIDS process cannot alter its rhythm to "wait" for architecture products.

In spite of mandated systems engineering and architecture standards, the relationship between system architecting and system engineering is poorly defined in DoD policy and instructions, and processes for development of architecture models as part of a systems engineering process are not prescribed. System engineering and architecture artifacts required by the DoN acquisition process do not promote or ensure integration of systems engineering with system architecting. The process and purpose of system architecting is poorly understood by system acquisition managers and decision-makers. And, as in the case of the JCIDS process, higher level, i.e., above product level, architecture development often does not keep pace with the acquisition process.

Translation of operational requirements to a set of system requirements is a vitally important part of the systems engineering process. Yet, ambiguity exists in definition of roles across DoN organizations, in terms and definitions, and in the necessary order and timing of events in this process. As a result, demonstration of system requirements completeness and traceability of system requirements to operational requirements is not consistently practiced.

Most significant among the causes for process disconnects are management-related and organizational. To make policy and resource decisions necessary for better process integration requires better understanding of system engineering and architectures than currently exists at the management level. Once policy and resource positions are developed, a DoN organization whose size and fragmentation by product line, near-term or long-term vision, and operational versus acquisition communities makes consensus difficult to achieve. Maintaining a Joint perspective adds yet another dimension of complexity.

Technical reasons are among the lesser for poor integration and execution of systems engineering, JCIDS, and architecture processes. Rigorous methods exist to model system requirements and architectures in a manner that supports mandated DoD Architecture Framework products while maintaining close-coupling with the systems engineering process. These methods can capture the behavior of complex, Naval warfare systems, maintain traceability to higher level requirements, and incorporate plain language views of

requirements. The community of skilled practitioners may not be large enough to rapidly increase the use of these methods.

There are several initiatives underway that could improve both integration and effectiveness of systems engineering, architecture, and JCIDS processes. Systems engineering both at the DoD and DoN level has received increased attention in policy and guidance over the last several years. The Defense Acquisition Guidebook (2004) devotes 98 of 520 pages to systems engineering. Most other Service and industry standards and guidebooks are less than five years old. The DoN recently adopted a common systems engineering technical review guide for all acquisition programs. Systems Engineering Plans are being given more attention.

In Joint Staff Instruction CJCSI 3170.01F (2007) which documents JCIDS policy, the *Summary of Changes* section notes that changes from the last version reflect "…lessons learned and changes as a result of implementation of the JCIDS process." Both the instruction and related manual (CJCSM 3170.01C, 2007) have been updated frequently in an effort to improve effectiveness of the process.

DoD Architecture Framework 2.0, currently in draft, was begun before issuance of the current, DoD Architecture Framework Version 1.5 (2007). Committees working on those documents recognized areas in need of change, but the large community of stakeholders and the large investment in the current framework slows the process of revision.

As the previous three paragraphs illustrate, significant effort is being expended updating and improving

policy and guidance for systems engineering, JCIDS, and architectures. And, effective systems engineering methods have been demonstrated on significantly complex systems. In light of remaining acquisition issues at least partially traceable to systems engineering failures, there must still be hindrances to improved effectiveness of the abovementioned processes.

The systems engineering competency needs to be strengthened. A recent collaborative initiative between the ASN(RD&A) and the Naval Postgraduate School will significantly increase opportunities for DoN engineers to pursue Master's degrees in systems engineering. However, competence in systems engineering, perhaps more than some other engineering disciplines, requires a wealth of experience typically acquired over many years. DoN engineers need experience as systems engineering practitioners in addition to education.

The increased emphasis on Joint, network-centric warfare results in systems-of-systems that cross boundaries of Services, organizations within Services, funding, requirements, political, and more. This type of construct hampers the effective integration and implementation of systems engineering, JCIDS, and architecture processes simply by making decision-making among a diverse community of stakeholders, considerably more difficult. A solution to this problem must address, among other things, the political process of funding appropriation and an ever-increasing desire by the Congress to perform DoD management functions. To the extent a way can be found to parse

complex systems-of-systems for system development and still maintain requirements and funding integrity, this problem can be mitigated.

Overall, a plan to improve integration and effectiveness of systems engineering, requirements, and architecture processes must be treated holistically and with a long-term vision. There is an enormous investment in fielded systems as well as those currently under development. Future systems cannot completely escape the effects of those systems' paradigms. Progress will be evolutionary, not revolutionary.

B. AREAS FOR FURTHER RESEARCH

The DoD Architecture Framework is currently under revision. Once issued, analysis of the revised framework for improvements in the areas of integration with systems engineering, and inclusion of architecture development process guidance would be useful.

System architecting for highly complex systems is not inexpensive. Justification of funding for system architecting is made more difficult by a cost-benefit relationship that is difficult to quantify and validate. Exploration of cost justification methods for system architecting would contribute to a more rational basis for investing in this area.

The analytical basis for system architecting is challenging for practitioners to master and even more challenging for practitioners to discuss with managers and customers. This is true to the extent that just the inclusion of the term "architecture" in a management-level

brief is often a carefully considered decision. Yet, a means of successfully communicating the concepts of system architecting to varied audiences is critical to maintaining support for this type of work. An area for further research could be exploration of alternative means to discuss system architecting among less technical audiences.

The architecture modeling example presented in Chapter VI was simplified by orders of magnitude relative to actually modeling a warfare system with enough fidelity to support system development. Research for this paper did not reveal significant information pertaining to estimation of time or resources to perform architecture models. Some historical data for architecture product development has been collected among U.S. Navy Systems commands.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Assistant Secretary of the Navy for Research, Development, & Acquisition, Chief Systems Engineer. (2006). *Naval "Systems of Systems" Systems Engineering Guidebook* Office of the Secretary of the Navy.

Bahill, A. T., & Henderson, S. J. (2005). *Requirements development, verification, and validation exhibited in famous failures. Systems Engineering, 8*(1), 1-14.

Bahill, T., & Dean, F. (1997). *The requirements discovery process*. Proceedings of the Seventh Annual International Council on Systems Engineering, Los Angeles, CA, USA.

Blanchard, B., & Fabrycky, W. (2006). *Systems engineering and analysis* (4th ed.). Upper Saddle River, NJ, USA: Pearson Prentice Hall.

C4 Architecture & Integration Division, J-6, The Joint Staff. (1992). *C4I for the warrior*. Washington, D.C.

Carson, R., Aslaksen, E., Caple, G., Davies, P., Griego, R., Kohl, R., et al. (2004). Requirements completeness. *Proceedings of the Fourteenth Annual International Symposium of the International Council on Systems Engineering,* Toulouse, France.

Chairman of the Joint Chiefs of Staff. (2007). *CJCS Instruction 3170.01F: Joint Capabilities Integration and Development System*.

Chairman of the Joint Chiefs of Staff. (2007). *CJCS Manual 3170.01C: Operation of the Joint Capabilities Integration and Development System*.

Chairman of the Joint Chiefs of Staff. (2006). CJCS Instruction 3010.02B: *Joint operations concepts development process (JOPSC-DP)*.

Cockburn, A. (2001). *Writing effective use cases*. New York: Addison-Wesley.

Cole, R. (2006). The changing role of requirements and architecture in systems engineering. *System of Systems Engineering, 2006 IEEE/SMC International Conference on,* Los Angeles, CA, USA. 5 pp.

Collins, M. (2008). Enabling systems engineering with an integrated approach to knowledge discovery and architecture framework. 1-2-20.

Daniels, J., Botta, R., & Bahill, T. (2005). A hybrid requirements capture process. *Proceedings of the Fifteenth Annual International Symposium of the International Council on Systems Engineering,* Rochester, New York, USA. 1-14.

Daniels, J., & Bahill, T. (2004). The hybrid process that combines traditional requirements and use cases. *Systems Engineering, 7*(4), 303-319.

Defense Acquisition University. (2001). *Systems engineering fundamentals*.

Defense Information Systems Agency. (1994). *Department of Defense technical architecture framework for information management, volume 7*).

Department of Defense. (2004). *Defense acquisition guidebook*.

Department of Defense. (2003). *DoD architecture framework, version 1.0*.

Department of Defense. (2007). *DoD architecture framework, version 1.5*.

Department of Defense. (2004). *DoD directive 4630.5, Interoperability and supportability of Information Technology (IT) and National Security Systems (NSS)*.

Department of Defense. (2003). *DoD directive 5000.1, The Defense Acquisition System*.

Department of Defense. (2004). *DoD instruction 4630.8, Procedures for Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS)*.

Department of Defense. (2008). *DoD instruction 5000.2, Operation of the Defense Acquisition System*.

Department of Defense. (2008). *National defense strategy*.

Joint Chiefs of Staff. (2004). *The national military strategy of the United States of America*.

Joint Chiefs of Staff, J-8, Force Application Assessment Division. (2006). *White paper on conducting a capabilities-based assessment (CBA) under the Joint Capabilities Integration and Development System (JCIDS)*.

Department of the Navy.(2004). *Naval systems engineering guide*.

Department of the Navy, Assistant Secretary of the Navy, RDA, Chief Systems Engineer. (2008). *Department of the Navy enterprise architecture hierarchy version 0.2 (working draft)*.

Environmental Protection Agency. (2004). *The life of a cell phone*.

Fein, G. S. New meaning for 'OODA' loop. *National Defense Magazine, 2003*(October), 12/30/08.

Forsberg, K., & Mooz, H. (1991). The relationship of systems engineering to the project cycle. *Proceedings of the First Annual Conference on NCOSE,* Los Angeles, CA.

GEIA EIA 632 - processes for Engineering a System(1999). Government Electronics Information Technology Association.

Gonzales, D., Landree, E., Hollywood, J., Berner, S., & Wong, C. (2007). *Navy/OSD collaborative review of acquisition policy for DoD C3I and weapons programs*. Santa Monica, CA: Rand National Defense Research Institute.

Government Accountability Office. (2008). *Defense acquisitions; DoD's requirements determination process has not been effective in prioritizing joint capabilities.* GAO-08-1060.

Government Accountability Office. (1998). *Defense information superiority; progress made, but significant challenges remain.* GAO/NSIAD/AIMD-98-257.

Haggerty, K., & Haggerty, L. J. *Introduction to structured methods*. El Segundo, CA: H&A System Engineering. Retrieved December, 2008 from www.hasys.com

Hanley, J. T., Fitzsimmons, M. F., Kurtz, J. H., Roark, L. M., Roske, V. P., & Cuda, D. L. (2006). *Improving integration of department of defense processes for capabilities development planning* No. IDA Paper P-4154). Alexandria, VA: Institute for Defense Analysis.

Hatley, D., Hruschka, P., & Pirbhai, I. (1987). *Process for system architecture and requirements engineering*

Hatley, D., Hruschka, P., & Pirbhai, I. (2000). *Process for system architecture and requirements engineering* (First ed.). New York: Dorset House.

IEEE std 1233, 1998 edition, IEEE guide for developing system requirements specifications (1998). New York, NY, USA: IEEE.

ISO/IEC 15288 (IEEE std 15288-2008), systems and software engineering - system life cycle processes.(2008). (Second Edition, 2008-02-01 ed.). New York, NY, USA: IEEE.

International Council on Systems Engineering (INCOSE). (2007). *Systems engineering handbook, a guide for system life cycle processes and activities* (Version 3.1 ed.).

ISO/IEC 26702 (IEEE std 1220-2005), systems engineering - application and management of the systems engineering process (2007). (First edition 2007-07-15 ed.). New York, NY: IEEE.

ISO/IEC 42010 (IEEE std 1471-2000), systems and software engineering - recommended practice for architectural description of software-intensive systems(2007). (First Edition, 2007-07-15 ed.). New York, NY, USA: IEEE.

Langford, G. (2006). *Systems engineering for product development (SI4021) course notes*. Monterey, CA: Naval Postgraduate School.

Maier, M., Emery, D., & Hilliard, R. (2004). ANSI/IEEE 1471 and systems engineering. *Systems Engineering Journal, 7*(3), 257-270.

Maier, M., & Rechtin, E. (2002). *The art of systems architecting* (2nd ed.) CRC Press.

Mar, B. W. (1994). Requirements for development of software requirements. *Proceedings of INCOSE 1994,* San Jose, CA.

McGroddy, J. C. (2004). *Letter report to the FBI*. Washington, D.C.: National Academies Press.

National Aeronautics and Space Administration. (2007). *NASA systems engineering handbook*. (Revision 1 ed.).

Office of the Assistant Secretary of Defense, Networks & Information Integration (OASD, NII). (2005). Architecture development and analysis survey; the state of DoD architecting. *Command Information Superiority Architectures (CISA) Worldwide Conference,* Omaha, NE. 1-2-44.

Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering Enterprise Development. (2008). *Systems engineering plan preparation guide* (Version 2.01 ed.)

Office of Secretary of the Navy. (2008). *SECNAV Instruction 5000.2D: Implementation and operation of the Defense Acquisition System and the Joint Capabilities Integration and Development System.*

Osvalds, G. (2006). Use of architecture for engineering systems; the good, the bad, and the ugly. *Proceedings of the 16th Annual International Symposium of the Infternational Council on Systems Engineering,* Orlando, Florida, USA. 1-9.

Rhodes, D., Hastings, D., Richards, M., & Shah, N. (2007). Architecture frameworks in system design: Motivation, theory, and implementation. *Proceedings of the 17th Annual International Symposium of the International Council on Systems Engineering,* San Diego, California, USA. 1-10.

Richards, M. G., Shah, N. B., Hastings, D. E., & Rhodes, D. H. (2007). *Managing complexity with the department of defense architecture framework: Development of a dynamic system architecture model*. Cambridge, Massachusetts: Massachusetts Institute of Technology Engineering Systems Division.

Riverside Publishing Company. (1984). *Webster's II new riverside dictionary.* New York: Berkley Books.

Roughead, G. (2008). *CNO guidance for 2009; executing our maritime strategy*

Schindel, W. (2005). Requirements statements are transfer functions: An insight from model-based systems engineering. *Proceedings of the Fifteenth Annual International Symposium of the International Council on Systems Engineering,* Rochester, New York, USA. 1-15.

Sheard, S. A. The frameworks quagmire. *Crosstalk, the Journal of Defense Software Engineering, 1997* (September). Retrieved 12/29/08 from http://www.stsc.hill.af.mil/crosstalk/1997/09

Siegers, R. (2005). The ABCs of AFs: Understanding architecture frameworks. *Proceedings of the Fifteenth Annual International Symposium of the International Council on Systems Engineering,* Rochester, New York, USA. 1-13.

Space & Missile Systems Center, U.S. Air Force. (2005). *SMC systems engineering primer & handbook; concepts, processes, and techniques*.

U.S. Senate. (2008). Defense acquisitions; better weapon program outcomes require discipline, accountability, and fundamental changes in the acquisition environment.

Wilcynski, B. (2007). DoDAF v2.0. McLean, VA.

The White House. (2006). The national security strategy of the United States of America.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Mr. Carl Siel
   ASN(RD&A) Chief Systems Engineer
   Washington, D.C.

4. Ms. Patricia Hamburger
   Director, SEA 05H, HSI and Warfare Systems Engineering
   Naval Sea Systems Command
   Washington, D.C.