



**NAVAL
POSTGRADUATE
SCHOOL**

MONTEREY, CALIFORNIA

THESIS

**ANALYSIS OF MISSION EFFECTIVENESS:
MODERN SYSTEM ARCHITECTURE TOOLS
FOR PROJECT DEVELOPERS**

by

Thomas E. Moulds

December 2017

Thesis Advisor:
Co-Advisor:

Warren K Vaneman
Kristin M Giammarco

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE December 2017	3. REPORT TYPE AND DATES COVERED Master's thesis		
4. TITLE AND SUBTITLE ANALYSIS OF MISSION EFFECTIVENESS: MODERN SYSTEM ARCHITECTURE TOOLS FOR PROJECT DEVELOPERS			5. FUNDING NUMBERS	
6. AUTHOR(S) Thomas E. Moulds				
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. IRB number ___N/A___.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) Fail early, fail often, but ensure that when failure occurs, a learning period is part of the systems development process. Understanding the reasons a system can fail during the development process is key to maximizing mission effectiveness. Would it not be valuable to have a process that allows the designers to recognize when a system is failing to meet the user's requirements early in the development process? Furthermore, would it not be useful for that process to be iterative, to allow the impacts of changes to be seen in real-time, as the concept is defined and the system is designed? What would it be worth to have the ability to accomplish this inside the engineering safety net of Model-Based Systems Engineering? This research shows an alternative process to classic systems engineering and optimization analysis, where system design decisions are statically and dynamically modeled in a Model-Based Systems Engineering environment and "what if" types of changes are answered and analyzed using embedded simulation. This research demonstrates the process with the use case of a highly relevant real world problem of countering the threat of small commercial unmanned systems to the security of naval installations.				
14. SUBJECT TERMS Model-based system engineering, analysis of alternatives, concept Development, Simulation, trade-space, system engineering			15. NUMBER OF PAGES 103	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**ANALYSIS OF MISSION EFFECTIVENESS: MODERN SYSTEM
ARCHITECTURE TOOLS FOR PROJECT DEVELOPERS**

Thomas E. Moulds
Civilian, Department of the Navy
B.S., Virginia Polytechnic Institute and State University, 1989

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING MANAGEMENT

from the

**NAVAL POSTGRADUATE SCHOOL
December 2017**

Approved by: Dr. Warren K. Vaneman
Thesis Advisor

Dr. Kristin Giammarco
Co-Advisor

Dr. Ron Giachetti
Chair, Department of System Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Fail early, fail often, but ensure that when failure occurs, a learning period is part of the systems development process. Understanding the reasons a system can fail during the development process is key to maximizing mission effectiveness. Would it not be valuable to have a process that allows the designers to recognize when a system is failing to meet the user's requirements early in the development process? Furthermore, would it not be useful for that process to be iterative, to allow the impacts of changes to be seen in real time, as the concept is defined and the system is designed? What would it be worth to have the ability to accomplish this inside the engineering safety net of Model-Based Systems Engineering? This research shows an alternative process to classic systems engineering and optimization analysis, where system design decisions are statically and dynamically modeled in a Model-Based Systems Engineering environment and "what if" types of changes are answered and analyzed using embedded simulation. This research demonstrates the process with the use case of a highly relevant real-world problem of countering the threat of small commercial unmanned systems to the security of naval installations.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	BACKGROUND	2
	1. Real-World Problem.....	3
	2. Concept Development.....	3
	3. Systems Engineering.....	4
	4. System Modeling.....	5
	5. Model-Based Systems Engineering	7
	6. Operational Concept.....	10
B.	PROBLEM STATEMENT	12
C.	RESEARCH QUESTIONS	12
D.	OBJECTIVES	14
E.	OVERVIEW OF THE APPROACH	14
	1. Research.....	14
	2. Overview	14
	3. Tools	15
	4. Scope.....	15
F.	ORGANIZATION	16
G.	CONCEPT DEVELOPMENT PROCESS	16
	1. Introduction.....	16
	2. AoA Guidance	17
H.	MODEL-BASED SYSTEMS ENGINEERING	19
	1. Similar Areas of Research.....	21
II.	METHODOLOGY	25
A.	GENERAL OVERVIEW	25
B.	DETAILED CHALLENGES.....	25
C.	MODELING METHODOLOGIES	28
D.	PROCESS TOOLS AND DEFINITIONS	28
	1. Modeling Languages.....	28
	2. Simulation.....	32
	3. Tools	32
	4. A Tailored MBSE Process Flow	33
	5. System Definition of Capability and Functionality.....	34
	6. Sub-system Definition with Action Flows.....	35
	7. Mission Analysis Phase with Integrated Simulation.....	35
	8. Assessment of Trade-Space with Simulation.....	36
	9. Recursive Refinement Based on Simulation Results	36

10.	Success-Oriented Exit Process	37
III.	COUNTER UNMANNED SYSTEMS CASE STUDY	39
A.	COUNTER UNMANNED SYSTEMS PROCESS USE CASE	39
1.	System Definition	40
2.	Sub-system Definition.....	47
3.	Create and Define the SV-4A Action View.....	49
4.	Mission Analysis.....	52
5.	Assessment	57
6.	Recursive Refinement.....	61
7.	Exit of Process	61
B.	ANALYSIS OF THE PROCESS.....	61
IV.	CONCLUSIONS/FUTURE WORK	63
A.	SUMMARY OF THE ANALYSIS	63
B.	CONCLUSIONS	63
C.	RECOMMENDATIONS FOR FUTURE WORK.....	66
	APPENDIX. C-UXS MODEL	67
	LIST OF REFERENCES.....	75
	INITIAL DISTRIBUTION LIST	79

LIST OF FIGURES

Figure 1.	The Systems Engineering Process. Source: Bahill (1998).....	4
Figure 2.	Model-Centric Systems Engineering Process.....	5
Figure 3.	MBSE Definitization Cycle.....	13
Figure 4.	Conceptual Modeling Process and the Systems Development Process. Source: Topper (2014, 421).....	21
Figure 5.	UML and SysML Relationship. Source: Open Management Group (2017).....	29
Figure 6.	SysML Basic Unit of Structure.....	30
Figure 7.	Generic Tailored MBSE Process.....	34
Figure 8.	OV-1 C-UAS.....	44
Figure 9.	C-UxS Mission Area.....	45
Figure 10.	Initial CV-2 Snapshot.....	46
Figure 11.	Initial OV-5N Snapshot.....	46
Figure 12.	Decomposed CV-2 Snapshot.....	48
Figure 13.	Decomposed OV-5N Snapshot.....	48
Figure 14.	SV-4 Top Level Action Flow.....	49
Figure 15.	SV-4 Subset Radar Action Flow Decomposition.....	50
Figure 16.	SV-4 Subset Fuse Action Flow Decomposition.....	51
Figure 17.	SV-4 Subset Operator Action Flow Decomposition.....	52
Figure 18.	Discrete Event Simulation User input Blocks.....	53
Figure 19.	User Input Script.....	53
Figure 20.	Simulation User Input Block.....	54
Figure 21.	Threat Setup Script.....	55
Figure 22.	Radar Detection Script.....	56

Figure 23.	Fuse Track's Script	56
Figure 24.	Model Coverage in Simulation	58
Figure 25.	Model Results in Console Window	58
Figure 26.	CV-2: JCA-based capability view	67
Figure 27.	OV-5N: C-UxS Conduct Counter UxS Operations	69
Figure 28.	SV-4: Counter UxS Flow (Top Level View).....	69
Figure 29.	SV-4: Counter UxS Flow (Radar View).....	70
Figure 30.	SV-4: Counter UxS Flow (Electronic Emissions View).....	70
Figure 31.	SV-4: Counter UxS Flow (Electro optic View).....	71
Figure 32.	SV-4: Counter UxS Flow (Fuse View).....	72
Figure 33.	SV-4: Counter UxS Flow (Operator View)	73

LIST OF TABLES

Table 1.	Mapping of SysML Diagrams to LML Diagrams and Entities. Source: Vaneman (2016).	31
Table 2.	Example of Requirements and Performance Metrics	35
Table 3.	Viewpoint and Models Developed.....	41
Table 4.	Requirements and Performance Metrics.....	43
Table 5.	Initial Mission Values	59
Table 6.	First Phase Mission Effectiveness Results.....	60

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AOA	analysis of alternatives
AV	all view
C-UxS	counter unmanned system
CONUS	continental United States
COTS	commercial-off-the-shelf
CV	capabilities view
DOD	Department of Defense
DODAF	Department of Defense Architecture Framework
DOTMLPF	doctrine, organizations, training, materiel, leadership and education, personnel and facilities
DRM	design reference mission
EM	electronic measures
EO	electro optical
ICD	initial capabilities document
IDEF0	integrated computer aided manufacturing definition
IT	information technology
JCA	Joint Capability Area
JCSFL	Joint Common System Functional List
LML	life-cycle modeling language
LSI	Lead System Integrator
MBSE	model-based systems engineering
NAVAIR	U.S. Naval Air Systems Command
OCONUS	outside the continental United States
OMG	open management group
OPNAV	Naval Operations
OV	operational view
PDR	preliminary design review
PEO-U&W	Program Executive Office—Unmanned Aviation and Strike Weapons
RCD	rapid capability development

RPED	rapid prototyping engineering development
SME	subject matter expert
SSoT	single source of truth
SV	system view
SysML	system modeling language
UAS	unmanned aerial system
UxS	unmanned system (all)
UML	unified modeling language
USV	unmanned surface vessel

EXECUTIVE SUMMARY

All systems are designed for mission success, but many are delivered to the customer with inherent and undiscovered design problems, limiting system effectiveness. Understanding the reasons a system can fail during the development process, while time remains to fix the issues, is key to maximizing mission effectiveness. In fact, possessing an understanding of the reasons a system can fail is more important than understanding how the system succeeds, particularly if a system can fail in a catastrophic manner. This research demonstrates the means by which an iterative and interactive design process allows the designers to recognize, early in the development process, a system's failure to meet the users' requirements. This research illustrates the conduct of this design process inside the engineering safety net of Model-Based Systems Engineering (MBSE). This research presents an alternative process to classic systems engineering and optimization analysis, in which system design decisions are statically and dynamically modeled in a MBSE environment, and "what if" types of changes are answered and analyzed using embedded simulation. This process is demonstrated in this research with a use case involving a highly relevant real world problem of countering the threat of small commercial unmanned systems to the security of naval installations.

Although requirements, assumptions, constraints, and stakeholders change and evolve rapidly, the current Department of Defense (DOD) 5000.2 process does not allow for periodic reassessment of the concept. This is the case even when the concept that was initially determined to be viable by the analysis of alternatives (AoA) may not be so when the program reaches preliminary design. With no AoA style reassessment built into the process, even if a stakeholder's initial request is developed, that developed product may not meet the stakeholder's true needs in the end. By embedding the assessment into the system model, mission effectiveness is reassessed periodically against the design baseline. In this way, corrections can be made earlier in the process, and the needed change is discovered early. This may be viewed by some as requirements creep, but the goal is to ensure that the requirements remain valid. To help guide this research, the following questions were asked:

- How can MBSE be used to forecast and investigate mission effectiveness, caused by material and design limitations, to inform and influence the early stages of the system design process?
- How can multiple runs of the simulation that vary the component level effectiveness be used to determine overall system sensitivity once the architectural model is complete with embedded mission effectivity analysis?
- How can the results of the system sensitivity results and analysis be used to optimize design and reliability requirements?
- How can one use sensitivity analysis techniques to adjust the project's path forward by having a continuous positive impact on the early stages of the development process?

To demonstrate this process, updates were made to the U.S. Naval Air Systems Command (NAVAIR) Counter Unmanned Aerial System (UAS) Architecture Framework model using the MBSE tools. This modeling process with embedded simulation offers clearer insight than classic fault analysis methods.

This study uses a MBSE tool to define the Counter UAS capability, functionality, and integrated actions. The effectiveness of the actions are based on the component capabilities and the interactions between these components as work flows through the system and results in a positive or negative outcome. Based on multiple simulation runs in which system input values are varied, predicted system performance is assessed. In addition, the modeling engine embedded in the tool allows for scripting the decision branches with additional randomized capabilities to varied path selection decisions based on the range provided from user input. The user can vary the inputs in order to determine the effectiveness of a model, and can automate variation to understand the sensitivity of the model to the specific capabilities and grouping of capabilities. Areas found to be highly sensitive can then be further investigated and optimized. For example, a counter-unmanned system (C-UxS) may be highly sensitive to a very high-end radar, thus driving cost and size, or it could have lower sensitivity, allowing for additional trade-space in the mixing of multiple lower cost sensors. The goal for a specific site is to allow the selection

of the correct combination of sensors to match cost with required mission effectiveness. The assumption is that a solution that is optimal in one location may not be ideal for another.

This research developed an MBSE process with embedded simulation that is used to assess the mission effectiveness metric early in the process as well as throughout the design lifecycle. A realistic, though truncated, use case for a C-UxS using the MBSE process with simulation was provided by this research.

This use case is intended to demonstrate a process that can be leveraged for all system development efforts. The actual MBSE process with integrated simulation for a large development effort would be much more extensive with all the additional capabilities, functionality, and action cases, to include the interactions between multiple system effectiveness metrics target values. Additionally, the ability to use the single source of truth with the built in simulation at any time in the lifecycle or a site-specific instantiation of the C-UxS allows for continuous knowledge of the systems mission effectiveness metric.

The use case analysis illustrates that, with just a static view, the developers' initial solution did not meet the stakeholders' requirements. The knowledge gained by multiple runs through the solution space via simulation helped guide the process to a successful outcome. The interaction between sensors, fusion, and operator seem simple when looked at statically, but the dynamic interactions are complex for even this simplistic use case. For example, the initial three concept simulation runs did not meet the mission effectiveness requirements, although they all appeared perfectly viable from a static perspective. Therefore, a different focus was placed on system composition to create a more balanced sensor suite that in turn allowed the design to meet the requirement. Without the iterative dynamic analysis throughout the process, the stakeholders may have had to settle on a non-optimized solution that too strongly favored a radar solution over a balanced solution. The key for this research is not the numerical values determined to be the feasible optimized solution, but the fact that the simulation-added knowledge allows for a continual optimization process based on the MBSE process with embedded simulation.

This thesis provides a MBSE process with integrated simulation that allows engineers the flexibility to try many times and fail early, so they can succeed in the long term. Multiple designs can be quickly built, including some that have a higher risk to reward ratio, and then the performance of each can be evaluated relative to the others. The act of understanding the reasons that one solution is more effective than another helps one build a greater understanding of the system. It also allows the designer a better understanding of the possible sensitivity of a solution to a single technology. The process defined above provides a framework to conduct the definition and design phase of system development using a defined and iterative process built on previous MBSE development research on developing large complex systems. This research and the enhanced MBSE process helps in the development of future large, complex systems during system definition and design phases by providing validation of the system model through simulation and analysis. This research shows that the MBSE process is available in all phases of the system lifecycle, as long as the model is maintained as part of the lifecycle process. A general MBSE process with integrated simulation is defined and verified by this research, and a representative C-UxS use case exercised the proposed process. This allowed validation of the process, as a use case provides a clear example of how the assessment provides a more effective overall product.

The discoveries of the related research, the enhancement to the current MBSE process, and the example use case addressed many of the primary research questions that were proposed. The objectives of the thesis effort are met with the development of the MBSE process with embedded simulation and the C-UxS use case. The beneficiaries of the research will be developers of large complex systems in dynamic environments who are able to use the process as a function of fielding and maintaining an effort. The multiple research questions are discussed below, with corresponding details identified for each, based on the research.

- 1) How is MBSE used to forecast and investigate mission effectiveness, caused by material and design limitations, to inform and influence the early stages of the system design process? The question is addressed in two parts - the first is defining the process of MBSE with embedded simulation, and the second is the use case, which

clearly shows the model being used in a relevant use case. The MBSE process with embedded simulation provides a powerful framework for complex system development.

2) Once the architectural model is complete with an embedded mission effectivity analysis, how can multiple runs of the simulation, with varying component level effectiveness probabilities of design choices, be used to determine overall system sensitivity? The question is addressed with the use case, specifically with the inputs of the six runs shown in Table 2 and the results shown in Table 3. In the use case, the design choices are the sensor mix and the capabilities selected from each sensor type. The variation shows where the use cases are most sensitive. In this particular use case, unbalanced systems perform poorly compared to balanced systems.

3) How can the results of the system sensitivity results and analysis be used to optimize design and reliability requirements? This effort leveraged previous work by Perez (2014), in which fault analysis is shown to be viable. This use case was based on capability and cost, but an additional dimension of fault and reliability could have been added based on past work and this research effort.

4) Based on the architectural model with the simulation, how can one-use sensitivity analysis techniques to adjust the project's path forward have a continuous positive impact on the early stages of the development process? The recursive nature of the defined MBSE process with embedded simulation takes what one knows and allows for simulation of the unknown, in a representative environment. The varying inputs made to the model clearly show in a relative manner the sensitivity of the input. Based on the results of the simulation run, the path forward becomes relatively clear. The designer can back out a negative change and try a different trade or continue to refine a positive change. When progress stalls, one may have to make seemingly random changes, simply to recognize the pattern of positive and negative effects and determine a new course of action.

This thesis research provides a MBSE process with integrated simulation that allows engineers the flexibility to test many solutions and fail early, allowing them to

succeed in the long term. The opportunity to fail without catastrophic consequences is truly a very powerful tool in system design.

ACKNOWLEDGMENTS

A big thank you to my wife, Danielle Moulds, and to my family, for supporting all my many days and nights of conducting my research.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Fail early, fail often, but ensure that when failure occurs, a learning period is part of the systems development process. This learning period should take place in a timely manner, such that time remains to fix the system before delivery to the customer. Most agree that all systems are designed for mission success, but many are delivered to the customer with inherent design problems, limiting the systems' effectiveness. Understanding the reasons a system can fail during the development process is key to maximizing mission effectiveness. In fact, understanding system failure is more important than understanding how the system succeeds, particularly if the system can fail in a catastrophic manner. Would it not be valuable to have a process that allows the designers to recognize when a system is failing to meet the user's requirements early in the development process? Furthermore, would it not be useful for that process to be iterative, to allow the impacts of changes to be seen in real-time, as the concept is defined and the system is designed? What would it be worth to have the ability to accomplish this inside the engineering safety net of Model-Based Systems Engineering (MBSE)? This research shows an alternative process to classic systems engineering and optimization analysis, where system design decisions are statically and dynamically modeled in a MBSE environment and "what if" types of changes are answered and analyzed using embedded simulation. This research demonstrates the process with the use case of a highly relevant real-world problem of countering the threat of small commercial unmanned systems to the security of naval installations.

As systems have grown in size and complexity, classic systems engineering methods and system optimization methods have become increasingly cumbersome and ineffective. As an alternative to classic systems engineering and optimization analysis, system and component responses can be statically and dynamically modeled in a MBSE environment to gain an understanding of system effectiveness, similar to classic discrete event simulation and fault analysis methods predicting system suitability. The results can inform the system designer of key areas on which to focus effort to maximize system effectiveness and minimize failure.

Typically, system optimization for a DOD development program starts with the analysis of alternatives (AoA) phase of the acquisition lifecycle. The AoA is conducted by an independent team to help move a program from idea, sometimes known as concept demonstrator, to a Program of Record. However, the AoA effort does not directly flow into the development process, even when performed optimally by the subject matter experts (SMEs) with the knowledge they possess at the time. Much of the AoA effort ends when the AoA phase is complete, and key knowledge is lost. This research demonstrates how moving MBSE techniques to the earliest phases of the lifecycle, including the AoA and concept development phases, can facilitate a deeper understanding of the problem statement and solution trade-space, while allowing the knowledge to be carried over into the later development phases.

The supposition for this research is that optimizing solution trade-space needs to be a continuous effort, similar to managing cost and schedule, and not performed only during the AoA, but throughout the development lifecycle. If effort is initiated early on to capture and automate the trade-space analysis process inside the core systems engineering development tool, then, as assumptions, constraints, and capabilities evolve over time, the development effort can adjust accordingly to provide the best value to the customer.

To help understand how MBSE can be used as part of the AoA trade-space analysis process and to continually re-look at the trade-space optimization, this research builds a system model of an example Counter Unmanned System (C-UxS), adds scripts for automated trade-space analysis, runs multiple simulations, and reports on the findings.

A. BACKGROUND

This background section establishes a common understanding on which this research is based. First, to tie the process to actual use, a real-world problem is described as the basis for a use case that exercises the defined process. This research builds on previous topics of research such as concept development, systems engineering, systems modeling, MBSE, and defining the use case's operational concept, as expanded upon

below. The process defined in Section 3 builds on previous research described in the background and the related research sections.

1. Real-World Problem

In a world of low-cost unmanned aerial systems (UAS), such as the very advanced DJI Phantom 4, which is available to terrorists on web stores with just one-click and one thousand dollars, the threat of low cost commercial-off-the-shelf (COTS) technologies for unmanned systems is a serious concern. Though the Phantom 4 is a small UAS, this class of technology presents a multi-domain problem: air, surface, sea, and ground, so one uses “UxS” as short hand for this considerable threat. Specifically, there are numerous open source articles citing the use of commercial small UAS “out of the box” and similar COTS technologies used in unmanned surface vessels (USV) as improvised weapons. For this research, smaller sized COTS UAS, also known as Group 1, are modeled. The selection of small UAS is based on the significant challenges they present. Small COTS UAS are difficult to defeat consistently due to the rapid technological advancement cycle, their worldwide availability, and their small bird-like profile. These improvised weapons can be used in congested airspace, which adds complexity to detection and shortens engagement windows. Small COTS UAS may seem almost toy-like, but the reality is that they can easily be turned into dangerous autonomous weapons.

2. Concept Development

The systems engineering process is one of the principal methods used to achieve the goals of product development, and serves as the basis for this research, which focuses on the intersection between products and systems, and specifically on the intersection of complex products and systems. The intent for the generic product development process is to develop it in a manner that meets all of the stakeholder’s needs and is value-added to the end user. In other words, the positive value is significantly greater than the negative value, and the product is desirable in spite of some inherent risk in its use. For example, cars or motorcycles can be involved in deadly accidents, but there is value in their transportation features that overrides this risk.

3. Systems Engineering

As defined by INCOSE, “Systems engineering is an important investment in the development of products, and the higher complexity of a product, the better value of that investment. As defined, systems engineering is an interdisciplinary approach and means to enable the realization of successful systems” (INCOSE 2017). The key term in the INCOSE definition is “successful,” and there are multiple processes, methods, and tasks to help a development effort be successful. INCOSE further defines systems engineering as:

Systems engineering is an engineering discipline whose responsibility is creating and executing an interdisciplinary process to ensure that the customer and stakeholder’s needs are satisfied in a high quality, trustworthy, cost efficient and schedule compliant manner throughout a system’s entire lifecycle. This process is usually comprised of the following seven tasks: State the problem, investigate alternatives, model the system, integrate, launch the system, assess performance, and re-evaluate. These functions can be summarized with the acronym SIMILAR: State, Investigate, Model, Integrate, Launch, Assess and Re-evaluate. This Systems Engineering Process is shown in Figure 1. It is important to note that the Systems Engineering Process is not sequential. The functions are performed in a parallel and iterative manner. (INCOSE 2017)

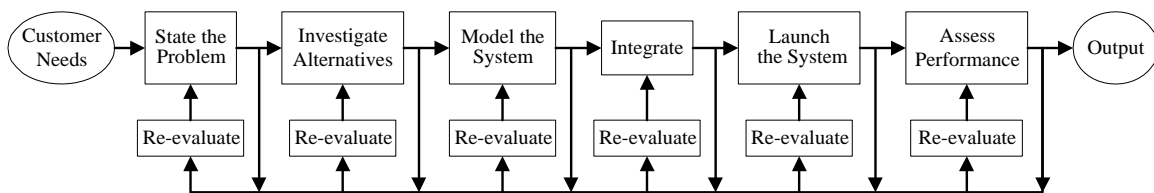


Figure 1. The Systems Engineering Process. Source: Bahill (1998).

Relating the discussion above and the liner recursive systems engineering process, the first and second tasks are the initial tasks of concept development. As shown in Figure 1, the third task of the systems engineering process, Model the System, is the primary emphasis for this research. It uses the model to allow for constant reevaluation of

the design solution compared to the alternatives. For static development efforts, investment in building a model may not be required, but for dynamic efforts, it may be essential.

4. System Modeling

An alternative view of this linear recursive Systems Engineering Process definition, shown in Figure 2, provides additional emphasis on the modeling task. Model the System now encompasses the processes from defining the need to launching the system, which promotes the modeling from a sub-task in a linear process to a recursive wrapper encompassing the primary steps of the systems engineering development process.

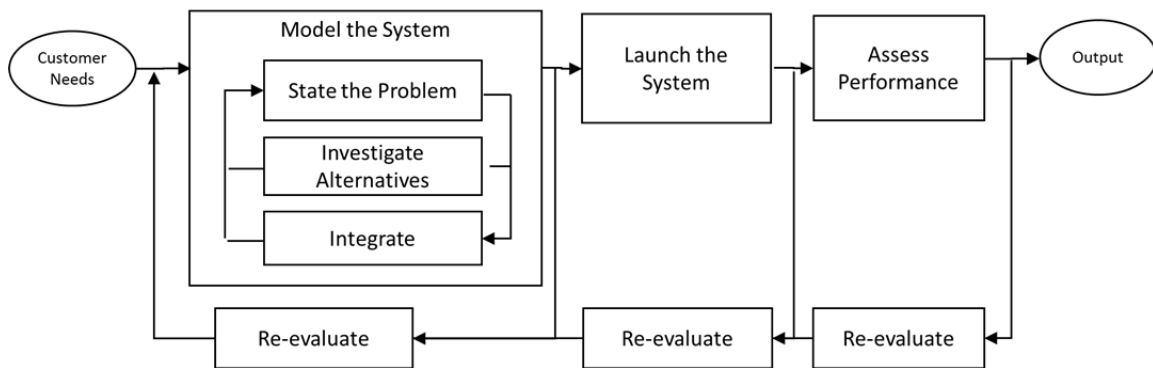


Figure 2. Model-Centric Systems Engineering Process

As shown, this is the manner by which the systems engineering process is transitioning from a linear process to a model-centric-based process. Now that the model is proposed to be the heart of the system development process, it is important to understand the “what and why” of modeling.

As stated above, the goal is to design an effective and desirable system. To reduce the risk of creating a poor design, modeling allows the developers’ team a better understanding of the system. According to Rumbaugh, the models are important to accomplish the following:

1. Capture and state requirements and domain knowledge so that all stakeholders may understand them.
2. Think about the design of the system.
3. Capture design decisions in a mutable form separate from the requirements.
4. Produce usable work products.
5. Organize, find, examine, filter, manipulate, and edit information about large systems.
6. Explore several solutions operationally, economically, and environmentally.
7. Master complex systems. (Rumbaugh 1999, 16–17)

Sussman (2000) also felt it is very important to good systems design to use viewpoints modeling techniques to better understand complex systems, to interactively experiment with different variations of the system, and to simulate the system in a representative environment. Adding to these insights by Rumbaugh and Sussman, this research expands the thought that models are also important to allow for efficient and detailed exploration of engineering trade-space.

Expanding on Rumbaugh's second point above, models positively influence the thought process on the design of a system. The memorable graphical views and the large amount of context that the right side of the brain can process and store are illustrated by the colloquialism, "A picture is worth a thousand words" (Ramos 2012, 103). Personal experience has shown that when one displays a spreadsheet with numerous values to a large audience, one gets limited response (the left brain); however, if the material is a graphical diagram, then there is an increase in audience response. Presenting an interactive graphic for which audience members are able to suggest inputs can yield an enthusiastic discussion that pushes design space (interaction between the right brain and left brain). In addition, after a vigorous discussion, participants remember and ponder the discussion and offer additional inputs long after the event. Also, if one can extend the event into a process, then one thereby enhances the design process and insight of one

team by capturing it into a single model for others to experience, rather than losing that knowledge in a stovepipe work environment.

Merely modeling the system is not sufficient; all the developmental modeling must be performed simultaneously in an integrated development environment. By requiring the systems engineering team to use what some call a “Single Source of Truth” or a “Single Version of Truth” while refining the system, reduces the chance of incompatible changes that appear later during integration events. Both integrated development environment concepts are defined as “Single Source of Truth” (SSoT). Defined by Grealou, “SSoT is the practice of structuring information models and associated schemata, such that every data element is stored exactly once” (Grealou 2016, 1). From a development perspective, at an engineering level, this means that engineering artifacts only have one instance, in the relevant master system, following a specific process or set of processes. “SVoT enables greater data accuracy, uniqueness, timeliness, alignment, etc.” (Grealou 2016, 1) The concept of SSoT for a development effort is extremely powerful, as there are currently many various disconnected models used for analysis of alternatives, tracking critical capabilities, and design development, allowing multiple incompatible versions of the truth to exist. For example, based on additional constraints imposed mid-development, design trades that were true during the analysis of alternatives modeling efforts, may not be true at this later stage. Additionally, if the model used for analysis of alternatives was not maintained past that initial portion of the design phase, the details of why a design trade was true at that time, but not true at the latter stage, are now lost. As a result, sub-optimal trades introduced to the system during the AoA could potentially persist into the design phase.

5. Model-Based Systems Engineering

As shown in Figure 2, a growing approach in systems engineering involves basing the systems engineering process on the model itself, and removing the need for paper-based documentation. This is the emergence of MBSE as the primary framework for complex system development. As defined by Vaneman (2017a, 5), “MBSE is the formalized application of modeling (both static and dynamic) to support systems design

and analysis, throughout all phases of the system lifecycle, through the collection of modeling languages, structure, Model-Based processes, and presentation frameworks used to support the discipline of systems engineering in a ‘Model-Based’ or ‘model-driven’ context.”

The four tenets of this definition, according to OMG (2012) are as follows.

- **Modeling Languages**—Serves as the basis of tools, and enables the development of system models. Modeling languages are based on a logical construct (visual representation) and/or an ontology. An ontology is a collection of standardized, defined terms and concepts and the relationships among the terms and concepts (Dam 2015).
- **Structure**—Defines the relationships between the system’s entities. These structures allow for the emergence of system behaviors and performance characterizations within the model.
- **Model-Based Processes**—Provides the analytical framework to conduct the analysis of the system virtually defined in the model. The Model-Based processes may be traditional systems engineering processes such as requirements management, risk management, or analytical methods such as discrete event simulation, systems dynamics modeling, and dynamic programming.
- **Presentation Frameworks**—Provides the framework for the logical constructs of the system data in visualization models that are appropriate for the given stakeholders. These visualization models take the form of traditional systems engineering models. These individual models are often grouped into frameworks that provide the standard views and descriptions of the models, and the standard data structure of architecture models. The Department of Defense Architecture Framework (DODAF) and the Zachman Framework are examples of frameworks that may be encountered. (OMG, 2012)

This research effort uses MBSE techniques to focus on the “model the conceptual system” phase of the program while fully understanding that the intent of the modeling effort is to show relevance to the complete lifecycle of a system, not merely to illustrate the concept development phase. Modeling is performed early in the development process to refine the problem statement and support the AoA process in an iterative manner until a viable design solution is agreed on by the stakeholders and end users. The power of

MBSE of complex systems is the investment in building a detailed model early in the concept development process to continue to provide value throughout the system's entire lifecycle through the disposal phase. This research, as stated above, focuses on modeling the system in a manner by which the model can be leveraged to drive the decision-making process during concept development. This requires a discussion about specifics on modeling the system.

Model the system: Models will be developed for most alternative designs. The model views for the preferred alternative will be expanded and used to help manage the system throughout its entire lifecycle. Many types of system models are used, such as physical analogs, analytic equations, state machines, block diagrams, functional flow diagrams, object-oriented models, computer simulations, and mental models. Systems Engineering is responsible for creating a product, in this case a complex system, and also for the process to produce it. So, models should be constructed for both the product and the process. Process models allow us, for example, to study scheduling changes, create dynamic PERT charts and perform sensitivity analyses to show the effects of delaying or accelerating certain subprojects. Running the process models reveals bottlenecks and fragmented activities, reduces cost and exposes duplication of effort. Product models help explain the system, and are used in tradeoff studies and risk management. As previously stated, the Systems Engineering Process is not sequential: it is parallel and iterative. This is another example: models must be created before alternatives can be investigated. (Bahill 2009, 2–3)

As an important part of MBSE different views, the concept of risk-informed design plays a large role during concept development. In fact, DOD 5000.2 forces the program manager to address risk as part of the AoA. The original intent of risk-informed design, as initially defined by the National Aeronautics and Space Administration (NASA), is to make informed design trades in order to continuously reduce risk to the crew and the mission, but this technique has been expanded to describe risk inside a MBSE approach (Moulds 2016).

The goal for both risk management and system architecting is to limit the number and effect of unknowns causing failures during the system's lifecycle. Management of what you know and what you do not know helps limit the chance that uncertainty drives the process and goals of development (Antunes 2015). "To people who lived centuries

ago, risk was simply the inevitable nature of chance; an occurrence beyond the realm of human control” (Mun 2015, 25). Today, with the use of modern MBSE methods, all areas that impact mission effectiveness can be discovered and mitigated early in the lifecycle of a system. In terms of system development, if the “hard stuff” is deferred, then the residual risk does not change, leaving a high risk of failure, or at least uncertain, to the very end (Maier 2009). For the purpose of this research, ineffective solutions, risks, faults, reliability issues and failures affecting mission performance are synonymous and represented in the model in a uniform manner. This means that one should first focus on correcting components and processes that have the greatest impact on mission effectiveness in order to have the greatest return on investment. This paper describes how the use of classic methods inside a modern system architecture modeling tool help in discovering, understanding, and documenting issues early enough in the program, thus optimizing resources and maximizing the impact on the development program. Engineers often aim to solve the exciting problems first, delaying resolution for the more mundane, yet relevant (and potentially costly), ones.

6. Operational Concept

To start the development process, the DOD has generated multiple urgent need statements and has purchased several COTS products for concept demonstration. Additionally, the DOD has hosted numerous events, such as Black Dart, where a live exercise environment allows developers to test their C-UxS technologies against live systems. The Design Reference Mission (DRM) has not selected any specific operational concept, but rather elected to develop a generic framework for C-UxS. This generic framework covers the entire kill chain of find, fix, track, target, engage, and assess, but it does not identify any particular technology. The operational requirement is to develop a solution to deter or defeat the representative UxS before they can enter the restricted zone as defined by the end user.

The concept and the architecture is based on the kill chain defined above, but segregated so a technology in one sub-domain is not so tightly coupled with a technology in a separate sub-domain that the solution becomes vendor locked.

1. Find: This sub-domain is normally the trigger when the process transitions from a system waiting for an occurrence to one that is actively processing an event. The sub-domain has numerous names, such as detect, battlespace awareness, collection capability, and the sense function.
2. Fix: This sub-domain is the transition between a raw detection and an establish track. The sub-domain has numerous names, such as classify, process exploitation capability, and performs tracking functions such as form track, fuse track measurements, correlate tracks, and associate tracks.
3. Track: This sub-domain is the monitoring phase in which a detection event meets a minimum criteria such that it should be monitored and persist in the system. This sub-domain covers the transition from the battle space awareness capability to the command and control capability, and completes the perform tracking functionality started on the Fix sub-domain described in item 2.
4. Target: This sub-domain is primarily the classic decision loop. A track is evaluated based on risk, and when the risk reaches a defined threshold, an appropriate course of action is selected. This is part of the decision or the control part of the command and control capability and the decision part of the mission execution functionality.
5. Engage: This sub-domain is one of the easier ones to understand and describe. It is part of the force application capability and the perform engagement group.
6. Assess: This sub-domain is the final step of the active process in which all knowledge from the event is stored and analyzed. This sub-domain also completes the transition from active process back to the waiting phase. This is part of the Understand Command and Control capability and the mission analysis functionality.

Other capabilities that are part of this DRM, but not explicitly part of the kill-chain are force support, logistics, communications and computers, protection, corporate

management and support, and interactions with the UxS. Similarly, other functionalities that are also part of the DRM are mission planning, enterprise IT, and network infrastructure.

From an implementation perspective, the C-UxS is composed of multiple interconnected capacities that either together or alone provide the required functionality, and as an integrated system provide the required capability. The government has the additional requirement to “own the middle” so a solution does not create a vendor monopoly, but at the same time allows for proprietary functionality for find and engage. In addition, other functionalities such as fusion, which technically “lives in the middle,” can also be proprietary, but still be subject to change by the Government Lead System Integrator (LSI).

B. PROBLEM STATEMENT

Requirements, assumptions, constraints, and stakeholders change and evolve rapidly, but the current DOD 5000.2 process does not allow for periodic reassessment of the concept, even when the concept that was initially viable as part of the AoA may no longer be viable when the program gets to preliminary design. Because of this lack of easy reassessment built into the process, many times the initial request is developed, but in the end it is not the solution that is needed. By imbedding the assessment into the system model, mission effectiveness can be reassessed periodically against the design baseline. In this way, corrections can be made earlier in the process, as early as the change in the need is discovered. Some would call this requirements creep, but the goal is to ensure that the requirements are valid.

C. RESEARCH QUESTIONS

- How can MBSE be used to forecast and investigate mission effectiveness, caused by material and design limitations, to inform and influence the early stages of the system design process?
- How can multiple runs of the simulation with varied component level effectiveness probabilities of design choices be used to determine overall

system sensitivity once the architectural model is complete with embedded mission effectivity analysis?

- How can the results of the system sensitivity results and analysis be used to optimize design and reliability requirements?
- How can one use sensitivity analysis techniques to adjust the project's path forward by having a continuous positive impact on the early stages of the development process?

As depicted in Figure 3, early in the development of a complex system, the process presents the development team seemingly limitless possibilities with unknown correlation with mission effectiveness. The AoA process uses modeling and simulation linked to appropriate metrics to help the team understand cause and effect, but this process typically does not continue past the AoA stage.

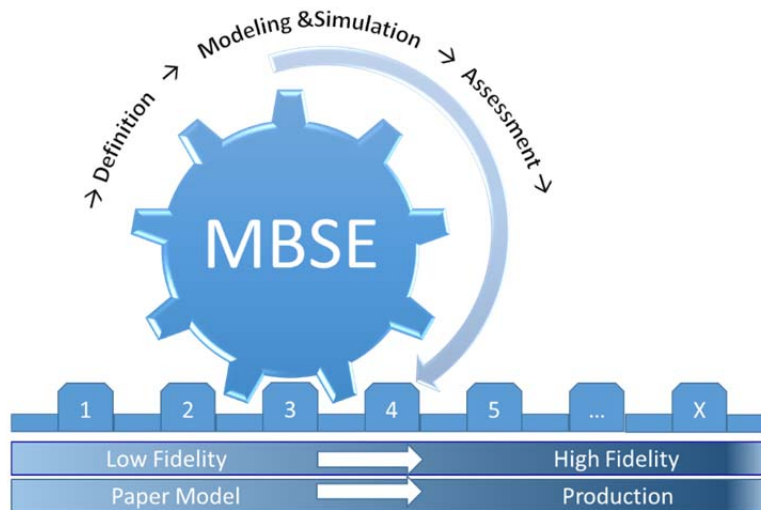


Figure 3. MBSE Definitization Cycle

This research examines whether the “what if” portion of the AoA process can be embedded into the system modeling process. The premise is that once embedded, it can persist throughout the system lifecycle, allowing consistent “what if” questions to be posed again as the system is refined and constrained. The architectural model provides

both a common knowledge base of the system along with embedded integrity checking and mission effective analysis.

D. OBJECTIVES

The objective of this research is to define an MBSE process that ensures that trade-space can be evaluated continuously throughout the lifecycle of a system. This process is based on existing MBSE research and current U.S. Navy engineering practices. Without a complete, repeatable, and embedded method of re-evaluating trade-space, the development effort could result in a sub-optimal design. This makes the real world problem and resulting research a critical process to define and demonstrate.

E. OVERVIEW OF THE APPROACH

1. Research

This research examines the means to merge the AoA modeling process with MBSE developmental and lifecycle processes to capture and expand the “what if” process of concept development. Often the AoA is based on one set of constraints or one view of an ideal system, but shortly after development starts, a new set of constraints emerges. The process compensates, but not necessarily with the same information (model) the AoA used, allowing for sub-optimization of the designed solution. The premise is that if the AoA and the following development process are based on the common model, then, when constraints are added, the effect of the constraints are shown as a function or routine model analysis.

To demonstrate this process, updates are made to the model of the NAVAIR Counter UAS Architecture Framework using the Innoslate MBSE tools. This modeling process with embedded simulation offers insights that are not revealed by classic fault analysis methods.

2. Overview

This study used the MBSE tool to define the Counter UAS capability, functionality, and integrated actions. The effectiveness of the actions are based on the

component capabilities and the interactions between these components as work flows through the system and results in a positive or negative result. Based on multiple simulation runs during which system input values are randomly varied, predicted system performance is assessed. In addition, the modeling engine embedded in the tool allows for scripting the decision branches with additional randomized capabilities to varied path selection decisions based on the range provided from user input. The user can vary the model inputs to view their impact on the outputs in order to understand the sensitivity of the model to specific capabilities and grouping of capabilities. Areas that are found to be highly sensitive can then be further investigated and optimized. For example, a C-UxS system may be highly sensitive to a very high-end radar, thus driving cost and size, or it could have lower sensitivity allowing for additional trade-space in the mixing of multiple lower cost sensors. The goal for a specific site is to enable the selection of the correct combination of sensors to match cost with required mission effectiveness. The assumption is that a solution that is optimal in one location may not be ideal for another.

3. Tools

The goal of the effort is to perform both the AoA and the system definition inside a MBSE tool. For this effort, a MBSE tool that supports modeling, scripting flow, and simulation is used.

4. Scope

While the concept and understanding of predicting mission effectiveness is almost limitless, this paper focus on the idea that model-based methods currently used to define a system as part of development, can also be used to address mission effectiveness analysis and system sensitivity to specific component capabilities. The simulation effort is limited to the current capability of the MBSE tool. The classes of the sensors are limited to three and the fusion engine is rule based. The use of a single tool with a single example system to demonstrate feasibility limits the scope of the effort to a manageable level, but does not attempt a complete proof of equivalence. In addition, the following assumptions guided the development of the example C-UxS model:

(1) UxS will become smaller, cheaper, and more capable as technology evolves; proliferation will increase as UxS become more capable and less expensive, related new technologies will emerge/evolve that enhance UxS operations. (2) Future decisions will provide adequate resources and organizational structure to support C-UxS capabilities development. (3) Current and future capabilities, to include surface-to-air systems, air-to-air systems, Command and Control Systems, are adequate to deal with large UAS. (4) The cyber domain and electromagnetic spectrum will be more contested in the future. (5) Adversaries will challenge the United States in these areas due to evolving technology and proliferation. (Army 2016)

The model expands on cross-domain solutions where it makes sense, recognizing that the C-UxS mission set exists in every domain, not only in the air. The full DRM is on request for the C-UxS real-world problem.

F. ORGANIZATION

Chapter I of this research provides background for the challenge and defines the research questions and overview. Chapter II provides a detailed walk through of complex system development research in systems engineering, MBSE, and related use of MBSE tools for informing development efforts based on risk informed decision making. Chapter III presents the tools selected, the model development, model analysis, and key decisions to make based on the process. The goal of this chapter is to illustrate to the reader the process so one can re-use it, but it also demonstrates the process with a simple example of a generic C-UxS system. Chapter IV reveals the conclusions and provides recommendations for further research. Complete views of the C-UxS Use Case Model are provided in the Appendix.

G. CONCEPT DEVELOPMENT PROCESS

1. Introduction

A key part of the development process is concept generation, as reflected in both commercial best practices and in the DOD 5000.2 acquisition process. As defined by Ulrich, “A product concept is an approximate description of the technology, working principles, and form of the product. It is a concise description of how a product satisfies the customer needs. A concept is usually expressed as a sketch or as a rough, three-

dimensional model and is often accompanied by a brief technical description. The degree to which a product satisfies customers and can be successfully commercialized depends to a large measure on the quality of the underlying concept” (Ulrich 2012, 118). Ulrich also re-enforces the heuristic that “a good concept is sometimes poorly implemented in subsequent development process resulting in a commercial failure, but a poor concept can rarely be manipulated to achieve commercial success” (Ulrich, 2012, 118). The exception to this heuristic is that, at times, the DOD manipulates poor concepts into operational systems, or adds constraints on the process later in the design phase that cause an effective concept to morph into an ineffective one. A continual lack of insight does not allow the potential path to failure to be known until it is too late. Ulrich concludes, “The development of a non-optimal concept is unfortunate because good concept generation leaves the team with confidence that the full space of alternatives has been explored” (Ulrich 2012, 219). Typically, a model of the system is not developed during concept development, but if modeling can be part of the process, than knowledge gained in concept development can stay with the process throughout the lifecycle.

2. AoA Guidance

From the DOD perspective, “the AoA is an important element of the defense acquisition process” (DAU 2012). Similarly, an “AoA is an analytical comparison of the operational effectiveness, suitability, and lifecycle cost (or total ownership cost, if applicable) of alternatives that satisfy established capability needs” (DAU 2012). An important part of an AoA is one of the first DOTMLPF (Doctrine, Organizations, Training, Materiel, Leadership and Education, Personnel, and Facilities) assessments, as understanding limitations of non-material solutions early on allows the AoA process to maximize the understanding of the feasible material solutions.

7. The guidebook goes on to state that, “The AoA is not a point analysis, but should be revisited” (DAU 2012). The concept of revisiting the analysis is often easier said than done, particularly on large programs in which the AoA is often more of a paper-based effort, and therefore hard to revise. Additionally, the AoA team members who are able revise the concept, to have already moved onto the next project. The ability to

automate this process and meet the requirement to revisit the analysis could prove to be very powerful, as one now only needs continuity in the tools, not necessarily in the people.

Additional guidance from the guidebook states, “The AoA is used to identify the most promising end-state materiel solution, but the AoA also can play a supporting role in crafting a cost-effective and balanced evolutionary acquisition strategy. The alternatives considered in the AoA may include alternative evolutionary paths, each path consisting of intermediate nodes leading to the proposed end-state solution. In this way, the analysis can help determine the best path to the end-state solution, based on a balanced assessment of technology maturity and risk, and cost, performance, and schedule considerations. In other words, doing the AoA inside the model allows for more than just the AoA itself, but it establishes a strong foundation and memory for a cost effective and balanced acquisition strategy” (DAU 2012).

The MITRE System Engineering Guide: Performing Analyses of Alternatives provides a very good guidance for AoAs, which is condensed below:

Why do we perform AoAs? AoAs are performed to allow decision makers to understand choices and options for starting a new program or continuing an existing program.

Commercial industry also uses “alternative analyses,” but they are usually more focused on lifecycle cost. The plan is important. It should include the following information:

1. Understand the technology gaps and capability gaps—what needs are the intended system supposed to meet?
2. Develop viable alternatives
 - a. Define the critical questions
 - b. List assumptions and constraints
 - c. Define criteria for viable/non-viable
 - d. Identify representative solutions (systems/programs)
 - e. Develop operational scenarios to use for comparisons/evaluation
3. Identify, request, and evaluate data from the representative systems/programs (determined to be viable)
4. Develop models - Work through scenarios

Know the baseline before starting the AoA, know your stakeholders, beware premature convergence, know your AoA Team, understand the

mission, obtain technical descriptions of the materiel solutions, anticipate problems, and be persistent! (MITRE 2017, 438)

A recent Government Accountability Office (GAO) report on defense acquisitions “attributes premature focus on a particular solution or range of solutions as a failing of AoAs” (GAO 2009). The GAO report goes on to state that, “If stakeholders are already enamored of a particular solution, completing a full AoA may be difficult” (GAO 2009).

The current best practice from this GAO report and practical experience only recommends that an AoA is completed before program requirements are set, but this research looks beyond the AoA guidance to determine the manner in which the AoA process can be part of the full system development process.

H. MODEL-BASED SYSTEMS ENGINEERING

MBSE was envisioned to transform systems engineering’s reliance on document-based work products to an engineering environment that is based on models. This transformation means more than using model-based tools and processes to create hard-copy text-based documents, drawings, and diagrams. Data in a MBSE environment is ideally maintained within a single repository, has a singular definition for any model element, and allows for the static and dynamic representations of a system from several different perspectives and levels of decomposition (Vaneman 2017a, 8). As stated above, this single repository can also be thought of as a Single Source of Truth (SSoT) for system development that is accessible to multiple tools and processes. An additional discriminator between document-based work products and Model-Based tools is that connections between paper documents is usually in the minds of the authors, but not captured in the model via interconnect views. Specifically, in a MBSE environment, each entity is represented as data, only once, with all necessary attributes and relationships of that entity portrayed. This data representation then allows the entity to be explored from the various engineering and programmatic perspectives (viewpoints). According to Vaneman’s paper, a viewpoint visualizes abstracts from one perspective in a way useful to programmatic decision-making. Vaneman defined the compilation of viewpoints (e.g., capability, operational, system, programmatic viewpoints) as representing the entire

system, where the system can be explored as a whole, or from a single perspective (Vaneman 2017a, 8).

According to Topper (2013, 419), “the goal of this conceptual model, which is now to be built as a function of the MBSE process, is to build a complete, coherent representation of a system and its operating domain, including interactions with other systems and with its environment that is common across the stakeholder community.” Concept development is the early phase in system development where brainstorming turns into prototypes and prototypes can be assessed for individual and collective merit. These prototypes can be physical or modeled in a tool, which may or may not include simulation. The purpose behind conceptual modeling is to garner an understanding of successes and failures in the solution space. It also must show that there is at least one possible solution, including documented analysis that the proposed solutions are actual solutions to the problem. The process Topper developed and used to build the conceptual model described below, involves creating the following artifacts:

- Domain model: This artifact describes the system and the environment. It captures the high-level components of the system and its operating environment and establishes the normalized referential framework particularly important for multi-disciplined stakeholder organizations.
- Use cases: These written descriptions of what the system will do capture its expected behaviors and its interactions with external actors.
- Functional model: The functional model describes how the system will accomplish its goals. It breaks the use cases into greater detail and shows activity flows and state transitions among components. Complex functionality, an increasingly common characteristic of modern systems, is difficult to address using traditional assessment techniques. In conjunction with other artifacts presented in this section, new techniques, outlined in the Functional Thread Analysis section, enable and enhance analysis, testing, and evaluation of complex systems, which are difficult to assess using traditional analytical methodologies and tools.
- Structural model: This specification of system structure allocates attributes and operations to system components, expanding and adding detail to the domain model. (Topper 2013, 420)

Note that with modern MBSE tools and languages, use case can be represented in a variety of ways such as Unified Modeling Language (UML), written use cases,

sequence diagrams, and action diagrams. Topper’s conceptual modeling process described above is shown in Figure 4.

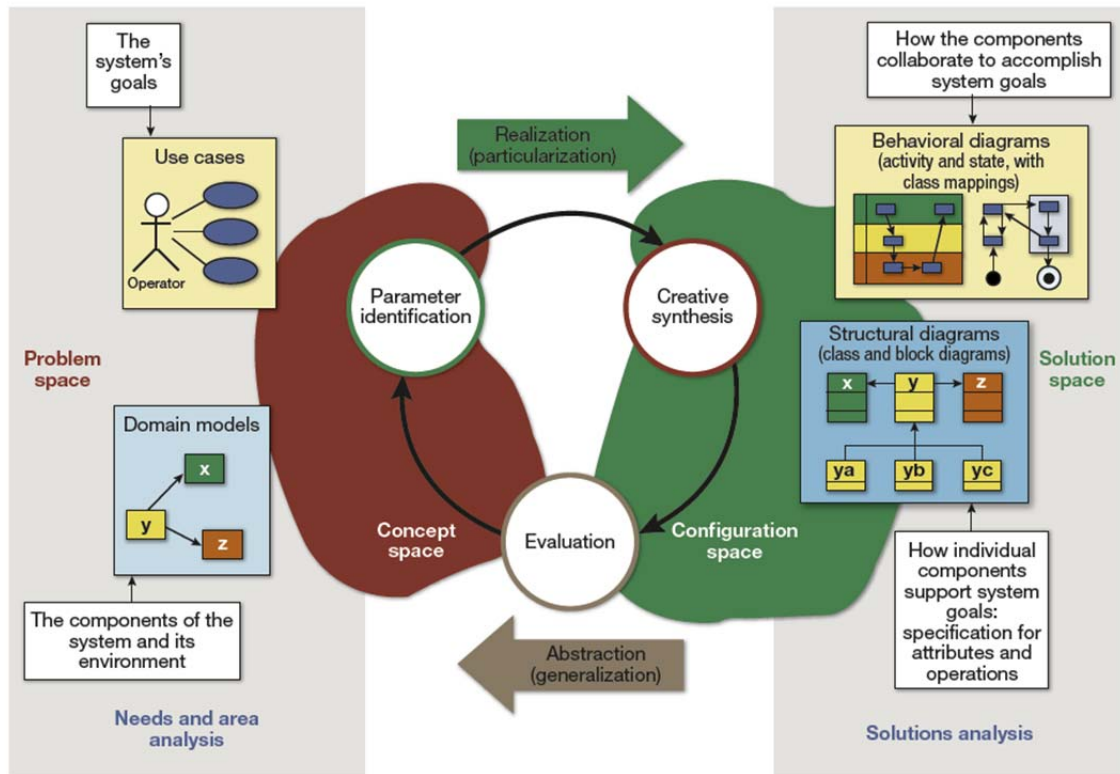


Figure 4. Conceptual Modeling Process and the Systems Development Process.
Source: Topper (2014, 421).

The process defined by Topper above is very detailed in the modeling of the systems and the interactive process when modeling the systems, but does not specifically add simulation as part of the trade-space analysis. This research defines a process that specifically incorporates simulation as part of the modeling process.

1. Similar Areas of Research

The use of MBSE is an area with rapidly advancing techniques, processes, and tool capabilities. Based on research, MBSE has been ongoing since at least 2010. The areas that are most relevant to this research are shown below.

a. *MBSE Supporting Development of Systems Architectures in Naval Ship Design*

Research by Tepper (2010) in the MBSE Supporting Development of Systems Architectures in Naval Ship Design paper shows that the use of dynamic techniques in a static model adds value, particularly if the results remain inside the model. Tepper states that dynamic models can support an analysis of alternatives (AoA) by conducting system design trades based on defined use cases inside the model to assess if the system capability satisfies mission requirements (Tepper 2010). Key decision-making artifacts of MBSE process are trade-space analysis, understanding the impact of changes, and the capability to have version control of changes along with the rationale of making the decisions. Additionally, Tepper's research built in a version management process embedded in the tools that provides traceability for changes made and historical record of the alternatives. Tepper concludes with "the designer is able to see how a small change in one aspect of the design can drastically affect the whole" (Tepper 2010, 18). Though this research is based on system architectures in U.S. Naval ship design, similar effects have been seen in the design of any complex system, where small changes have rippled through the design causing extensive and unneeded rework.

b. *MBSE Supporting Risk-Informed Design Methods*

Research by Perez (2014) into the application of MBSE tools and processes to Risk-Informed Design (RID) provides the capability to perform risk analysis early in the life-cycle. The research focused on spaceflight projects, but the concept of performing risk analysis inside the model is valid in many areas. Perez describes, "risk-informed design uses a 'minimum functionality' approach, whereby a minimal, single-string system design is first envisioned that only meets basic performance requirements without any regard to overall reliability or safety" (Perez 2014, 6). A key enabler for this research is that Perez was able to model risk inside the MBSE tool for the first time, which sets the stage for moving the AoA and the analyses inside the tool. Perez's research is an excellent example of a MBSE based process applied to a basic system model with scripted simulation for part of the Altair lunar lander system (Perez 2014). Perez's

research is also a prime example that, theoretically, MBSE is capable of supporting a full risk assessment and that a tool like Innoslate is able to actually support it.

c. MBSE Supporting of Complex Systems Development

Research by Topper (2014) operated on the premise that the MBSE technique facilitates complex design and documentation processes. As stated earlier, the key to the benefit based on this research states that these MBSE techniques best support complex systems. Topper goes on to state that “the resulting model is more useful than traditional documentation because it represents structure, data, and functions, along with associated documentation, in a multidimensional, navigable format.” Benefits extend beyond traditional system definition and documentation since language-based models also support automated analysis methods, such as functional thread extraction. The definition of functional thread analysis is relevant to this research and is defined by Topper (2014, 424) as the following: “The state of a complex system changes continuously as the designed functionality is executed within changing mission phases and environmental conditions. These systems can invoke a large number of functional threads to accomplish (or fail) a required task, and as system complexity grows, it can be difficult to identify critical threads and accurately assess key system performance requirements.” Note that the emphasis is on the ability to accurately assess key system performance requirements.

Topper’s conclusions state that, “The increase in system complexity precipitated by the advent of network-centric systems, MBSE techniques offer a way to capture, archive, and use information that is essential for complex system design, analysis, implementation, and test and evaluation (T&E) throughout a system’s lifecycle” (2014, 430). In Topper’s research, the “conceptual model includes entities, their important attributes and interrelationships, how they operate and behave, and any assumptions made about them.” Topper goes on to state that, “MBSE provides a basis for future analysis studies, model development, simulation efforts, system requirements definition, and program information management” (2014, 430). Topper (2014) believes that a robust conceptual model does the following:

- Facilitates communication and collaboration among project stakeholders by standardizing and documenting a common reference blueprint for the

project. This basis allows the team to exhaustively explore the system's conceptual and configuration spaces, and identify and assess key parameters in the evaluation of system alternatives.

- Promotes reuse of components and analytical results among projects across a shared domain.
- Enables information management and integrates business and engineering processes into a single model. A conceptual model of the project, particularly one that reuses components from previous projects and includes elements from the enterprise architecture as well as the system, allows managers to better estimate the scope, schedule, and resources needed to develop and deploy a complex system.
- Documents traceability from needs to results, supporting verification and validation. (Topper 2014, 430)

II. METHODOLOGY

Based on the literature review conducted on systems engineering, risk-based design, and MBSE, previous research is leveraged to define a MBSE process with embedded trade-space analysis that supports appropriate concept development, design, implementation, fielding, and support throughout the lifecycle of a system. This chapter identifies the basic MBSE process, discusses the specific challenges associated with design of complex systems in a rapidly changing world that need to be addressed by a MBSE approach, and details the embedded trade-space analysis approach, leveraging the research on MBSE process.

A. GENERAL OVERVIEW

A C-UxS system integrates capabilities to support, detect, classify, track, and defeat COTS unmanned system technologies. As no single capability or standalone system provides the required capability, the C-UxS system is considered a System of Systems with multiple complimentary capabilities or sub-systems integrated together. The C-UxS system is defined using MBSE processes and the Innoslate tools discussed in earlier chapters to fulfill the C-UxS mission as defined by the DRM with the appropriate level of mission effectiveness. Using the MBSE process, this research provides views of the system, as it is modeled, during the concept development (AoA) phase of the project. This research shows how embedding discrete events simulation into the Innoslate model using the scripting tools increases the overall understanding of the available trade-space. Analysis of the results is provided in a summary table showing the cause and effect of different trades indicated by the changes in the theoretical mission effectiveness reference metrics.

B. DETAILED CHALLENGES

There are numerous challenges associated with defining a C-UxS system reference architecture that can support the full lifecycle and multiple instantiations. This complex problem space renders it crucial to follow a defined and repeatable process while documenting capabilities, functionality, assumptions, and trade-space decisions.

With the high rate of change of COTS UxS threats and C-UxS capabilities, the process must be responsive, transparent, and flexible. If shortcuts to the process are made, then the C-UxS system does not support the capabilities required by the users as they are needed, and the design of the C-UxS Architecture does not fully meet the supportability and extendibility requirements. More importantly, if the initial trades made early in the process are not well understood and captured inside the development model, then the system is not able to evolve past the AoA baseline to support future threats, even if it was able to support them when initially fielded using the initial AoA based trades.

The first challenge is that in early phases of concept development, the trade-space can have many dimensions, such as types and number of capabilities, so the combination that provides the best mission effectiveness value is not obvious. This challenge manifests itself when modeling a system and optimizing the composite architecture for maximum mission effectiveness when the exact value of each independent capability can be a range, not a point value, and all the possible combinations of each capability and the manner in which the capabilities are integrated can be very high. This can easily be an unsolvable problem from an optimization perspective, so the system designers are required to venture best guesses and assess them. In addition, the mission effectiveness of a system does not have to be optimal; it merely has to be equal to or above the required mission effectiveness value set by the end user. In some cases, a sub-optimal mission effectiveness value may be the best value when cost and reliability are considered. For system designers, it is very hard to resist the temptation to not proceed with the best technology, so a clear understanding needs to be developed and maintained when making selections and knowing the way in which those selections impact total system performance.

The second challenge is combinational and dynamic complexity that can make determination of the direct impact of a change almost impossible. Combinational and dynamic complexity are concepts defined by the operational research community, but they can apply to the engineering community, as the line between system definition and system analysis is blurred. For this research, combination complexity is defined as the point where multiple combinations of systems, each with unique capabilities, are

integrated into a system of systems, but the unique aspect of each capability must still be accounted for in the design. For this research, dynamic complexity is defined as the common changes to input that can cause unique changes to output for each independent system that is part of the system of systems. In other words, changes to input can affect each system differently. This is a challenge, because what may be an obvious optimization trade to one part of the system or during one part of the process, may not be an optimal trade overall. For example, decisions made during the AoA phase may be misunderstood during the design phase and a subsequent trade may inadvertently cause the system to be less optimized rather than more optimized. Though not directly studied in MBSE, this has been assessed in other areas. For example, Vaneman (2017b) states, “the organization’s ability to master these transient periods is fundamental to achieving steady state operations more efficiently, thus reducing losses due to sub-optimal performance” (Vaneman 2017b). In terms of the DOD acquisition process, the DOD has been very successful in converging on an optimized solution during the AoA phase and during the design phase. However, those two optimized solutions are frequently different, and the understanding of the basis on which they differ is lost, as the end result is the trades based on a down selection of many different possible combinations and the dynamic effect of those combinations.

The third challenge is that each site can slightly vary, so a one-size-only solution may work for one site, but fail at another site. The variation to the external conditions at different sites can be obvious, (e.g., obvious differences in terrain), or be harder to visualize differences in regulations, (e.g., radio frequency spectrum interference and acoustic noise).

The fourth challenge is that the COTS technology has a very fast refresh rate. In this case for C-UxS, there may be a new threat system introduced every six to 12 months, a time period that could easily be inside the typical development timeline, so a design solution that was valid during the AoA may not remain valid when the system is fielded. The awareness of when a threat improvement significantly moves the needle is critical to ensure that the C-UxS system remains relevant.

Together these challenges need a robust modeling process, automated trade-space analysis, and flexibility to do initial “what ifs” and to validate that capabilities and architecture designs made in the past are still valid in the present.

C. MODELING METHODOLOGIES

The modeling technique used defines the system from both a static and a dynamic perspective with the focus on the dynamic action views. For context, static modeling is used to represent the static constituents of a system model such as the hierarchy, capabilities, functionality, and static interfaces between capabilities and functionalities. Examples of common static diagrams in a DODAF vernacular are Capability View-2 (CV-2), a Operation View-2 and -5a (OV-2, OV-5a). Dynamic modeling is used to represent behavior of the static representation of a system, as well as interaction and emergent aspects as a system is exercised. Examples of common dynamic diagrams in a DODAF vernacular are System View-4 (SV-4) and Operation-5a (OV-5b).

In addition, the dynamic models allow a “glimpse into the black-box” by taking the inner-workings of the internal structure into account. They also allow for inputs from one period to result in outputs for another period (Kao 2014).

D. PROCESS TOOLS AND DEFINITIONS

1. Modeling Languages

For this research, the primary two modeling languages used are Systems Modeling Language (SysML) and Life Cycle Modeling Language (LML).

a. System Modeling Language

The Systems Modeling Language is best defined and described by the Open Management Group (OMG) SysML web page (OMG 2017), “What is SysML” and is included below:

SysML is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities. In particular, the language provides graphical representations with a semantic foundation for modeling system requirements, behavior, structure, and

parametrics, which is used to integrate with other engineering analysis models. (Open Management Group 2017)

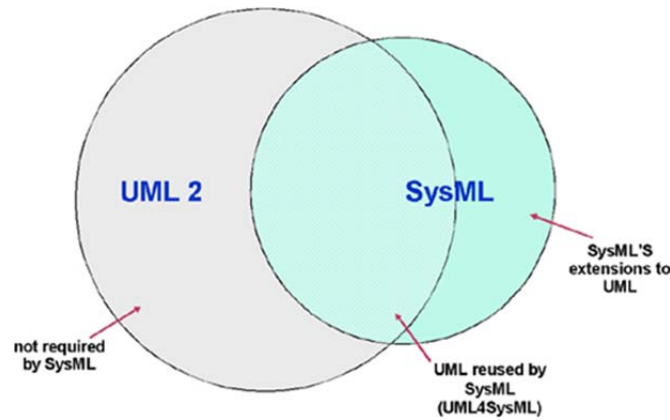


Figure 5. UML and SysML Relationship.
Source: Open Management Group (2017).

The behavior diagrams include the use case diagram, activity diagram, sequence diagram, and state machine diagram. A use-case diagram provides a high-level description of functionality that is achieved through interaction among systems or system parts. The activity diagram represents the flow of data and control between activities. A sequence diagram represents the interaction between collaborating parts of a system. The state machine diagram describes the state transitions and actions that a system or its parts perform in response to events.

SysML includes a graphical construct to represent text-based requirements and relate them to other model elements. The requirements diagram captures requirements hierarchies and requirements derivation, and they satisfy and verify relationships to allow a modeler to relate a requirement to a model element that satisfies or verifies the requirements. The requirement diagram provides a bridge between the typical requirements management tools and the system models.

The parametric diagram represents constraints on system property values such as performance, reliability, and mass properties, and serves as a means to integrate the specification and design models with engineering analysis models. (Open Management Group 2017)

The modeling method for this research uses many of the SysML extensions it provides to UML, as shown in Figure 5. Specifically, it uses the performance properties as defined in the Parametric Diagram. The parametric diagram, as shown in Figure 6, could lead one to believe that it is a single physical representative block for each

capability that is easily defined. However, for this effort, and for most modeling efforts, the parametric data is defined in multiple ways and at times in different ways, for each capability as part of the modeling and scripting effort.

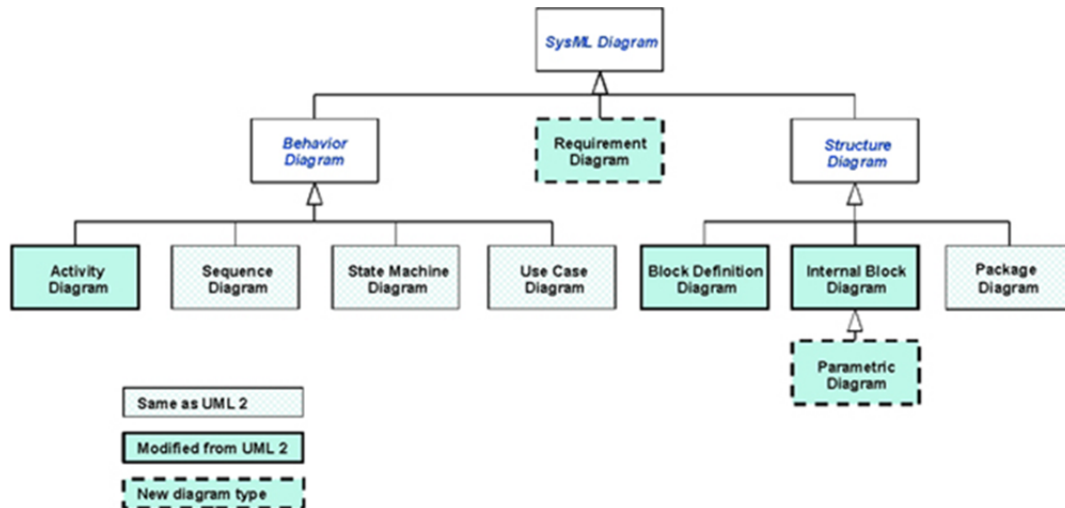


Figure 6. SysML Basic Unit of Structure

b. Lifecycle Modeling Language

The Lifecycle Modeling Language is best defined and described by the Lifecycle Modeling Organization (2015) Lifecycle Modeling Language Specification 1.1:

The basis for the LML formulation is the classic entity, relationship, and attribute meta-meta model. This formulation modifies the classical approach slightly by including attributes on relationships, to provide the adverb, as well as the noun (entity), relationship (verb), and attribute (adjective) language elements. Since LML was designed to translate to object languages, such as UML/SysML, these language elements correspond to classes (entity), relations (relationship), and properties (attribute).

Extending the above reference from Lifecycle Modeling Language Steering Committee, Vaneman (2016, 5) states, “Once mapped, the LML visualization models can be associated to the corresponding LML entity, and by extension provides an ontology for SysML. Providing this ontology will prove important to practitioners as they will be able to better represent the complexities of a system.” In other words, LML attempts to

simplify the modeling effort; where DODAF may have had multiple names for very similar nodes leading to confusion by the modelers, LML has just one node.

c. Mapping of SysML to LML Diagrams

The good news is that the complexities and simplification of UML, SysML, and LML are now primarily handled by the tools that support the modeling process. This research uses primarily LML model language, but note that the tag of “Use Case” is overloaded and has different meaning to different architecture practitioners. For reference, a mapping between the SysML diagrams and the LML entities is shown in Table 4. This mapping provides an understanding of commonality among the SysML and LML visualization models (Vaneman 2016, 6).

Table 1. Mapping of SysML Diagrams to LML Diagrams and Entities.
Source: Vaneman (2016).

SysML Models	LML Models	LML Entities
Activity	Action Diagram	Action, Input/Output
Sequence	Sequence	Action, Asset
State Machine	State Machine	Characteristic (State), Action (Event)
Use Case	Asset Diagram	Asset, Connection
Block Definition	Class Diagram, Hierarchy Chart	Input/Output (Data Class), Action (Method), Characteristic (Property)
Internal Block	Asset Diagram	Asset, Connection
Package	Asset Diagram	Asset, Connection
Parametric	Hierarchy, Spider, Radar	Characteristic
Requirement	Hierarchy, Spider	Requirement and related entities

Vaneman goes on to state:

The Lifecycle Modeling Language defines a new approach to MBSE that simplifies the ontologies and logical constructs found in previous MBSE methods and languages. Coupling SysML and LML provides an environment with an ontology that allows system concepts to be better represented by denoting underlying properties, relationships, and interrelationships. LML provides a means to improve how we model

system functionality to ensure functions are embedded in the design at the proper points and captured as part of the functional and physical requirements needed for design and test. (Vaneman 2016, 6)

Therefore, while it is important to understand the basis of the modeling languages and the mapping between the UML extensions that are available for use, one does not need to fully understand all the details provided in the three specifications.

2. Simulation

The simulation engine is based on a combination of discrete event simulation for known decision points selected by the user, and Monte Carlo simulation in which system performance effect can be randomized. Real Time Discrete Event Simulation is a model, both mathematical and logical, of a designed system with decision paths at precise points in the simulation flow. The discrete event simulator predicates key system and project metrics based on user input along with the randomization of the variable events. The Monte Carlo simulation definition that best fits this process is randomizing of decisions that impact the results, but are not specifically modeled. For this effort, the simulation capability allows the variation of design choices that impact the key mission effectiveness metric to indicate the best point in the solution space where mission effectiveness meets the required value.

3. Tools

a. Innoslate

This effort was modeled in the Spec Innovations Innoslate tool. Innoslate includes the modern end-to-end design, modeling, and traceability capabilities systems in industry standard LML, SysML, and IDEF0. These models, (e.g., the activity diagram), can easily be simulated with integrated discrete event and Monte Carlo simulators, coupled with additional handwritten scripts to better specify flow. The Innoslate tool allows for local execution of the model and the ability to export the results to a comma separated value file for additional analysis.

The clean interface, simple relationships, and modern diagram visualizations make managing model entities easier than ever. Currently, there are over nine different diagrams to visualize behavioral models including executable Action, Sequence, N-squared, and IDEF0. Physical models have eight different diagrams including Asset, Class, Use Case, and Organization Chart. All of the diagrams are drag droppable, allowing for quick model design and construction. The diagrams conform to the LML, SysML, or the IDEF0 standard. Innoslate includes both a discrete event simulator and fully scalable discrete event and Monte Carlo simulator to execute system models and verify model correctness. These simulators can calculate a system's time, cost, and resource levels, and produce easy to read graphical outputs (including Gantt charts, cost curves, resource usage). The model maturity checker evaluates the model according to best-practice heuristics developed by research at Naval Postgraduate School and Stevens Institute of Technology. (Innoslate 2016)

As discussed above, the Innoslate tool provides an easy to use graphical interface, allowing us to build a well-defined model based on the UML, SysML, and LML well defined rules.

4. A Tailored MBSE Process Flow

The following sections describe the methodology to support system definition and assessment of trade-space decisions. This addresses the formal definition of a system and the interactive method of assessing and refining the system definition. The generalized recommended process is shown in Figure 7. The methodology for this research is broken up into four steps: system definition, sub-system definition, mission analysis, and assessment for the generic tailored MBSE Process. As discussed previously, the focus area is on mission analysis and assessment, but because the system must be defined before it can be analyzed and assessed, this research also develops a simple modeling plan, a use case reference mission, and a high-level set of requirements. The research and the use cases are based by a tailored version of the DODAF 2.0 products and views as defined in the All View -1 (AV-1) and available in the Appendix. As system definition and decomposition is not the focus area, the discussion is limited, but is included in the model for completeness.

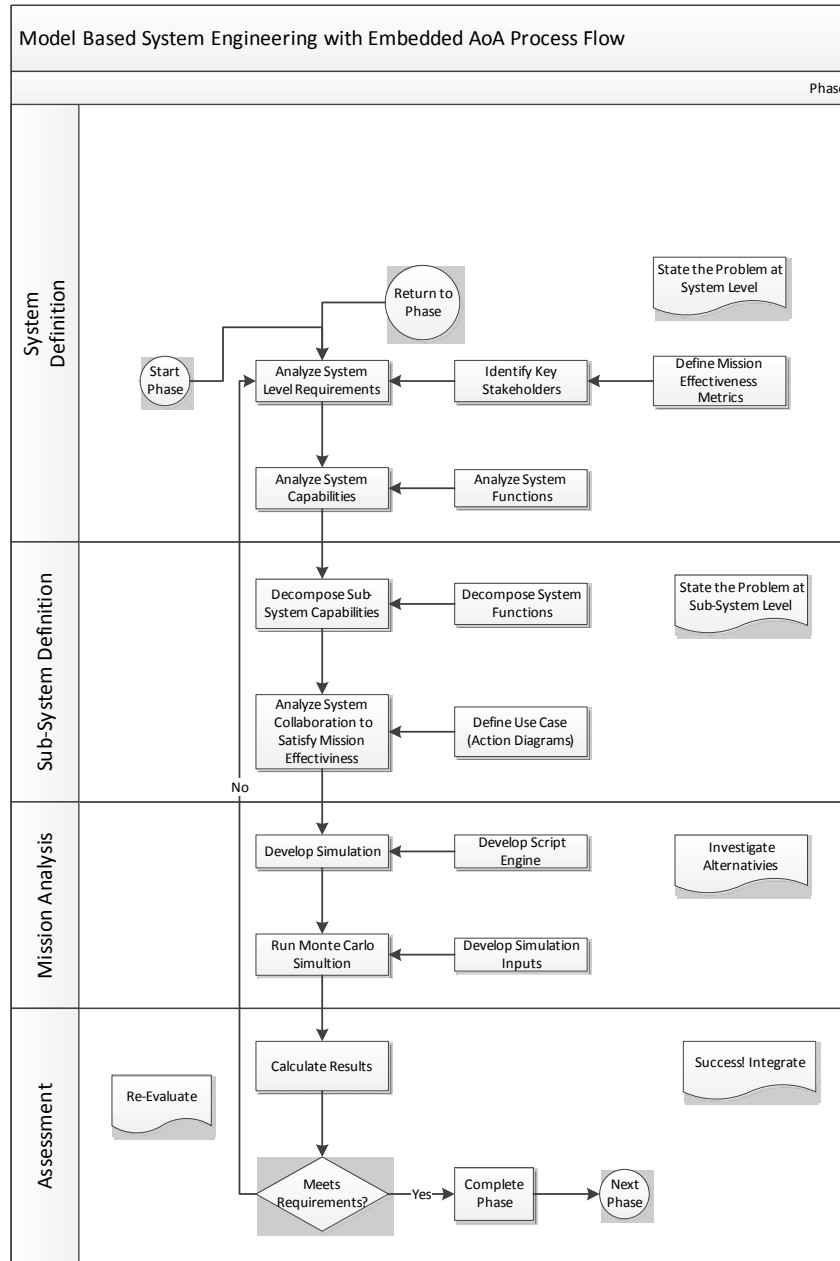


Figure 7. Generic Tailored MBSE Process

5. System Definition of Capability and Functionality

The system definition phase starts with the All View that presents the overall model development plan: define system level requirements, identify key stakeholders, develop the initial set of design reference missions, create the initial capability view defining the systems capabilities, and create the initial operational view defining the

system functionality. It also defines the metrics and target values for determining mission effectiveness later in the process. A clear, concise system level definition is critical for overall success of the effort, as the analysis of the effectiveness to the trade-space decisions is based on the simulation results compared to the system level required performance metrics. An example of the format for a Requirements and Performance Metrics table is shown in Table 1. The system stakeholders are able to use this table to capture and define the constraints for cost and mission effectiveness.

Table 2. Example of Requirements and Performance Metrics

Capability	Weight (1-3)	Description of Metric	Metric Target Value
System Requirement 1	1	Plain English description of metric	Value of required performance
System Req N	3	Plain English description of metric	Value of required performance

6. Sub-system Definition with Action Flows

The sub-system definition phase states the problem at the sub-systems level by decomposing the capability view and the operational view to the leaf level, and defining the appropriate action views for use case definition. At this point, the basic flow of the initial system is created and defined in the action views as well as in the decomposition of the action views. The action views perform a capability or a sub-function of the capability. All capabilities are fulfilled by an action that provides the required functionality. The inclusion of which capabilities (CVs) and functionality (OVs) are fulfilled by an action view provides the system traceability that can be used in the completeness and coverages metrics.

7. Mission Analysis Phase with Integrated Simulation

The mission analysis phase develops rules for the action flow defined by the SV-4's action views. By creating input blocks to define user inputs and environment variability with the random function, and by adding scripts to define characteristics of the

flow to the simulation engine as attributes to the action diagram logic blocks, the discrete event simulation predicts mission effectiveness. The discrete event simulation can be run at all phases of the process, with multiple operator selected combinations of capabilities, with results displayed at the end of each simulation run and exported to a common separated value (CSV) export file. Innoslate's discrete event simulation run off the "advance functional flow diagram" allows for the operator input and scripted instructions to describe low-level flow. Note that the case study in Chapter IV describes one pass through evaluation process, however for a real program this would be continuous evaluation occurring as new trades are made as part of the development process. In this case study, the example pass is completed during the concept demonstration phase.

8. Assessment of Trade-Space with Simulation

Using both the MBSE tool and Excel spreadsheet calculations, the results are calculated and compared to the requirements and the system level metric target values. This step includes defining the input and recording the results into the spreadsheet, building an analysis summary to allow for comparisons, and documenting the results. Inside the spreadsheet, analysis utilizes pivot tables to organize raw data and summarize results. Based on the results, as compared to the mission effectiveness metrics, the process is repeated by re-entering the sub-system definition phase to update and/or refine the model to improve results. The process is continued throughout the lifecycle to account for system updates, changes in environment, and needs of the stakeholders.

9. Recursive Refinement Based on Simulation Results

The power of the process is that designers are not required to know all the optimal trades upfront prior to conducting an initial pass through the process. One can start with reasonable trades based on what is known at the time. The process is an iterative cycle in which the current information known about the design of desired system is modeled, simulated and assessed. Though the goal is to implement positive changes to the system based on the process, though many times implicit changes thought to be benign to system performance have a significant impact, which is why catching them early is critical.

10. Success-Oriented Exit Process

The actual exit of the MBSE lifecycle process is not until the system is retired, but there are times when one may exit a phase of the lifecycle and hand it over to another entity for care and feeding. If the process is followed correctly and all the knowledge is already captured inside the Single Source of Truth (SSoT), then no further action is needed, but, if not, then this knowledge needs to be added. Because the exit of one phase is most likely the entrance to another phase, the handover from the previous knowledge manager to the current knowledge manager is key. For this effort, the exit of the process is the documentation of results by the use case to show the manner by which the above process fully supports the needs of the development process. The use case example in the next chapter shows one phase with six input refinement iterations to show how the MBSE is refined based on the feedback from the simulation and assessment efforts. A real effort would have many more refinement iterations over the lifecycle, but it is felt that demonstrating a single phase with multiple input refinement iterations has sufficient granularity to demonstrate the positive effect on the overall system development process.

THIS PAGE INTENTIONALLY LEFT BLANK

III. COUNTER UNMANNED SYSTEMS CASE STUDY

The previous section identified a tailored MBSE process for including a continuous AoA process within the SSoT model. This process uses a case study for a Navy C-UxS architecture to confirm that the process is applicable and effective. This use case makes one pass through the process shown in Figure 7 to demonstrate all the steps. The C-UxS, which requires flexibility and adaptability to support multiple unique sites provides a challenging case study for the proposed process. The proposed MBSE process with embedded simulation provides a iterative lifecycle framework that can be used from concept development through critical design activities phase. The research is that the MBSE process of defining the system provides valuable context and the simulation process of performing “what if” provides knowledge, but if completed together as a single process it combines the context and knowledge for much greater understanding of the system.

A. COUNTER UNMANNED SYSTEMS PROCESS USE CASE

The process identified in the previous chapter is applied to the C-UxS acquisition environment. The below sections go through each step of the process and employ the use case for a U.S. Navy architecture. For reference, Step One provides broad context back to the DoD domain. Step Two decomposes the broad context defined in Step One down to details relevant to the C-UxS problem statement and suitable to define a relevant action diagram. Step Three defines the actions that are relevant to the system metrics in the C-UxS action diagram. Step Four links the system definition phase (context) with the system analysis phase by adding simulation flow and simulation scripts to the model. Finally, Step Five completes the process (knowledge) by assessing the simulation output based on the context of the model. For additional general context, the Appendix contains the full model views, and, on request, the model is available on-line via the Innoslate web application. The full DRM is also available on request for additional mission context.

1. System Definition

The system definition phase uses the All View template for the overall model development plan. That includes: define system level requirements, identify key stakeholders, develop the initial set of design reference missions, create the initial CV-2 defining the systems capabilities, create the initial OV-5N defining the system functionality, and define the metrics and targets values for determining mission effectiveness later in the process. A clear, concise system level definition is critical for overall success of the effort, as the analysis on the effectiveness to the trade-space decisions is based on the simulation results compared to the system level required performance metrics. Note that NAVAIR used the fit for purpose view referred to as the OV-5N, to show the system level operation view and decomposition, because the initial operation view is closer to a system of systems, than a subcomponent to a system, which is the traditional definition of an OV-5.

a. Overall Model Development Plan

(1) Scope

This architecture to counter all types of unmanned systems provides a government-owned, defined (open) architecture that supports current and future Navy acquisitions efforts. The architecture depicts and describes the capability requirements, operational activities, and the system views. The architecture provides the appropriate functional decomposition, functional interaction, and functional interoperability as appropriate. To enable commonality, the architecture builds on existing capability and functional decomposition and definitions, along with linkages in existing data dictionaries and data definitions.

This architecture is intended for CONUS, OCONUS, and Maritime applications. The system is intended to initially counter Group 1 unmanned aerial systems (UAS); however, this architecture may be expanded to encompass unmanned ground, surface, and underwater systems as necessary.

(2) Viewpoints and Models Developed

Table 3 lists the Department of Defense Architecture Framework (DODAF) version 2.02, August 2010 view required to satisfy the C-UAS purpose and intent. Please note the OV-5N is a fit for purpose view developed by the U.S. Navy to show functionality at the system or system of system level.

Table 3. Viewpoint and Models Developed

Applicable Viewpoints	Models	Titles
All	AV-1	Overall Plan
Capability	CV-2	System Capability Decomposition
Operational	OV-1	Operational View
Operational	OV-5N	System Functionality Decomposition
System	SV-4	System Action Flow

(3) Assumptions

The architectures and associated data can be leveraged and reused by subsequent capability developers and program offices for the development of solutions for C-UxS.

The required capabilities are reasonably covered by the Joint Capability Areas (JCAs), which provide a common vernacular and context. The JCAs are pruneable and extendable to allow for best fit to the C-UxS architecture. The JCAs are used as the bases for the CV-2 decomposition.

The required functionality is reasonably covered by the Joint Common Systems Functional List (JCSFL), which provides a common vernacular and context. The JCSFL

is pruneable and extendable to allow for best fit to the C-UxS architecture. The JCSFLs are used as the bases for the OV-5N decomposition.

(4) Constraints

Developing the Countering Unmanned Systems ICD Architecture thoroughly and expeditiously to support the Counter-UAS Rapid Deployment Capability (RDC) and Rapid Prototype Engineering Development (RPED). The Speed to Fleet Initiative effort was limited to process development, process proof, and initial definition of the generic counter UAS architecture.

(5) Purpose and Perspective

The countering unmanned system architecture depicts and describes the capability requirements, operational activities, and system functionality involved in countering unmanned systems across all domains and operational environments. Because these are an initial set of architectures, they provide a basis for identifying follow-on capability developmental efforts.

b. Define System Level Requirements

The C-UxS system is designed to be flexible in nature, but with a Government-owned core, so vendor specific solutions can be modified without requiring involvement of said vendors. The C-UxS System is initially designed to protect land-based sites against group 1 small COTS UASs, as the example C-UxS. The architecture is flexible enough to allow for growth in support of maritime-based sites (ships) and COTS-based equipment in unmanned surface vessels, underwater vessels, and ground vehicles.

This use case focuses on the trade-space analysis for detect, sensor type, capabilities, numbers, and combinations. This is demonstrated by the capabilities and functionality included with the C-UxS action diagram (SV-4).

The requirements and performance metrics table is shown in Table 4. This is where the system stakeholders can define the constraints for cost and mission

effectiveness. In a real world program, this requirement table would be more detailed, but for this effort, it is very simple to better show cause and effect on the process.

Table 4. Requirements and Performance Metrics

Required Capability	Weight (1-3)	Description of Metric	Metric Target Value
1.0 The C-UxS shall classify threats over an area covering 10 by 20 kilometers	2	Coverage of the sensors by type of covered area divided by the total area	> 90%
2.0 The C-UxS shall have a deployed cost of the sensors be less than \$500,000.00	1	Cost of the material and installed cost of the sensors divided the allowed cost of the sensors	< 90%
3.0 The C-UxS shall defeat 80% of the threats	3	Number of defeated threats divided by the total threats	< 40%

c. Identify Key Stakeholders

The stakeholders for this use case are defined as the funding authority (OPNAV), the acquisition authority (PEO-U&W), the subject matter experts (SMEs), the representative end users (user community), and the test and evaluation team. For this use case, the user community has provided an initial starting point area size that needs to be protected and the required level of protection to allow for continuation of normal operations. The funding authority has provided the deployed cost of the sensors. The SMEs have provided type and realistic capability and cost of the sensors for analysis. The test and evaluation team has verified that the metrics are relevant and collectable.

d. Initial Mission Definition

The C-UxS OV-1 (see Figure 8) describes, at a high level, the operational scenario addressed by this architecture. The desired outcome for this architectural effort is to enable the development and acquisition of a suite of capabilities to prevent and mitigate the adversaries' use of the UxS that capitalizes on both material and non-

material approaches. Due to the diversity of this threat and the intentional domain agnostic approach to the architecture, capabilities resulting from this architecture should inform solutions in other domains and mission sets. This results in a suite of modular (open) capabilities that can be rapidly integrated to keep pace with the rapid growth of UxS technologies.

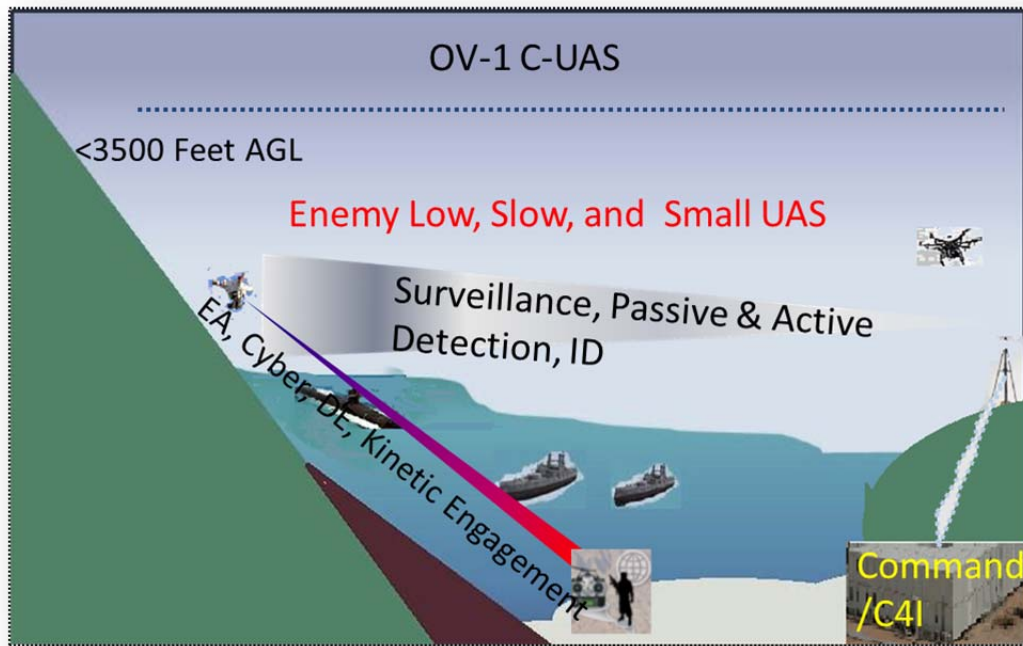


Figure 8. OV-1 C-UAS

To keep this use case generic and unclassified, a simple circular area with an approximate diameter of 16 kilometers is used as the representative area to be protected. The circle has five zones; the outer three zones are a restricted area where small UASs are not allowed and where the sensor coverage starts. The fourth zone is the act and engage zone in which any UAS is considered as a threat to be acted on with appropriate defeat methods. The fifth and final zone is the failure zone, in which any entering threat is considered compromised, and all work must stop. The mission is a success if the required percentage is detected, classified, and defeated outside the failure zone. The mission is considered a failure if the required percentage is not defeated before entering the failure area. A simple diagram of the mission area is shown in Figure 9.

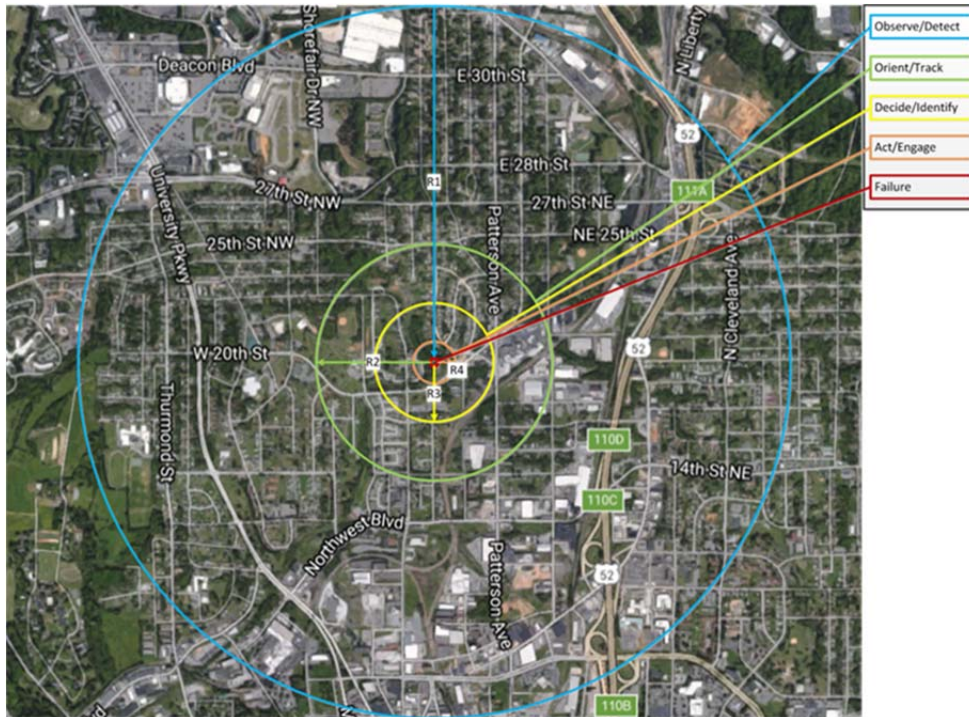


Figure 9. C-UxS Mission Area

e. Create the Initial CV-2 Defining the Systems Capabilities

The CV-2 defines and decomposes the required capabilities at the appropriate level. As stated in the assumptions, the JCAs provide a common vernacular and context across the DOD and are applicable to the C-UxS. JCAs are pruned and extended to allow for the best fit to the C-UxS architecture, and are used as the basis for the CV-2 decomposition shown in Figure 10 initial CV-2 snapshot. The entire CV-2 is too large to be shown in its entirety here, although it is shown in the Appendix, and is available in the associated model. However, the snapshot demonstrates the context in relation to the area relevant to the research. For this phase of the process, the CV-2 containing level 0 is the capability area heading, and level 1 and level 2 provide the additional context in terms of the DOD JCA. The use of the JCA and the additional context allows the stakeholders to verify that the initial context is correct. In this snapshot, the SMEs have selected Battlespace Awareness, Force Application, Logistics, and Command and Control as required high-level capabilities. An additional required high-level capability for the use case, not shown in the snapshot, is the Interact (External) capability. The most relevant

capabilities of Battlespace Awareness, Force Application, Command and Control, and Interact with the threat UxS are decomposed in the next step of the process.

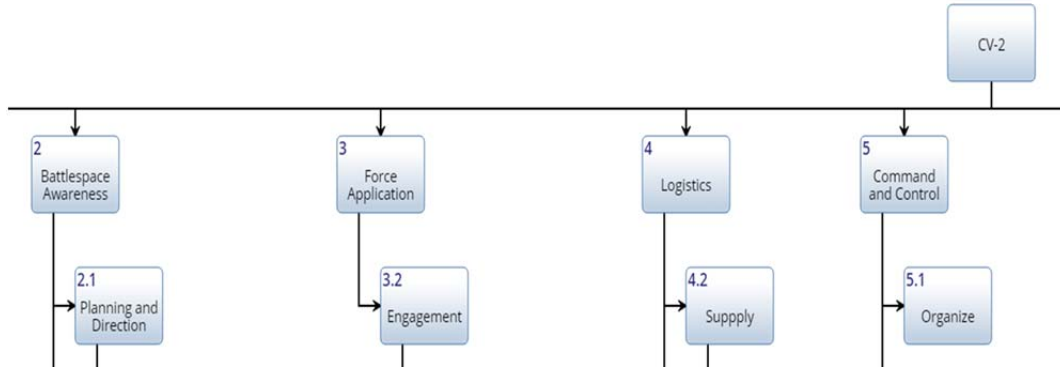


Figure 10. Initial CV-2 Snapshot

f. Create the Initial OV-5N Defining the System Functionality

The OV-5N defines and decomposes the required functionalities at the appropriate level. Like the JCAs, the JCSFLs provide a common vernacular and context across the DOD and are applicable to the C-UxS. JCSFLs are pruned and extended to allow for best fit to the C-UxS architecture. The JCSFLs are used as the basis for the OV-5N decomposition shown in Figure 11. The entire OV-5N is too large to be shown in its entirety here, although it is available in the Appendix and in the associated model, but the snapshot demonstrates the context in relation to this research. For this view, level 0 is the OV-5N heading, and level 1 and level 2 provide the context in terms of the DOD JCSF.

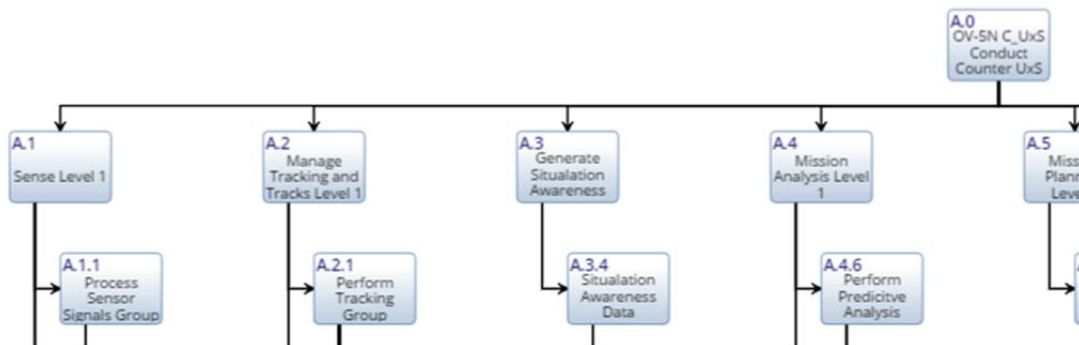


Figure 11. Initial OV-5N Snapshot

In this snap shot, the SMEs have identified Sense, Manage Tracking and Tracks, Generate Situational Awareness, and Mission Level Analysis as required high-level functionality. Additional required functionality in the use case, not shown in the snapshot, is Mission Execution Functionality. The most relevant capabilities of Sense, Manage Tracking and Tracks, Generate Situational Awareness, Mission Level Analysis, and Mission execution are decomposed in the next step of the process.

2. Sub-system Definition

The sub-system definition phase states the problem at the sub-systems level of the C-UxS by decomposing the CV-2 and the OV-5N to the leaf level and defining the appropriate SV-4A C-UxS Flow high-level action views for use case definition. The C-UxS Flow shows the basic flow of the system and the manner in which the AoA simulation capability is integrated. The decomposition, simulation, and analysis of the action view C-UxS flow is the focus of the use case research and is expanded upon below.

a. Decompose CV-2 to Leaf Node Capability

In this phase, the CV-2 had to be decomposed to the leaf level so the action view components could be built with the relevant level of detail. This expanded the CV-2 from two levels to five levels, introducing the additional detail of specific types of sensors such as the radar. The example leaf node shown in Figure 12 is the radar leaf capability that is part of the Battlespace Awareness Capability Group. Additional required leaf node capabilities for this research are Electronic Emissions sensing, Electro-optical sensing, Fuse tracks, Operator interactions, and Defeat, as shown in the Appendix.

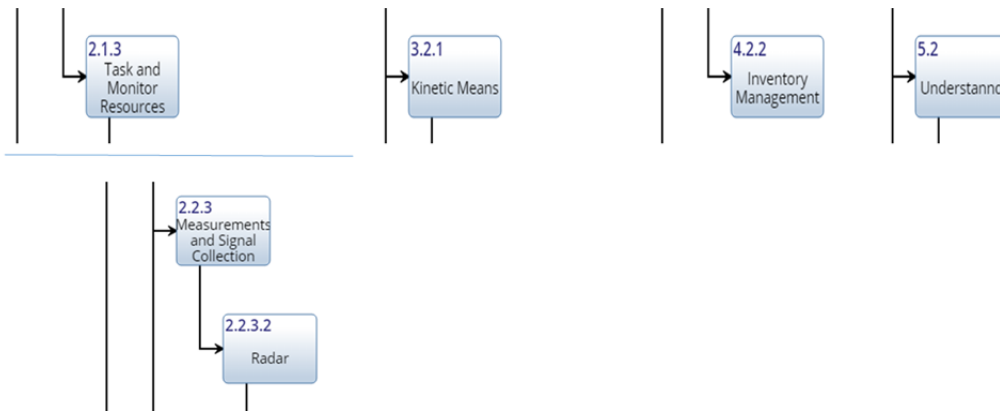


Figure 12. Decomposed CV-2 Snapshot

b. Decompose OV-5N to Leaf Node Functionality

Similarly, the OV-5N had to be decomposed to the leaf level so the action view flow could be built with the relevant level of detail. This expanded the OV-5N from two levels to three levels. The example leaf node shown in Figure 13 is the Search with Active Sensor, Search with Passive Sensor, Fuse Track Measurements, and Evaluate and Assess Engagement. Additional required leaf node functionalities for this research are Conduct Manual Engagement, Conduct Automatic Engagement, and Fly Threat Small UAS, as shown in the Appendix.

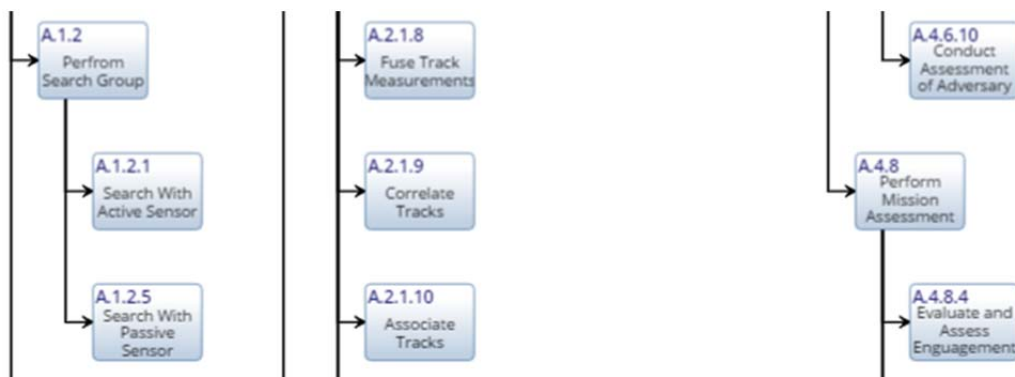


Figure 13. Decomposed OV-5N Snapshot

3. Create and Define the SV-4A Action View

The action view is the first crossover point between the standard MBSE modeling process and the process presented in this paper. The action view allows the static capabilities to be combined in a meaningful way for the C-UxS problem, and then dynamically simulated. For this effort, the goal is to balance cost, coverage, and mission effectiveness based on the top-level requirements and metrics defined above. To do so, the action view defines the interactions between the sensors, fusion, operator, defeat, and the threats. The top-level action flow is shown in Figure 14 and defines the interactions and flow of the functionality across the capabilities.

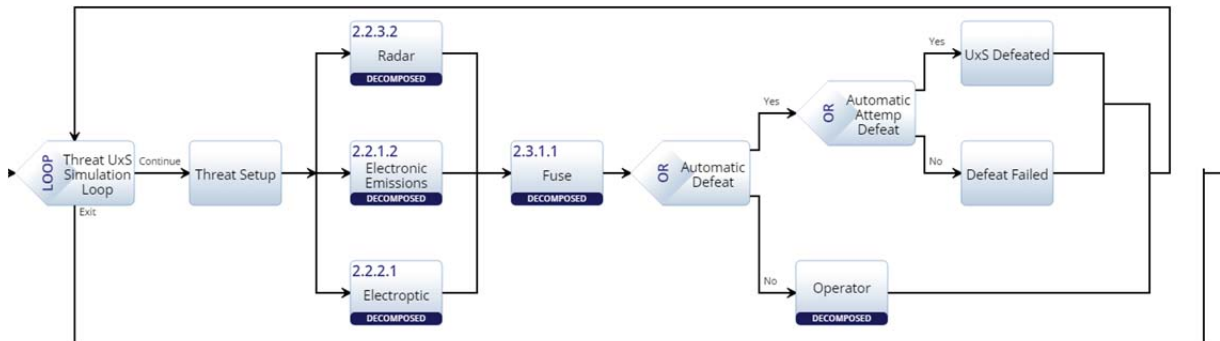


Figure 14. SV-4 Top Level Action Flow

Next, the flow of each top-level action is decomposed to the sub-level flow. For example, Figure 15 shows the radar decomposition. The decomposition allows the number of radars to be varied along with the capabilities of the radar to detect small UASs, and, if detected, to classify the detections as threat small UASs.

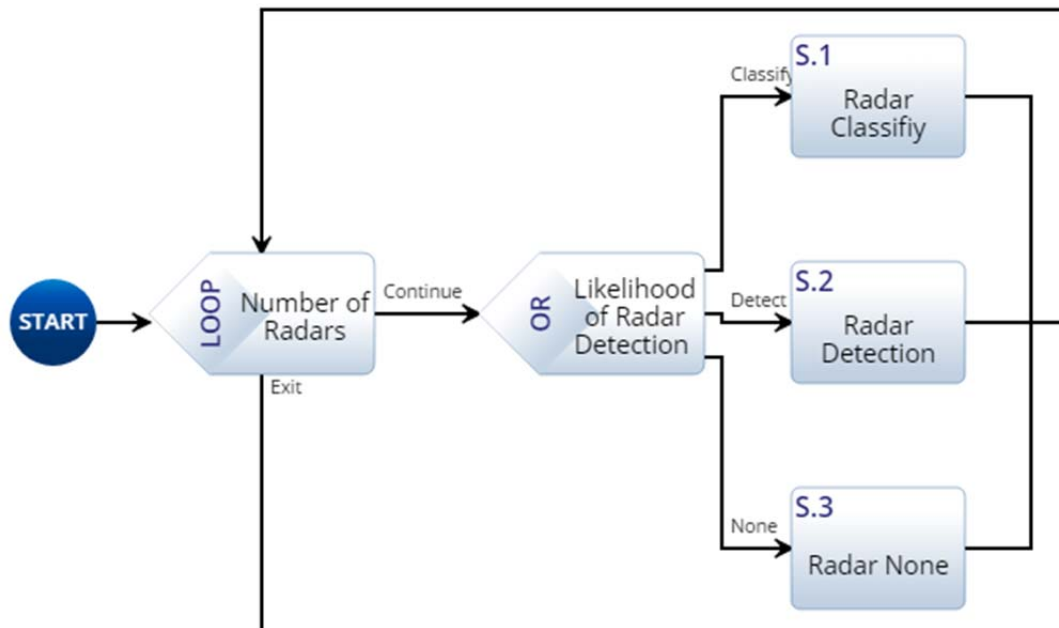


Figure 15. SV-4 Subset Radar Action Flow Decomposition

After radar is decomposed, Electronic Emission, Electro-optics, Fuse, and Operator are decomposed in the mode and shown in the Appendix. The Electronic Emission and Electro-optics decompose in a manner very similar to radar. The Fuse capability takes the output of the sensors and uses rules to command action or to recommend action to the operator, as shown in Figure 16.

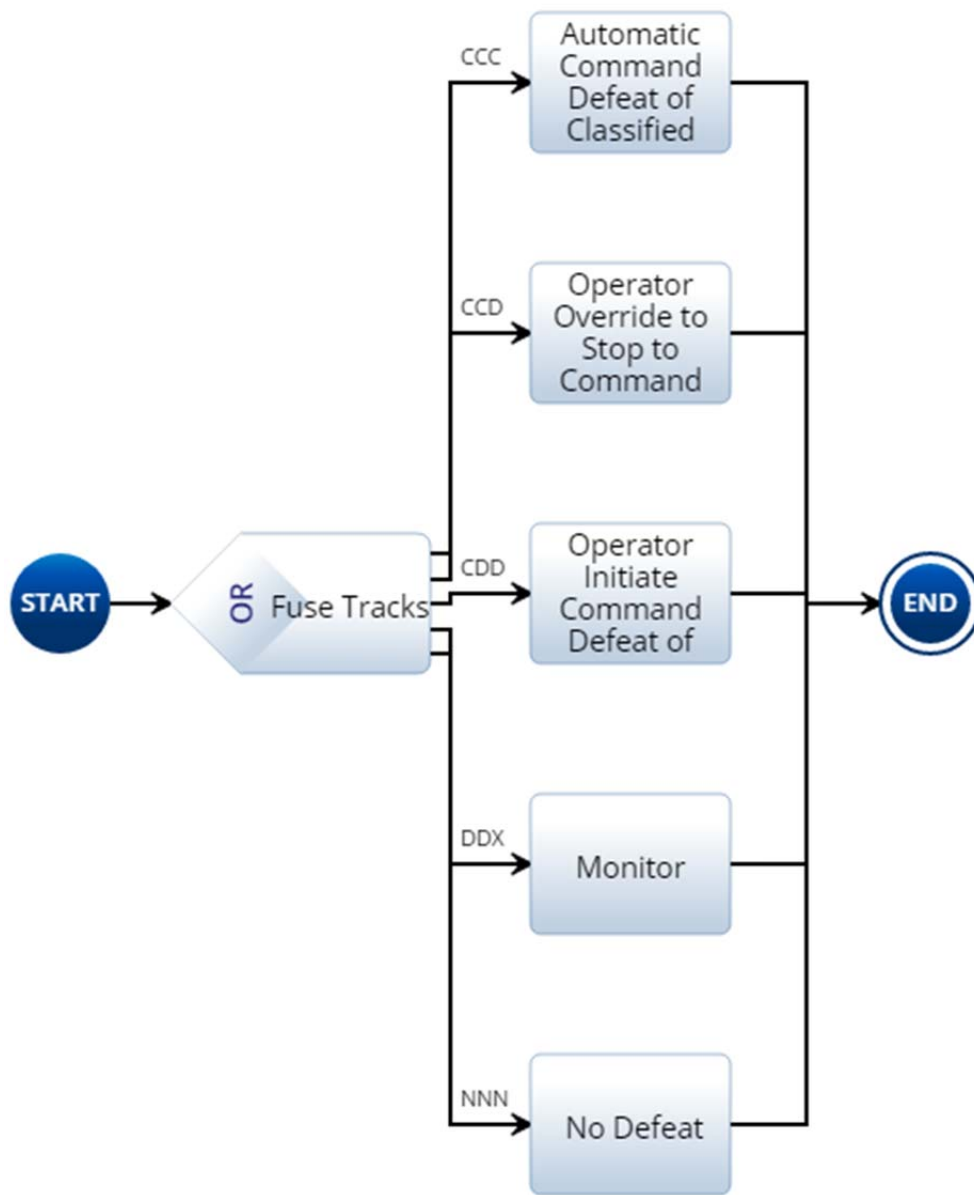


Figure 16. SV-4 Subset Fuse Action Flow Decomposition

The automatic defeat is based on output from Fuse, as show in Figure 14, when a UAS is classified as a threat by all three sensors types. The success of the automatic defeat is based on a simple probability, where there is an 80% chance of defeat based on a timely automatic response and clear fused data from heterogeneous sensors. If the sensor data is not unanimously seen as classified by Fuse, then an operator is brought into

the loop as shown in top level Figure 14 and decomposed in Figure 17. Figure 17 only shows a partial view due to its size, but based on rules it allows four choices by the operator. The full view is shown in the Appendix for additional context.

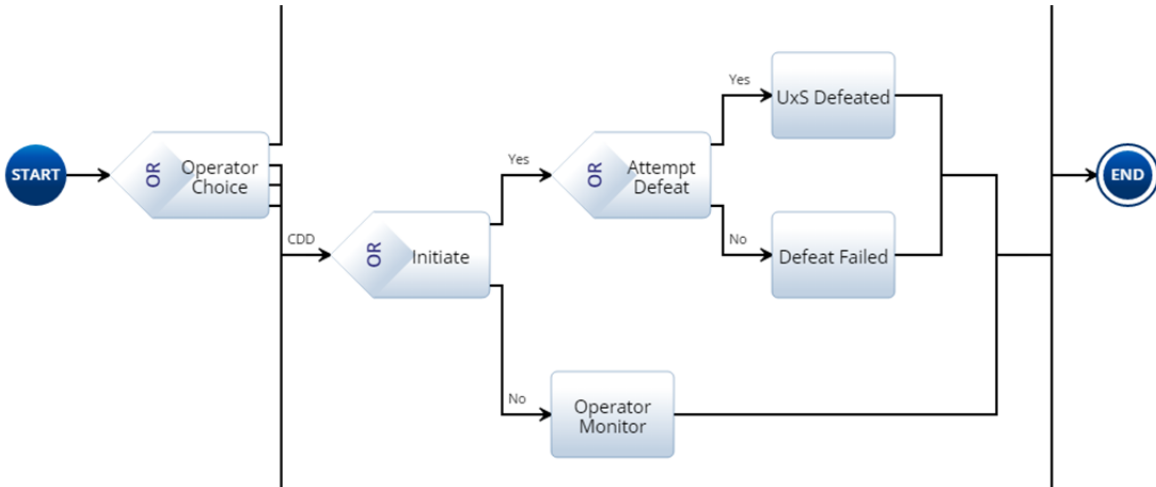


Figure 17. SV-4 Subset Operator Action Flow Decomposition

The first operator choice is operator override, and is based on two sensor types classifying a threat, with the third detecting it. The second operator choice is to initiate a defeat attempt, and is based on one sensor classifying a threat and at least one other sensor detecting it. The third is the operator continuing to monitor detected, but not classified, threats.

4. Mission Analysis

The mission analysis phases take place during the AoA phase (concept development), prototyping phase, and final design phase to show the way by which metric achievement progresses. This first phase demonstrates the concept development phase design (AoA) compared to the metric target values. The user assesses and optimizes different concepts and continues to refine those concepts as more detailed information is available. Based on the metrics in Table 3, the user can vary the sensor type, numbers of sensors, and capability of the sensors. A constraint on the case study is that each sensor type must cover the entire operations area. Cost is a function of

(coverage * capability), such that the higher the coverage and capability, the higher the cost per sensor. A cost weighting value makes radars an order of magnitude more expensive than electronic emission sensor and electro-optic sensors. Rules for the decision process are developed to govern the action flow as defined by the SV-4's action views. These rules remain constant throughout the case study for the purpose of simplicity.

The first step is to set up the simulation to allow for user input of key areas of assessment. The user input blocks are shown in Figure 18, and the user input script is shown in Figure 19.

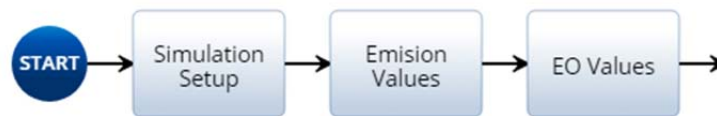
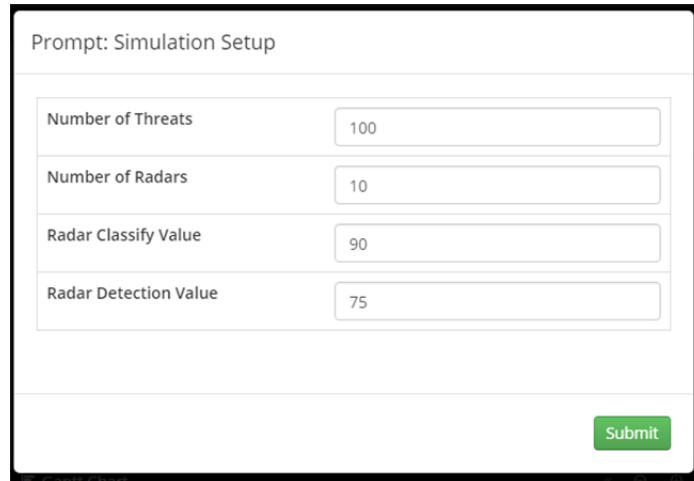


Figure 18. Discrete Event Simulation User input Blocks

```
1 function onStart() {
2
3 }
4
5 function prompt(p) {
6     p.addNumberInput("Number of Threats","Threat_Number");
7     p.addNumberInput("Number of Radars","Radar_Number");
8     p.addNumberInput("Radar Classify Value","Radar_Classify");
9     p.addNumberInput("Radar Detection Value","Radar_Detect");
10
11 }
12
13 function onEnd(response) {
14     globals.put("Threat_Number", parseFloat(response.Threat_Number));
15     globals.put("Radar_Number", parseFloat(response.Radar_Number));
16     globals.put("Radar_Classify", parseFloat(response.Radar_Classify));
17     globals.put("Radar_Detect", parseFloat(response.Radar_Detect));
18 }
```

Figure 19. User Input Script

This allows the user to vary the environment (threats), the sensor types, sensor numbers, and sensor capabilities to classify or detect a threat. Figure 18 shows input blocks for threats, radar parameters, emission values, and Electro-optic (EO) values. Figure 19 show an example for the threat and radar parameter script input. Figure 20 shows the input block during execution of the script.



The image shows a user input form titled "Prompt: Simulation Setup". It contains four input fields with the following values: "Number of Threats" (100), "Number of Radars" (10), "Radar Classify Value" (90), and "Radar Detection Value" (75). A green "Submit" button is located at the bottom right of the form.

Parameter	Value
Number of Threats	100
Number of Radars	10
Radar Classify Value	90
Radar Detection Value	75

Figure 20. Simulation User Input Block

Next, the threat parameters are created by a randomized function in the threat set up block shown in Figure 14, and defined by script shown in Figure 21.

Edit Threat Setup's Script

```
Custom Script
1 function onStart() {
2
3 }
4
5 function onEnd() {
6     var r = Math.random() * 100;
7     globals.put("Random_RCS", r);
8     print(globals.get("Random_RCS"));
9     r = Math.random() * 100;
10    globals.put("Random_EMS", r);
11    print(globals.get("Random_EMS"));
12    r = Math.random() * 100;
13    globals.put("Random_EOS", r);
14    print(globals.get("Random_EOS"));
15 }
16 }
```

Figure 21. Threat Setup Script

The threat signatures are scaled between 0 and 100, to be compared to the capabilities of the sensors as entered by the user and compared in the sensor block. The sensor detection is based on the threat signature (user input), and the distance factor of the threat to the sensor (random) compared to the sensor capacity to detect and classify (user input). The threat is visible to each sensor a single time. The script for radar detection is shown in Figure 23. The scripts for the other two sensor types, emission and EO, are performed in a similar manner.

```

1 function onStart() {
2   print(globals.get("Radar_Classify"));
3   print(globals.get("Radar_Detect"));
4   print(globals.get("Radar_Solution"));
5 }
6
7 function onEnd() {
8   // RCS is constance for each instance of a threat
9   var r = globals.get("Randon_RCS");
10  // Distance between threat and sensors is random. Closer is better
11  var d = Math.random() * 100;
12
13  // increase the detect capabilit based on the distance factor
14  r = r + d;
15
16  // if not classified by another Radar, see if it should be Classified or Detected
17
18  //First Classified
19  if (globals.get("Radar_Solution") == "Radar_Classified") {
20    return 'Classify';
21  } else if (r >= globals.get("Radar_Classify")) {
22    globals.put("RadarSolution", "Radar_Classified");
23    return 'Classify';
24  }
25
26  // Second Detected, only gets here is not already classified
27  if (globals.get("Radar_Solution") == "Radar_Detected") {
28    return 'Detect';
29  } else if (r >= globals.get("Radar_Detect")) {
30    globals.put("RadarSolution", "Radar_Detected");
31    return 'Detect';
32  }
33
34  // Not classified or Detected beore or now
35  globals.put("Radar_Solution", "Radar_None");
36  return 'None';

```

Figure 22. Radar Detection Script

Next, the rules for the fuse capability shown above in Figure 15 are scripted as shown in Figure 23.

```

2 function onEnd() {
3   var amount = globals.get("Radar_Output") + globals.get("EM_Output") + globals.get("EO_Output");
4   if(amount <= 3) {
5     globals.put("Fuse_Output", 1);
6     return '14875626895750004|CCC';
7   } else if(amount <= 7) {
8     globals.put("Fuse_Output", 2);
9     return '14875626895760005|CCD';
10  } else if(amount <= 11) {
11    globals.put("Fuse_Output", 3);
12    return '14976318761890002|CDD';
13  } else if(amount < 30) {
14    globals.put("Fuse_Output", 4);
15    return '14976321191940003|DDX';
16  } else {
17    globals.put("Fuse_Output", 5);
18    return '14984925848980001|NNN';
19  }
20 }

```

Figure 23. Fuse Track's Script

The fuse script roles up the output from the sensors to a single value that effectively recommends an action to the system, such as automatic defeat as show in Figure 14 or passes information onto operator to manage final process as partially shown in Figure 17. The shorthand notation in the script cases of “CCC” indicates that all sensor types classified the threat. “CCD” indicates two of the three sensor types classified the detections as a threat and that all detected the threat. “CDD” indicates that all three sensor types detected the threat, with one classifying it as a threat. “DDX” indicates only two sensor types detected the threat. “NNN” indicates the threat was never detected by any sensor. This is a very simple fusion rule set for this case study, but the fusing capability can be greatly influenced by the trade-space if a more advanced version is used, but that is beyond the scope of this research.

The final step in the action flow is to activate the defeat mechanism as shown in Figure 14 for automatic, or, as shown in Figure 17, for operator choice of override automatic defeat, initiate attempt defeat, operator monitor, or no action, based on the Fuse_Output variable. Whether the UxS is defeated or not is based on a simple probability, where it is biased such that the systems is more likely to be defeated when all sensors classify, and less likely to be defeated when only one sensor type can classify.

The discrete simulation runs are based on the simulation input tables, which varies the input of the type of sensors, sensor capabilities, and number of each sensor type, for the simulation input values. Then each simulation run simulates 1,000 threats and the results are summed by the script and shown in the console window. Results are transferred by hand back to the spreadsheet for further analysis. With the model and scripts set up, the assessments are then completed.

5. Assessment

This case study focuses on the first recursive refinement phase completed in a multi-phase effort to demonstrate the effects of trade-space choices and the simulation refinement process. The first phase demonstrates the team’s knowledge during the AoA phase, and demonstrated the six simulation runs used to refine the design choices. The early work completed during the Rapid Development Capability (RDC) effort discovered

that a homogenous solution does not work, so this effort is starting at the point that they left off with the heterogeneous sensor mix and is refined from that starting point. The screen shot of the model coverage is shown in Figure 24, and the console output is shown on Figure 25.

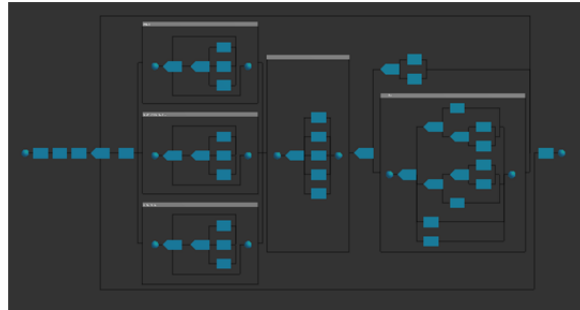


Figure 24. Model Coverage in Simulation

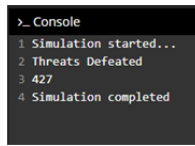


Figure 25. Model Results in Console Window

a. Input

The AoA Phase starts the process with the SME's best understanding of the simulation capabilities. The SMEs devised six different system combinations and types to compare capabilities of higher cost systems to lower cost systems, with coverage and cost kept constant to the threshold value of 95% coverage as shown in Table 2.

Table 5. Initial Mission Values

Run	Radars					
#	#	Classify	Detect	Coverage per	Cost	Total Coverage
1	4	15	10	50	\$374,850	100%
2	3	35	16	65	\$260,852	98%
3	3	30	10	65	\$300,983	98%
4	4	35	15	50	\$270,725	100%
5	4	30	18	50	\$281,260	100%
6	4	32	15	50	\$283,220	100%

Run	EM					
#	#	Classify	Detect	Coverage per	Cost	Total Coverage
1	3	40	30	65	\$102,375	98%
2	10	10	5	20	\$213,750	100%
1	3	40	30	65	\$102,375	98%
4	20	15	10	10	\$191,250	100%
5	12	20	15	16	\$163,200	96%
6	20	15	10	8	\$153,000	80%

Run	EO					
#	#	Classify	Detect	per	Cost	Coverage
1	6	60	20	33	\$28,512	99%
2	6	60	20	33	\$28,512	99%
1	80	30	15	2.5	\$53,550	100%
4	24	60	20	8	\$27,648	96%
5	38	40	20	5	\$41,040	95%
6	38	40	20	8	\$65,664	152%

b. Results

For each of the runs, the input values were entered into the Innoslate discrete event simulation tool, run, and the results captured from the console window and recorded below in Table 3. The use of the “print to console” capability greatly simplified the “what if” process. For this use case, the first three runs did not achieve the desired metric values, so the focus was shifted to a more balanced approach. This shift in focus

led to finding at least three viable concepts for the AoA that can be carried forward, as shown by Runs 4–6, where Run 5 showed the most promise.

Table 6. First Phase Mission Effectiveness Results

Run #	Threats Defeated	Total Effectiveness	Sensor Cost
1	682	68%	\$505,737
2	778	78%	\$503,114
3	685	69%	\$456,908
4	810	81%	\$489,623
5	870	87%	\$485,500
6	864	86%	\$501,884

c. Analysis

For this use case, the cause and effect of changing the different sensor combinations in a static viewpoint prospective are not readily apparent. Though the model looks simple, each part adds significant complexity, with only the rule-based fusion engine being deterministic. In this use case, both the threats and the operator responses are dynamic and non-obvious. For example, early experimental runs, that were thought to have optimal sensor combinations, did not achieve expected results. Subsequent runs benefitted from these early failures to meet the required mission effectiveness, resulting in improved results. This demonstrated that knowledge and understanding gained by the trial and error process provided by the simulation to this relatively simple use case, is a valuable part of system design. This use case only shows the first phase, but one can easily understand that changes to the model, specifically refinement of the fusion engine, can greatly impact the mission effectiveness. For example, a simpler fusion technique may favor a single powerful sensor type, while a more complex fusion technique might allow for less capable sensors. In the analysis of this use case, sensor capability was directly related to system cost. For this effort, Runs 5 and 6 showed the most promising results, which helped us understand the appropriate mix of capabilities for future systems.

6. Recursive Refinement

As mentioned above, some mission effectiveness was driven by the choices made in the fusion engine design and rule set. The next natural step in the refinement process may be to leave the sensor combination the same, but to test different variations of the fusion engine rule set. Alternatively, the next natural step in the refinement process may be to add additional operator aids to assist with the defeat process, or to additionally automate the process so the variability of the operator is completely removed.

7. Exit of Process

The actual exit of the MBSE lifecycle process and this use case is ultimately not until the system is retired, but following the process once through the cycle to build the model, action diagrams, and scripts allowed assessment of the proposed iterative process. One can easily envision the ways in which it will enhance the design process and later phases of the lifecycle.

B. ANALYSIS OF THE PROCESS

Chapter III provided MBSE process with simulation that can be used to assess the solution early in the process as well as throughout the design lifecycle. Chapter IV provided a realistic, though truncated, use case for a Counter Unmanned System using the MBSE process with simulation. The mission effectiveness metric assessed for each sensor combination at the end of each run compared the simulation output to the target values of the metrics. Based on the knowledge gained, the stakeholders and the designers could adjust the sensor parameters for the next run and the target values for the next phase. This allowed for gradual convergence on a viable solution for the stakeholders in the first phase. Though not shown by the use case, the process cycles can account for changes in the capabilities, environment, and threats, all of which have an impact on the overall effectiveness of the solution. For example, the radar capabilities used in the first phase may prove too optimistic after initial testing is complete, therefore the system then does not meet the mission effectiveness metric and may drive cost. Correspondingly, the opposite, advancement in artificial intelligence and image processing, may occur and

make optical sensors much more effective than first thought, greatly increasing mission effectiveness and reducing cost.

This use case intends to demonstrate a process that can be leveraged for all system development efforts. The actual MBSE process with integrated simulation for a large development effort will be much more extensive with all additional capabilities, functionality, and action cases, to include the interactions between multiple system effectiveness metrics target values. Additionally, the ability to use the single source of truth with the built in simulation at any time in the lifecycle or a site-specific instantiation of the C-UxS allows for continuous knowledge of the systems mission effectiveness metric.

IV. CONCLUSIONS/FUTURE WORK

A. SUMMARY OF THE ANALYSIS

The use case analysis shows that neither the static view nor the dynamic analysis initial solution would have met the stakeholders' requirements without the knowledge gained by multiple runs through the solution space via embedded simulation. The interaction between sensors, fusion, and operator seem simple when viewed statically, but the dynamic interactions are shown to be complex for even this simplistic use case. For example, the initial three concept simulation runs did not meet the mission effectiveness requirements, although they all appeared perfectly viable from a static perspective. Through the inner loop of the iterative process, one learned to apply a different focus, one that placed more of a balanced sensor selection approach, which in turn allowed the design to meet the requirement. Without the iterative dynamic analysis throughout the process, the stakeholders may have settled on a non-optimized solution that too strongly favored a radar solution over a more optimal, balanced solution. The key for this research is not the numerical values determined to be the feasible optimized solution, but the fact that iterative simulation added knowledge inside the engineering model. In turn, that knowledge allowed for system optimization inside the MBSE process.

B. CONCLUSIONS

Some say failure should be avoided and ignored, however many innovative ideas, including Thomas Edison's light bulb, are built on a series of trials and errors that did not initially succeed. When asked by a reporter, "How did it feel to fail 1,000 times?" Edison replied, "I didn't fail 1,000 times. The light bulb was an invention with 1,000 steps" (Edison 1890). Edison went on to state that he learned 1,000 things not to do, as part of the development process. This process of trial and error still exists today. The only difference is that the systems are much more complex, so the failures can be much more costly. In the book "Black Box Thinking," Syed (2015) presents the idea that failure is a fact of life, but one can choose to either learn from failure or to pretend that the failure was out of his personal control and not learn from it. Deming, in his breakthrough book

“Out of a Crisis” (1982) also stressed that understanding failure was the key to future success and the foundation of process improvement. For aviation, learning from failure has always been part of the culture, but almost all of these failures occur after the system is designed, and sometimes too late to be cost effectively fixed. This effort moves the power of learning from failure into the early phases of the process, exactly when it is the prime time for “the learn and fix cycle” to take place. The power to learn by failing is even greater than the power of early success, as failure causes one to consider the reasons for the failure. Knowledge is truly gained during this process of understanding the reasons for failure. This thesis therefore provided a MBSE process with integrated simulation that allows engineers the flexibility to test many solutions and fail early, allowing them to succeed in the long term. Multiple designs can be quickly built, including some that have a higher risk to reward, and then performance of each can be evaluated relative to each other. The act of understanding the reasons one solution is more successful than another helps one build a greater understanding of the system. It also allows the designer a better understanding of the sensitivity of a solution to a single technology. The process defined above provides a framework to conduct the definition and design phase of system development using a defined and iterative process built on previous MBSE development research on developing large complex systems. This research and the enhanced MBSE process will contribute to the development of future large, complex systems during system definition and design phases, by providing validation of the system model through simulation and analysis. This benefit is available in all phases of the system lifecycle, as long as the model is maintained as part of the lifecycle process. A general MBSE process with integrated simulation and analysis was shown in Figure 7 and described in Chapter III. A representative Counter Unmanned System use case exercised the proposed process. This allowed validation of the process while a use case provided a clear example of the improved overall product provided by the assessment.

The discoveries of the related research, the enhancement to the current MBSE process, and the example use case addressed many of the primary research questions that were proposed. The objectives of the thesis effort were met with the development of the

MBSE process with embedded simulation and the C-UxS use case. The beneficiaries of the research will be developers of large, complex systems in dynamic environments who are able to use the process as a function of fielding and maintaining the effort. The multiple research questions are discussed below, with corresponding details identified for each, based on the research.

- 1) How can MBSE be used to forecast and investigate mission effectiveness, caused by material and design limitations, to inform and influence the early stages of the system design process? The question is addressed in two parts—the first is the process shown in Figure 7 and described in Chapter III; and the second is the use case, which clearly shows the model development, the action flow, and the scripting to interactively assess mission effectiveness. The MBSE process with embedded simulation provides a powerful framework for complex system development.
- 2) How can multiple runs of the simulation that vary the component level effectiveness be used to determine overall system sensitivity once the architectural model is complete with embedded mission effectivity analysis? The question is addressed with the use case, specifically with the inputs of the six runs shown in Table 2 and the results shown in Table 3. In the use case, the design choices were the sensor mix and the capabilities selected from each sensor type. The variation shows the areas in which the use case was most sensitive. In this case, the unbalanced system performed poorly compared to balanced systems.
- 3) How can the results of the system sensitivity results and analysis be used to optimize design and reliability requirements? This effort leveraged previous work by Perez (2014) where fault analysis was shown to be viable. The use case was based on capability and cost, but an additional dimension of fault/reliability could have been added based on past work and this research effort.
- 4) How can one use sensitivity analysis techniques to adjust the project's path forward by having a continuous positive impact on the early stages of

the development process? The question was not explicitly addressed by the use case, as multiple phases would have been required, but it was addressed by the iterative portion on the MBSE process show in Figure 7. Specifically, the recursive step described in Chapter III, Section 6 takes existing knowledge and simulates it in a representative environment. Following performance assessment, the system is recursively refined based on the knowledge gained from the assessment. The changes clearly show, in a relative manner, if the change had a positive or negative impact on the system performance. The path forward becomes relatively clear - back out a negative change and try something else or continue to refine a positive change. At a time when progress stalls, one may be required to make seemingly random changes, simply to ascertain the pattern of positive and negative effects to determine a new course of action.

C. RECOMMENDATIONS FOR FUTURE WORK

As stated above, only the first phase of a development effort was performed in the C-UxS use case for a single action view. The process used should be expanded to follow a real program through multiple phases and with a broader use of action views to further refine the process. The third research area on using the process to relating reliability requirements was not expanded upon due to the limitation of the use case, but this is an interesting area for further research. Additionally, this research used a single tool set, but multiple tool sets are available that may provide additional insight. A broader investigation is required prior to recommendation of a single tool set.

APPENDIX. C-UXS MODEL

The views created for the C-UxS Model are provided, for completeness, in this Appendix. The entire mode is available in the Innoslate web application on request, Thomas.Moulds@Navy.Mil.

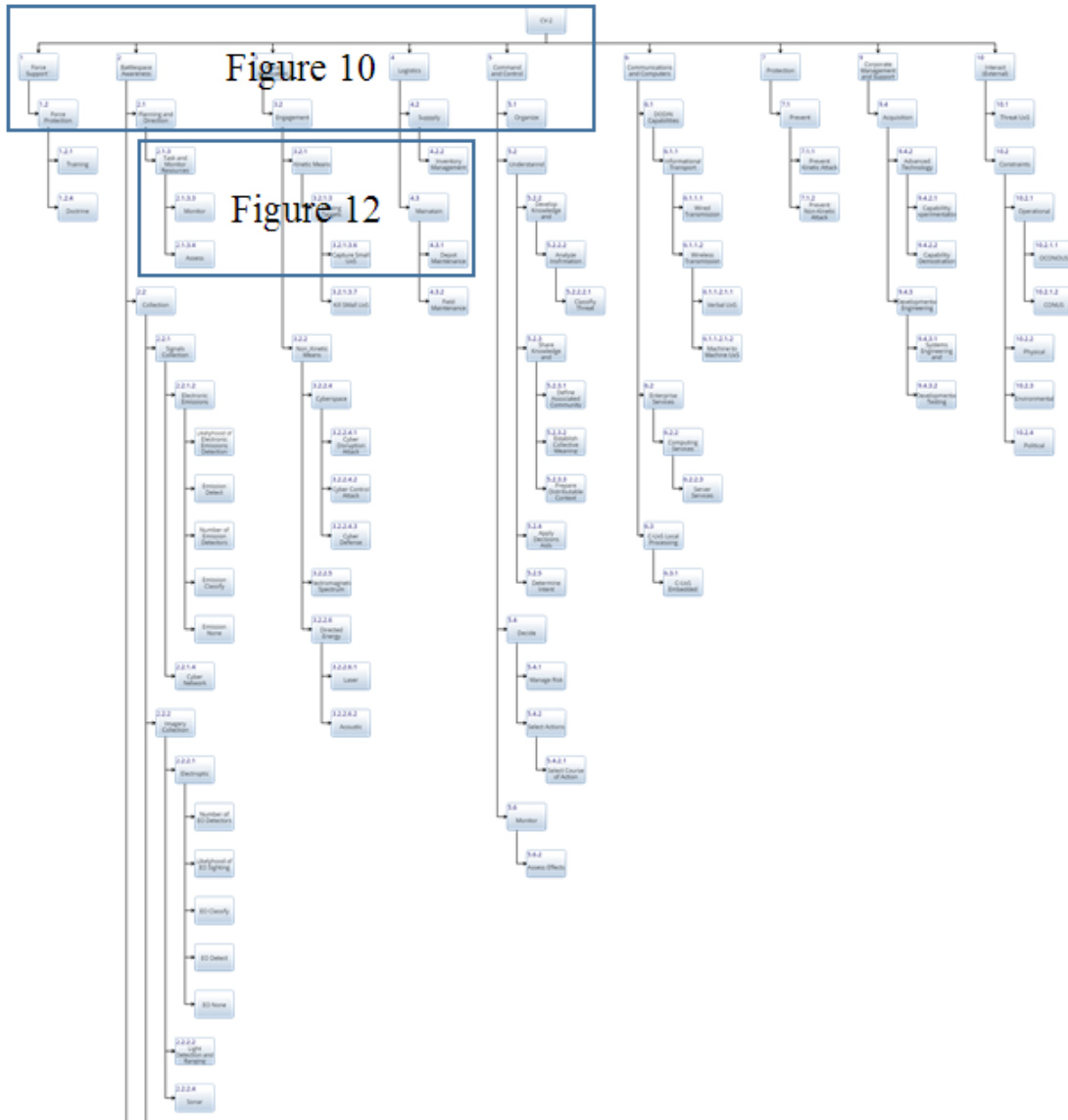


Figure 26. CV-2: JCA-based capability view

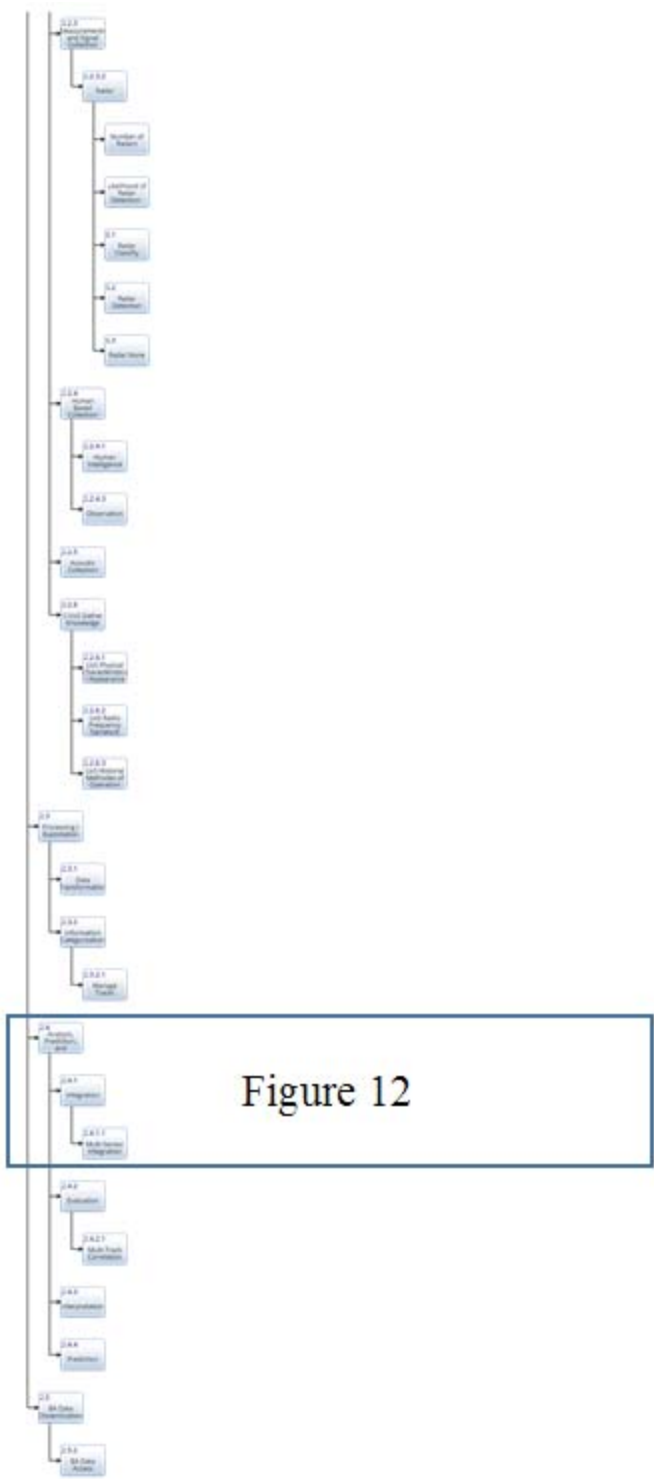


Figure 12

Figure 26 cont'd. CV-2: JCA-based capability view

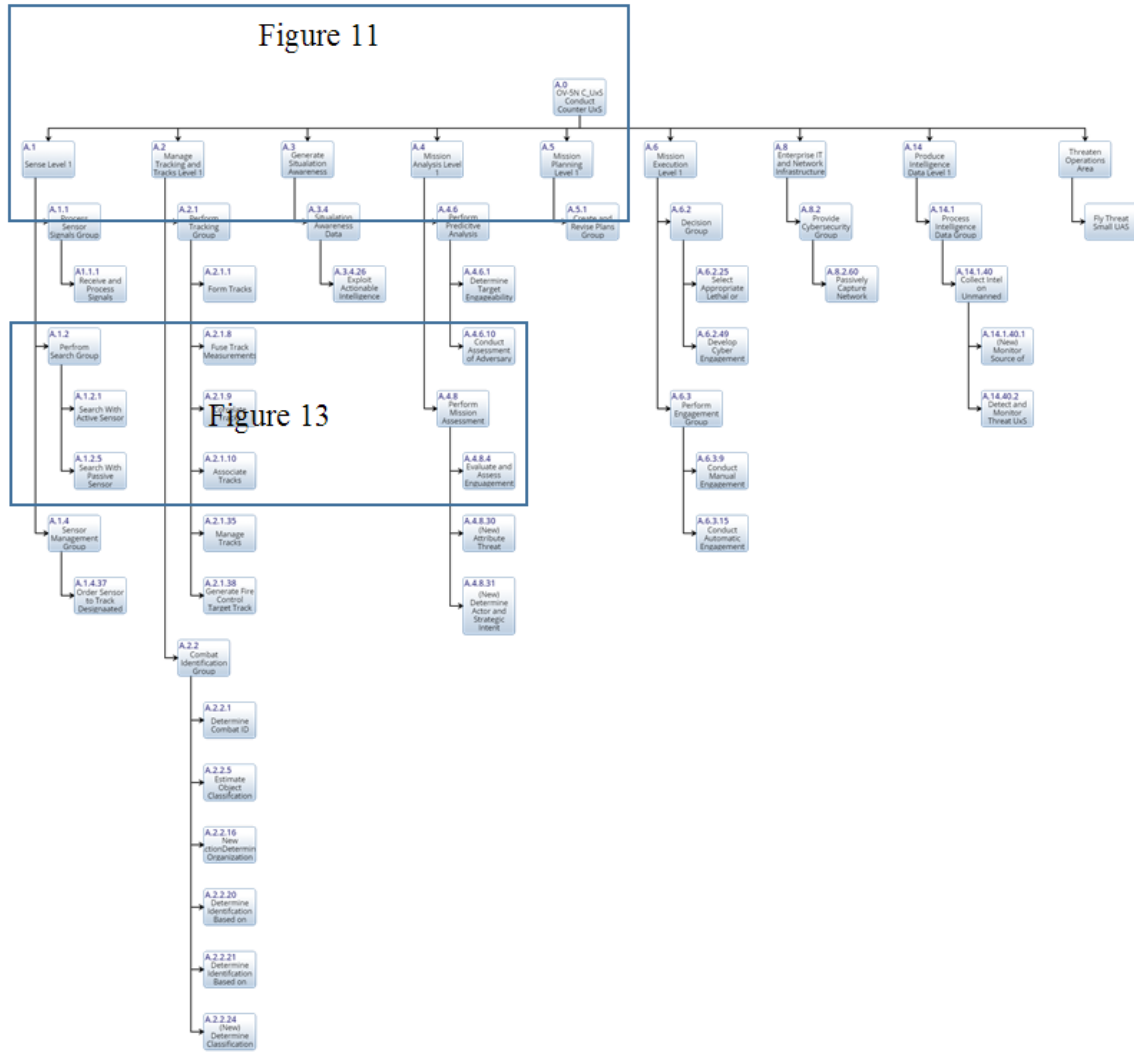


Figure 27. OV-5N: C-UxS Conduct Counter UxS Operations

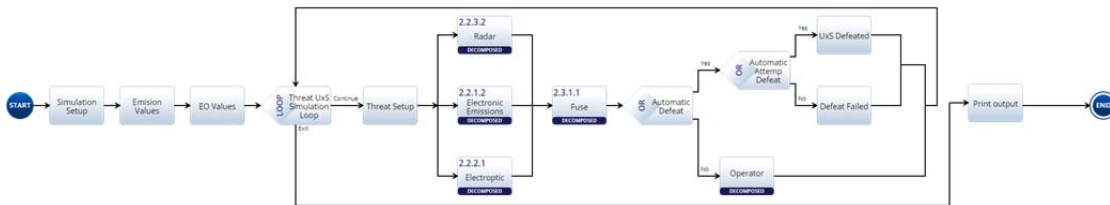


Figure 28. SV-4: Counter UxS Flow (Top Level View)

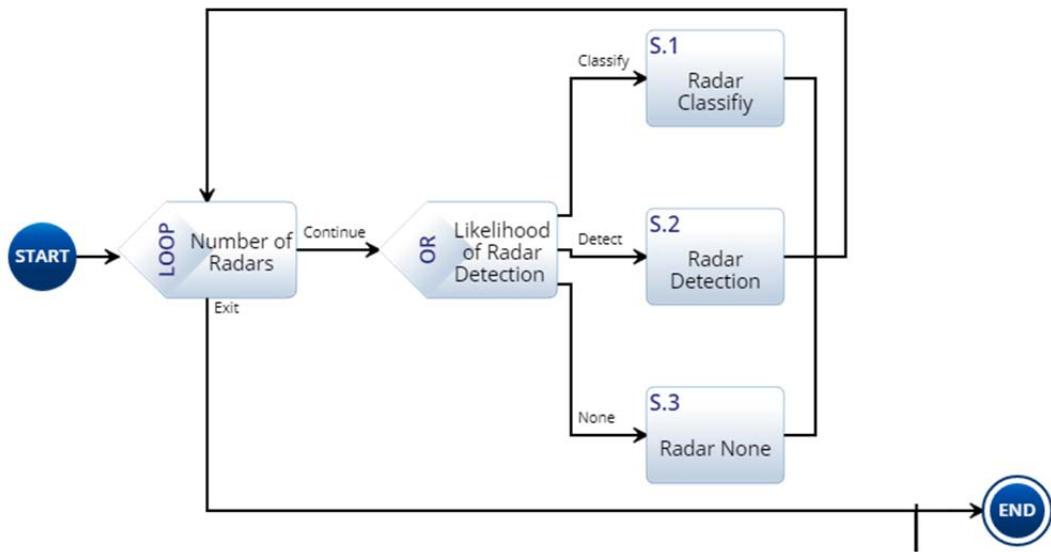


Figure 29. SV-4: Counter UxS Flow (Radar View)

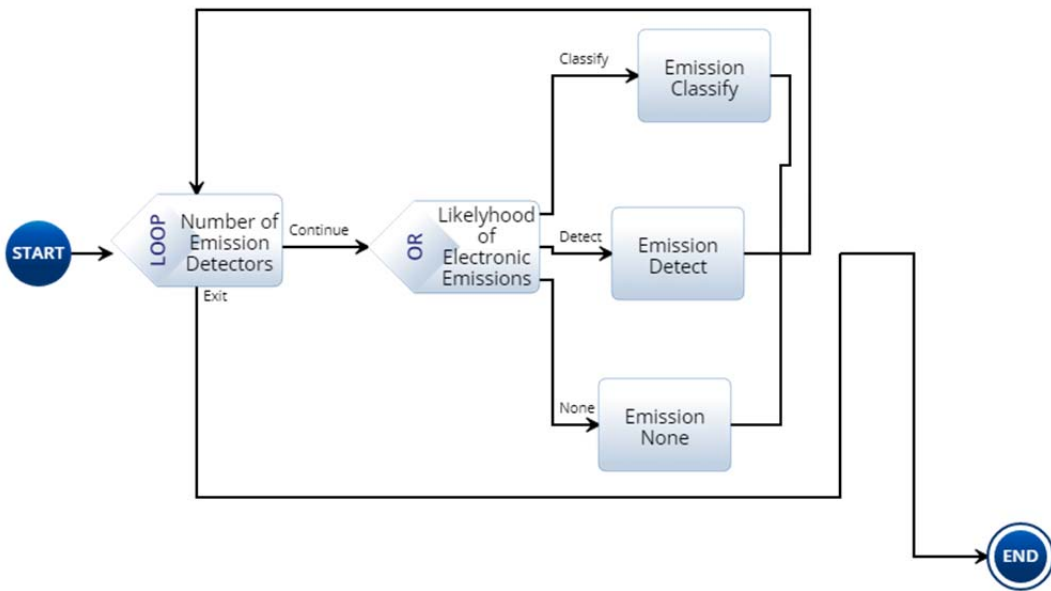


Figure 30. SV-4: Counter UxS Flow (Electronic Emissions View)

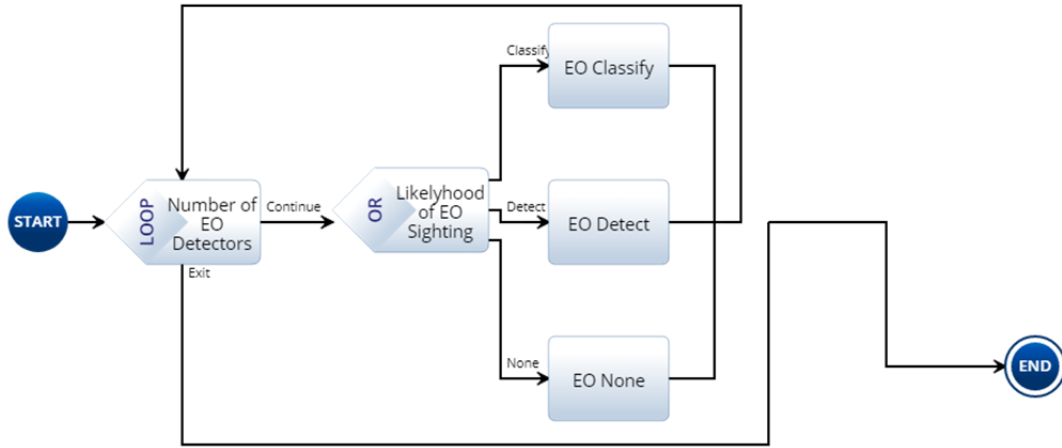


Figure 31. SV-4: Counter UxS Flow (Electro optic View)

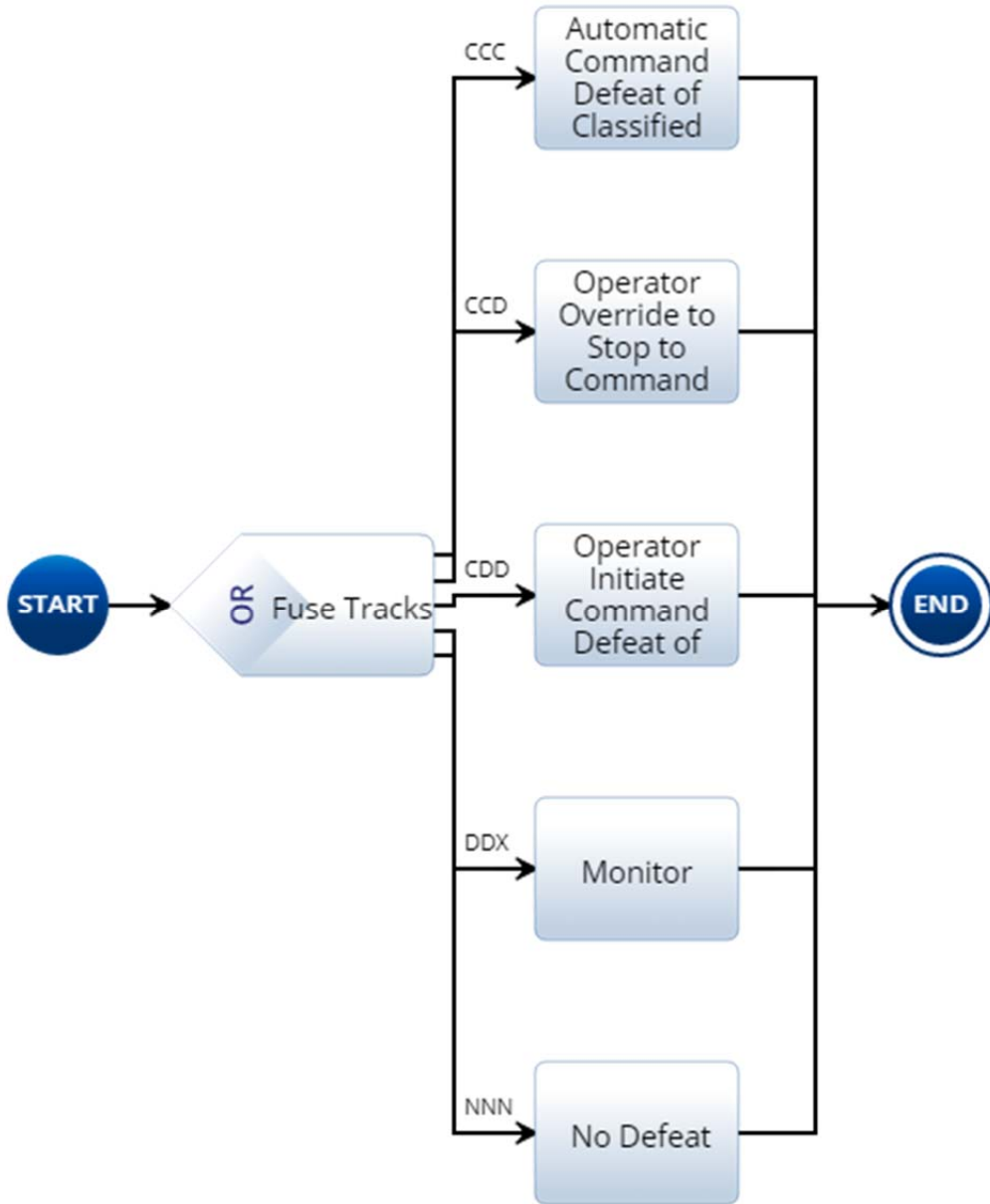


Figure 32. SV-4: Counter UxS Flow (Fuse View)

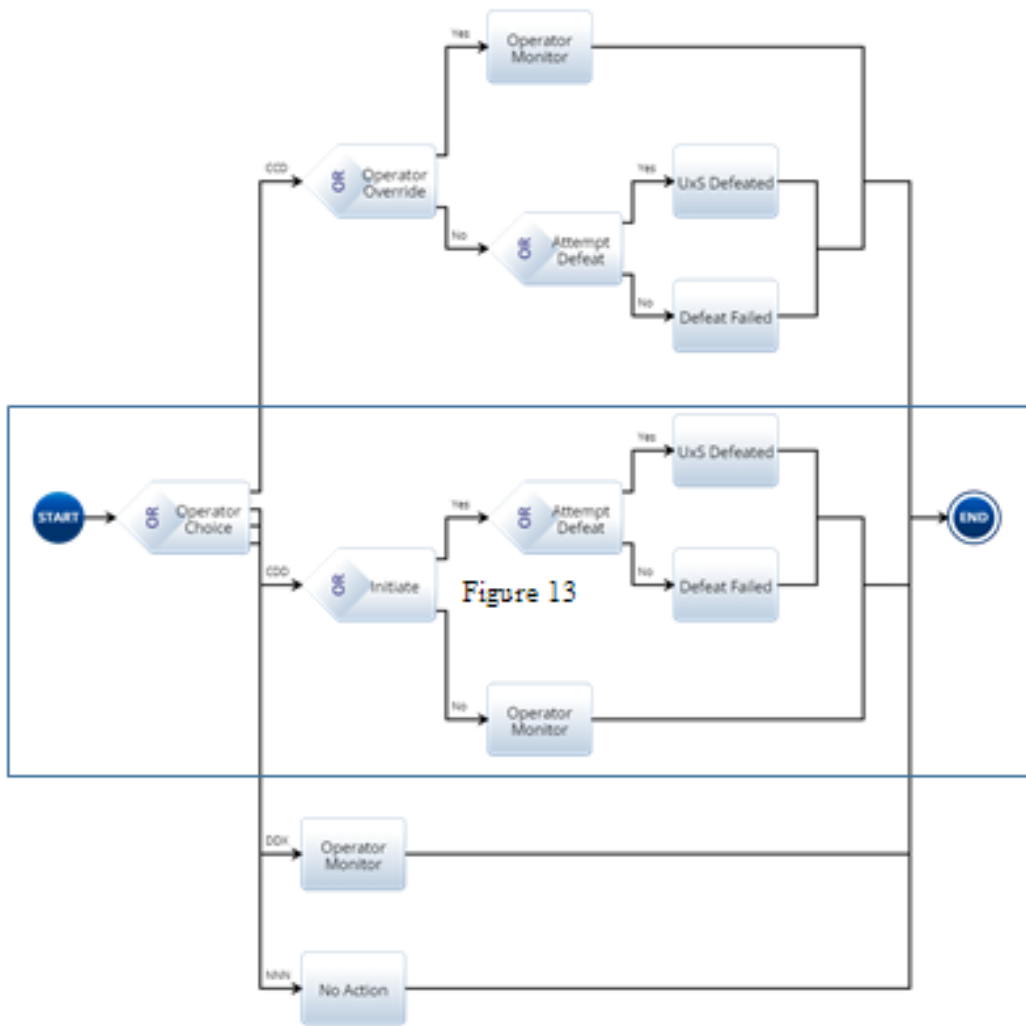


Figure 33. SV-4: Counter UxS Flow (Operator View)

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- Antunes, Ricardo, and Vicente Gonzalez. 2015. A Production Model for Construction: A Theoretical Framework. *Buildings* 5(1): 209–228.
- Bahill, A. Terry, and B. Gissing. 1998. “Re-evaluating Systems Engineering Concepts Using Systems Thinking,” *IEEE Transaction on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 28(4):516–527, 1998.
- Bahill, Terry. 2009. “What is Systems Engineering? A Consensus of the INCOSE Fellows.” Created spring 2001, revised 2004, 2006 and 2009.
<http://www.incose.org/AboutSE/WhatIsSE>
- Deming, Edward W. 1982. *Out of the Crisis*. Cambridge MA: MIT Press.
- Department Acquisition University. 2012. “Defense Acquisition Guidebook.” *Defense Acquisition University/ Tools Catalog/ Defense Acquisition Guidebook*. Accessed 9 November 2016. <https://dag.dau.mil>.
- Edison. 1890. Interview with Thomas Edison, Harper’s Monthly Magazine.
- GAO Report. 2009. “Many Analyses of Alternatives Have Not Provided a Robust Assessment of Weapon System Options.” *GAO Reports*. Accessed 7 April 2016.
www.gao.gov/products/GAO-09-665
- Grealou, Lionel. 2016. “Single Source of Truth vs Single Version of Truth.” *In*. Accessed 3 May 2017. <https://www.linkedin.com/pulse/single-source-truth-vs-version-lionel-grealou>
- INCOSE. 2017 “What is SE”. *INCOSE Publications*. Accessed 5 Jan 2017.
<http://www.incose.org/AboutSE/WhatIsSE>
- Kao, Chiang. 2014. “Network Data Envelopment Analysis: A Review.” *European Journal of Operations Research*, 239(1): 1–16.
- Lifecycle Modeling Organization. 2015. “LML Specification 1.1.” *Lifecycle Modeling Language*. Accessed Jun 2, 2017. <http://www.lifecyclemodeling.org/specification/>
- Maier, Mark W., and Eberhardt Rechtin. 2015. *The Art of System Architecting (3rd edition)*. Boca Raton: CRC PRESS.
- MITRE. 2017. “System Engineering Guide, Analysis of Alternatives,” *MITRE.org*. Accessed 5 June 2017, <https://www.mitre.org/publications/systems-engineering-guide/acquisition-systems-engineering/acquisition-program-planning/performing-analyses-of-alternatives>

- Mun, Johnathan. 2015. *Readings in Certified Quantitative Risk Management* Third edition California, USA. Thompson-Shore, ROV Press, and IIPER Press.
- Object Management Group (OMG). 2012."OMG Systems Modeling Language (OMG SysML)." sysMML Spec. Accessed May 12, 2017. www.omg.org/spec/SysML/20120401/SysML.xml
- . 2017. "What is Systems Modeling Language (OMG SysML.)?" *What is sysML*. Accessed March 2, 2019. <http://www.omgsysml.org/what-is-sysml.htm>
- Perez, Rafael M. 2014, "Application of MBSE to Risk-Informed Design Methods for Space Mission Applications." SPACE Conferences and Exposition.
- Ramos, Ana Louise, Jose Vasconcelos Ferreira, and James Barcelo. 2012. "Model-Based Systems Engineering: An Emerging Approach For Modern Systems," *IEEE Transactions On Systems, Man, and Cybernetics—Part C: Applications and Reviews*, 42(1): 101–111.
- Rumbaugh, J., and I. Jacobson, G. Booch. 1999. *The Unified Modeling Language Reference Manual*. Reading, MA Addison-Wesley Longman.
- Subcommittee NDIA. 2011. "Final Report of the Model-Based Engineering (MBE)," Systems Engineering Division M&S Committee.
- Subcommittee NDIA Systems Engineering Division M&S Committee, 2011. "Final Report of the Model-Based Engineering (MBE)."
- Sussman, J. 2000. *Introduction to transportation systems*. Dedham, MA: Artech house.
- Syed, Matthew. 2015. *Black Box Thinking*. New York, New York: Penguin.
- Tepper, Nadia A. 2010. "Exploring the Use of Model-Based Systems Engineering (MBSE) to Develop Systems Architectures in Naval Ship Design." Cambridge, MA: Massachusetts Institute of Technology.
- Topper, J. S., and N. C. Horner. 2013. "Model-Based Systems Engineering in Support of Complex Systems Development." JOHNS HOPKINS APL TECHNICAL DIGEST, VOLUME 32, NUMBER 1.
- Ulrich Karl T., and Steven D. Eppinger. 2012. *Product Design and Development*. 5th ed.
- United States Army 2016 Counter - Unmanned Aircraft System (C-UAS) Strategy Extract October 5, 2016, [Http://www.arcic.army.mil/App_Documents/Army-CUAS-Strategy.pdf](http://www.arcic.army.mil/App_Documents/Army-CUAS-Strategy.pdf)

Vaneman, Warren K. 2016. Enhancing Model-Based Systems Engineering with the Lifecycle Modeling Language. Proceedings of the 10th Annual IEEE Systems Conference, April 18–21, 2016, Orlando, FL.

———. (draft) 2017a. Model-Based System Engineering De-Mystified.

———. (draft) 2017b. Measuring and Controlling System Efficiency during Transitional Periods.

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