

Model Based Systems Engineering Methods Applied to the Mission Engineering Modeling Use Case



NSWC Crane
300 Hwy 361
Crane, IN

2022

Distribution Statement:	DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.
-------------------------	---

CRANFIELD UNIVERSITY

Jonathan Kidner

**Model Based Systems Engineering Methods Applied to the Mission
Engineering Modeling Use Case**

Cranfield Defence and Security
Expeditionary Warfare Systems Engineering

This thesis is submitted in partial fulfilment of the requirements for the
Masters of Science in Expeditionary Warfare Systems Engineering

Supervisor: David Galvao Wall

RECORD OF REVISIONS

REV	Date of Revision	Description/Reason for Revision (Pages and or paragraphs affected)	Originator's Name (Type/Print)
0	3MAR022	Initial Draft	Jonathan Kidner
1	22AUG22	First External Review	Jonathan Kidner
2	24AUG22	Reviewed Paper	Thomas Raley
3	26AUG22	Reviewed Paper	Stephen Graves
4	2SEP22	Reviewed Paper	John Miller
5	19SEP22	Final Draft and Edits Adjudicated Pushed out for Final Approval	Jonathan Kidner
6	27SEP22	Submitted to PAO	Jonathan Kidner

Table of Contents

Table of Contents	i
Table of Figures	v
Table of Tables	viii
Acronym List	ix
Abstract	xi
1. Introduction	1
1.1. Hypothesis	1
1.2. Aim	1
1.3. Intended Audience	2
1.3.1. Document Structure	2
1.3.1.1. Introduction	2
1.3.1.2. Mission Engineering	2
1.3.1.3. Model Based Systems Engineering Literature Review	2
1.3.1.4. MBSE in support of Mission Engineering	3
1.3.1.5. Example Mission	3
1.4. Background	3
1.4.1. System Complexity	3
1.4.2. Modeling and Simulation in Systems Engineering	4
1.4.3. Model Based Systems Engineering	6
1.4.4. Mission Engineering	8
1.4.5. Mission Level Simulation Framework	8
2. Literature Review	11
2.1. State of Academic Literature Review	11
2.2. Mission Engineering Literature Review	12
2.2.1. History of ME/Wargaming	12
2.2.2. Transition to Mission Engineering	14
2.2.3. Types of Simulation for Mission Engineering - LVC	19
2.2.4. Mission Engineering Use Cases	21

2.2.4.1.	Relation to Testing	22
2.2.4.2.	Analysis of alternatives/Gap analysis	23
2.2.5.	Technical Mission Priorities	30
2.2.6.	Key Performance Parameters (KPPs)	35
2.2.6.1.	Force Protection (1 st KPP)	36
2.2.6.2.	System Survivability (2 nd KPP)	37
2.2.6.3.	Sustainment (3 rd KPP)	37
2.2.6.4.	Energy (4 th KPP)	39
2.2.6.5.	Net-Ready Performance	41
2.2.6.6.	Determining KPPs	43
2.2.7.	Mission Engineering Review Summary	45
2.3.	Model Based Systems Engineering Literature Review	46
2.3.1.	Summary of Systems Engineering	46
2.3.2.	History of MBSE	50
2.3.2.1.	Modeling Language	50
2.3.2.2.	MBSE Language Taxonomy	52
2.3.2.3.	General Terms	62
2.3.3.	Current MBSE Methodologies	65
1.	INCOSE Object-Oriented Systems Engineering Method (OOSEM)	66
2.	Rational Unified Process (RUP) and IBM Rational Telelogic Harmony	71
3.	CORE from Vitech	73
4.	JPL State Analysis	78
5.	System of System Interface Modeling Method	81
6.	Hypermodeling and SAIC Profile	82
7.	Weilkiens SysMod Process	84
2.3.4.	MBSE Methodology Summary	85
2.3.5.	Presentation Views – Architecture Frameworks	85
2.4.	Literature Review Summary	90
3.	MBSE in support of Mission Engineering	92
3.1.	MBSE Related to Mission Engineering	92

3.1.1.	Abstraction Linkage from Systems Engineering to Mission Engineering	93
3.1.2.	Survey Data	94
3.2.	Modeling Methodology Applied to Mission Engineering Use Case	98
3.2.1.	Mission Description	98
3.2.2.	Model structure	99
3.2.2.1.	The Reference Modeling Process	99
3.2.2.2.	Library	99
3.2.2.3.	Stakeholders	100
3.2.2.4.	Top Level Functional Requirements	100
3.2.2.5.	Mission Engineering Level	101
3.2.2.6.	System Engineering Level	102
3.2.2.7.	Distro A Model Template	102
3.2.3.	Model Process	102
3.2.3.1.	Stakeholder Identification and Process Overview	103
3.2.3.2.	Mission Narrative and OV-1 Generation	104
3.2.3.3.	Mission Requirements Capture	106
3.2.3.4.	Mission System Definition - Structure	107
3.2.3.5.	Mission System Definition - Behavior	113
3.2.3.6.	Note on ALH and dot notation	116
3.2.3.7.	Mission Architecture and Parametrics	117
3.2.3.8.	Requirements Satisfaction	124
3.2.3.9.	Simulation Results	126
3.2.3.10.	Simulation Results Analysis	131
3.2.4.	Application of Results	132
3.2.5.	Example Model Summary	133
3.3.	Future Work	134
3.3.1.	Example of Growth	136
3.3.2.	Summary of Future Growth	139
4.	Summary and Conclusions	140
5.	Appendix A: Bibliography	144

6. Appendix B: No Magic Implementation of UAF Viewpoints in Cameo – Reference Guide 154

Table of Figures

Figure 1 Software Growth in Aircraft Civilian (Top) Military (Bottom) [117]	4
Figure 2 US DoD Acquisition Life Cycle [7] (Page 10).....	5
Figure 3 Generic Life Cycle [13]	5
Figure 4 Interconnectivity of a System to Mission Development Team [17]	7
Figure 5 Reported Market Areas for Industry Respondents (Left) Reported Respondent Organizational Roles. (Right)[23]	12
Figure 6 Spectrum of Wargaming Adjudication Methods vs Rigidity Picture by Author	14
Figure 7 Strengths, Weaknesses and Overlaps between Wargaming [25]	15
Figure 8 Orthogonal Axis for Technical Rigor (Author)	16
Figure 9 Three Axes of Mission Engineering [21]	17
Figure 10 M&S Category Hierarchy [119]	18
Figure 11 Mission Engineering Guide Process [21]	24
Figure 12 MEG Metrics Relations (Modified from [21])	25
Figure 13 MEG Process [21] Feeding the JCIDS Process for Net-Ready Systems [42]	26
Figure 14 Pictorial Depiction of the Analysis of Alternatives Process [40]	27
Figure 15 Relation of System and Mission Metrics [21]	29
Figure 16 MEG Process [21] Feeding the JCIDS Process for Net-Ready Systems [42] which informs MBSE [95] [98]	30
Figure 17 System Roles for the Energy KPP [42]	40
Figure 18 Pictorial Representation of the KPP Analysis Process [42]	44
Figure 19 Linear Systems Engineering Process Diagram (SE Vee) [52]	48
Figure 20 Iterative Systems Engineering Process [55]	49
Figure 21 UML-SysML Profile Relations [68]	51
Figure 22 SysML Diagrams and Origins [67]	52
Figure 23 Diagram Interrelations [72]	54
Figure 24 Pictorial Representation of a set of Ports (top), and Item Flow (bottom) [62]	56
Figure 25 Pictorial Representation of a Constraint Block [62]	56
Figure 26 Pictorial Representation Several Types of Actions [62]	57
Figure 27 Pictorial Representation of a General Allocation Relationship [62]	58
Figure 28 Pictorial Representation of a Requirements Diagram with Multiple Types of Requirements [62]	60
Figure 29 Map of Model Elements Interrelations [74]	61
Figure 30 Design Pyramid [76]	64
Figure 31 OOSEM Process Steps (top) [13] OOSEM Process Flow (bottom) [76]	68
Figure 32 Example of Generalization Inheritance [65]	69
Figure 33 Alternate Methods of Inheritance [88]	70
Figure 34 IBM Harmony Systems/Embedded Design Process [89]	71

Table of Figures

Figure 35 Overall Constructive Framework for Systems Engineering [92]	74
Figure 36 System Level Constructive Framework Adapted to Mission Level (Adapted From [92])	75
Figure 37 Vitech CORE 4 Step Process [77]	76
Figure 38 Vitech CORE 4 Domains [95]	77
Figure 39 Vitech C4ISR Model Decomposition Process (Left) [96] and Recursive Layer Process (Right) [79]	77
Figure 40 JPL State Analysis Structure [98]	78
Figure 41 JPL Information Flow Structure [98]	80
Figure 42 History of the Interrelated Architecture Frameworks [108]	86
Figure 43 UAF Viewpoint Matrix for v 1.2 [110], [111]	87
Figure 44 DoDAF Viewpoint Mapping to UAF 1.2 [111], [112]	88
Figure 45 UAF 1.2 Metamodel [111]	89
Figure 46 Systems Complexity Axis Continuum.....	93
Figure 47 Mission Engineering, Systems Engineering Overlap [115]	95
Figure 48 SERC Overlap of Mission Engineering and Systems Engineering [115]	96
Figure 49 Comparison between Systems and Systems of Systems as Applied to Systems Engineering [115]	97
Figure 50 Top Level Model Structure.....	99
Figure 51 Model Library Package Breakout	100
Figure 52 Stakeholder Package Breakout	100
Figure 53 Top Level Functional Requirements Breakout.....	101
Figure 54 Mission Engineering Level Package Breakout.....	101
Figure 55 System Engineering Level Breakout.....	102
Figure 56 Mission Engineering Process.....	103
Figure 57 Generic Mission Engineering Use Case.....	103
Figure 58 IAD Mission OV-1	104
Figure 59 Mission Message Sequence	105
Figure 60 Requirements Captured in Model	106
Figure 61 Mission Structure bdd.....	108
Figure 62 Surrogate Block Creation and Traceability	110
Figure 63 Surrogate Block Definition with Additional Simulation Information.....	111
Figure 64 Surrogate Block Value Property Traceability	112
Figure 65 C2 Behavior State Machine (Top Left), Top Level Activity (Top Right), Message Reception (Bottom Left), and Track Storage Management (Bottom Right).....	114
Figure 66 Threat Aircraft State Machine	115
Figure 67 Search Radar State Machine (Top Left) and Behavior (Bottom Left and Above)	116
Figure 68 Self Referential Association Interface	118
Figure 69 Mission ibd	119

Table of Figures

Figure 70 Mission Level Parametric Diagram	120
Figure 71 Connector Parametric Diagram	121
Figure 72 Mission Surrogate Black Box Related to Functional Roles (Top) Physical Form (Bottom)	123
Figure 73 Completed IAD Mission bdd	125
Figure 74 Initial Solving for IAD Mission Simulation	126
Figure 75 IAD Mission Simulation Configuration Diagram	127
Figure 76 Power Received by Search Radar of Mission for IAD Mission Simulation.....	128
Figure 77 Threat Distance to Target and Engagement Range for IAD Mission Simulation	128
Figure 78 Power Received by Track Radar for IAD Mission Simulation.....	129
Figure 79 State Summary for IAD Mission Simulation.....	130
Figure 80 Backlog of Track Data in C2 Node (Left) Process Track Activity with Processing Delay (Right).....	132
Figure 81 Multiple Surrogate Blocks of the Same System for Different Customers	135
Figure 82 Send Behavior on the Sending System	136
Figure 83 Improved Communications Behavior	137
Figure 84 Improved Communications Model Structure	137
Figure 85 Dashboard View of an Association Block Interaction	138

Table of Tables

Table 1 Comparative Strengths vs Weaknesses for Live Virtual Constructive Test Methods (Aggregated [33]–[35])	20
Table 2 JCIDS, ME, T&E Objective Statement Comparison.....	21
Table 3 JCIDS vs MEG Summary Comparison	28
Table 4 US NDS, DARPA SBIR, and OUSD (R&E) Technical Area Priorities	31
Table 5 Total System Inventory for Single Location Management.....	38
Table 6 Net Ready Abstraction to Measure Linkage Table [42]	42
Table 7 Elements of Definition Mapping to Elements of Usage [75]	62
Table 8 RUP SE Architecture Model Level and Purpose [79].....	72
Table 9 Summary of Reviewed MBSE Methodologies and Key Takeaways	85

Acronym List

act	Activity Diagram
bdd	Block Definition Diagram
CBRN	Chemical Biological Radiological Nuclear
CONOPS	Concept of Operations
DIS	Distributed Interactive Simulation
DoD	Department of Defense
DoDAF	DoD Architecture Framework
F2T2EA	Find-Fix-Track-Target-Engage-Assess
GCRA	Government Capability Reference Architecture
GMRA	Government Mission Reference Architecture
HLA	High Level Architecture
ibd	Internal Block Diagram
INCOSE	International Council on Systems Engineering
JCIDS	Joint Capabilities Integration and Development System
JPL (NASA)	Jet Propulsion Laboratory
KPP	Key Performance Parameter
KPP	Key Performance Parameter
LVC	Live Virtual Constructive
MATREX	Modeling Architecture for Technology, Research, and Experimentation
MDAO	Multi-Domain Analysis and Optimization
ME	Mission Engineering
MEG	OSDs Mission Engineering Guide
MoD	Ministry of Defense
MoDAF	Ministry of Defense Architecture Framework
MoP	Measure of Performance
NAF	NATO Architecture Framework
NASA	National Aeronautics and Space Administration

Acronym List

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

NDIA	National Defense Industrial Association
NILE	Naval Integrated LVC Environment
OMG	Object Modeling Group
OOSEM	Object Oriented System Modeling
OSD	Office of the Secretary of Defense
par	Parametric Diagram
SE	Systems Engineering
seq	Sequence Diagram
stm	State Machine Diagram
SysML	Systems Modeling Language
T&E	Testing and Evaluation
UAF	Unified Architecture Framework
UML	Unified Modeling Language
UPDM	Unified Profile for DoDAF and MoDAF
USD(R&E)	Under Secretary of Defense for Research and Engineering

Acronym List

Abstract

This report examines the possibility of applying methodological advancements and digital process improvements within Model Based Systems Engineering to the Mission Engineering use case. This premise is grounded in the hypothesis that Systems Engineering, through evolution into more complex System of Systems Engineering, has approached close enough to Mission Engineering to be technically linked. If true, then it is now possible to create a digital thread from initial wargames, through the Mission Engineering Guides and Joint Capabilities Integration and Development System processes, into requirements able to be digested and built upon in a SysML based Systems Engineering Model. This would help improve the traceability and accuracy of top down requirements, as well as facilitate bottom up component trade studies at a mission level, as part of system design and development.

The research and analysis to determine the veracity of this hypothesis consists of two literature reviews. The first literature review regards Mission Engineering drivers and guiding documents, deriving from them the requirements any useful model must satisfy. The second literature review examines current Model Based Systems Engineering methods, scraping the literature for those applicable to Mission Engineering as opposed to those simply supporting engineering level tools. The information garnered in the second literature review provides the method that the requirements identified in the first review may be satisfied.

Once the literature reviews are completed, the two topics are formally linked through analysis of each's aims, objectives, and tasking. At this stage, it is apparent that there is significant similarity of intent and requirements between the two topics, substantiating that advances in MBSE could be applicable to the Mission Engineering use case. As further proof of concept, the identified methodology is applied to an example Integrated Air Defense mission. Through this application it is further verified that MBSE methodologies, processes, and tools, can conduct Mission Engineering with two caveats; an exterior performance simulation framework is necessary, and any MBSE methodology, particularly for Mission Engineering, is an evolving entity that is not suitable for static standardization at this time.

Through the literature reviews, synthesis, and proof of concept application, it is determined with high confidence that advances in Model Based Systems Engineering can benefit the Mission Engineering use case as a subset of System of Systems Engineering. This conclusion provides a much greater breadth of development and information for Mission Engineers to draw from when optimizing and developing their capability in support of the warfighter.

Abstract

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

Keywords – Mission Engineering, Model Based Systems Engineering, Capability Gap Assessment, Modeling and Simulation, Systems Modeling Language (SysML), System of Systems Engineering, Wargaming

Abstract

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

1.Introduction

“All Models are wrong, but some are useful”

-George Box

1.1. Hypothesis

This paper posits that the disciplines of Mission Engineering and Systems Engineering are fundamentally similar enough that Mission Engineering can leverage advances in Model Based Systems Engineering (MBSE) to develop a Model Based Mission Engineering methodology. The model produced by this methodology will be able to support physics-based mission analysis tools, while maintaining traceability to the authoritative source of truth for the systems composing the mission.

1.2. Aim

This paper identifies requirements and best practices from system modeling and simulation literature for use at the mission level, from a Mission Engineering perspective. The information gleaned through this literature review will be used to build a MBSE SysML model in Cameo Systems Modeler and integrate it with MATLAB as a surrogate mission engineering simulation tool. Such a model provides a traceable linkage from department level publications, priorities, and doctrine, to define a mission engineering environment. This definition includes required areas of simulation to produce system requirements, and the top layer of a model-based systems engineering schema to interface with MBSE models being used in material solution development. The path will represent an engineering view of the traditional capability gap identification and solution process, enabled through the adoption of digital engineering and MBSE at the engineering level.

While inclusion of mission considerations as part of a system design cycle is not novel, those considerations are fundamentally viewed from a Systems Engineering perspective. The approach here is to view System Engineering tools and methodologies from the perspective of Mission Engineering, in order to create a mission centric approach without the inherent priorities of straight adoption of a system modeling approach.

This will take place within the confines of Distribution Statement A using academic resources, and intentionally gives wide berth to that information. As such, specific solutions or comparisons of simulation tools will be avoided as no meaningful discussion can take place without restricted information. The literature review will be limited to identifying considerations that should inform an activities tool selection. This is in an effort to make the models used for decision making less wrong and more useful.

1.3. Intended Audience

This paper is geared primarily towards informing newly minted Mission Engineers on the basics of the discipline. There are excerpts intended to be useful to more experienced Mission Engineers responsible for ontological and schema development within their organizations as well.

1.3.1. Document Structure

There are four major sections to this document; an introduction setting the stage for the use of Mission Engineering and MBSE, a literature review covering Mission Engineering, a literature review covering the evolution and current state of the art MBSE methods, and an example mission demonstrating the hypothesis that Systems Engineering, through evolution into more complex System of Systems Engineering, has approached close enough to Mission Engineering to be technically linked. Each section (to a depth of X.X as long as that section contains further sub headings) will include a summary at the beginning giving a few sentences about the contents of that section.

The summary at the head of sections and chapters will take this form. The chapter will resume below the bar.

1.3.1.1. Introduction

The first portion of the paper will briefly introduce the concepts, impetus, and background to Mission Engineering and MBSE and why these approaches are supplanting older methods.

1.3.1.2. Mission Engineering

This section introduces Mission Engineering as a concept, describing its genesis, relation to wargaming, how it supports capability gap analysis and requirements generation, and the common key performance parameters (KPPs) to be considered for modeling. These KPPs will form the foundational interface between Mission Engineering and MBSE, as they are mission level requirements from which platform and system level requirements can be derived from.

1.3.1.3. Model Based Systems Engineering Literature Review

This section includes literature review, beginning with the introduction of Systems Engineering, the evolution of MBSE, a review of the Systems Modeling Language (SysML) used, and the highlights of a literature review. From a much more extensive literature review, seven of the most pertinent MBSE methods for Mission Engineering are identified and key takeaways are derived. These takeaways are the components that will be synthesized in Chapter 4.

Intended Audience
Introduction

1.3.1.4. MBSE in support of Mission Engineering

This section takes the foundation laid in the previous two chapters and formalizes the link between MBSE and Mission Engineering that validates the application of MBSE developments to the Mission Engineering domain. The link is formalized by comparing the processes for Mission Engineering and MBSE supported by survey results provided by SERC. The chapter then walks through the structure, behavior, simulation, and trade study of an example model using the methodology identified in section 2.3. This section is a proof of concept that the methodology derived from advancements in MBSE is suitable for Mission Engineering, with the caveat that there is not currently a universally applicable standard within practicable reach.

1.3.1.5. Example Mission

This example mission and associated model demonstrate the utility of the highlights of the literature review conducted in previous sections and also shows how they can be implemented. This includes the requisite connective tissue that is critical to any executable model but is often left out of current literature. For this paper, the example mission statement is an Integrated Air Defense (IAD) system deployed to an austere location to protect high value assets in the rear of the operational zone. This system includes a pair of networked sensors offset from the command node to improve survivability, communicating over a wireless network. There is an inbound hostile aircraft that must be detected by the search radar and the tracking radar prior to entering its weapons engagement zone of 200km. Because the base is austere, fuel consumption and associated logistics are a key concern, limiting the fuel consumption to 110kg of fuel per hour. As multiple of these sites are required to provide sufficient coverage, the operational cost shall not exceed \$2000 per day. The goal is to determine if the systems specified can conduct the mission within the given constraints. All information and parametrics are pulled from Distribution A sources for the purposes of this paper, an analysis in practice should leverage existing tools as part of the simulation framework.

1.4. Background

1.4.1. System Complexity

The first aspect to address is the rationale of using Modeling and Simulation (M&S). Traditional design has involved prototypes and live testing, to that process, effective M&S is a relatively new addition. However, there is a key driver to changing that traditional paradigm and increasing the role M&S plays. That driver is complexity; more complex systems are now operating in a more complex environment. A cursory review on the Government Accountability Office shows that, after adjustments for inflation, the F-15 cost \$3,078 million to develop in 1972. By the time the F-22 development costs were estimated in 2000, they were standing at \$18,880 million, to be increased to \$20,400 million as of the same year [1], [2]. The F-35 is

estimated to beat out both of them, though it is too early to tell by how much, as while the F-35 has entered operational capability within the navy in 2019, development continues and costs grows [3], [4]. However, this is in large part the result of an exponential increase in system complexity in both civilian and military air platforms, as shown in Figure 1.

While this trend does include the increasing reliance on computers as a facet of modern life, it none the less increases the complexity. The above, therefore, supports the case that systems are getting more expensive as a result of increasing complexity, and the need for a paradigm shift as a result.

1.4.2. Modeling and Simulation in Systems Engineering

This paradigm shift comes in the form of increased use of M&S, though M&S only really becomes effective when part of a digital design cycle discussed later. Although there is ample rationale for the use of M&S in the design and acquisition life cycle, it is not a perfect solution. Recent trends in engineering analysis have highlighted both the utility, shortcomings, and future challenges of modelling and simulation (M&S) for the US Department of Defense (DoD)[5], [6]. Fortuitously, many of the shortcomings of currently employed M&S can be

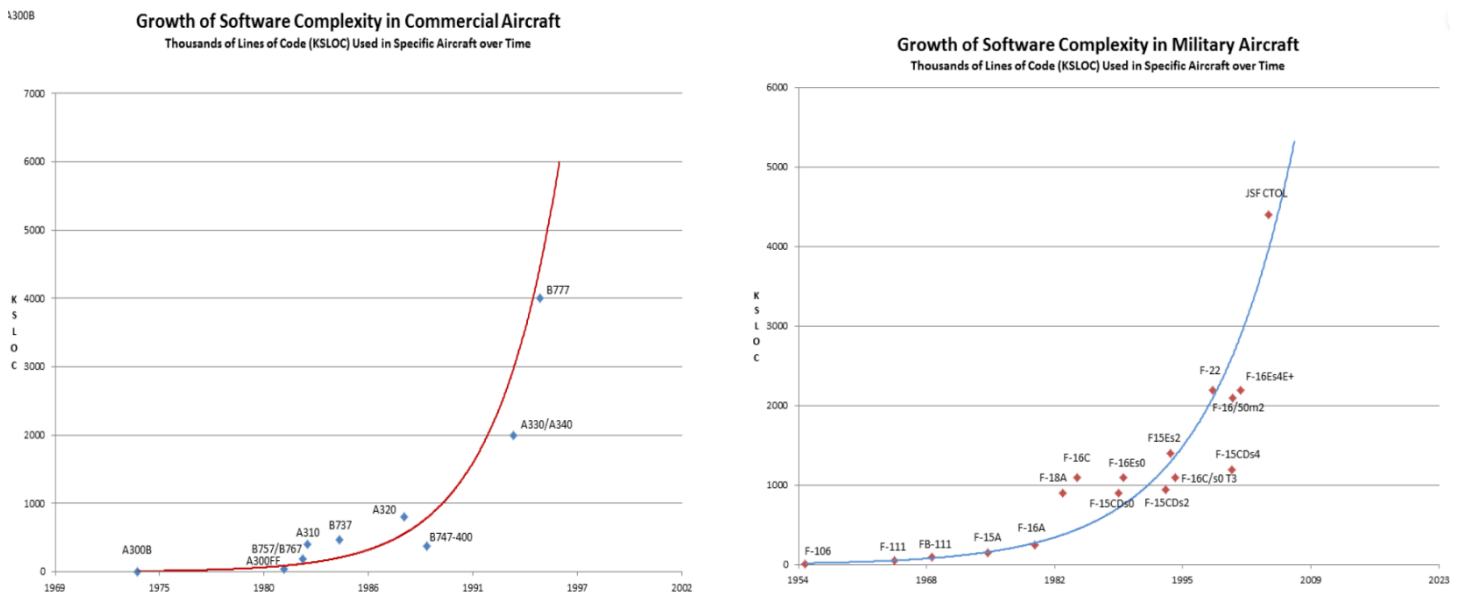


Figure 1 Software Growth in Aircraft Civilian (Top) Military (Bottom) [117]

rectified by the integration of existing M&S capabilities together and with other means of evaluation [7], [8]. However many of these capabilities, and associated development, are centered on system design and acquisition.

This is demonstrated by how the DoD uses M&S: “The Department of Defense (DoD) continually seeks to acquire, sustain, and manage material and non-material solutions to

address capability needs of the war fighter in military operations and to provide efficient support and readiness in peacetime. A growing number of military capabilities are achieved through a system of systems (SoS) approach. ... Systems engineering (SE) is recognized as a key contributor to successful systems acquisition and is equally important for SoS.” [9]

Within the DoD, the exact definition and use of systems engineering is “acquisition life cycle and is based on DoDD 5000.01, DoDI 5000.02; DoDI 5000.85, Major Capability Acquisition; and DoDI 5000.88” [10] However, its purpose can be surmised as the process to shepherd a material solution to a capability gap from identification through operational support, to the end of its life when it is disposed. This approach does not emphasize the design of the system. Rather it emphasizes the capabilities of the system and historically interfaces with the designing organization through formal program reviews at the milestones laid out in Figure 2 [9], [11].

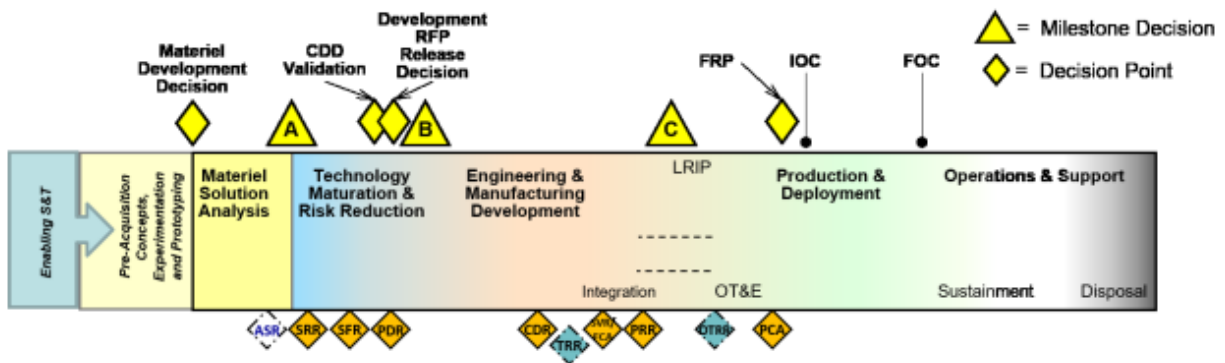


Figure 2 US DoD Acquisition Life Cycle [7] (Page 10)

The DoD is not the only one conducting M&S within a scope relevant to this paper. The definition provided by the DoD contrasts with the design-oriented approaches laid out by a number of other organizations, for our purposes here restricted to the International Council Of Systems Engineering (INCOSE) and National Aeronautics and Space Administration (NASA), as two design-oriented organizations encompassing both industry and government.

“The INCOSE Systems Engineering Handbook gives an overview of the life cycle of a system with a breakdown of phases as Concept, Development, Production, Utilization, Support, and Retirement as referenced from ISO 15288 and ISO 24748-1... These system life cycle concepts will not change with MBSE, so an MBSE approach must adhere to these existing stages.” [12]

Generic life cycle (ISO/IEC/IEEE 15288:2015)

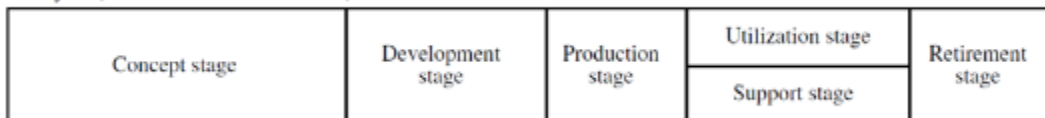


Figure 3 Generic Life Cycle [13]

INCOSE formally defines systems engineering as “a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods.” [14] This definition includes the “shepherding” of the system through the word “use”, though it is a much-reduced emphasis compared with that in the DoD. This difference is due to the design emphasis of INCOSE vs the acquisition emphasis of the DoD.

The Final Report of the Model Based Engineering (MBE) Subcommittee National Defense Industrial Association (NDIA) Systems Engineering Division M&S Committee, 2011 highlights these differences, [15] noting that the question about even the general role that a model should play is somewhat contentious, revolving around the difference between developmental model’s vs contract management model. Within the US DoD, a MBSE model represents the “baseline” of a system for the purposes of requirements, bids, deliverables, and for participation in traditional SETR events. While neither is “wrong”, they will require different modeling approaches/methodologies.

1.4.3. Model Based Systems Engineering

Despite demonstrating that there are a wide breadth of objectives and philosophies for Systems Engineering, an important note for later, there is a lack of specificity as to practical implementation. Traditional design and acquisition process is document centric, with diagrams used ad-hoc [5]. There is a shift now away from documents created in word, like this one, and towards a central model that acts as the authoritative source of truth. This approach, known as Model Based Systems Engineering (MBSE), will be explored in more depth later, there are two aspects warranting mention now:

- It is overall a cheaper approach for large projects because it assists in catching errors prior to release [16].
- An authoritative digital representation of a system is a key enabler of larger M&S efforts.

This second point stands on the fact that models are only as good as the information provided to them. Scrubbing documents for information to put into a model is a time consuming and error prone task that rarely yields all the requisite information for highly detailed physics-based simulations. Having a digital model of the system that can be queried for information and directly linked to the simulation software is a game changer. The facilitation of better and faster communications is critical due to a as of yet not discussed factor about the increasing cost and complexity: the supporting team of people also grows. The team can grow to be quite large and interface with organizations not normally in the direct development process, as shown in Figure 4:

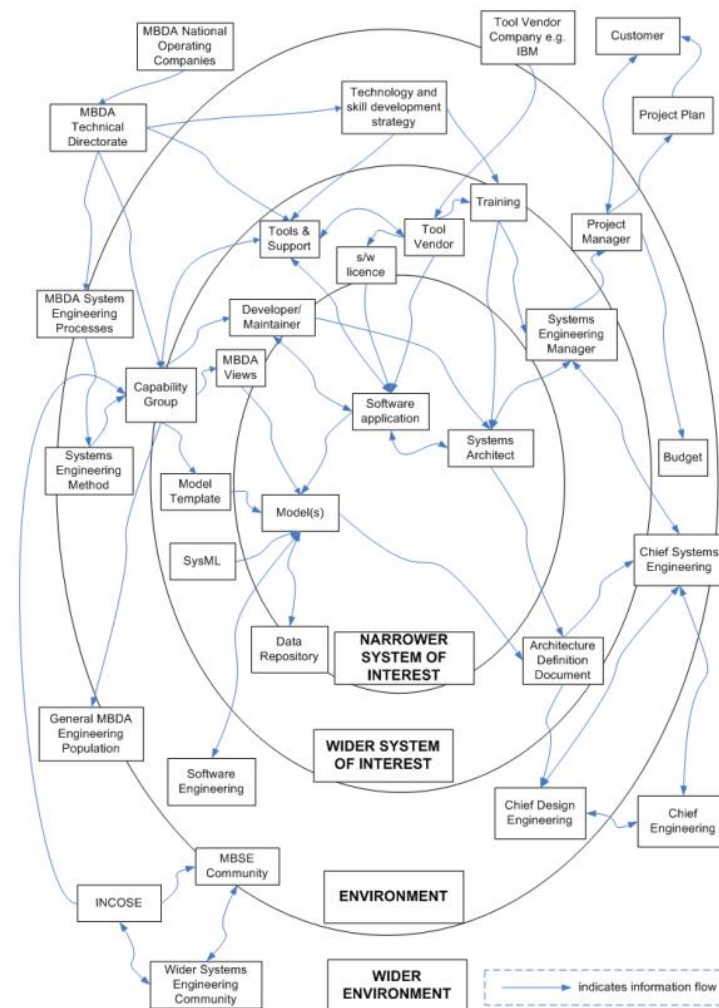


Figure 19: System Context Diagram

Figure 4 Interconnectivity of a System to Mission Development Team [17]

The growth in team, and increased number of involved organizations (from more prominent involvement of existing participants, to the developers of new enabling tools) also grow the requirements list necessary to fully scope out a prospective system. Fortunately, MBSE tools are particularly good at linking requirements to locations within the model, on account of MBSE models being descriptive models rather than analytical ones. The identification of requirements, and the detailed processes on how that is done well, is beyond the scope of this paper but include considerations such as later testing and support, which may not normally be given consideration at this stage on account of keeping the requirements more manageable [17].

1.4.4. Mission Engineering

In the previous sections, the paper has discussed the following key points:

- Systems are becoming more expensive as a result of getting more complicated.
- There are widely varying definitions and philosophies on Systems Engineering.
- M&S has been a part of the design and acquisition process for some time, but has challenges getting accurate and complete datasets in a timely manner.
- Model Based Systems Engineering is a shift away from document-based information that has potential to feed higher quality simulations.

These considerations are key to understanding how Mission Engineering plays into the mix. As systems become more complicated, it is imperative to identify proper requirements and usage so that development time is focused and scoped appropriately. This imperative extends to the operational environment, which is increasing in interconnectivity between the five formal domains and electronic warfare as an effective sixth [18], [19]. The widely varying definitions of Systems Engineering mean that there is no single approach optimized for Systems Engineering. Therefore, there is a diverse base of information to draw on, provided that a firm case can be made that such application is not misuse of Systems Engineering theory. In other words, that Mission Engineering is an extension of Systems Engineering. If that link can be made then not only are existing mission level analysis tools able to be better fed in terms of data from system models, but a mission integrated with system design can leverage all the higher fidelity physics-based design tools. The opening of the aperture in respects to tools available to mission analysts opens the question of what a mission-oriented simulation framework will look like.

1.4.5. Mission Level Simulation Framework

While beyond the immediate scope of this paper, understanding the concept of a federated mission level simulation is crucial for any mission level model. The simulation framework is the dynamic simulation component of mission engineering, where the model developed here is the structure that feeds information into the simulation framework. To support mission analysis, a framework must be able to fulfill both operational customer requirements and systems engineering requirements. This means operating with a high fidelity across the 5 warfighting domains: [20]

- Air
- Cyberspace
- Land
- Maritime
- Space
- Electromagnetic Spectrum (not formally a domain)

Typical Mission Engineering questions must also be answered. Examples from the DoD Mission Engineering Guide: [21]

- What domain(s) (Air, Surface, Subsurface, Space, Cyber, etc.) are involved in this study?
- How much fidelity is required in the model to adequately respond to the mission description/analytic question?
- What models are required to conduct the analysis? (e.g., 6-DOF, Physics-Based)
- What models are already accessible? Do the required models already exist?
- What is the pedigree of the model (verification, validation, accreditation, etc.)? Did we build the right model for use?

On top of these requirements and questions is one major difference between Mission and Systems Engineering: the dynamic, interoperable nature of missions. Systems as deployed are relatively static designs, where force designs are often task oriented dynamically as dictated by the mission. Therefore, direct application of System Engineering to Mission Engineering use cases lead to a stiff and suboptimal solution.

The question about which domains is more complicated than may be immediately apparent. A mission objective may be completed within a single domain, or if the mission may be better accomplished through interplay of multiple domains. The tradeoff is increased resource demand, resource availability, and inclusion of potential new players. The Russian Federation's invasion of Ukraine demonstrated the utility of the cyber domain in an otherwise traditional air-land-battle attack. The wide use of commercial satellites and social media to inform Open Source Intelligence (OSINT) adds a new dimension to the conflict in terms of giving anyone with an internet connection a front row seat to the conflict. While a detailed exploration into the effects of these changes on future doctrine is beyond the scope of this effort, suffice to say that the new options to strategically flank an opponent means that a fundamentally basic question of mission engineering, where the mission takes place, is now itself a multivariate trade study at a strategic level. A robust mission engineering framework must be able to account for these factors here as part of the trade, rather than relying on the mission designers to make that decision beforehand.

With the complexity that the growing number of domains impose upon mission engineering, the question of requisite fidelity vs risk level becomes more important. It is not feasible to compute every permutation of a mission, from a time/cost/information availability/classification perspective

As the literature review will demonstrate, there are two sets of tools for different customers. There are the wargaming tools which are primarily focused on requirements generation and wargaming. These tools have steadily been adding more and more fidelity to their entities, without compromising the ability to play out missions. The other toolset is composed of

engineering tools, such as Computer Assisted Drawing (CAD), that have been steadily adding cross discipline simulation to improve system design of increasingly complex systems. These two toolsets often have limited interoperability within their own camps. But that capability is maturing as well, with networked simulators over DIS or HLA on the Mission customers, and software like Model Center linking CAD with MBSE models [22].

There is rationale behind the assumption that the eventual independent development of the simulation framework is a realistic assumption. This rationale is another one drawn from the systems engineering world, Multi Domain Analysis and Optimization. Multi Domain Analysis and Optimization (MDAO) is a case where the Systems Engineering world had to overcome a similar problem when different disciplines had to start interoperating thermal, electrical, and structural models among others, into a cohesive environment. Any engineer familiar with ANSYS or COMSOL will have seen varying degrees of this process. Those same tools and companies have also started to extend their environments to MBSE models, providing the infrastructure for eventual integration [22]. There are already groups exploring this integration topic. However, they, like the definitions above for Systems Engineering, are light on the practical model creation. So, while these efforts and precedent give sufficient confidence that a federated simulation framework will manifest, this paper provides a transparent, detailed implementation to supplement the coming literature review.

2. Literature Review

This section covers the open source literature pertaining to Mission Engineering and Model Based systems Engineering (MBSE). It includes an introduction to the state of the discipline and its effects on the skew of sources available away from the academic. The Mission Engineering Section covers the history of Mission Engineering spanning from the origination of wargaming all the way through current practices and the documentation governing current ideas of the “what” and the “how”. The MBSE section details the evolution of Systems Engineering into MBSE, the industry standard modeling language (SysML), and the various literature deemed relevant to the Mission Engineering use case. The information laid out here will form the foundational outline of the current objectives, and the means of achieving them.

2.1. State of Academic Literature Review

The primary source for references in the paper are governmental and defense industry. This is a result of mission engineering and MBSE being more useful to large technical organizations (NASA, defense industry, DoD, etc.) than they are to academia. This is a function of the size and scope of the organizations themselves, creating a situation where journal published papers are sparse. This topic is further restricted due to sensitivities, as detailed examples of how a military or company is planning future engagements or products is not usually widely advertised. While the sources not published in a journal are still peer reviewed within the government (which are required to be internally peer reviewed to be publicly releasable), from authoritative sources (such as INCOSE, or other methodology owners), widely recognized textbooks (SysML distilled, or a Practical Guide to SysML) or represent public face of a large defense contractor (such as Boeing), these sources are still considered authoritative for their respective fields. However, given the deviation from traditional academic centric sources the clarification is needed.

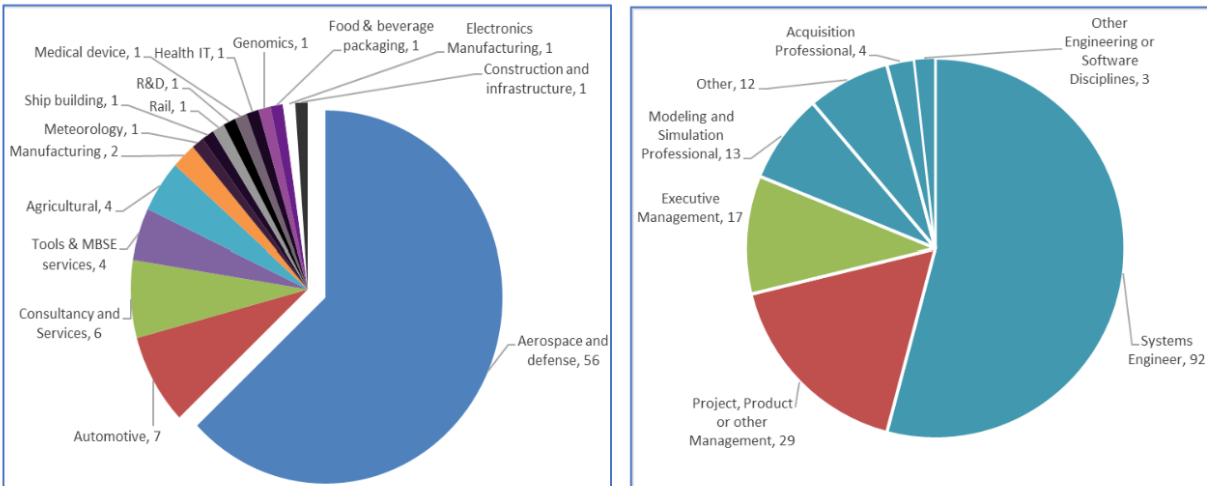


Figure 5 Reported Market Areas for Industry Respondents (Left) Reported Respondent Organizational Roles. (Right)[23]

Figure 5 highlights the impetus of the issues described in the above paragraph, and represent a study conducted by an academic affiliated institution with industry input on behalf of a government request, and is thusly taken as representative of the publishing space and not itself a skewed statistic.

2.2. Mission Engineering Literature Review

The Mission Engineering literature review walks through the genesis of Mission Engineering from its origins in the mid 1800's through modern implementation. Modern implementation contains both governing documents for the DoDs efforts for focus areas and process, as well as a discussion of the simulation methods supporting execution. It completes with a review and description of the JCIDS KPPs that will be used in the later example mission. This section details the objectives that any modeling methodology must satisfy which are considerations that guide the following MBSE sections.

2.2.1. History of ME/Wargaming

Wargaming, in the sense of prediction in warfare, is as old as warfare itself [24]. Modern wargaming originally began as a way to train officers from non-aristocratic backgrounds in high level tactics [24]. The goal of these exercises was, and continues to be, examining the potential impact of proposed equipment and tactics on the outcome of a given confrontation [24]. Well-executed wargames have delivered significant competitive advantage in numerous conflicts – although wargaming does not, and cannot, guarantee success [25]. Wargaming in its current paradigm began in the 1820's when two Prussian officers, von Reiswitz and his son developed a formalized map and board pieces. These are still the hallmark of a modern styled wargame today, [25] even as those maps and pieces have become digitized.

Mission Engineering Literature Review
Literature Review

In the United States, wargaming was first officially adopted by the US Navy, spearheaded by William McCarty Little, in the 1880's, and has been a staple ever since [24].

A core component of wargames is the method of adjudication. Chapter 3 of the UK's Ministry of Defense Wargaming Handbook [25] discusses the types of wargames used, and touches on the adjudication methods available. Proper adjudication is one of the most important aspects of any wargame as it, along with entity definition, are the foundational building blocks upon which lessons learned from any wargame rest.

The techniques used for adjudication are:

Role Play:

"Defence wargames sometimes include an element of role play, but are rarely role-play-only. Role play can exert a strong influence towards free, or even consensual/minimal, adjudication. Constraining role-play actor interactions can reduce the influence, but that risks lessening the benefits of role play (free thinking creativity). The ultimate expression of role play is completely open-ended games featuring consensual adjudication. There is some evidence, [42] when considering human conflict situations, that role play is a better predictor of outcomes than either a single 'expert', or game theory, or simulated interaction and unaided judgement for forecasting decisions in conflicts." [25]

Moderation:

"Moderation is used to steer a wargame to achieve specific training objectives, or to lessen extremes in an analytical event. Moderation is generally used during semi-rigid adjudication, and can influence the decision in either direction (towards an average expected outcome). However, moderation has perils, since shifting towards average outcomes can all too easily sideline important conclusions about the vulnerability of plans to chance and bad luck." [25]

Operational Analysis:

"Operational analysis informs the adjudicator(s), typically by presenting a spread of outcomes such as the best, worst and most likely cases. Using this to inform the decision influences adjudication in the direction of a rigid outcome." [25]

Computers:

"Computer assistance in the form of a 'plug-in' model or spreadsheet 'combat calculator' likewise informs the adjudicator's pending decision. Like operational analysis, the influence is towards a rigid outcome. Computerised simulations exert an even stronger influence in the direction of rigid adjudication. Commonly used, computerised simulations can provide the entirety of the adjudication function." [25]



Figure 6 Spectrum of Wargaming Adjudication Methods vs Rigidity Picture by Author

Each approach has benefits and drawbacks. However, for design trade studies, the repeatability of computerized wargames is important. As war becomes more technologically complex, demonstrated by the introduction of cyber and electromagnetic (EM) domains, the adjudication becomes more complicated and computer adjudication becomes more and more useful. In general, the computerized simulations light weight enough to be used in wargames make some very broad simplifications regarding aspects like human psychosocial and political characteristics. These simplifications are often necessary due to the restricted time for learning and execution available, and the need for real time feedback to the users [24].

Predictive analysis can be used to supplement limited relevant historical precedence [26]. This includes human factors, to the extent they can be described, but can be hamstrung by limited data and limited direct comparison. This is why so much emphasis is placed on recent conflicts like the Nagorno-Karabakh and Ukrainian wars, as relatively symmetric wars with modern equipment [27], [28], [29]. Computers represent a method to help offset the increased computational requirements inherent to predictive analysis.

2.2.2. Transition to Mission Engineering

Up to this point, the main purpose of wargames has been to act as training and decision aids to military commanders. As the reliance and complexity of the enabling technologies that modern militaries rely on increases, previously held levels of abstraction of the equipment need to be reduced. From a technical standpoint, when the human body represented the aggregate sensor, shooter, and transportation, wargames had experts on the technical systems by virtue of the participants being human. Modern militaries have, in many engagements, exceeded the human ability to sense, shoot, and carry with the introduction of radar, hypersonic missiles and mechanized/motorized/armored forces. This introduces error into traditional wargaming because the critical technical understanding is no longer present, which given that many of the mentioned areas require subject matter expertise to advance is entirely understandable. Nonetheless, the error introduced by abstracted equipment and technologies needs correcting, which must entail the integration of engineers and system representations into the core of decision making and force design. This integrated vision is articulated in the Defense Acquisition Guidebook which defines Mission Engineering (ME) as “the deliberate planning, analyzing, organizing, and integrating of current and emerging operational and system capabilities to achieve desired warfighting mission effects” [21]. This definition contains both the traditional wargaming planning, analyzing, and organizing to achieve desired warfighting mission effects,

Mission Engineering Literature Review
Literature Review

and the integration of system capabilities as an intrinsic method to achieve those effects. The integration of technically complex system considerations into the traditional wargaming process is the core fundamental of Mission Engineering.

Mission Engineering does not replace all wargaming, as there are trades to be made in matching the optimal approach for a system's complexity to the type of analysis. Figure 7 shows a relationship between three approaches: computerized modeling, wargaming, and synthetic training environments. Modeling is typically more technical, typically smaller scope efforts, and constitutes either static or dynamic models. Wargaming is human decision centric and can, as discussed in the introduction, be more freeform and less technically rigorous. Synthetic environments are much more suited to training purposes with high unit cost and can be used to inform both wargames and modeling with its outputs by including humans in the loop where appropriate.

Figure 7 highlights a dimension of the trade space for determining the most suitable approach. Mission Engineering falls somewhere around the star. It is not the cheapest or fastest to set up, and the technical considerations limit the scope of problems that can be analyzed. However, it is still capable of incorporating decision makers into the design process in a meaningful manner. Additionally, there is another axis to this plot that needs to be considered for the purposes of Mission Engineering- technical rigor.

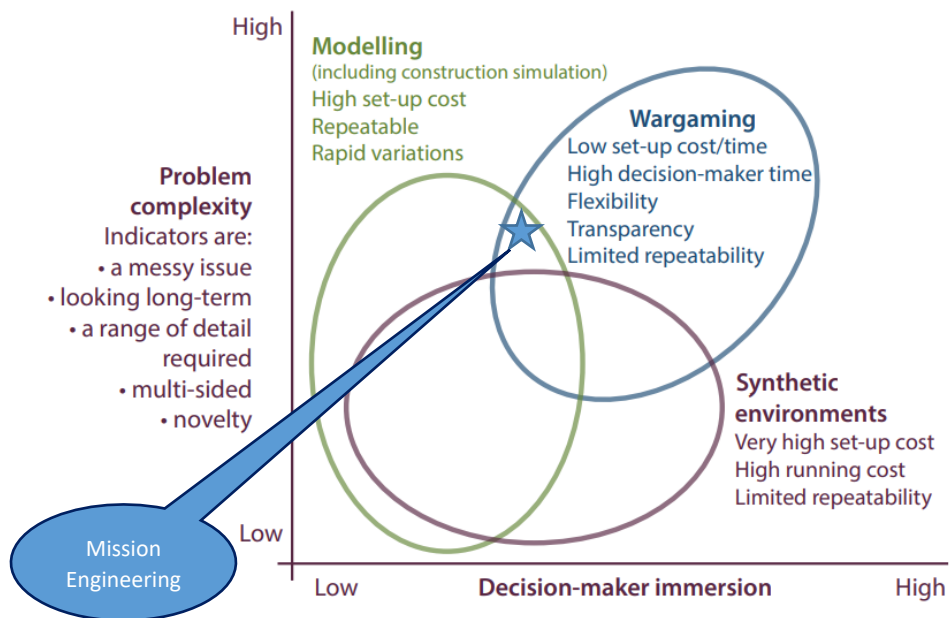


Figure 7 Strengths, Weaknesses and Overlaps between Wargaming [25]

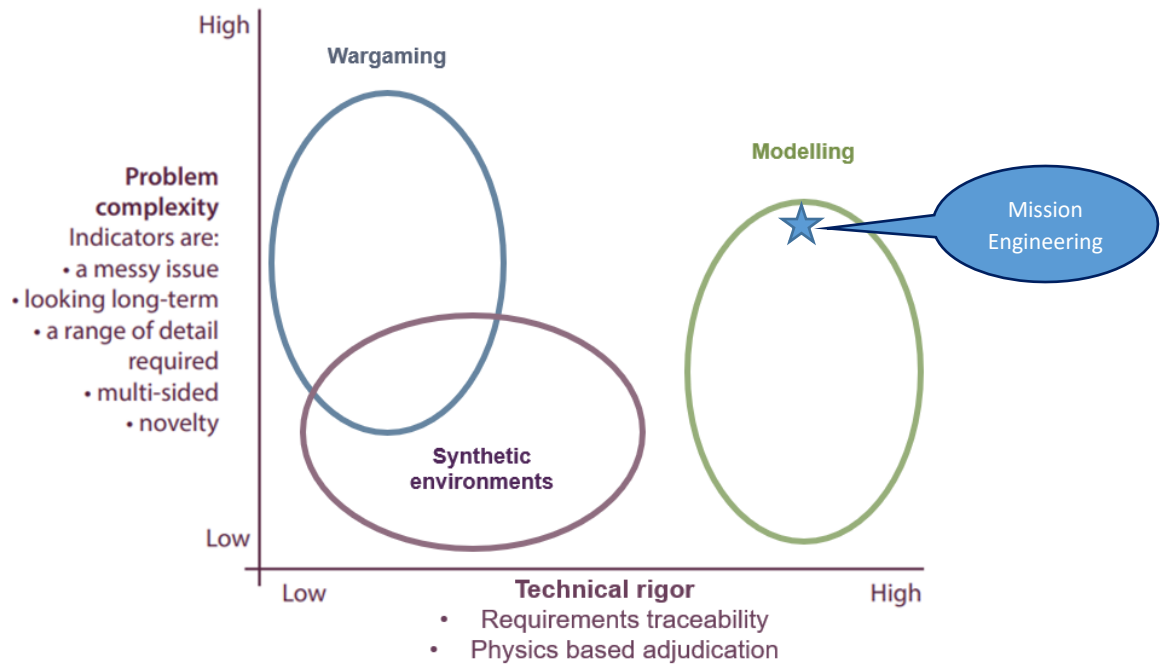


Figure 8 Orthogonal Axis for Technical Rigor (Author)

Figure 8 represents the other critical aspect of modern warfare as discussed previously: the technical rigor of the wargame. Technical rigor is the depth to which system performance and behavior are considered in the study. With the complexity inferred by this consideration, there are several tradeoffs, primarily in terms of cost, time to conduct the study, and scope ceiling that can be examined. The tradeoff of technical/analytical rigor with forecasting time and cost is captured in a more axis centric view in the US Mission Engineering Guide:

Mission Engineering Performed at Many Levels

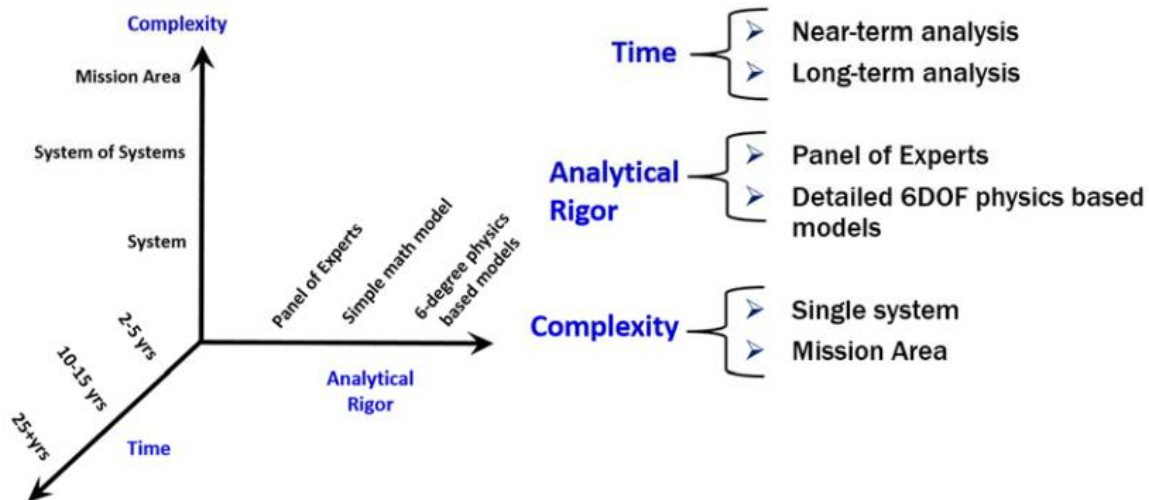


Figure 1-2. Three Axes of Mission Engineering

Figure 9 Three Axes of Mission Engineering [21]

While it is unfortunate that two-dimensional paper does not represent four-dimensional trade spaces well, the result is fairly intuitive: the complexity that a human can easily ingest is static, and under that limit, decision complexity, technical complexity, scope complexity, and time induced complexity are traded at the expense of the others. This human cap is core to why there are, and will continue to be, different types of analysis at different stages of a decision-making process, for both military and commercial ventures. And it is why Mission Engineering is not replacing wargaming, but offering another evolutionary tool in the search of an opponent's vulnerable flank, physical or technological.

Between these three figures, it has been highlighted that Mission Engineering seeks to blend the decision-making focus of wargaming, informed by smaller scoped technical models (linked to system design that will be explored in a later chapter), for the purposes of forecasting capabilities and solutions. This subsequently takes the form of an engineering-model-informed computerized wargame with notional systems, as the only practical means to represent the level of complexity involved.

This computerization allows for an increase in technical scope as complexity aids mitigate the effects of the associated increase in complexity. Formally, mission engineering sits between engagement modeling (one on one), and campaign modeling (force on force), as shown below in Figure 10.

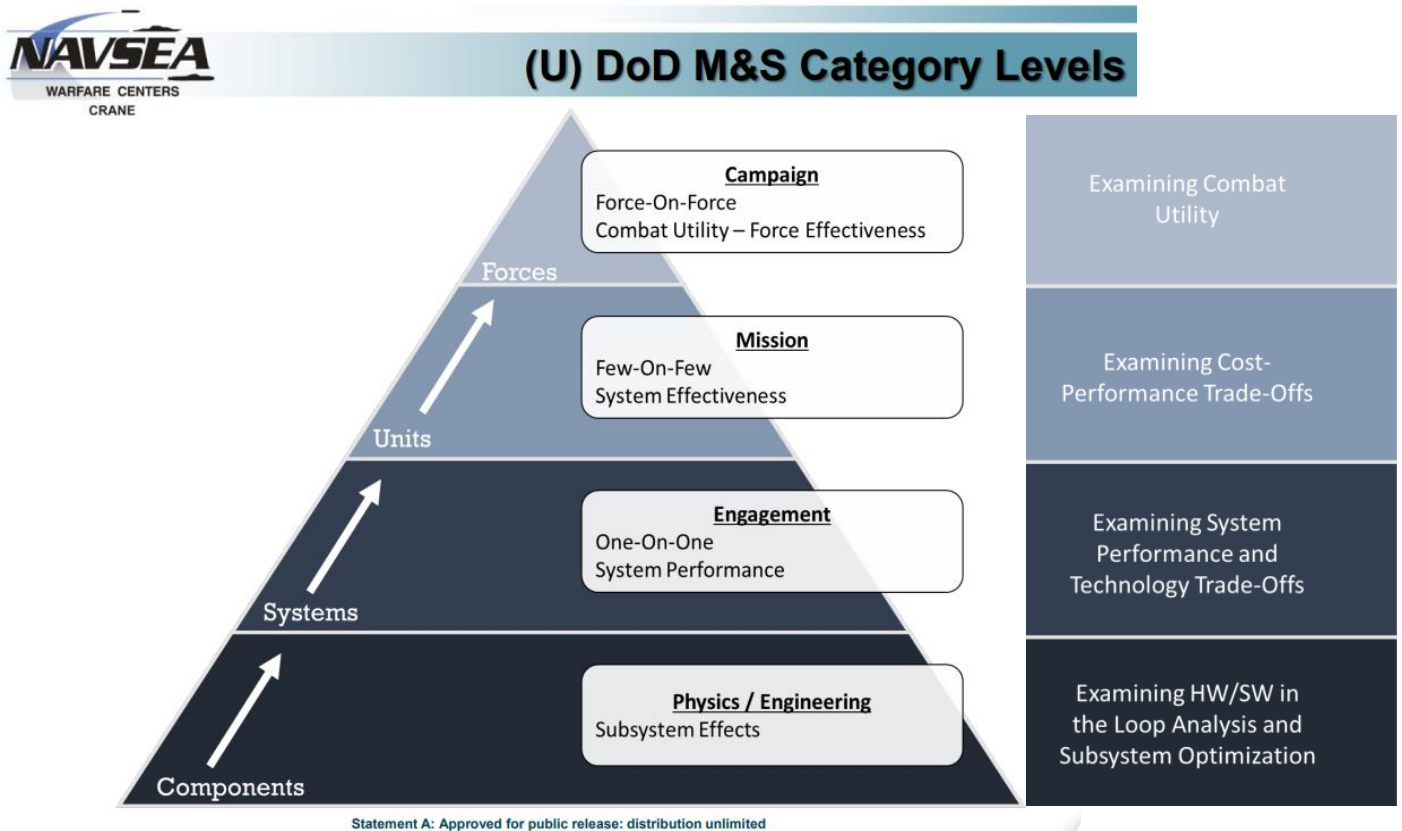


Figure 10 M&S Category Hierarchy [119]

Proper use of physics and engineering models of systems defined by MBSE has informed engagement level engineering. The idea that multiple modeled engagements can stack into one model of a mission at the cost of increased resource demands, while maintaining traceability is the thesis of this paper. The current trend indicates that this is the next logical step, provided that there is an increased simulation capability to support it. A wide spread adoption of a simulation capability that fuses live, virtual, and constructive entities to apply real world fidelity where required, and rapidly developed and tested constructive entities to mitigate cost and schedule creep would be a necessary key enabler of the application of this work in the future.

2.2.3. Types of Simulation for Mission Engineering - LVC

Having now outlined what mission engineering is, that it's a heavily computerized/simulated form of wargaming, it is appropriate to touch on the different ways that Mission Engineering can have an impact within the larger simulation construct that supports acquisition.

There are three main categories of simulations: [30]

Live: Simulation involving real people operating real systems

- Involve individuals or groups
- May use actual equipment
- Should provide a similar area of operations
- Should be close to replicating the actual activity

Virtual: Simulation involving real people operating simulated systems. Virtual simulations inject Human-In-The-Loop in a central role by exercising:

- Motor control skills (e.g., flying an airplane)
- Decision skills (e.g., committing fire control resources to action)
- Communication skills (e.g., members of a C4I team)

Constructive: Simulation involving simulated people operating simulated systems. Real people can stimulate (make inputs) but are not involved in determining outcomes. Constructive simulations offer the ability to:

- Analyze concepts
- Predict possible outcomes
- Stress large organizations
- Make measurements
- Generate statistics
- Perform analysis

Mission Engineering as defined so far is entirely constructive. This can be extended to include the virtual mode of simulation through use of physical hardware, either computer based or human based, so long as it is still set in a constructive environment. By the time live fire is included, the line between Mission Engineering and Testing and Evaluation begins to get blurred. These concepts are mutually supportive, but it is arguable that combining them transcends either individually, and becomes something more. This is the premise of Live Virtual Constructive (LVC).

LVC is used to reduce T&E cost for organizations, both commercial and governmental. For instance, the US DoD hosts several different implementations of LVC, NILE [31] for the Navy and MATREX for the Army [32], but also non-defense groups such as NASA who utilized LVC to

Mission Engineering Literature Review
Literature Review

analyze airspace concerns brought about by proliferation of unmanned air systems (UAS) in national air space (NAS) and the risk they pose to space launches. [33]

As LVC will be the larger technical approach to systems that Mission Engineering will support (as a significant portion of the Constructive component) it is beneficial to further explore LVC, how it relates to the system design cycle, mission engineering simulation requirements, training, and structure. The reason that LVC is so broadly applicable is the core premise that no one method of testing is without compromise. The comparative strengths and weaknesses of each are outlined below.

Table 1 Comparative Strengths vs Weaknesses for Live Virtual Constructive Test Methods (Aggregated [33]–[35])

Method	Strengths	Weaknesses
Live	– Comparatively low-level assumptions/abstraction – Real world performance and response – Training potential – Environmental factors accurately represented	– High overhead cost – Requires fielded hardware – Long lead time for planning – Few reps/small data generation – Competition for platforms
Virtual	– Some training potential – Lab scale testing possible – Real time hardware response	– Requires prototype hardware (minimum) – Higher level of abstraction – Narrow scope – Validation and Verification challenges
Constructive	– Large quantities of data creation – Operate with models (no prototypes need constructed) – Flexibility in missions analyzed – Large, cross domain testing – Does not tie up scarce platforms	– Accuracy is tool-based – Highest levels of abstraction – Computational demands – Distributed testing can induce latency – Validation and Verification challenges

In support of system design, LVC represents a way to conduct cost mitigation in preparation for a live test event, [34] particularly for joint live test events or exercises. What the key questions or Critical Operational Issues and Concerns (COICs) are for an event will drive heavily what blend of LVC is used at a given point within the life cycle.

For Mission Engineering, constructive is the most applicable type of testing. At this stage the necessary assumptions and abstractions can be mitigated, computational demands accounted for, and careful tool selection provides a sufficient level of accuracy. Using these tools Mission Engineering can analyze a variety of missions with a larger pool of systems at relatively low

Mission Engineering Literature Review
Literature Review

costs to create large quantities of information as part of the trade space exploration. The data generated serves the required outputs of mission engineering better than a live test event that focuses on the environmental protection of a system meeting the requirements. This reality is compounded by the fact that Mission Engineering is weighted to pre milestone A where typically the requirements have not been set, nor the system developed.

For systems engineering, the blend is a function of system maturity. As a system progresses from whiteboard, digital prototype, bench top prototype, through to fielded system, it will shift from constructive to more live focused. Simulation will never replace live testing, similar to how live testing will never generate the information gleaned from real combat. Though through credible use of the former two, the latter may be avoided.

For all the benefit that LVC can have, there are still some challenges it faces, particularly in the realm of distributed testing. While having a nationwide backbone of digitally connected centers capable of bringing their unique capabilities to bear in a joint digital battlespace is an incredibly useful tool, it requires a wide spanning, interoperable network. As will be discussed later under the net-ready Joint Capabilities Integration and Development System (JCIDS) Key Performance Parameter (KPP), this is not trivial (see Key Performance Parameters (KPPs)). There are a number of different protocols and software, each tailored for what that particular center is specializing in, that must be brought into a cohesive framework. For the purposes of this paper, it is sufficient to say that this issue is being continually improved and already represents a mature capability. Comparisons of protocols and road maps have, had entire thesis documents written on this one subject, and more information can be found within references: [31]–[39]

2.2.4. Mission Engineering Use Cases

There are two primary places that mission engineering can benefit the acquisition process: gap identification through JCIDS and analysis of alternatives, and in designing developmental test (DT) and operational test (OT) events as part of the integrated testing strategy.

Table 2 JCIDS, ME, T&E Objective Statement Comparison

Joint Capabilities Integration and Development System (JCIDS) [40]	Mission [21]	Integrated Testing [40]
“The systematic method established by the Chairman of the Joint Chiefs of Staff for identifying, assessing, and prioritizing gaps in joint warfighting capabilities and recommending potential	“The deliberate planning, analyzing, organizing, and integrating of current and emerging operational and system capabilities to achieve desired warfighting mission effects.”	“Conduct a seamless test program producing credible qualitative and quantitative data useful to all evaluators, and addressing developmental, sustainment, and operational issues.

Mission Engineering Literature Review
Literature Review

solution approaches to resolve these gaps.”

Integrated testing allows for the collaborative planning of test events; where a single test point or mission can provide data to satisfy multiple objectives, without compromising the test objectives of participating test organizations. Test points in this context, mean a test condition denoted by time, three-dimensional location and energy state, and system operating configuration; where applying a pre-planned test technique to the system under test and observing and recording the response(s).”

Moving from right to left on Table 2 follows the progression of a system from need through to T&E phase prior to fielding, following the system design cycle. Simulation in general has reached ubiquitous use on complex systems design, as evidenced in multiple documents such as [10], [11], [21], [40], [41]. However, many efforts remain relatively stove-piped, though that observation is subjective. In the following section, the use of mission engineering as a method to directly link capability gap identification to T&E will be explored before making a similar connection to the systems engineering design process in the next chapter.

2.2.4.1. Relation to Testing

The goals of both integrated testing and mission engineering contain the same core principles- identify operational issues or design flaws that would prevent the desired warfighting mission effects, and provide results to inform decision makers. This relationship is made clearer when the requirements are listed.

Requirements for integrated testing Developmental Testing and Operational Testing: [40]

- Evaluation in complex joint mission operating environments with systems of different levels of maturity (integrating upgraded systems with legacy systems)
- Replication of the real-world environment as closely as practicable in a safe and affordable manner
- Use of validated tactics, techniques, and procedures (TTPs)
- Representation of Blue and Red Forces

Mission Engineering Literature Review
Literature Review

- Validated scenarios
- Threat and threat countermeasures
- Dedicated instrumented ranges. (Differences exist in the instrumentation fidelity required to control participants, collect data, and support real-time and post-event analyses)
- Data collection, management, archiving, and retrieval processes
- Embedded sensors and instrumentation

It may not have been immediately apparent whether the requirements listed above were for integrated testing or mission engineering. This highlights the amount of overlap between the two subjects, and supports the further application of mission engineering solutions to planning of DT and OT events. For the purposes of mission engineering alone, simulations have an even greater capability to meet some of the listed requirements than live testing. This is a result of a greater quantity of runs, and the higher fidelity of the real world being offset by limitations in range instrumentation and data collection. While the above were taken straight from the Defense Acquisition Guidebook there is a final specified requirement worth highlighting:

- Need for a distributive live/virtual/constructive (LVC) representation of the joint operational environments (the only affordable way to test and train in a complex system-of-systems environment).

Constructive simulation, the core tool of mission engineering, is called out as a specific instrument of integrated testing. While there is much literature on the benefits of including simulation in integrated testing, or testing and evaluation in general, the key takeaway is that mission engineering, as an evolution of and user of constructive simulation, can and should readily support T&E. This provides both context and interface requirements to what Mission Engineering must provide when it comes to Mission Engineering's bread and butter; gap analysis and analysis of alternatives.

2.2.4.2. Analysis of alternatives/Gap analysis

There are two aspects of system design that Mission Engineering directly supports: capability gap identification and the analysis of alternatives at a mission level. There are two primary documents that govern this support; the more system centric Joint Capabilities Integration and Development System (JCIDS), and the mission centric Mission Engineering Guide (MEG).

2.2.4.2.1. Mission Engineering Guide Process

The Mission Engineering Guides (MEG) definition has been stated already, but in summary, is a mission level approach to achieving effects against a threat. The key emphasis is on the prominence of the threat, rather than on the solution to that threat. While this distinction may

Mission Engineering Literature Review
Literature Review

seem esoteric at first given the apparent overlap, it reflects an important difference in approach.

The approach that the MEG outlines is shown in Figure 11, beginning with the problem statement/mission and ending with the document study reporting.

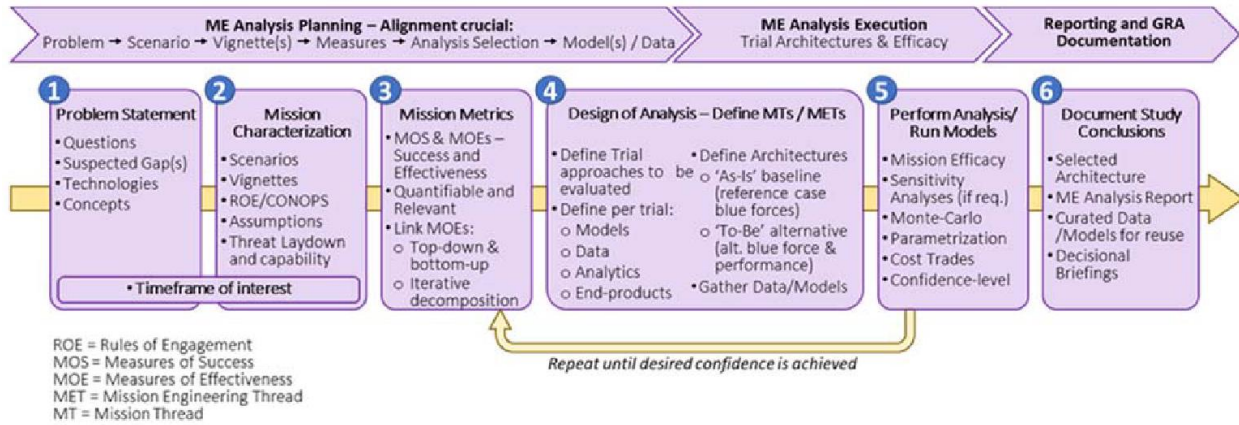


Figure 11 Mission Engineering Guide Process [21]

While much of the process is a quintessential trade study, the scoping of Mission Engineering process is important to note. The GRA documentation called out as an output focuses on mission level MOEs and inter-mission effects (e.g., an asset being used in one mission may be unavailable for another mission) [21]. These mission focused metrics can decompose to system level metrics used by the JCIDS process as shown in Figure 12:

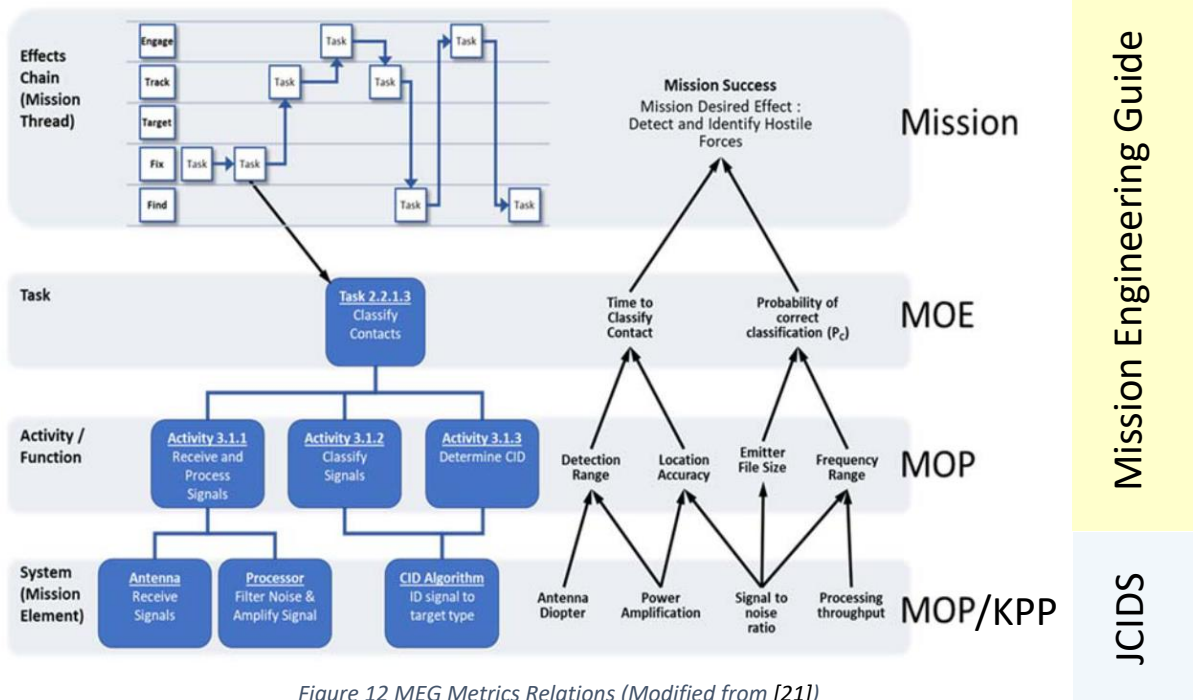


Figure 12 MEG Metrics Relations (Modified from [21])

While the MEG does a good job of outlining the process for in depth mission level trade studies as a part of Mission Engineering, the document is scoped similarly to the process in Figure 11. This means that it is light on details of how the output is used, and this is where the MEG feeds into the JCIDS process.

2.2.4.2.2. JCIDS Process

JCIDS process is built as a top-down analysis starting at the national level priorities and producing system level requirements. According to the JCIDS manual, the objective is to:

“Identifies capability gaps and potential materiel and non-materiel solutions. While it considers the full range of doctrine, organization, training, materiel, leadership and education, personnel and facilities (DOTMLPF) solutions, for purposes of this Guidebook, the focus is on the pursuit of "materiel" solutions” [40]

JCIDS is more system level focused, framing a material solution in preparation to entering the acquisition pipeline. These material solution concepts will need design and testing, which is the link to the integrated testing points made above. Particularly considering the overall objective is not individual systems but “the context of integrated architectures of multiple interoperable systems” [40] It consists of two main steps as outlined in Figure 13, Mission Analysis and Information Analysis. These steps capture the “experiment – data analysis” iterative cycle of traditional engineering design.

Mission analysis is intended to define the mission threads, questions to be asked, order of battle, and to define “success”. This scoped mission acts as the foundation for the information analysis step within which a more system-linked performance and trades can take place. This is, within JCIDS, a simplified process of Mission Engineering. That simplification is necessary as Mission analysis/Engineering is not the core focus of JCIDS. Minding that, JCIDS can still benefit from the more detailed process laid out in the MEG, as shown in Figure 13.

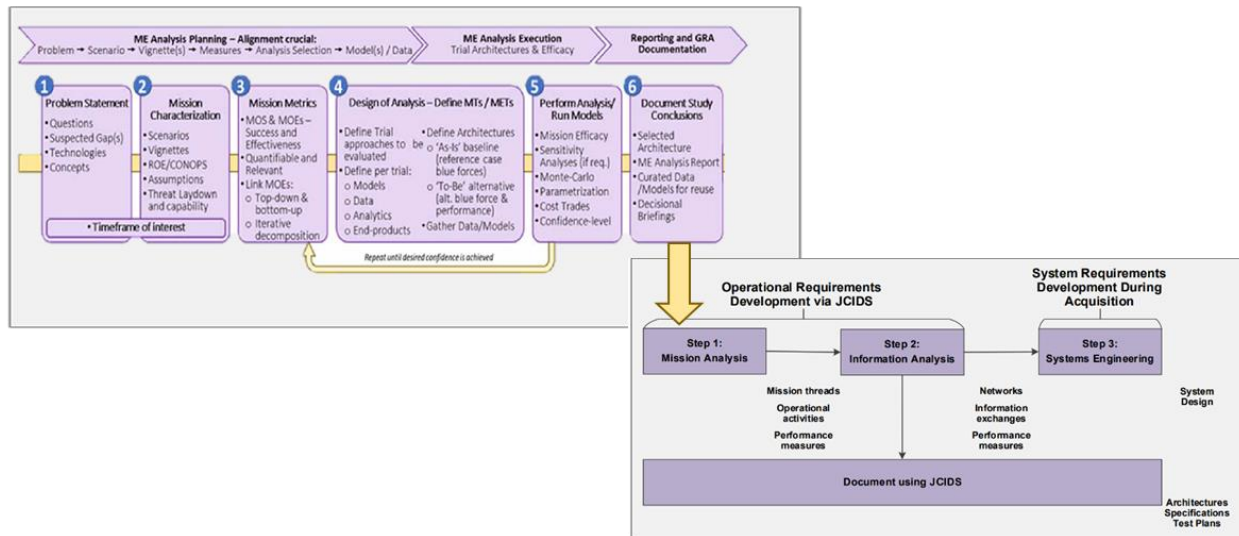


Figure 13 MEG Process [21] Feeding the JCIDS Process for Net-Ready Systems [42]

At the output of the information analysis step, the high-level outline of systems requirements is passed to the next stage of the acquisition process, identifying what technologies or systems are the best alternatives to meet the requirements. These outputs take the form of Key Performance Parameters (KPP) that a solution system must be design to, or designed to derivatives of those requirements. The KPPs will be discussed in detail later on, as they represent the interface with MBSE for a clean slate system design (see Figure 16).

As part of the design process for the solution systems prior to testing, there is a step called the Analysis of Alternatives (AoA) that compares potential solutions, formally defined as: “The AoA should describe and include the results of the supportability analyses and trade-offs conducted to determine the optimum support concept as part of the preferred system concept.” [40]

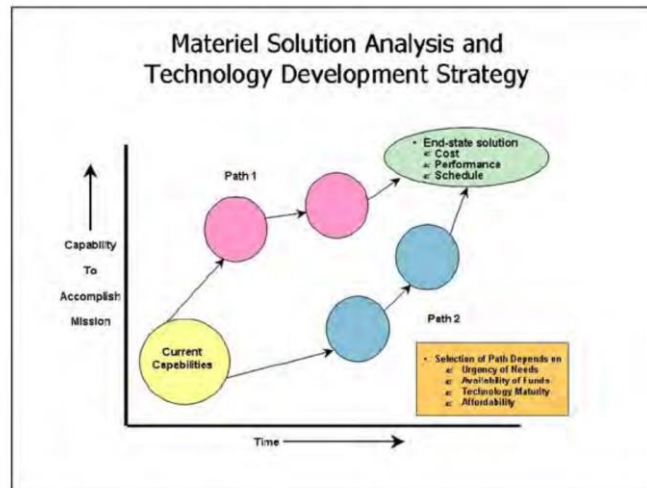


Figure 14 Pictorial Depiction of the Analysis of Alternatives Process [40]

Through documenting the requirements from the JCIDS process, and evaluating proposed high-level solutions to better refine the solution space, the foundation for the remainder of the systems engineering process has been laid. If these requirements are documented digitally in accordance with mission engineering's computerized approach, perhaps using SysML, then there is a technical linkage/digital thread from the process thus far directly to the solution model. This linkage also fulfills the requirements generation steps of Model Based System Engineering (MBSE). This stands in stark contrast with traditional wargaming, which has a written or verbal documentation passed through a chain of reviewers between requirements and the executing technical authority. A bidirectional technical integration of these functional areas would allow system developers and project managers to conduct AoAs throughout the system design cycle, and would better enable them to evaluate design trades and react to changes in mission sets. It would also help more closely align with the iterative approach of the JCIDS process as a whole. While this concept will be explored more in the MBSE Related to Mission Engineering section, it is worth emphasizing here that this approach helps reduce the linear, brittle waterfall approach often undertaken as part of the typical current design process.

Having outlined how the JCIDS, capabilities-based assessment, and AoA are related to mission engineering, this paper will proceed to further link the two use cases.

According to the US DoD Acquisition Guide:

“Key sources of T&E information, used during the formulation of a Materiel Solution, include the capabilities-based assessment (CBA), Analysis of Alternatives (AOA), JCIDS documents, etc. Items of particular interest to the T&E community include [40]:

- Mission description, scenarios, Concept of Operations (CONOPS), performance attributes and effectiveness metrics, targets and threats, operational environments, etc.
- Mission to task decomposition and scenario-based task performance standards
- Task to system/sub-system associations and functionality
- Alignment of mission Measures of Effectiveness (MOEs) with system performance attributes and measures

These are the exact products that mission engineering seeks to produce, making mission engineering as a subset of the overall requirements generation process uniquely suited to supporting T&E activities with the same products it develops for the design requirements process.

2.2.4.2.3. Relation of MEG to JCIDS

As this brief introduction to these two approaches highlights, the JCIDS and MEG processes are intertwined. The MEG refines the “Mission Analysis” step nested within JCIDS, with the latter shaping and consuming the outputs of Mission Engineering. It is beneficial to leverage the increased emphasis and capability of Mission Engineering over the existing JCIDS initial step, as the former provides a more holistic approach to the Mission Analysis step of JCIDS. The complementary focuses of the two approaches is summarized in Table 3.

Table 3 JCIDS vs MEG Summary Comparison

Area	JCIDS	MEG
Focus	System performance within a mission	Mission (System of Systems)
Solution space	New system	New system or improvements to current systems.
Customers	Acquisition	Acquisition and operational
Use Case	Generalized	Multiple specific missions and vignettes
Metrics	Key Performance Parameters	Mission level measure of success, measure of effectiveness, and measures of performance

The comparison of focus areas requires some additional explanation. The JCIDS KPPs, discussed later, are mission level because they specify a consideration of a threat, even though they are heavily biased towards the system perspective [42]. This is in contrast with the focus of the MEG, which is mission effect focused. This begs the question that if both of these are mission level considerations, at what point does pure System Engineering start. The cleanest delineation of that line from the literature review is consideration of the threat. A mission level KPP must include a specified threat, but the solution to that threat can be system design focused or exchanging for a different system. As will be seen in the example mission, it is

possible to derive a mission level KPP into a system level KPP as is done with fuel consumption being derived from detection distance against a specific target, along the lines of Figure 15. The concept of delineation of a system vs mission KPP along the lines of inclusion of a threat is more relevant to the body of literature than the acronym itself, as the usage of KPP as purely a mission or a system metric is inconsistent. For the purposes of this paper KPP is just that, a key performance parameter that is relevant, be it system or mission level.

While the approaches will be combined in this papers example mission, the slant towards integration with MBSE methods described in Figure 16, bias the content towards JCIDS KPPs.

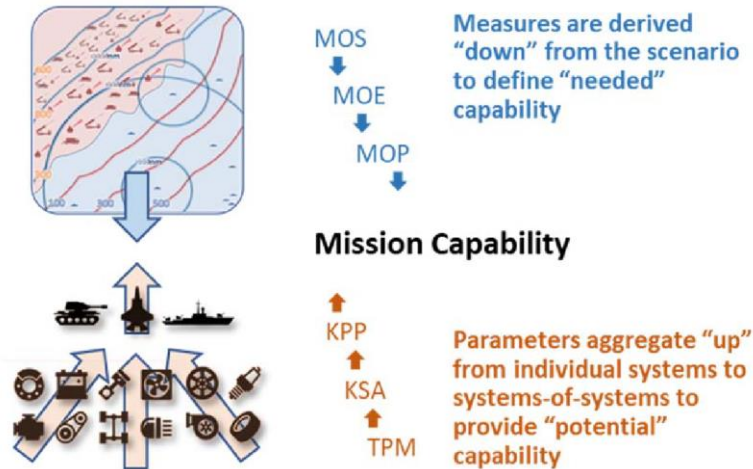


Figure 15 Relation of System and Mission Metrics [21]

The focuses on the more system centric JCIDS KPP's is a result of its position as the point of integration with MBSE for the mission side, as shown in Figure 16.

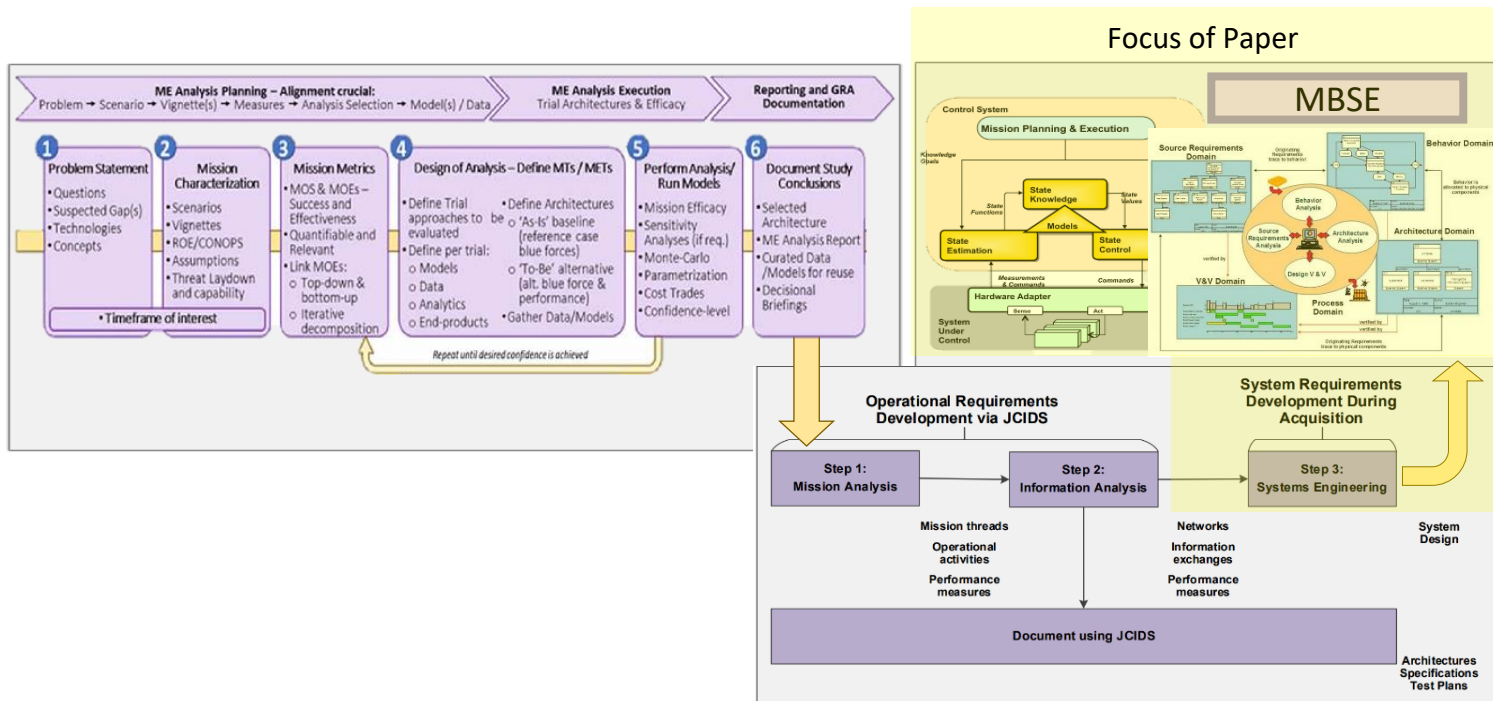


Figure 16 MEG Process [21] Feeding the JCIDS Process for Net-Ready Systems [42] which informs MBSE [95] [98]

2.2.5. Technical Mission Priorities

Having established that the products of mission engineering are useful to both the design process and T&E event planning, and how those products can be generated, it is time to outline the technical areas that are under investment, and will therefore be influencing the scope of mission engineering in the future.

Deep analysis of these high-level documents is not the main focus here, so a sub-section of three main guiding documents is cross-correlated for similarities which are then used to bound the simulation framework, and are taken as representative of the national interests for the United States at the time of writing (preserving the listed order).

Table 4 shows the priorities as listed by the National Defense Strategy (NDS), Defense Advanced Research Projects Agency (DARPA), and the Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)). The table has maintained the order in which each topic was listed on the respective citations, and the color coding highlights similar technical areas across all three organizations.

Table 4 US NDS, DARPA SBIR, and OUSD (R&E) Technical Area Priorities

Priority	NDS [43], [44]	DARPA 2019 SBIR/STTR BAA [43]	OUSD(R&E) Modernization Priorities [45]
1	Hypersonics (Joint lethality in contested environments.)	Artificial Intelligence	Artificial Intelligence/Machine Learning
2	Directed Energy (Joint lethality in contested environments.)	Autonomy	Biotechnology
3	Command, Control and Communications (C4ISR)	Communications	Autonomy
4	Space Offense and Defense (Space and cyberspace as warfighting domains)	Cyber	Cyber
5	Cybersecurity (Space and cyberspace as warfighting domains)	Directed Energy	Directed Energy
6	Artificial Intelligence/Machine Learning (Advanced autonomous systems)	Hypersonics	FNC3
7	Missile Defense (Missile defense)	Microelectronics	Microelectronics
8	Quantum Science and Computing (Advanced autonomous systems)	Quantum Sciences	Quantum Science and Computing
9	Microelectronics (Resilient and agile logistics)	Space	Hypersonic
10	Autonomy* (Advanced autonomous systems)	Nuclear Modernization	Space

There are significant similarities between the three organizations priority list, providing clear direction in the topic areas likely to appear in a mission engineering scenario. Therefore areas that the modeling and simulation framework needs to support as well.

There are two primary takeaways from Table 4: One, there is a significant consensus between the three lists, and two, the lists cover a wide breadth of technological areas. The first point acts as reinforcement that investment into a simulation capability able to address the technology areas is a worthy venture, since there is redundancy in funding sources. The second point is that such a simulation capability will be difficult to construct to sufficient levels of technical rigor. As it is pulling from a wide variety of sources of information, a new approach to

feeding the simulation framework is required, and, given the growing prevalence of Model Based Systems Engineering (MBSE), it is logical that MBSE will play the role of authoritative source of system truth.

The following section identifies how each of the common technical thrust areas will impact the types of information used in the example mission in this effort. Regardless of whether or not a technology thrust area is included in the example mission, all of the areas are extremely pertinent to high level strategy. Many are not included here due to reasons of scope or sensitivity, and do not reflect any comment on priority.

2.2.5.1.1. Artificial Intelligence (AI)

OSD Definition: “The DoD will leverage AI to enable U.S. forces to operate more effectively and efficiently. As a Department, we are evaluating which of our processes and procedures can be enabled via adoption of AI technology to meet warfighter needs and Defense priorities.” [45]

DARPA Objective: “Improve algorithms, address data quality, optimize human machine coordination and disrupt adversaries’ efforts.” [43]

Mission Engineering Potential and Use in this Paper: This area manifests itself as a use of the simulation framework, providing a constructive environment to develop AI/ML and conduct testing risk mitigation. While training specific types of AI/ML off of results generated through a design of experiments (such as the example mission) is practical, it is directly beyond the scope of this analysis of MBSE in support of Mission Engineering.

2.2.5.1.2. Autonomy

OSD Definition: “Autonomy extends and complements human capabilities. Advantages include persistence, size, speed, maneuverability, and reduced risk to human life. The DoD targets seamless integration of diverse unmanned/mixed team capabilities that provide flexible options for the Joint Force.” [45]

DARPA Objective: “Address teaming of autonomous systems; machine perception, reasoning and intelligence; human and autonomy systems trust and interaction.” [43]

Mission Engineering Potential and Use in this Paper: Artificial Intelligence can be a key enabler of Autonomy, because it shares similar suitability considerations. As such, autonomy will not be explored in the example for similar reasons to AI, primarily generational quantity, and scope of the paper. Autonomy development can both be some of the most beneficial use of constructive simulation (when exploring behavior) or least beneficial use (when exploring perception and classification of objects). It is worth highlighting, briefly, the role that automation of a mission engineering design of experiments could play in countering the issue of scope through additional quantity of runs and points in that trade space. This would provide a higher

resolution of the trade space topography. This level of automation does have a requirement, to be supplied with large quantities of high-quality data, highlighting the importance of an authoritative source of truth for the systems being examined in the trade space [46].

2.2.5.1.3. Directed Energy

OSD Definition: “Directed Energy (DE) matures to a deployable capability, our armed forces will have the potential to defend against several types of threats with great precision and minimal collateral damage, at minimal cost per engagement. High Energy Laser (HEL) technology development and advancements in hardware are making laser weapon systems increasingly viable.” [45]

DARPA Objective: “Address power scaling, jitter reduction, laser size and weight, adaptive optics, beam propagation and target tracking.”[43]

Mission Engineering Potential and Use in this Paper: When modeling DE in the example mission, DE systems will be defined primarily by power capacity and generation, jitter, power expenditure rate, size and weight.

2.2.5.1.4. Hypersonics

OSD Definition: “Hypersonic weapons travel five or more times the speed of sound. There is a focus on the tactical capability that these sorts of weapons bring to theater conflicts or regional conflicts. Very quick response, high speed, highly maneuverable, difficult to find and track and kill. We are modernizing our offensive and defensive force structure to both utilize and deter this capability.” [45]

DARPA Objective: “Address high temperature materials, hypersonic vehicle manufacturing, air breathing propulsion and hypersonic guidance and control systems.” [43]

Mission Engineering Potential and Use in this Paper: Due to a variety of reasons, primarily considerations around sensitivity, hypersonics will not be considered as part of this example mission.

2.2.5.1.5. Space

OSD Definition: “The U.S. way of war, across all domains, is dependent on timely and assured space effects. Adversary capabilities and advancements require us to move quickly to a more defendable and resilient space posture. Added protection and resiliency to our current spacecraft fleet is essential.” [45]

DARPA Objective: “Developing low earth orbit nano-satellites for missile warning, intelligence, surveillance, reconnaissance, navigation and communications.” [43]

Mission Engineering Potential and Use in this Paper: While space represents an entire warfighting domain, it will not be considered in the example mission due to sensitivity concerns.

2.2.5.1.6. Microelectronics

OSD Definition: “Microelectronics have been rapidly evolving as the demand for inexpensive and lightweight equipment has increased, and have been incorporated into countless DoD systems. Our modernization ability is jeopardized by foreign microelectronics (ME) production, actions, and investments. We must develop and deliver next generation microelectronic technologies to enhance lethality, ensure critical infrastructure, and achieve economic competitiveness.” [45]

DARPA Objective: “Develop economically competitive domestic manufacturing capabilities, improve radiation hardening and develop radio frequency (RF) technologies for specialty applications with nuclear, space and electronic warfare capabilities.” [43]

Mission Engineering Potential and Use in this Paper: Microelectronics are not properly scoped to participate in mission engineering level simulations due to primarily impacting the practicality of implementing the requirements generated through mission engineering. This type of distinction will be covered in the next section about properly scoping and developing key performance parameters at the mission level.

2.2.5.1.7. Quantum Science and Computing

OSD Definition: “Quantum computers pose an impending threat to secure communications. Continued US dominance in quantum information science will keep us ahead of these risks, and NSA crypto-modernization will protect our most sensitive communications against a quantum computer attack. Quantum sensing will deliver new and assured precision position, navigation, and timing capabilities, keeping our forces safe in GPS-denied theaters. Quantum networks will deliver drastically enhanced sensors for finding and fixing elusive targets, and will deliver resource multiplying effects for commercially developed quantum computers to solve DoD’s hardest analytical problems [45].

DARPA Objective: “Address quantum clocks and sensors, quantum communications technologies and develop enabling technologies for quantum computing in the areas of cryogenics and photon detection.” [43]

Mission Engineering Potential and Use in this Paper: Similar to Space and Hypersonics, anything relating to the design or efficacy of alt-PNT or cryptographic sciences is noted as a key thrust area, but not appropriate for discussion in a publicly releasable document.

2.2.5.1.8. FNC3 - Communications

OSD Definition: “Command, Control, and Communications technology encompasses the capability to acquire, process, and disseminate information across force elements. DoD requires a clear path to robust C4I with multiply redundant fully-networked “Comms.” Existing capabilities require sufficient protection against an increasing threat, in pervasiveness and effectiveness.” [45]

DARPA Objective: “Addressing high-performance, low-power embedded processing and developing algorithms for self-configuring, self-healing and resource allocation.” [43]

Mission Engineering Potential and Use in this Paper: The ability to conduct communications will be included in the example mission later in this paper as the mechanics of propagation are widely available.

2.2.5.1.9. Cyber

OSD Definition: “Cyber is a unique operational domain with significant security challenges and potential leap-ahead capabilities for military operations requiring enhanced command, control and situational awareness, and autonomous operations. Ability to gain and maintain the U.S. technological edge in cyberspace in the face of rapid evolution is essential to maintaining mission readiness [45].

DARPA Objective: “Address behavioral issues, develop self-securing networks and develop methodologies to assess cyber effects and consequences.” [43]

Mission Engineering Potential and Use in this Paper: Cyber effects pose unique challenges and opportunities for modeling in a constructive environment, as live fire cyber effects occur on the same hardware type as being used to simulate the rest of the scenario in a closed loop. As a result, cyber effects can be introduced particularly well when done in conjunction with other uses of constructive simulation such as during a wargame or an LVC event. Introducing accurate system hardware would allow for virtual cyber effects and increased accuracy, however for sensitivity reasons, no cyber effects will be modeled in the example mission.

2.2.6. Key Performance Parameters (KPPs)

With the different capability sets provided by Mission Engineering’s simulation framework for the topics above (when compared against traditional stove-piped systems engineering, T&E, or

CBA/JCIDS) it is pertinent to take a brief review of how gaps in those areas are closed, and indeed how those areas are prioritized in the first place.

The Joint Capabilities Integration and Development System (JCIDS) is a crucial document for systems requirement generation. It is a set of guidelines for translation of the outputs from wargames and experiments into system level considerations called key performance parameters (KPPs). As such it represents the first step of both document-based systems engineering and MBSE for any system to be developed from a set of mission requirements.

While the 2018 JCIDS document has been superseded by a 2021 revision, the distribution A JCIDS release that is included the appendices is the 2018 release. As such, it is the one more closely evaluated below due to the importance of Enclosure B. Enclosure B details the four top level KPP's that are required to be addressed when developing a system to complete a mission, even if its capabilities are degraded, so long as it is not mission killed. These KPP's are outlined in para 2.5.4 2018 JCIDS Manual: [42]

- Force Protection
- System Survivability
- Sustainment
- Energy

2.2.6.1. Force Protection (1st KPP)

As outlined in Annex B, Appendix G, Enclosure B of the JCIDS Manual (2018):

“The Force Protection (FP) KPP (one of the four mandatory KPPs) is intended to ensure protection of occupants, users, or other personnel who may be adversely affected by the system or threats to the system. Although the FP KPP may include many of the same attributes as those that contribute to System Survivability, the intent of the FP KPP is to address protection of the system operator or other personnel against kinetic and non-kinetic fires, CBRN, and environmental effects, rather than protection of the system itself and its capabilities.” [42]

Decomposition of the force protection KPP includes a number of review criteria. While Annex B of the JCIDS Manual (2018) outlines these in detail, the distilled list is:

- Is there a documented comprehensive analysis of the system, its planned use to include operating environment and tempo, threat, vulnerability assessment and solutions supporting the FP goals?
- Is there adequate protection from kinetic and non-kinetic fires, and from CBRN threats for the planned threat per the standards of Annex B?
- If applicable, are the occupants protected from environmental effects and accidental crashes?

There are additional details on survivability, threat methodologies, example language, and applicable attributes for EMS/non-kinetic, Cyber, and Kinetic via the Knowledge Management and Decision Support; however, that site is on SIPR, and therefore well beyond the scope of this paper. All information here is taken from JCIDS Manual (2018).

2.2.6.2. System Survivability (2nd KPP)

As outlined in Annex C, Appendix G, Enclosure B of the JCIDS Manual (2018).

“The System Survivability (SS) KPP (one of the four mandatory KPPs) is intended to promote the development of critical warfighter capability that can survive kinetic (i.e., traditional, non-traditional, and CBRN) and non-kinetic (cyber and electromagnetic spectrum (EMS)) threats across domains and applicable environment including space. The SS KPP supports three system-oriented objectives: prevention, mitigation in tactically relevant time, and recovery from threats and fires. Simultaneously, the SS KPP maintains survivability from diverse environmental factors (i.e., heat, dust, moisture, EMS-congested areas, etc.). The SS KPP promotes the analysis of probable threats and advances a SoS approach that leverages in-depth, overlapped, or redundant defensive capabilities through networked, autonomous, or semi-autonomous systems to ensure mission completion, despite the loss or degradation of individual components.”[42]

Decomposition of the SS KPP includes a number of review criteria. While Annex C of the JCIDS Manual (2018) outlines these in detail, the distilled list is:

- Is there a documented comprehensive analysis of the system, its planned use to include operating environment and tempo, threat (KE, EM, and CBRN), vulnerability assessment and solutions supporting the SS goals?
 - Can survivability be accomplished using camouflage, concealment and deception (CC&D), situational awareness, maneuver within a tactically relevant time, or hardening (i.e., physical armor, EM, cyber, and against environment hazards).
- Minimize down time between mission cycles, for both maintenance and battle damage repair.
 - Ensure that the system can be recovered, and identify the level of facilities necessary to enact maintenance and repair and their availability given the intended use of the system.
- Are the performance attributes consistent with the intended use as outlined in the CONOPS, and are the criteria realistic and measurable?

2.2.6.3. Sustainment (3rd KPP)

As outlined in Annex D, Appendix G, Enclosure B of the JCIDS Manual (2018):

“Sustainment KPP (Mandatory KPP). The Sustainment KPP is intended to ensure an adequate quantity of the capability solution will be ready for tasking to support operational missions.” [42]

This KPP spans the entire lifecycle of the system and “is derived from system availability requirements to support the required capability, assumptions for its design and operational use as specified in the CONOPS and/or OMS/MP tradeoffs between reliability, maintenance concepts, lifecycle cost, and the planned sustainment strategy.” [42]

Decomposition of the SS KPP includes a number of review criteria. While Annex D of the JCIDS Manual (2018) outlines these in detail, the distilled list is:

- Material availability
 - “Can be expressed mathematically as the number of operational available end items/total population”[42]
 - The table below outlines a management option for a given location to determine the correct quantity needed for the threat faces and the system capabilities proposed.

Table 5 Total System Inventory for Single Location Management

CONOPS	Training	Attrition	Prepositioned	Average Annual Down [1]	Total
CONUS					
OCONUS					
TOTAL					
Note 1: The average number of unavailable assigned assets based on assumed planned depot/ shipyard cycles					

- Operational Availability
 - One of the more complicated distillations of the KPPs, it calculates the average annual down time and is recorded above. The proportion of time a system spends “down” (in repair/maintenance) is a function of the density and tempo of operations, both combat and training, and while is approximated as a steady state percentage, it is complex variable in reality.
 - Can be represented by the equation uptime/ total time
- Reliability
 - Reliability factors into operational availability. It is calculated for a specific interval under specified conditions. “For continuous use systems (such as an aircraft), reliability should be measured in terms of its primary usage metric (such as operating hours, miles or flight hours). For discrete systems (such as a single use munition), reliability should be measured as a probability.” [42]

- Figure B-23 of [42] discusses types of reliability for different systems. The table can be summarized as:
 - Mission reliability = operating hours/mission failures
 - Logistics reliability = operating hours/logistics demand
 - Computers need no calculations because “materiel availability is not applicable”[42]
- Reliability can dictate a number of items that increase the logistical footprint or decrease the operational availability, and represents a critical attribute of a systems overall performance, particularly in scenarios in which an increased logistical foot print is not viable.
- O&S Costs
 - Measures the cost of the system across its entire life cycle, ensuring that the system is feasible for acquisition, as an inadequate system in hand is far superior to a perfect system that was never purchased due to cost.

One consideration for complex systems (such as the system of systems (SoS) that drive this paper) is that reliability is aggregated up from components, but O&S and operational/materiel availability are applied to the SoS as a whole. Additionally, Annex D, Appendix G, Enclosure B of the 2018 JCIDS includes additional questions to assist in determination of the metrics above that are too extensive to include here, but make for excellent references and will be used in the example mission in a later chapter.

2.2.6.4. Energy (4th KPP)

As outlined in Annex E, Appendix G, Enclosure B of the JCIDS Manual (2018):

“Energy KPP (Mandatory KPP). The Energy KPP is intended to ensure combat capability of the force by balancing the energy performance of systems and the provisioning of energy to sustain systems/forces required by the operational commander under applicable threat environments.”[42]

Energy on the current battlefield is almost entirely liquid fuel. That fuel has a large footprint, and a logistics trail all its own that balloon the requirements of the end system. The remainder of the energy used is electricity, which could be drawn from a local source, but often is itself reliant on liquid fuel for generators.

There are a several performance attributes associated with energy:

- Operational effect of the system as a function of its energy requirements. For a system of systems, identify those most critical to the mission effectiveness.
 - Describe this function as payload (mass or volume)/gallon for transportation
 - Consumption at peak consumption (gallons/kWh)
- The requirements change as a function of the type of “energy node” a system is.

Mission Engineering Literature Review
Literature Review

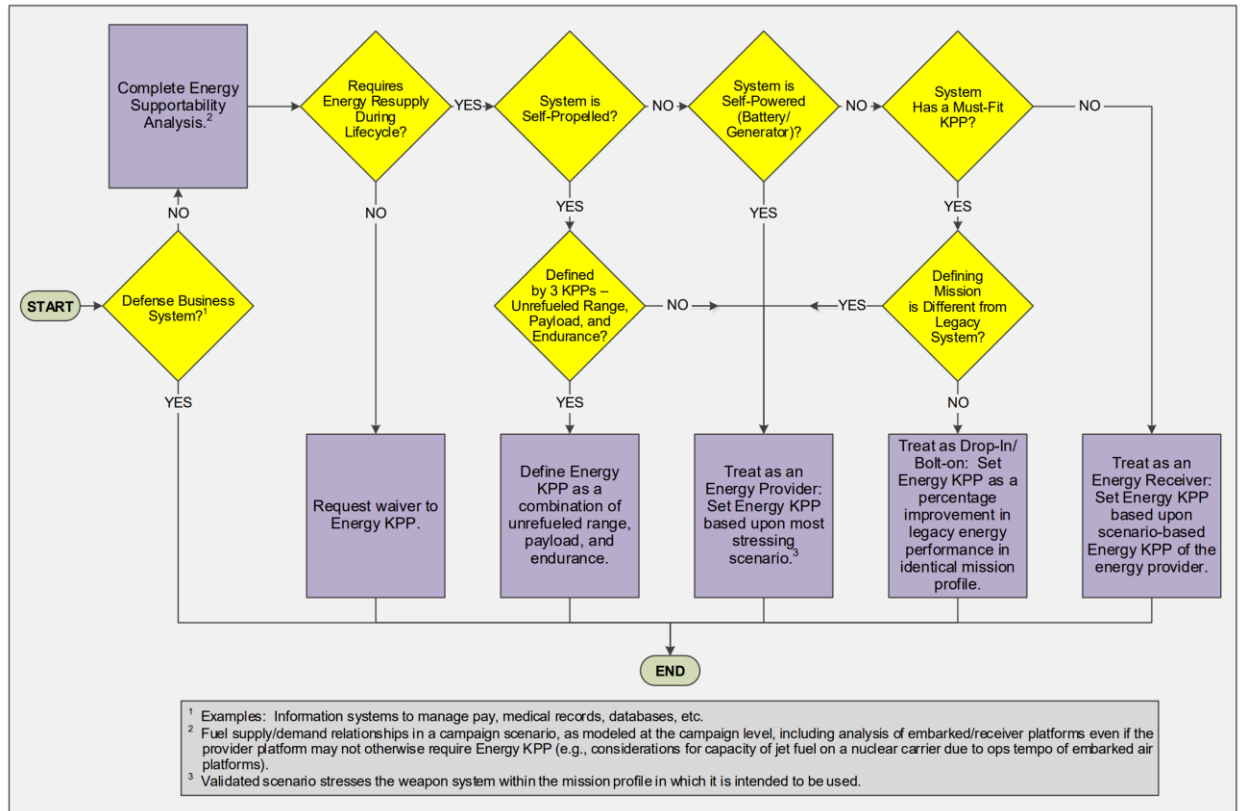


Figure 17 System Roles for the Energy KPP [42]

- Does the analysis of mission effectiveness based on energy performance identify the most critical node in the supply chain providing that power?
- Is the environment, threat, operating tempo, and known supply chain limitations considered?

In support of determining the value of these attributes, the JCIDS document outlines three steps and a number of sub steps. The three steps are:

1. “The first part of the methodology is to analyze the energy supply capacity available to the entire unit of maneuver, considering other consumers of the same energy logistics. Energy supply capacity is not a single number, but an accounting of the future capacity of the supply chain to deliver energy to the future unit of maneuver over time during the scenario.”[42]
2. “The second part of the methodology involves looking at the energy demands of the platform, the unit of maneuver, and other consumers of the same energy logistics. This analysis examines the desired performance of the system and its impact on energy sources, either as a receiver (drawing energy from other systems) or as a provider (supplying energy to other systems).”[42] (Figure 17)

3. “The third part of the methodology is to analyze the difference, in the context of the scenario and the threat, between the capacity to supply energy and the energy demand. How this difference is addressed will assist in determining the key attributes to be included in the Energy KPP, and their associated threshold and objective values.” [42]

Energy management is a critical consideration, as insufficient consideration for energy, say in the form of fuel, may leave an expensive armored vehicle stranded to be “retrieved” by farm tractors and lost, all for the want of a couple gallons of fuel.

2.2.6.5. Net-Ready Performance

There is also a 5th mandatory point to evaluate, with is the Net-Ready Performance Attribute. “The net-ready performance attribute ensures interoperability between individually developed and fielded capability solutions.” [42] This additional point of evaluation is detailed in Annex A to Appendix G of Enclosure B.

Net-ready performance applies to any information system acquired, procured, or operated by any DoD component. It requires identification of:

- Network interface compatibility (physical and protocol compatibility)
- Which networks it will be connected to? Not just DoD networks but allied networks as well.
- What metrics will describe the performance?
 - Throughput/bandwidth
 - Security
- Who will manage it? How? Under what configuration parameters?
- Susceptibility to EMSO and compliance with spectrum management in the intended operational environment.

How these questions for net-readiness should be supported for which part of the JCIDS is laid out in Table 6. This table is particularly important for two reasons: One, it outlines types of measurements for different levels of abstraction and how systems engineering is directly supporting Mission Analysis (the fundamental genesis of mission engineering). Two, the data sources and views that will convey the information to decision makers. These views (AV-1, OV-1, etc....) are DoDAF views, which has an extension for the SysML language implemented by Cameo that will be discussed in the next section.

Table 6 Net Ready Abstraction to Measure Linkage Table [42]

NR Development Step	NR Performance Attribute	Attribute Details	Measures	Sample Data Sources
Mission Analysis	Support to Military Operations	Mission Statement	Overall real-world military mission of the system	Executive Summary, CONOPS, AV-1
		Mission Activity	Joint Mission Activity	OV-5a/b or Document
		Measure	Measure to determine the success of the Joint Mission Activity supporting the military operation	SV-6/SvcV-6 or SV=7, or Document
		Conditions	Conditions under which the military operations must be executed	Document
		Network	Network the system must connect to in order to support its military operations	SV-1/SV-2 or Document
Information Analysis	Entered and Be Managed on the Network	Measure	Time to connect to and/or how system is managed on the network	OV-3, SV-6/SvcV-6 or SV-7
		Network	Network the system must connect to in order to support its military operations	SV-1/SV-2 or Document
		Measure	Time to connect to and/or how system is managed on the network	OV-3, SV-6/SvcV-6 or SV-7
		Conditions	Conditions under which the military operations must be executed	Document
		Information Element	Type/kind of data being exchanged between Joint entities	OV-3, SV-6/SvcV-6 or SV-7
	Exchange Information	Measure	Performance attribute/ measure of success per Joint information exchange	OV-3, SV-6/SvcV-6 or SV-7
		Conditions	Conditions under which the military operations must be executed	Document
		Information Element	Type/kind of data being exchanged between Joint entities	OV-3, SV-6/SvcV-6 or SV-7
		Measure	Performance attribute/ measure of success per Joint information exchange	OV-3, SV-6/SvcV-6 or SV-7
		Conditions	Conditions under which the military operations must be executed	Document
		Ensures that IS satisfies the attribute requirements	Provides traceability from the IS Measure to the derived operational requirements	OVs, SVs, StdV-1/ StdV-2, and/or SvcVs
		Accessed from the enterprise	Measures, Sample Architecture Data Sources	
		Which services do military operations require?		
Systems Engineering and Architecture	Supports all three attributes			

Mission Engineering Literature Review
Literature Review

Net-ready performance is the mission analysis layer most directly linked to systems engineering because it primarily relies on system design parameters, rather than on operational implementation. Many of these parameters are not adjustable in the field, which places additional emphasis that they be worked out in the lab and design process before-hand. For this reason, net-ready performance and analysis is based on SoS interoperability.

2.2.6.6. Determining KPPs

Having outlined the top level KPP's, the task remains to determine the applicability and scoring of a system within a mission. While there are methods to waive some of the supporting requirements for a variety of reasons (outlined in the JCIDS Manual (2018)), such waivers are beyond the scope of this analysis, and we will consider only applicable KPPs [42], [47].

While there are multiple overlaps between KPP decompositions, each applies to different aspects of an operational system. A common analysis process is:

- Determining applicability. Not all systems use all KPP's the same. For example, a cyber-warfare system does not have to consider occupant safety in the event of a crash
- Assess the threats likely to be encountered across all domains, both kinetic and non-kinetic. Threat assessments are critical to every system, but not every system will face the same threats or have the same vulnerabilities.
- Determine the level of risk. A determination is based on the assessed threats and identified vulnerabilities of the designed system. Is the current design an acceptable level of risk?
- Develop system requirements. "Develop clear, specific and achievable requirements for system components, features and technologies that will contribute to mission success against identified threats. If performance criteria are specified, ensure that they include threshold and objective measures that are realistic, testable, and measurable." [42] This is the step that links mission engineering to system engineering, and management of this step of the process will be one of the main focuses of the methodology developed here.
- Risk Management. Apply risk management through the design process based on the previous steps to guide development of the system. Consider cost, schedule, and performance to ensure that the solution is both realistic and feasible in the quantity needed.
- Present results of analysis to decision makers in a format they can consume.

Figure 18 represents the same process in a pictorial format with slightly different language. It is worth the redundancy of description here since this process is the foundation for all later work,

and if done improperly or misunderstood, it can have cascading effects for the system and capability.

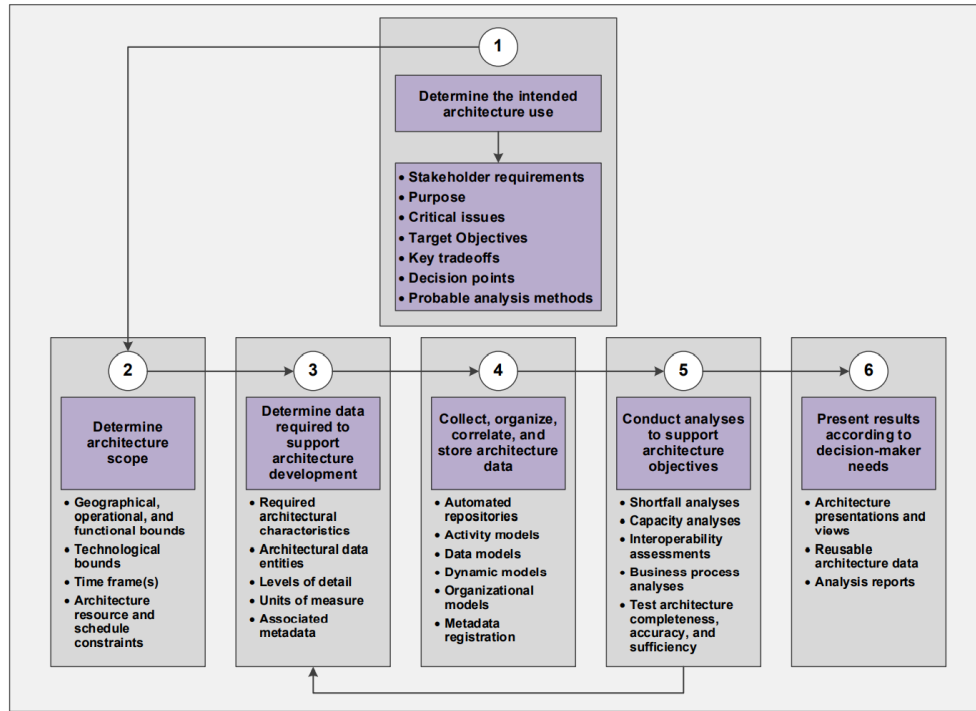


Figure 18 Pictorial Representation of the KPP Analysis Process [42]

Taking a more system centric view, the Acquisition guidebook has their own list of actions to be undertaken in support of refining the solution system outlined more generally through JCIDS [40]. These actions are:

- Study development cost by function
- Study feasibility of the prospective system in the intended operational environment,
- Conduct engineering trade-offs and analyses of alternatives,
- Study and refine viability of planned software and computers to meet KPPs,
- Simulate undeveloped equipment during software testing, and emulate the interoperability environment of the system during integration.

By leveraging the KPP process, and MBSE modeling processes, it is possible to develop robust system requirements that are traceable to the mission scenario that identified the capability gap that sparked their creation. Understanding this process is the core interface between mission engineering- and systems engineering.

2.2.7. Mission Engineering Review Summary

This chapter has covered one half of the postulated mission description, the mission engineering half. Recalling the definition of Mission Engineering as “the deliberate planning, analyzing, organizing, and integrating of current and emerging operational and system capabilities to achieve desired warfighting mission effects” [21] we can clearly see the evolution of the “planning, analyzing, organizing” [21] portions of the definition. This sets the stage to review the type of simulation, testing, and requirements that can be used to support those efforts. This will combine the previously discussed “how” and “why” with the “what”. Courtesy of a summary of current national level priorities, the discussion shifted to the top level KPPs as defined by the JCIDS process. These KPPs are crucially important for translating operational mission sets, gaps, and capabilities into forms that are usable by system engineers building the solution system. Those KPPs are briefly:

- Force protection – centers on safeguarding the people
- System survivability – focuses on increasing the resiliency of the system itself
- Sustainment – concentrates on ensuring that a system is repairable and maintainable at the appropriate level
- Energy – recognizes that modern warfare runs off of fuel, and accounts for its usage and supply
- Net-ready performance – no system operates on its own, making interoperability absolutely key on a joint battlefield that is decided during design and construction.

2.3. Model Based Systems Engineering Literature Review

The MBSE literature review chapter covers the evolution of Systems Engineering into MBSE as part of a larger effort of digitizing engineering environments and tooling. Included is a simplified overview of the SysML language sufficient for the example mission. It will then cover 7 current MBSE methodologies as they pertain to Mission Engineering and distill them into a single approach, to be synthesized in the example mission chapter. This section contains the “how” that will be used to answer the previous sections’ mission objectives.

2.3.1. Summary of Systems Engineering

The previous chapter discussed the existing process for high level requirements generation, why it’s done that way based on where it came from, and the current recommended starting points. However, the existing process is light on translating the 4 high level KPPs and the Net Ready requirement into system requirements that can be designed to. This process of requirements decomposition is central to systems engineering, and as those systems engineers are the intended recipients of the decomposed mission requirements, it is logical to utilize the same processes. To support this extension of practice, systems engineering as a discipline must first be described and explored, with emphasis on the recent trend towards digital engineering within the discipline. Digital engineering includes Model Based Systems Engineering and the associated Systems Modeling Language (SysML). Once the foundational premise and language is defined sufficiently to support analysis of current developing methodologies, those methodologies can be scraped/mined for approaches relevant to mission engineering that are the building blocks with which the mission engineering methodology will be synthesized in the next chapter.

There are a number of working definitions of systems engineering depending on the context it’s being applied in. To begin building the foundation of what systems engineering is, there are two broad perspectives that need to be analyzed, industry and government. Government is broken into both NASA and the US DoD separately as they possess different perspectives.

Beginning with the industry perspective, the International Council on Systems Engineering (INCOSE) is a definitive source for systems engineering and defines systems engineering as:

“Systems Engineering is a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods.” [48]

NASA defines it as:

“Systems engineering is the art and science of developing an operable system capable of meeting requirements within often opposed constraints” [49] Systems engineering must account for the ‘connections or interactions among the system functions and with the system environments. In many cases this includes physical, social, and logical interrelationships and environments” [50]

Lastly, the US DoD defines it as:

“Systems engineering is an interdisciplinary engineering management process that evolves and verifies an integrated, life-cycle balanced set of system solutions that satisfy customer needs” [51]

There is a common theme from all three, and that is the focus on multiple domains and interfaces over the entire life cycle of a system, leading to a very broad scope of considerations for the systems engineer. For both logical and physical systems these considerations revolve around:

- Documenting, tracking, and satisfying requirements
 - Elucidating performance and behavioral characteristics that satisfy those requirements
- Defining and managing the interactions/connections
 - Physical interfaces
 - Languages/Protocols
 - Ensuring continuity between components
- Detailing the structure that those interfaces are built on

This process takes place over the lifecycle of a system and is often visualized using a diagram called the “Systems Engineering Vee”.

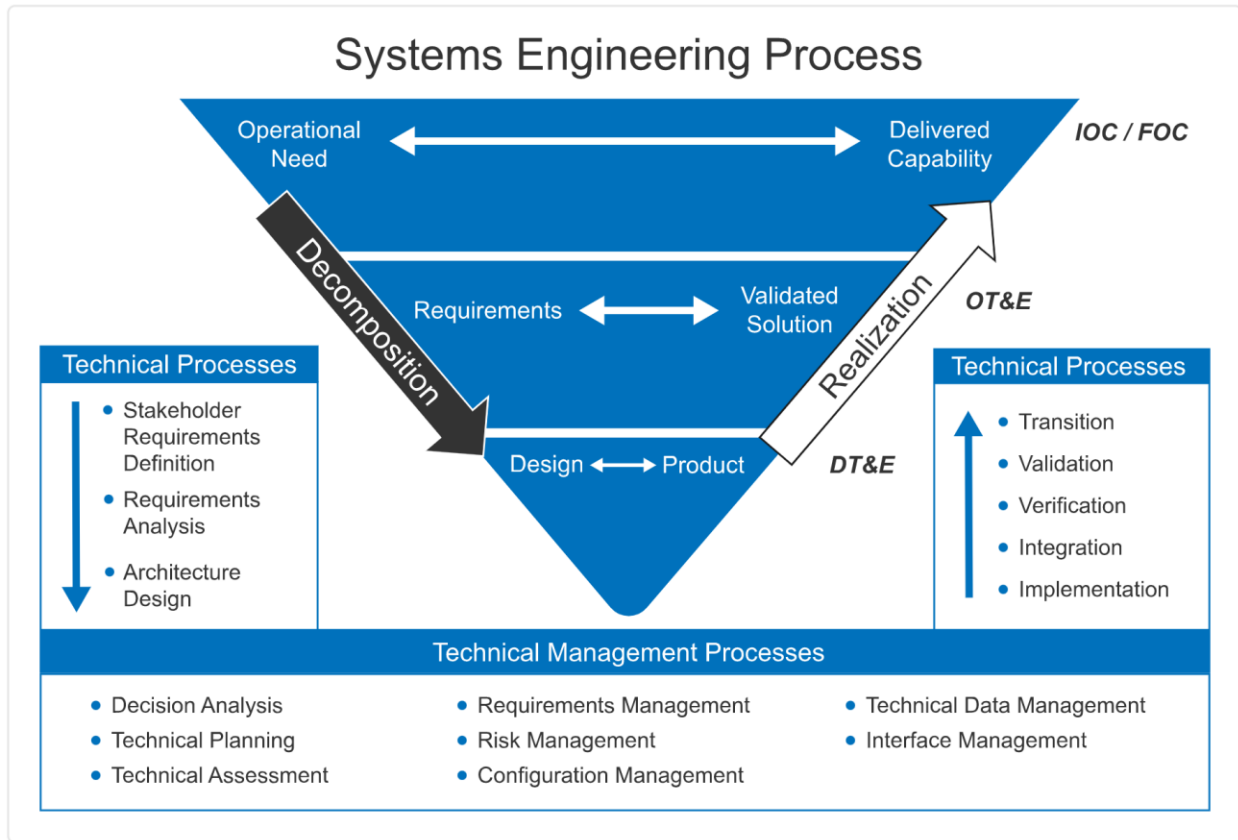


Figure 19 Linear Systems Engineering Process Diagram (SE Vee) [52]

This linear process with feedback has been used for a significant time within the US DoD acquisition community, and as such most of the points on the diagram support that process. However, partly due to advancements such as MBSE, this model’s utility and viability is beginning to be questioned, [53], [54] and it does not appear in the February 2022 revision of the Engineering of Defense Systems Guidebook [10]. This is indicative of the general shift in thought from a linear design process to a more iterative one. This will be discussed further after the introduction of MBSE, which possess the possibility of a better solution to managing the breadth of systems engineering.

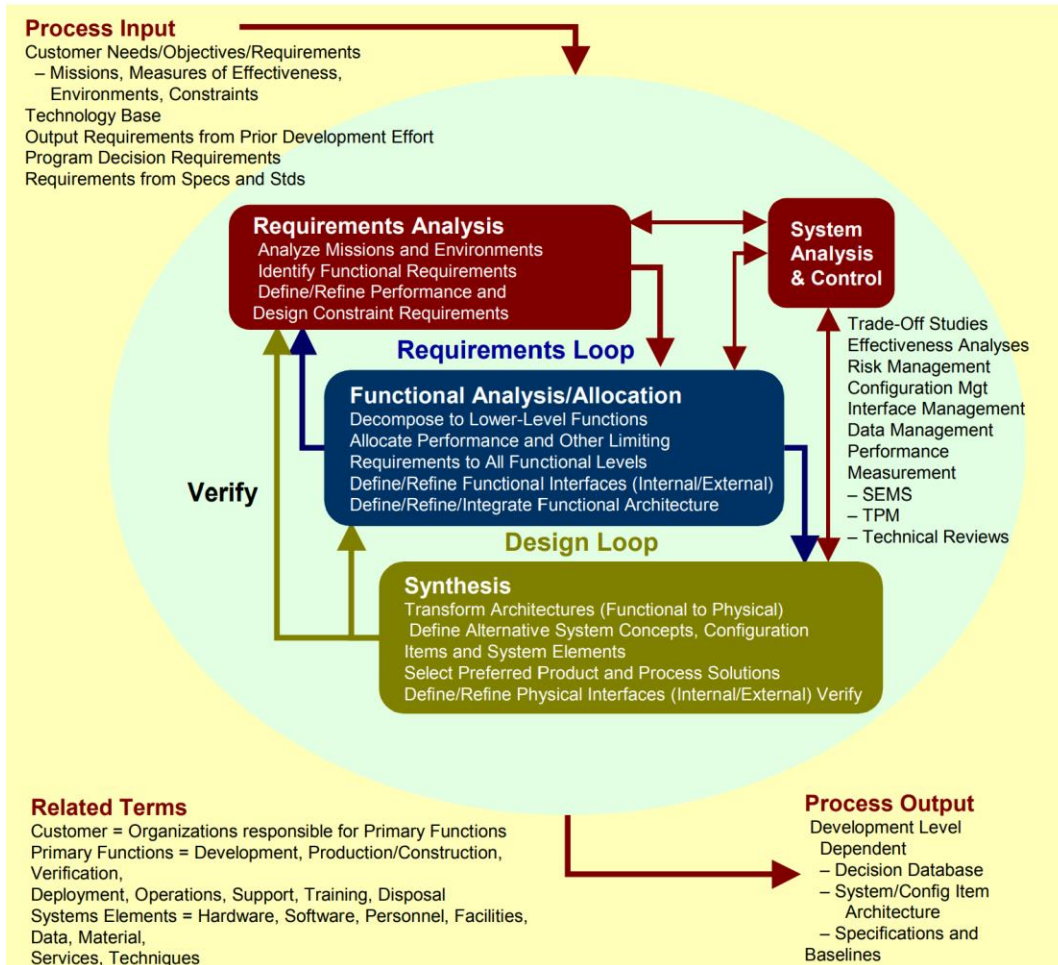


Figure 20 Iterative Systems Engineering Process [55]

The shift from linear design processes to iterative ones is not particularly new, first described in 2001 “Fundamentals of Systems Engineering” [56], and implemented in follow-on documents such as the one that produced Figure 20. While the iterative process is more complicated and difficult to manage it is more accurate to the nature of complex system development as no process is truly linear, and allows for earlier feedback to the stakeholders [57]. Both of the models for SE outlined above clearly breakout the location and roles of what has become mission engineering; a part of the requirements analysis stage, the additional detail of Figure 20, in particular the requirements loop, is worth retaining for the discussion in the next chapter.

Irrespective of the linear Vee form, the iterative loop form, or any other description of the SE process, the broad “mile wide, inch deep” position should be reminiscent to the origins of wargaming discussed in the previous chapter. Both have high level objectives and must trust the sub units (human or not) to operate properly and concern themselves with interoperability between them, even as those sub units represent disparate capabilities and interfaces. This connection has been made before in other thesis, [58]–[60], and some official US DoD

publications [10], [21]. However, no satisfactory technical linkages have been made, though the precedent has been established. The primary take away is that evolutions in one may be applicable to the other, and the evolution in systems engineering to be examined here is the advent of Model Based Systems Engineering (MBSE).

2.3.2. History of MBSE

MBSE is a relatively recent evolution of systems engineering, which itself only started really growing around the 1950's as system complexity grew [61]. Estefan (2007) tracks the evolution of systems engineering from a formal documentation in Mil-Std 499 to 2002, as the process shifted from timeline driven to a more robust event driven schedule [62]. Since 2002, MBSE has only continued to grow in adoption, but has yet to be formally codified within the DoD, instead remaining as a strong "suggestion" in support of larger goals [10], [63], [64].

Outside of the US DoD MBSE has undergone a wealth of development. Within MBSE there are three pillars to be addressed [65]:

1. Modeling language
2. Modeling Method
3. Modeling tool

2.3.2.1. Modeling Language

SysML, a Unified Modeling Language (UML)-based graphical programming language, has become a predominant tool within MBSE circles, and this ubiquity has extended to the DoD. This ubiquity within the DoD is why the language was selected for this paper. SysML evolved from UML following the development timeline below:

- Jan., 2001 – INCOSE Model Driven Systems Design working group started customizing UML for systems engineering applications [66].
- July, 2001 – INCOSE & Object Management Group (OMG) chartered OMG Systems Engineering Domain Special Interest Group (SE DSIG) [66].
- 2003 – Sanford Friedenthal & Cris Kobryn organized and co-chaired SysML Partners to develop SysML [66].
- Sept., 2007 – OMG SysML v. 1.0 specification was released [66].
- 2017 – SysML published by International Organization for Standardization (ISO) as a full International Standard (IS) [66].
- Dec. 8, 2017 – Work began on SysML v. 2.0 [66].
- Dec., 2019 – SysML v. 1.6 issued by OMG [67].

As SysML is an extension of UML, Figure 21, it allows systems engineering to draw on a longer heritage of software development and leverage the lessons learned there.

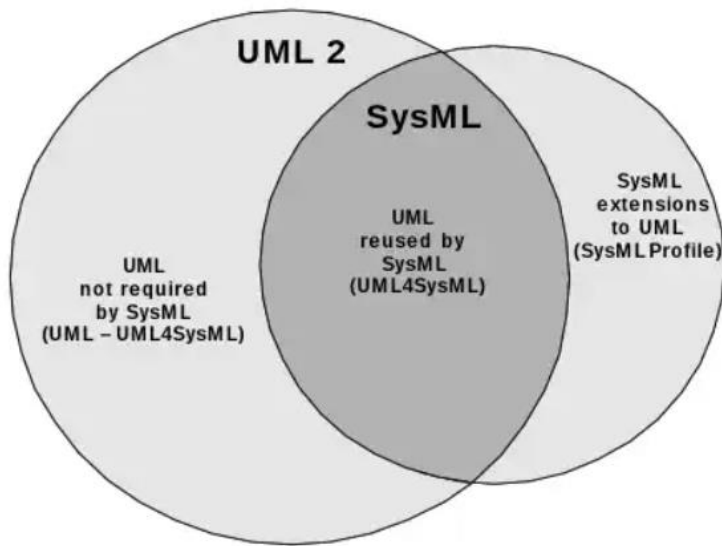


Figure 21 UML-SysML Profile Relations [68]

While the genesis of SysML and MBSE, MBSE's relation to traditional document centric systems engineering, rationale for MBSE over traditional SE, and standards evolution are important for understanding the full story of SysML's development, it is also a well-documented one. Rather than recreating that body of work here, a short list of summaries of relevant work will suffice for any reader interested in a better understanding:

- **Systematic Literature Review of System Engineering Design Methods**
Follows the development of system engineering design methods and process documentation to their implementation in MBSE at a high level [69].
- **INCOSE New Standards for System of Systems Engineering (SoS)**
Covers a brief synopsis of a number of ISO/IEEE standards on system live cycles and system of systems standards that are being adopted by industry [70].
 - ISO/IEC/IEEE 15288
 - ISO/IEC/IEEE 21839
 - ISO/IEC/IEEE 21840
 - ISO/IEC/IEEE 21841

The most relevant summarized in this document are 21840 and 21841

- **Thirteen years of SysML: a systematic mapping study (Wolny et. all) [71]**
Covers publications from 2005 to 2017 and analyzes a wide swath of the literature for inter-relations by author and location, via four prime research questions. This paper is a good source for grasping SysML publication space quantitatively, viewing trends, and locating additional papers. The relevancy here is that it highlights how the primary publication genre validation research vs proposals including solutions, such as this one

Model Based Systems Engineering Literature Review
Literature Review

[71]. This supports a point made at the beginning of the literature review about a lack of academic papers relevant to this particular problem set to cite, and hence the reliance on industry and governmental publications.

2.3.2.2. MBSE Language Taxonomy

While the origins of SysML is useful for understanding the structure and design choices made, it is generally well documented and reiterating it further will add little to the overall body of literature. What is needed here is a brief overview of the language itself to set the stage for its utilization in the coming chapter.

2.3.2.2.1. SysML Diagrams

SysML is built out of elements (blocks, interfaces, relationships, all constituents of the model itself) and views into the model to make it human readable (diagrams). As the diagrams represent the best way to make sense of the model, they are the logical first discussion point. There are nine kinds of diagrams in SysML, as outlined below:

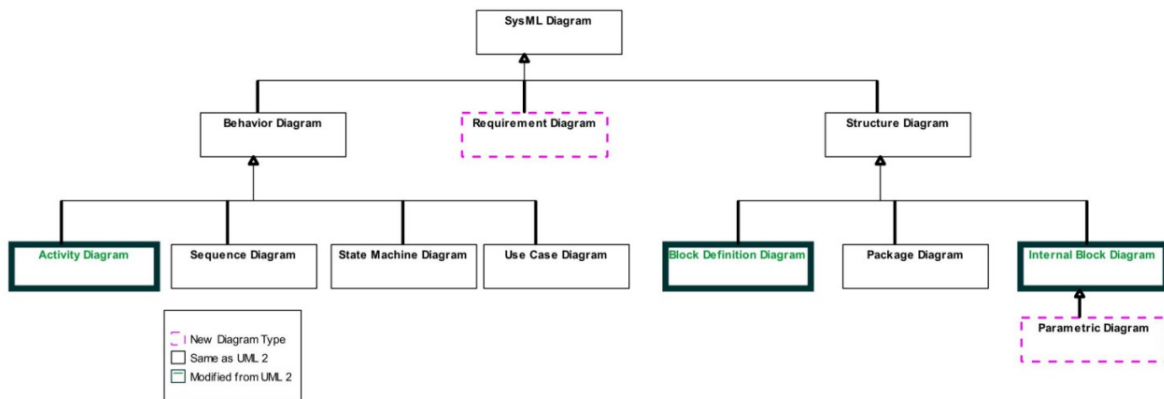


Figure 22 SysML Diagrams and Origins [67]

Most are inherited from UML, but there are some additional ones. Diagrams represent the way to view the model, but they are not the model themselves.

Each diagram in SysML serves a purpose (*italicized words are SysML specific verbiage*):

- The *package diagram (pkg)* is used to display the way a model is organized in the form of a package containment hierarchy. A package diagram may also show the model elements that packages contain and the dependencies between packages and their contained model elements [65]. The package diagram represents the best way to show the containment of the entire model, or a subset of the model. It is similar to a diagram showing file storage locations on a computer to identify the location of the elements, and what connections may exist between them.

- The *requirements diagram (req)* is used to display text-based requirements, the relationships between requirements (containment, derive requirement, and copy), and the relationships between requirements and the other model elements that satisfy, verify, and refine them [65]. Requirements diagrams are a more graphic alternative to requirements tables. Both show relations between requirements, and will be the primary view used during the initial and final stages of mission analysis. In the initial stage, it outlines the mission objectives, and in the final stage it links those high-level objectives with either more refined requirements or to elements in the model that satisfy the requirements.
- The *activity diagram (act)* is used to specify a behavior, with a focus on the flow of control and the transformation of inputs into outputs through a sequence of actions. Activity diagrams are commonly used as an analysis tool to understand and express the desired behavior of a system [65]. Activity diagrams work best for linear analysis or functional flow within a model. Of particular note about the actions that can be used on an activity diagram is the subtype opaque action, which allows for the insertion of text-based code to execute blocks that are too cumbersome to model in a visual language like SysML.
- The *block definition diagram (bdd)* is used to display elements such as blocks and value types (elements that define the types of things that can exist in an operational system) and the relationships between those elements [65]. This diagram is the primary view of macroscopic structure and composition.
- The *internal block diagram (ibd)* is used to specify the internal structure of a single block. More precisely, an *ibd* shows the connections between the internal parts of a block, the interfaces between them, and any items or information flowing across that connection [65]. *ibd*'s functions as the microscopic view of a system and the minutia of interoperability and function.
- The *parametric diagram (par)* is used to express how one or more constraints, specifically equations and inequalities, are bound to the properties of a system. Parametric diagrams support engineering analyses, including performance, reliability, availability, power, mass, and cost. Parametric diagrams can also be used to support trade studies of candidate physical architectures [65].
- The *sequence diagram (seq)* is used to specify a behavior, with a focus on how the parts of a block interact with one another via operation calls and asynchronous signals. Sequence diagrams are commonly used as a detailed design tool to precisely specify a behavior as an input to the development stage of the life cycle. Sequence diagrams are also an excellent mechanism for specifying test cases [65].
- The *state machine diagram (stm)* is used to specify a behavior, with a focus on the set of states of a block and the possible transitions between those states in response to event occurrences. A state machine diagram, like a sequence diagram, is a precise

specification of a block's behavior that can serve as an input to the development stage of the life cycle [65]. A state machine is usually used when there become enough activities that they need to be grouped for either performance or accuracy reasons.

- The *use case diagram (ucd)* is used to convey the stakeholders, CONOPS that a system performs and the actors that invoke and participate in them. A use case diagram is a black-box view of the services that a system performs in collaboration with its actors [65].

Both the state machine and the use case diagram will appear at the highest levels of a system or a mission context, to provide an overarching view of the behavior and structure respectively. The nature of mission engineering focusing on higher level usage of systems will additionally influence the distribution of diagram usage, primarily with regards to the state machine diagrams, a difference that will be discussed further in the next chapter.

These diagrams have multiple paths of interrelation as well:

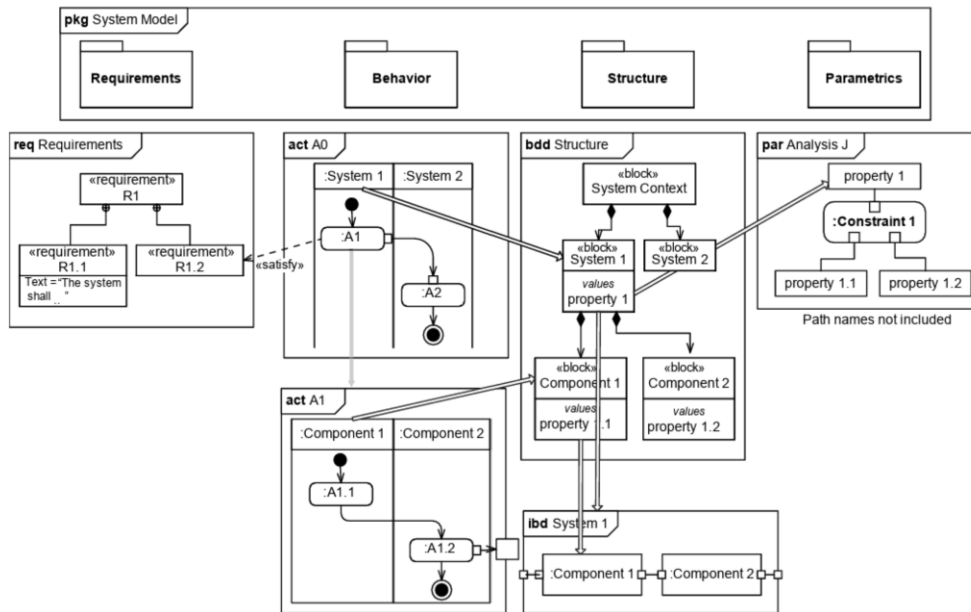


Figure 23 Diagram Interrelations [72]

Figure 23 depicts relations between elements across diagrams. Requirements are satisfied by a behavior from an activity diagram (act) allocated to a system for execution, or a value property on a block definition diagram (bdd). That system has parts (component 1, 2) and descriptive values (property 1) to represent what is being modeled. That system has interfaces that govern its internal structure (internal block diagram from perspective of system 1) and a component relation to other components within the system (internal block diagram from perspective component 1, 2). These relationships support analysis conducted using information from multiple blocks in a parametric diagram (par) stored within the highest-level block being

referenced (System 1 for Analysis J). Once the syntax of SysML is understood, this paragraph worth of information can be easily conveyed to another systems engineer with all the additional detail that was left out of this equally sized paragraph.

2.3.2.2.2. SysML Elements

In order to fully understand these diagrams, the other critical aspects are the elements and relations. As of SysML specification 1.6 there are nine elements that need to be defined to understand the methodologies in the next section in addition to subtypes of each that will be introduced as they appear (for a full list, reference the specification document, section 8.2):

Block

Diagram: *bdd*

Formal Definition: A “block defines a collection of features to describe a system or other element of interest. These may include both structural and behavioral features, such as properties and operations, to represent the state of the system and behavior that the system may exhibit.” [67]

Mission Engineering Example: Blocks typically represent systems or components as in regular systems engineering. However, in ME, it can also represent missions as a collection of systems with potentially unique structures/interfaces and a set of parameters that are of interest to that mission. As such, a block can be an aircraft, radar, antenna, or strike mission, either specifically or in general using functional/logical supertypes.

Common Subtypes: <<domain>>, <<context>>, <<subsystem>>, <<activity>>

Ports, Flows, Interfaces

Diagram: *ibd, bdd*

Formal Definition: Ports: “Ports are points at which external entities can connect to and interact with a block in different or more limited ways than connecting directly to the block itself. They are properties with a type that specifies features available to the external entities via connectors to the ports” [67] Ports can also be nested for more complex interfaces.

Item Flow/Flow: “Item flows specify the things that flow between blocks and/or parts and across associations or connectors. Whereas flow properties specify what “can” flow in or out of a block, item flows specify what “does” flow between blocks and/or parts in a particular usage context” [67]

Interface: “Defines a set of operations and receptions, a behavioral contract that clients and providers will conform to.” [65]

Mission Engineering Example: Ports implement interfaces by being typed by the interface. For example, in that interface there could be:

- The interface block itself that represents a physical connection such as an RJ45.
- An association block specifying the type of physical connection/wire (CAT5)
- Flow properties specifying the types of flows that can flow across it (TCP/IP)

Managing interfaces often becomes some of the most detail-oriented portions of the model.

Common Subtypes: <<Full>>, <<Proxy>>, Atomic vs non-atomic

Constraint Block

Diagram: *bdd, ibd, par*

Formal Definition: “Constraint blocks can be used to specify a network of constraints that represent mathematical expressions such as $\{F=m*a\}$ and $\{a=dv/dt\}$, which constrain the physical properties of a system. Such constraints can also be used to identify critical performance parameters and their relationships to other parameters, which can be tracked throughout the system life cycle.” [67]

Mission Engineering Example: As will be explored in the next section, the majority of actual calculations are better served outside of Cameo

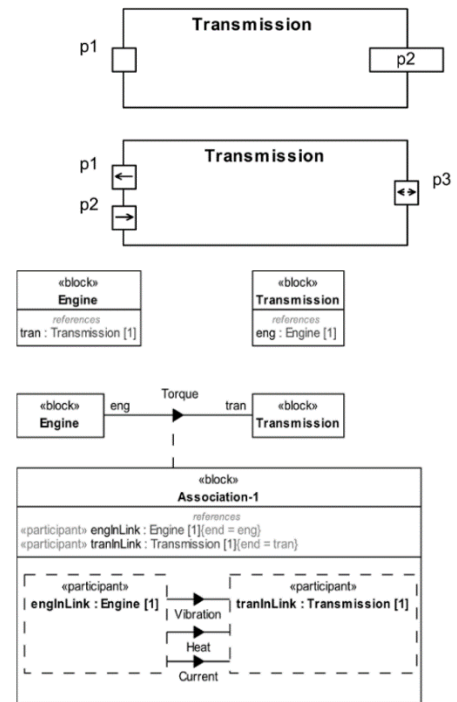


Figure 24 Pictorial Representation of a set of Ports (top), and Item Flow (bottom) [62]

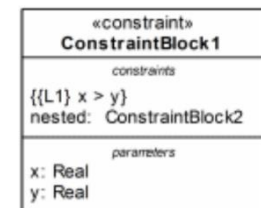


Figure 25 Pictorial Representation of a Constraint Block [62]

as methods of simulation are less developed in SysML and its tools. Cameo's strengths are structure or behavior, and constraint blocks generally serve only for light weight calculations or visual aids. Case in point, "simulation" is not mentioned in the SysML specification. However, Constraint blocks (and their behavioral analogue Opaque Actions subtype) serve an important role as the location to interact with exterior simulations. Constraints (and Opaque Actions subtype) can be written in a number of languages. However, the seeming inability to include additional libraries limit their utility in acting as an interface to a more complete version of that language's compiler.

Common Subtypes: <<moe>>, <<objectiveFunction>>

Activity, Actions

Diagram: *act*

Formal Definition: Action: "An action represents some form of processing or transformation that will occur when the activity gets executed during system operation." [65]

Send Signal: "A send signal action is a specialized kind of action that asynchronously generates and sends a signal instance to a target when it becomes enabled." [65]

Accept Event: "An accept event action is the partner of the send signal action in asynchronous behaviors; an accept event action is the element you use in an activity to convey that the activity must wait for an asynchronous event occurrence before it can continue its execution." [65]

Mission Engineering Example: Multiple actions of varying types are combined to form an activity. Activities can be called on by other activity diagrams through call behavior action. Activities are often best used for linear processes contained within a state as part of a state machine. The traditional kill chain process, such as F2T2EA [73] is a linear process from a particle view (think of fluid flow) of a single packet, as failure anywhere in the chain fails the engagement.

Common Subtypes: <<opaqueActions>>, <<callBehaviorAction>>, <<acceptEventAction>>, <<sendSignalAction>>

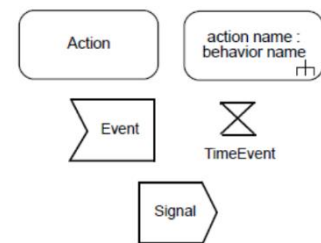


Figure 26 Pictorial Representation Several Types of Actions [62]

State Machines, States

Diagram: *stm*

Formal Definition: “The state machine represents behavior as the state history of an object in terms of its transitions and states.” [67]

Mission Engineering Example: A state machine is used when a system or mission is complex enough that activities need to be grouped and executed only when certain criteria are met. If the traditional F2T2EA process is viewed from a flow perspective, then the failure in one state (based on contained behavior, likely activity diagrams) would revert the process to the previous state/prevent progression i.e., if a target was found, but with insufficient accuracy to clear a weapons release, then the exit transition criteria to transition from “target” to “engage” are not met and the state does not change.

Common Subtypes: <<compositeState>>

Allocations

Diagram: bdd, allocationMatrix, tables

Formal Definition: “Allocation is the term used by systems engineers to denote the organized cross-association (mapping) of elements within the various structures or hierarchies of a user models. A typical example is the allocation of activities to blocks” [67]



Figure 27 Pictural Representation of a General Allocation Relationship [62]

Mission Engineering Example: ME uses allocation the same way that regular systems engineering does, to link form and function. This is particularly useful as the same multifunctional system may be allocated to different parts of the kill chain as a function of other available assets and the geographic dimension. A radar may be allocated to fulfill the “find” and “fix” in one mission, and only the “find” portion of another pending a number of factors external to the system design.

Common Subtypes: N/A

Use Case

Diagram: ucd

Formal Definition: “The use case can also be viewed as functionality and/or capabilities that are accomplished through the interaction between the subject and its actors. Use case diagrams include the use case and actors and the associated communications between them.” [67]

Mission Engineering Example: Use cases are generally early life cycle diagrams when the function of the system is still being determined. In mission engineering this is the CONOP, which

lays out a potential use of the system, which when linked together with other systems creates a web of systems with potential uses. A path through this web is a specific scenario or mission to be analyzed, which is the distinction between a “kill chain” and a “kill web” [65]. In general, these diagrams are made then referenced as a means of interpreting non-engineering missions, scenarios, CONOPS into a format that is within the larger ontology of the eventual systems engineering model. As a result, use case diagrams are primarily found at the top levels of a model.

Operations

Diagram: *act, stm, bdd*

Formal Definition: “An operation represents a behavior that a block performs when a client calls it. Most often, an operation represents a synchronous behavior.” [65] Operations contain other behaviors responsible for execution of a called methods.

Mission Engineering Example: Operations, while generally representing synchronous behavior, can be toggled to asynchronous and used in place of a base action. This has one primary advantage when dealing with closely tied system-function trade spaces: ownership. Actions are owned by the activity they appear on and there is no referencing them on other activity diagrams without duplicating the action or transferring ownership. Operations, as they are owned by a block, can appear on multiple activity diagrams while being owned by the system responsible for execution. A system can provide the same function to multiple missions, either concurrent or not, and this retention of ownership is worth the slight overhead incurred by use of operations in this manner (primarily remembering to toggle off asynchronous behaviors).

Common Subtypes: N/A

Requirements

Diagram: *req, bdd, table*

Formal Definition: “A requirement may specify a function that a system must perform or a performance condition a system must achieve. SysML provides modeling constructs to represent text-based requirements and relate them to other modeling elements.” [67]

Mission Engineering Example: Similar to allocations, requirements at the mission level remain largely unchanged. Top-down system requirements generated through a process such as JCIDS already flow through mission level requirements, just with a different level of design and analysis than lower system requirements. A mission requirement may be to ensure a communication linkage reliability, a given location, weather, and a set of equipment within

which to trade. The different types of requirements are useful to differentiate the different locations by which a requirement is intended to be satisfied (e.g., behavior for functional, value property for performance, port for interface).

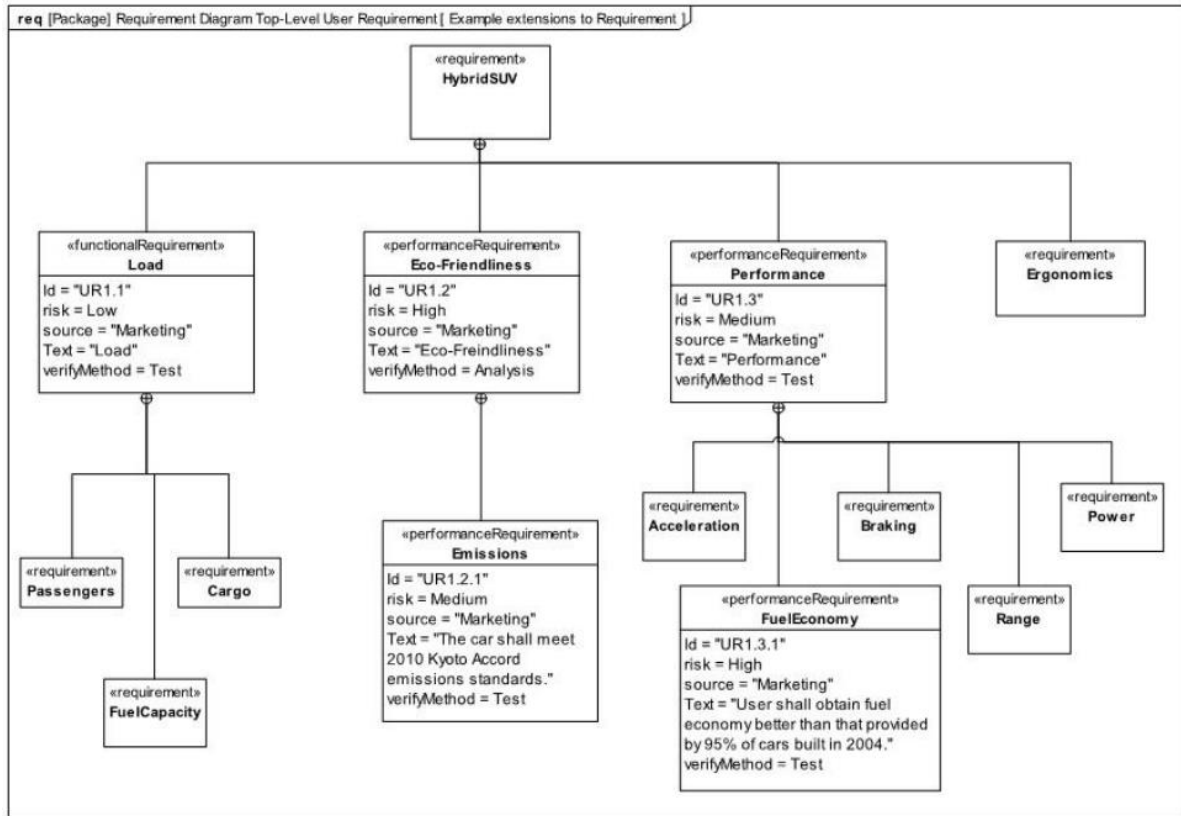


Figure 28 Pictorial Representation of a Requirements Diagram with Multiple Types of Requirements [62]

Common Subtypes: <<FunctionalRequirements>>, <<PerformanceRequirements>>, «interfaceRequirement» Table E-3 of SysML Specification 1.6 for more and their constraints

In keeping with the pictorial theme of SysML, the aggregated interrelation between these elements is best displayed as a map, rather than text. Figure 29 maps the elements discussed here, as well as some other ones in a language agnostic manner. The key takeaway from the diagram is that there is no obvious “start here”. Generally, the process will begin with requirements that are decomposed to the level where systems begin to get involved (policy to mission level). This figure is an important one to retain, as it lays the general linkages of the above elements, while diagrams will only show portions of the total relationships.

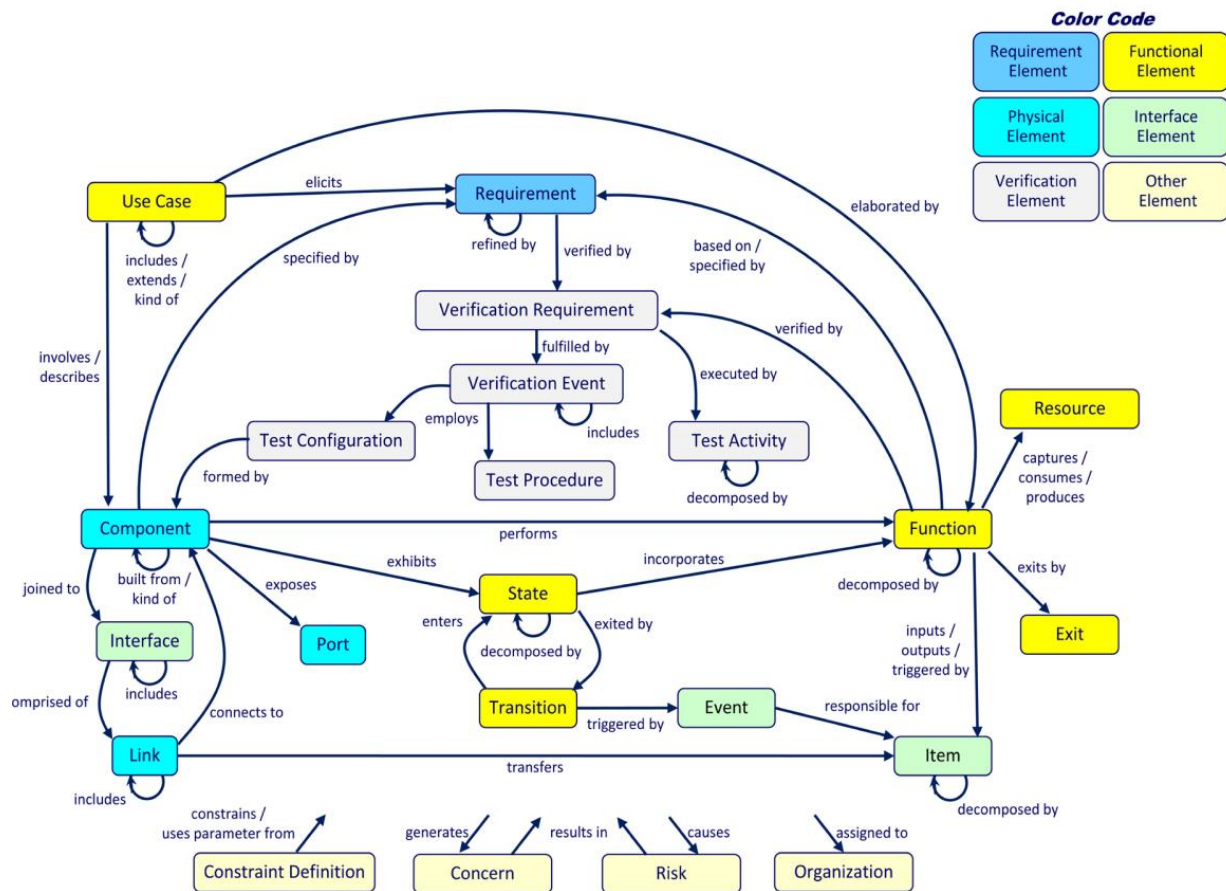


Figure 29 Map of Model Elements Interrelations [74]

In addition to the map, a quick reference table that links the definition listed above to the elements they type are summarized below in Table 7.

The elements listed previously in this section represent the elements of definition of a system. However, in SysML there is a key distinction between definition and the usage of that definition. As a result, there are other elements not expressly defined previously in this section. The information in Table 7 is a concise overview of these additional elements.

Table 7 Elements of Definition Mapping to Elements of Usage [75]

Elements of definition:		Elements of usage:
Actor	types	Reference Property or Lifeline
Block	types	Part Property, Reference Property, Lifeline, Atomic Flow Port, Flow Property, Object Node, or Constraint Parameter
Value Type	types	Value Property, Constraint Parameter, Atomic Flow Port, Flow Property, or Object Node
Constraint Block	types	Constraint Property
Flow Specification	types	Nonatomic Flow Port
Signal	types	Atomic Flow Port, Flow Property, or Object Node
Interface	types	Standard Port
Association	types	Connector
Activity	types	Call Behavior Action
Interaction	types	Call Behavior Action
State Machine	types	Call Behavior Action

Elements of usage:		Elements of definition:
Part Property	typed by	Block
Reference Property	typed by	Block or Actor
Value Property	typed by	Value Type
Constraint Property	typed by	Constraint Block
Constraint Parameter	typed by	Value Type, Block
Nonatomic Flow Port	typed by	Flow Specification
Atomic Flow Port	typed by	Block, Value Type, or Signal
Standard Port	typed by	Interface
Flow Property	typed by	Block, Value Type, or Signal
Connector	typed by	Association
Call Behavior Action	typed by	Activity, Interaction, or State Machine
Object Node (includes Pin and Activity Parameter)	typed by	Block, Value Type, or Signal
Lifeline	typed by	Block or Actor

2.3.2.3. General Terms

There are a number of general terms and concepts within MBSE taken from other fields and applied here, and are often not expressly defined within a system engineering context. For that reason, they need to be addressed before discussing the methodologies:

- Black box vs White box
- Design engineering vs reverse engineering
- Logical-Functional-Physical architectures

2.3.2.3.1. Black box vs White box

Black box is a view of a system that has: functions/operations to be performed (but not the “how”), key properties, and external interfaces [76]. A white box shows that information and more, such as the internal characteristics of a system including component parts or methods to behaviors. Those components can themselves be in black boxes or white boxes. The general approach, when modeling from the top down, is to build a black box, then add definition, at which point the black box becomes a white box comprised of black boxes. The process repeats. The end product will be comprised of white boxes with two primary exemptions: lowest level of required fidelity and the platform level within the mission.

The lowest level of the model is, by definition, without parts or internal specification (as that would require components, which themselves are then the lowest model element). The lowest level of a model must be reactive to the needs of the customer, as there is a trade between fidelity and computational performance. There will be a central model with as much detail as possible, from which the shallower models (for when performance is key) branch off.

The primary reason that platforms within a mission are black boxes is the expectation that, for every mission customer, the high-level black box mission view is a desired output (the OV-1 analogue). The other reason is the desired lowest-levels of fidelity used are defined ad hoc per the customer.

2.3.2.3.2. Design Engineering vs Reverse Engineering

There are two broad categories each Engineering, design engineering and reverse engineering. The first involves building a new system, in part or full, while the second is modeling an existing system to better understand its function.

Design engineering can be broken down into two categories; “clean slate” and “middle-out”. Clean slate requires a new system, new problem, and the need to be solved in isolation without interaction with other systems, something that is becoming increasingly rare in systems engineering and long since nonexistent in mission engineering [77]. As a result, the primarily relevant type is “middle-out” engineering, and is usually the type referred to as design engineering here. The JCIDS process is built to support design engineering, with the top-down requirements generation and implementation,

Reverse engineering is simply the practice of modeling to document, digitize, and understand a legacy system in its current form, and does not imply foreign system exploitation. At the current juncture, with the number of legacy systems in inventory, reverse engineering is a dominant portion of modeling at the mission level, and the current level of uniformity within MBSE (or lack thereof) means that many of the models that have been built under either engineering type are not compatible. A sign of effective adoption at system level design of

these systems is the shift away from reverse engineering to design engineering at the mission level.

However, the method and tools to conduct both types of engineering are largely similar. The methodology outlined by the end of this paper will be applicable to both.

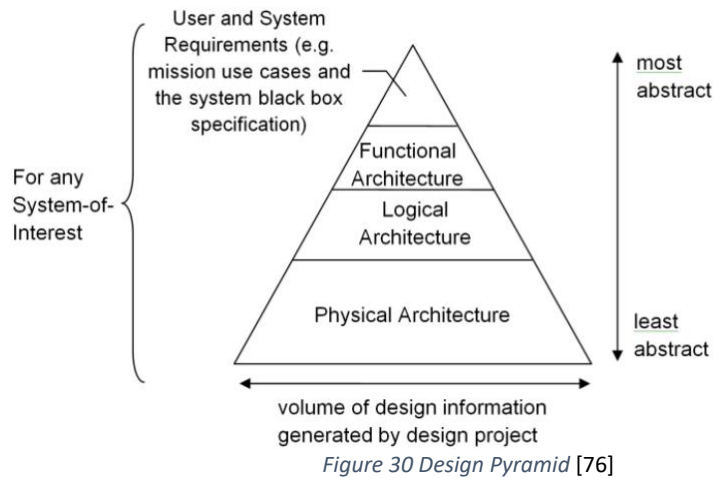
2.3.2.3.3. Logical- Functional-Physical Architectures

There are three levels of any design model: logical, functional, and physical [76]. During the design process, in accordance with a process such as JCIDS, a set of desired capabilities are specified, which are then broken down into functional requirements (both structural and behavioral). For instance, a functional architectures set of

requirements could be to transit from point A to point B within 3 hrs. carrying 200 lbs. of material. Once those requirements are established the logical options can be established. The logical options could be a: car, truck, ship, helicopter, or fixed wing aircraft. These are classes of vehicles and are not specific representations of an implementable solution, but are defined only so far as needed for whatever analysis is being conducted, unconstrained by detail but still constrained by common sense. Once the optimal class of travel is identified (including the designed to specifications), say truck, then the physical architecture can be implemented. This is the detailed systems engineering problem that will yield the system that will implement the specifications and be physically built.

As an early insight to the next chapter, below is an example of how this hierarchy is traversed as part of mission engineering.

- The mission provides the context from which the functional architecture is derived.
- The system of interest is identified, which will likely be the only logical system at the beginning because it is rare for an entire order of battle to be in design stages



concurrently, as most large entities (including the military) have existing systems upon which a new system is an incremental improvement .

- The rest of the missions' systems are identified and physical models either built or gathered for use as contextual systems.
- The logical system is optimized within context of the other physical models.
- The specifications of the optimal logical system are passed to the designing organization, often a contractor.
- The designing organization develops and produces the product, ideally generating a physical model as an early deliverable.
- The physical architecture model is passed through the mission analysis process as a subtype to the optimized logical system to ensure that the design continues to meet requirements and provide opportunities for insight.
- The designing organization delivers the product, which transitions to a program of record at which time the product becomes context for the next generation of systems of interest, and the process repeats.

This process is obviously tailored towards a design process, rather than a reverse engineering process. As the adoption of MBSE is relatively recent compared to the age of many legacy systems within the DoD, there are a large number of systems developed without MBSE models of any kind but are still crucial to the context of the system of interest. In terms of the functional-logical-physical hierarchy in a reverse engineering scenario, it is possible to directly link functional-physical as there is no express need to conduct a class level optimization, as the system already exists. It is worth noting that, as the library of modeled systems develop, there is benefit to having a logical structure for configuration management and alignment to the simulation engines inputs. The concept and implementation of a logical class supertype for simulation and definition of subtype black boxes of a physical architecture will be explored in the next section.

2.3.3. Current MBSE Methodologies

As outlined thus far, adoption of MBSE practices is proliferating, yet no unified methodology exists, and due to differences in domain/objectives, it is likely that one will not exist in the near future. As such, there are a number of existing methodologies that have been developed and are sufficiently relevant to this use case to warrant exploration. The exploration undertaken in this section pertains to the methodologies and their most relevant attributes to mission engineering. The highlights will be assembled in the next chapter into a cohesive method. This is not a comprehensive review of these methodologies, but rather a selection of features that have been determined to be relevant through a review of the methodology and of relevant

survey papers. The following seven methodologies have been identified through a combination of experience and literature searching [49], [65], [78]–[81].

1. INCOSE Object-Oriented Systems Engineering Method (OOSEM)
2. IBM Rational Telelogic Harmony-SE, IBM Rational Unified Process for Systems Engineering (RUP-SE)
3. CORE Vitech MBSE
4. JPL State Analysis (SA)
5. System of Systems Interface Modeling
6. Hypermodeling and SAIC Validation Profile
7. Weillkiens Systems Modeling Process (SysMOD)

Prior to delving into the details of each methodology, it is worth briefly reviewing the top important KPPs for a methodology:

- KPP 1: Capture and analyze traceable requirements in a manner that can be simulated.
- KPP 2: Define system structure (both logical and physical components and interfaces) and behavior (operations, states and transitions), in accordance with the needs of the simulation framework.
- KPP 3: Facilitate simulation and analysis by linking the simulation framework to the model serving as an authoritative source of system truth in support of analysis of alternatives.
- KPP 4: Conduct 1-3 at both the system and system of systems level within the SysML environments available at an US DoN Enterprise level (Cameo Systems Modeler) [82].

There is the ever-present question of model verification and validation (V&V), which is something that is an ingrained part of mission engineering. V&V represents a significant challenge that first needs a methodology to validate and verify. V&V is therefore part of possible future works, and is only considered notionally here.

It is worth emphasizing that more general reviews of these methodologies have been conducted elsewhere, and this analysis is restricted to the sections most applicable to Mission Engineering use-case, as a full review of each would be redundant.

1. INCOSE Object-Oriented Systems Engineering Method (OOSEM)

OOSEM follows a modified Vee model from the top down, with emphasis on object reuse, while following all four KPPs outlined above, plus model V&V. It possesses a heavy emphasis on logical structure as the first level of implementation to satisfy requirements [81].

While OOSEM shares similarities with other methodologies (such as Harmony or RUP SE [72]), it has a greater emphasis on structure. OOSEM is the implementation of object-oriented coding

practices in the systems engineering model, leveraging encapsulation, specification, and element reuse. This is particularly conducive given the genesis of SysML from UML discussed previously, as the SysML language itself is object oriented at its core. While this is useful for MBSE, it is doubly useful for mission engineering given the heavy emphasis on simulation, and that many external simulation software at the mission level are object oriented themselves (such as AFRL's AFSIM [83]). OOSEM is one of the most influential methodologies of those discussed, and further reading is recommended, including Chapter 17 of "A Practical Guide to SysML"[72], Section 9.4 of the INCOSE Systems Engineering Handbook [13], or several related academic papers: [76], [84]–[87]

The synopsis of what is to be gleaned from OOSEM is:

- Top-down approach and process
- Practices for encapsulation, inheritance, and reuse as part of object-oriented structure modeling

OOSEM Top-Down Approach

OOSEM follows the process outlined in Figure 31 (top) which details the steps needing to be taken and is read starting with a mission and progressing from left to right, first on top of the diagonal line, then below in the gray zone. Figure 31 (bottom) outlines the same process, but better shows the flow of actions across the cycle.

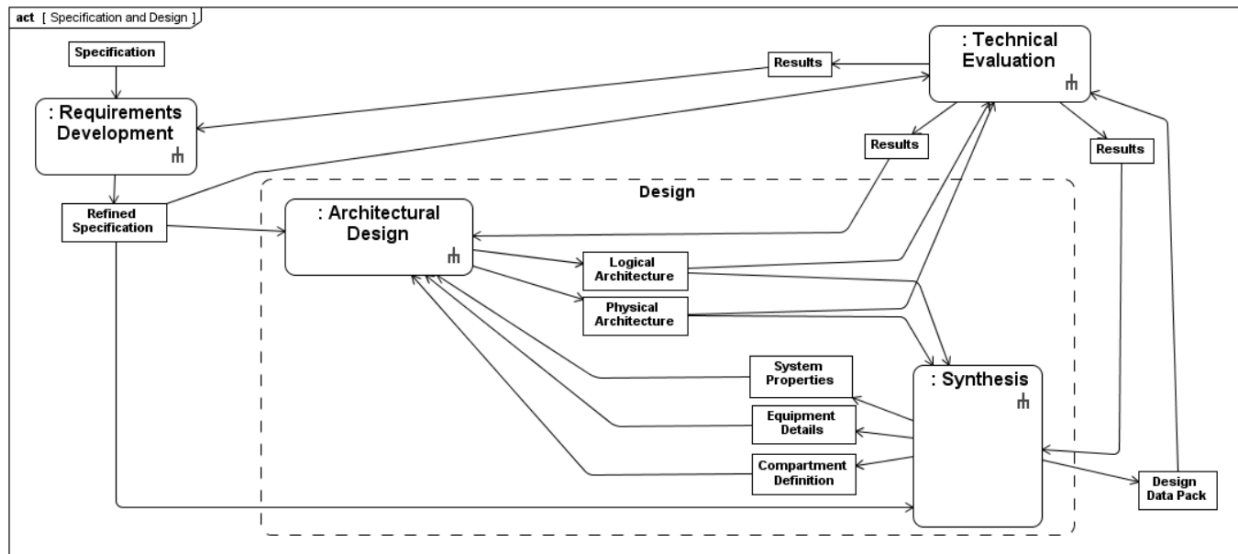
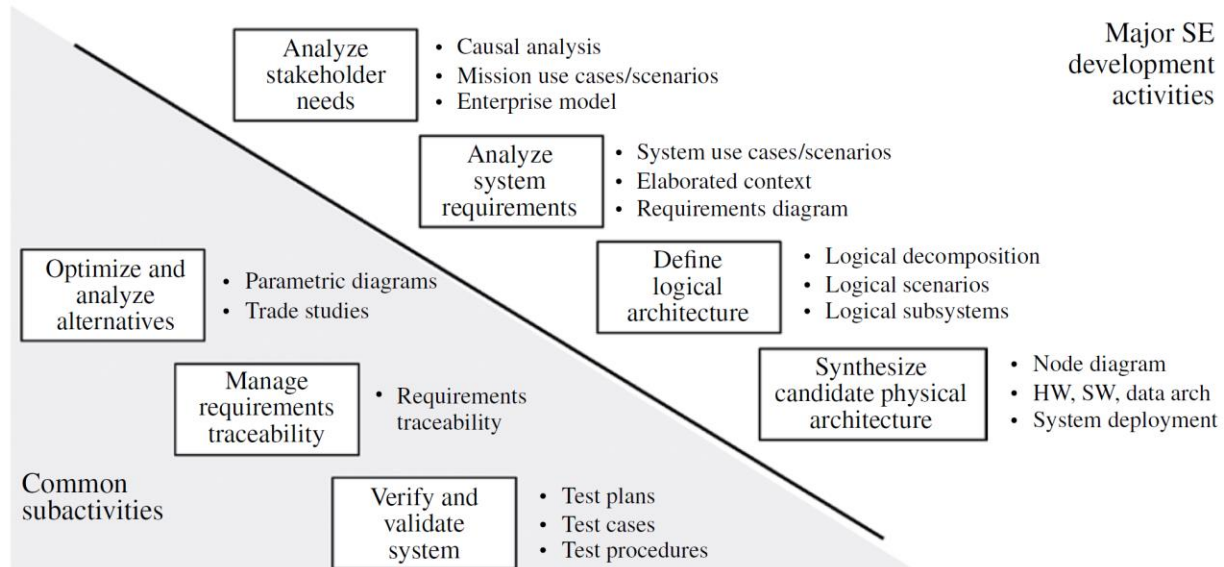


Figure 31 OOSEM Process Steps (top) [13] OOSEM Process Flow (bottom) [76]

Aggregating Figure 31 yields the following listed steps:

1. Start with a mission and mission KPPs outlined in previous chapter
2. Analyze stakeholder needs, and refine those KPPs into less abstract requirements
3. Translate those needs into system requirements
4. Define logical architecture (systems, scenarios, behavior)
5. Analyze logical architecture systems within the scenario
6. Synthesize 1 to N (1...*) physical architecture subtypes to the logical architecture
7. Analyze physical architecture subtypes, optimize best solution.

8. Ensure that the optimized physical architecture subtype is traceable to original requirements, and that traceability is documented
9. Verify the system design through testing

While this method could work for behavioral analysis of a mission, the emphasis on inheritance, for example, does not carry well to behaviors where subtypes are less well represented. Behavioral analysis will be better served by more functional decomposition centered methodologies such as IBMs Harmony.

Inheritance and Reuse

Inheritance uses the generalization relationship introduced within the *SysML Elements* section. Figure 32 shows an example of three layers of inheritance, all still logical black box elements. (The blocks are logical, as there is no specification to the values inherited by the bottom layer of blocks. Additionally, they do not display inherited values with numbers assigned to them, they have not redefined them, and therefore analysis could not be conducted. However, this is not conflicting with inheritance, and is beneficial if the system of interest is a type of star sensor and the analysis focused there vs the star tracker/star mapper layer).

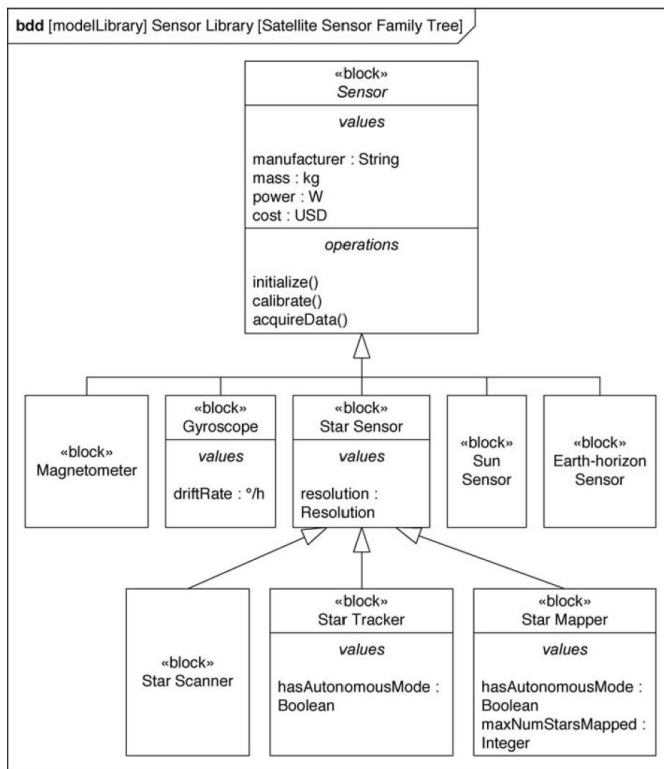


Figure 32 Example of Generalization Inheritance [65]

The advantage of this structure is two-fold. The first advantage is a consistent and common definition of elements of the same type (e.g., all star trackers subtyped from the shown star

tracker block will have the same defining characteristics) allowing construction of a traceable “genetic” tree of a component. The second advantage is the single point of class definition eases interfacing with exterior simulation software, as the definition of “sensor” in a software such as AFSIM need only be consistent with the sensor block in the model, rather than spread across elements in the model. This enables configuration management as the external tool changes and introduces compartmentalization to reduce unapproved modification of data.

Although generalization is the most common inheritance, there are other types as well. Figure 33 shows two additional methods: instantiation and extension.

Instantiation is creating an instance of the template block. There is no room or flexibility to change either behavior or interfaces. This is suboptimal for most applications due to the inability to modify or refine a subtype from the supertype. The use case for this is when system properties are the only aspect being evaluated (e.g., the effect of varying the velocity of an aircraft on its overall survivability in a given mission on a given path). Additionally, as a close look on Cameo’s operation will reveal, the simulation tool kit converts all elements to instances prior to execution anyway. If traceability within the design of experiments (DoE) is paramount, it would save a step to use instances for narrow scope trades.

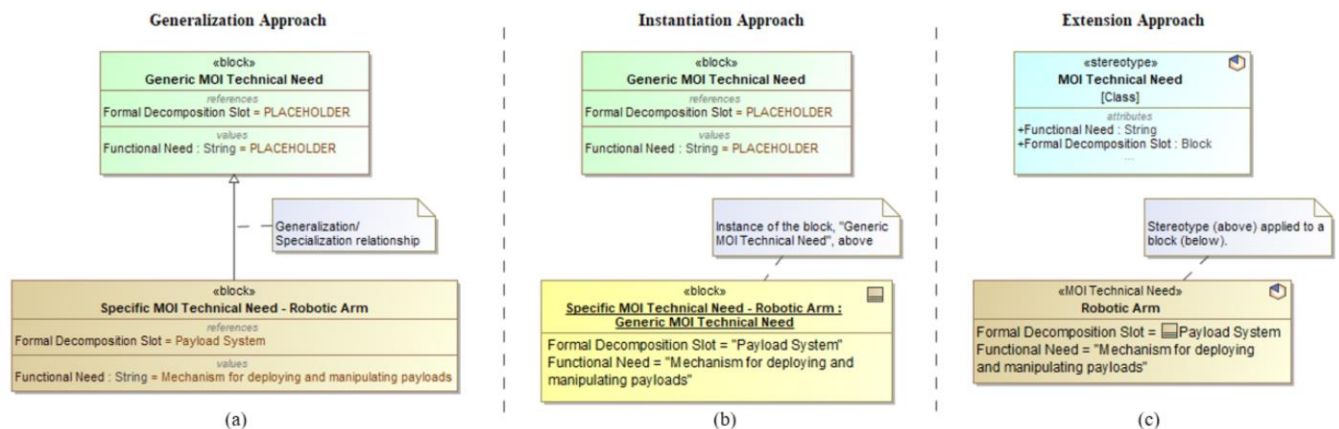


Figure 33 Alternate Methods of Inheritance [88]

While each inheritance approach has its uses, the Generalization approach will be used here due to the increased flexibility of specification compared with instantiation and extension. Instantiation is unsuitable because the child needs the ability to have different interfaces, as ports are not specified on the parent. Extension doesn’t work either due to the limitations that are induced by stereotypes being restricted to properties. While extension may be more flexible for deep models, the implementation of functional roles that the generalization here will be applied to is “wide”, as opposed to “deep”, which avoids some of the complexities associated with multiple layers of generalization. However, extension for purposes of meta-navigation, as discussed in Hypermodeling and SAIC Profile, will be used.

2. Rational Unified Process (RUP) and IBM Rational Telelogic Harmony

Both the Rational Unified Process and the Rational Telelogic Harmony and methods are created by members of the IBM Company for systems engineering. A standout difference between them is the available tooling, as the RUP is centered on IBM's Rhapsody tool and the Harmony is a tool-agnostic approach which is much more applicable for use here. While there is the ability to read in Rhapsody models into Cameo, the additional room for error that such translations introduce makes adoption of the methodology as written incompatible with KPP 4. From the aggregate of IBM's methodologies, the key points are:

- Functional decomposition and focus on behaviors
- Levels of abstraction
- Intra-requirements relationships (between relationships)

Harmony

IBM Harmony is a tool-agnostic methodology, following a service-request approach [81]. The approach is focused on an executable model, a key attribute for a model to support the simulation capability as identified in KPP 3. It also follows an iterative version of the traditional SE Vee as shown in Figure 34:

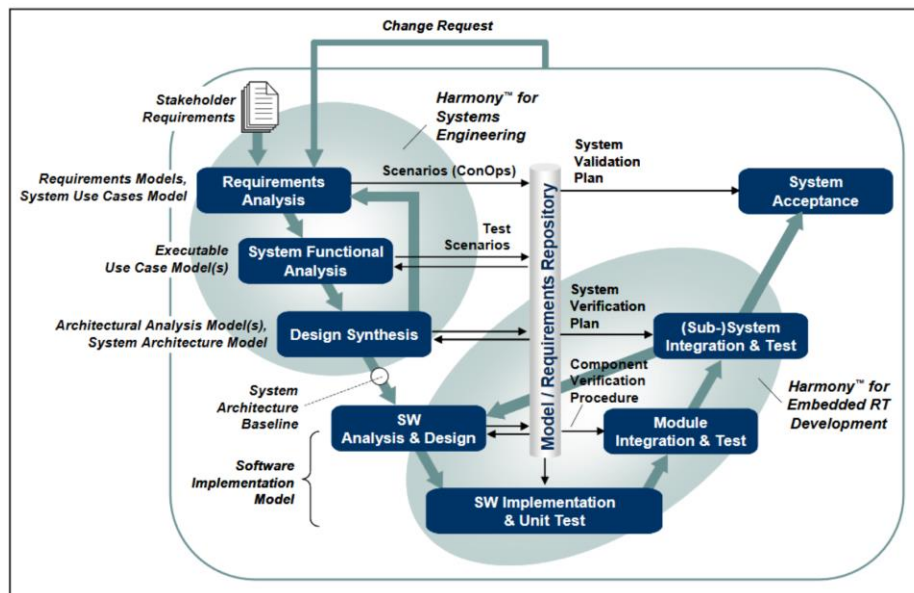


Figure 34 IBM Harmony Systems/Embedded Design Process [89]

The emphasis of Harmony is behavioral modeling using activity and sequence diagrams, and the behavioral relations to those requirements through functional decomposition. As an example of the variable emphasis placed on different diagrams by different methodologies, Harmony

makes very little use of *bdds*, linking behavior directly to the detailed view of an *ibd* [89]. The process Harmony undertakes terminates at outlining the physical structure:

1. Identify/derive required system functionality.
2. Identify associated system states and modes.
3. Allocate system functionality/modes to a physical architecture.

This emphasis on behavior as the interface between structure and requirements fits the behaviorally heavy Mission Engineering use case, and will be central to the larger process outlined in the next chapter. This will be combined with OOSEMs logical system layer as part of modeling out the system of interest (SoI), a concept which will be fully introduced in the next chapter.

RUP SE

The RUP SE emphasizes business modeling, following a spiral model with some object-oriented elements [81]. However, at the project stage that mission engineering is conducted at, a spiral model's focus on risk reduction reduces the agile, experimental nature of the analysis of alternatives through additional work, counterproductively introducing unnecessary levels of risk into the process.

From a practical design level, RUP SE centers on use case flow design activities, which work excellently for software, as the original RUP SE was designed for, but does not satisfactorily address the logical-physical hardware. However, RUP SE is a process in addition to a methodology, and there are a number of points that are relevant and worth highlighting in the simulation and requirements of the central mission engineering use case.

- Abstraction
- Requirements relationships (allocated vs derived requirement)

Abstraction

Table 8 RUP SE Architecture Model Level and Purpose [79]

Model Levels	Expresses	Examples	
		Worker	Logical
Context	System black box—the system and its actors (through this) are a black-box view for the system; it is a white-box view for the enterprise containing the system.	Role definition, activity modeling	Use case diagram specification

Analysis	System white box—initial system partitioning in each viewpoint that establishes the conceptual approach.	Partitioning of system	Product logical decomposition
Design	Realization of the analysis level in hardware, software, and people	Operator instructions	Software component design
Implementation	Realization of the design model into specific configurations	Hardware and software configuration	

Table 8 outlines the four layers of abstraction in RUP SE. While it spans the entire spectrum from Mission (Context) to Implementation, the Context and Analysis levels are the two most relevant to mission engineering. The Design level is also relevant and is fed the output of “system config/architecture” outlined in Figure 20.

Requirements Relationships

SysML provides a number of relationships, and similarly to other language specifications, allows for multiple ways to code the semantics. One of the prime areas of uncertainty is the intersection of either multiple systems collaborating to fulfill a larger functional requirement, or a single system of systems being treated as a single multifunctional platform fulfilling a requirement independently. This distinction is a matter of perspective, and will be addressed later, but once a systems perspective is identified, RUP SE provides a compelling breakdown of the sole vs shared ownership of a functional requirement. “A locality or subsystem requirement is allocated if a locality or subsystem is assigned sole responsibility for fulfilling a system requirement. A locality or subsystem requirement is derived if it is identified by studying how the subsystem or locality collaborates with others to meet a system requirement.” [79] It is worth noting that this relationship only holds between requirements, as a derived relationship requires both ends to be a requirement entity [65], [72].

3. CORE from Vitech

Vitech has been a contributor to the SysML specification for several years, and while their methodology is nominally tool agnostic, it is partial to the CORE toolset owned by Vitech [67], [79]. The methodology consists of concurrent, iterative design cycles modeling towards a previously identified level of detail, metaphorically “peeling back the onion” as it is commonly referenced to in papers discussing the methodology [78], [79], [81], [90] [91].

The methodology is written general enough that the approach can be applied to mission engineering, and there are multiple process points to be taken away, but no specific syntactical points. These higher-level process points are:

Model Based Systems Engineering Literature Review
Literature Review

- Overall constructive framework (breaking simulation out of the structural SysML model)
- Four step process
- Onion model and concurrent processes

Overall Constructive Framework

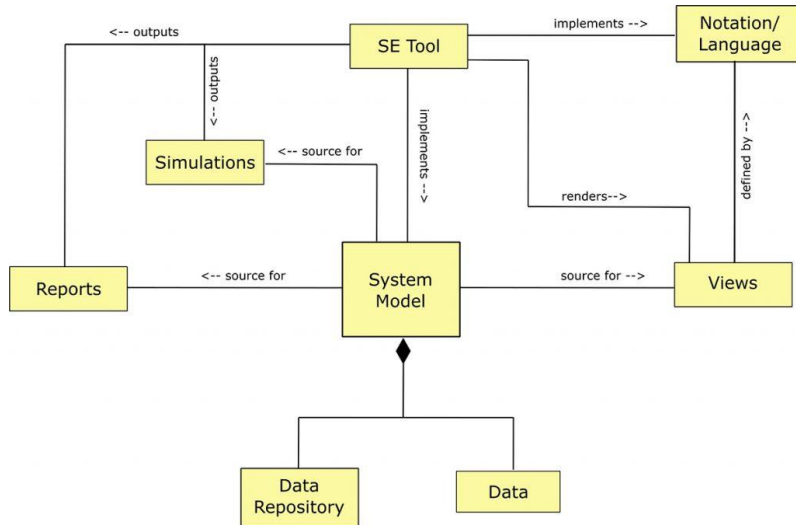


Figure 35 Overall Constructive Framework for Systems Engineering [92]

The overall framework described in Figure 35 highlights that the system model is not the only important pillar, even as part of a system design. The breakout of simulation from the SysML model into another model is not unique to CORE, appearing in other methodologies as well. However, all of this focus primarily on system level modeling, where simulation is CAD level of abstraction [93], [94]. This theorem holds true for mission engineering as well, if modified as below:

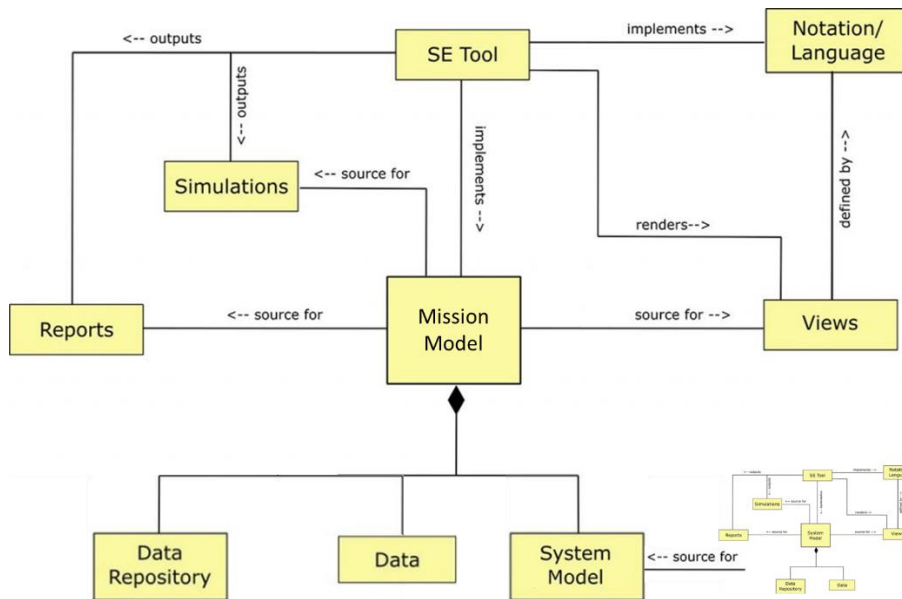


Figure 36 System Level Constructive Framework Adapted to Mission Level (Adapted From [92])

This modification is consistent with the work outlined thus far as one of the core underpinnings is that mission engineering is related to systems engineering. This figure will be revisited in its adapted form at the start of the example mission. It highlights how the information will flow at a macro level, particularly that the simulation is outside the model, and that the reports and views need to be communicable outside the model so that non-technical consumers of the information can understand it. This last point, on how information is to be consumed and the consequential standards that models must conform to, will be explored in the next section addressing diagramming schemas such as UPDM, UAF, and DoDAF.

Vitech CORE 4 Step Process

CORE follows a 4-step process when constructing design models [77]. The four steps are:

- Requirements Analysis
- Functional Behavior Analysis
- Architectural Synthesis
- Validation and Verification (beyond the immediate scope of the paper, but included for completeness)

These steps run in the above order, and iteratively checks with prior steps to ensure compatibility and consistency, as shown below in Figure 37:

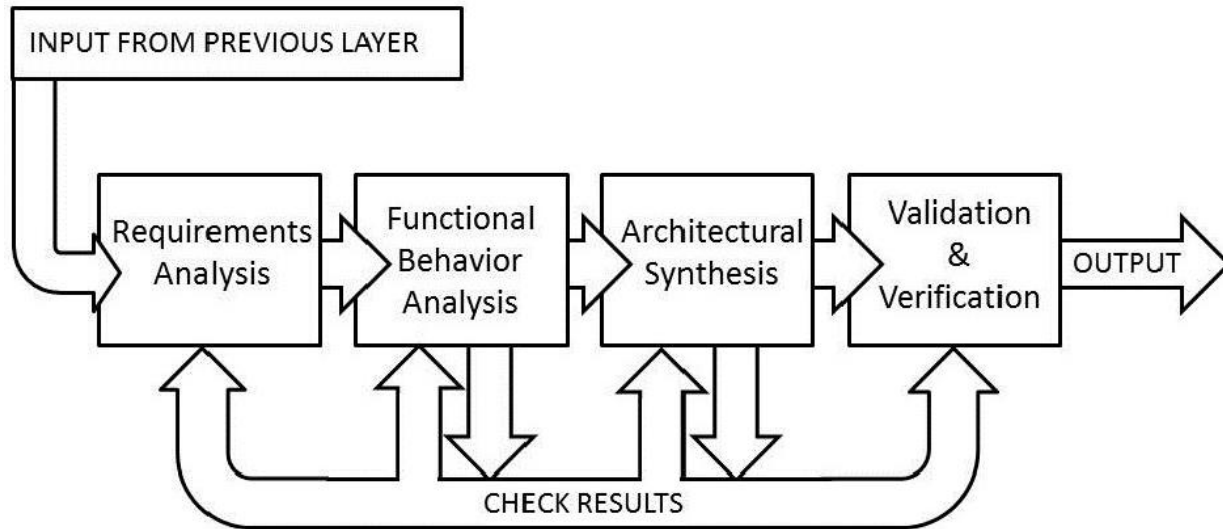


Figure 37 Vitech CORE 4 Step Process [77]

While the process is largely similar to other processes (such as IMB’s Harmony discussed above), few other methods state the process as concisely as CORE does. In contrast to Harmony’s close focus on activity to *ibd* linkage, CORE looks at a larger scoped implementation of functional decomposition. This functional decomposition is the behavioral analysis preceding the architecture (for each layer of the system, which is iteratively decomposed), something that several methods for systems engineering miss. Supporting this point, it is worth noting that INCOSE reaches system design/synthesis through system requirements rather than behavioral analysis. The conclusion here is that both should be addressed concurrently, yet INCOSE does not specify that the requirements are born of behavior, but rather system parameters such as weight. Neither approach on its own adequately captures the complexity of mission behavior, or system complexity, so it is appropriate to apply both concurrently. As will be addressed in the next chapter, this concurrent approach is the one used here.

The four steps of the CORE model map to four domains within the modeling architecture, as outlined in Figure 38. The primary takeaway from the figure are the domain transitions and how they relate to each other. In practice the domains will all be in the same model, including the V&V domain to the extent that the V&V strategy is documented. The relationships between requirements, behavior, and architecture in particular are worth expressly stating here in Figure 38.

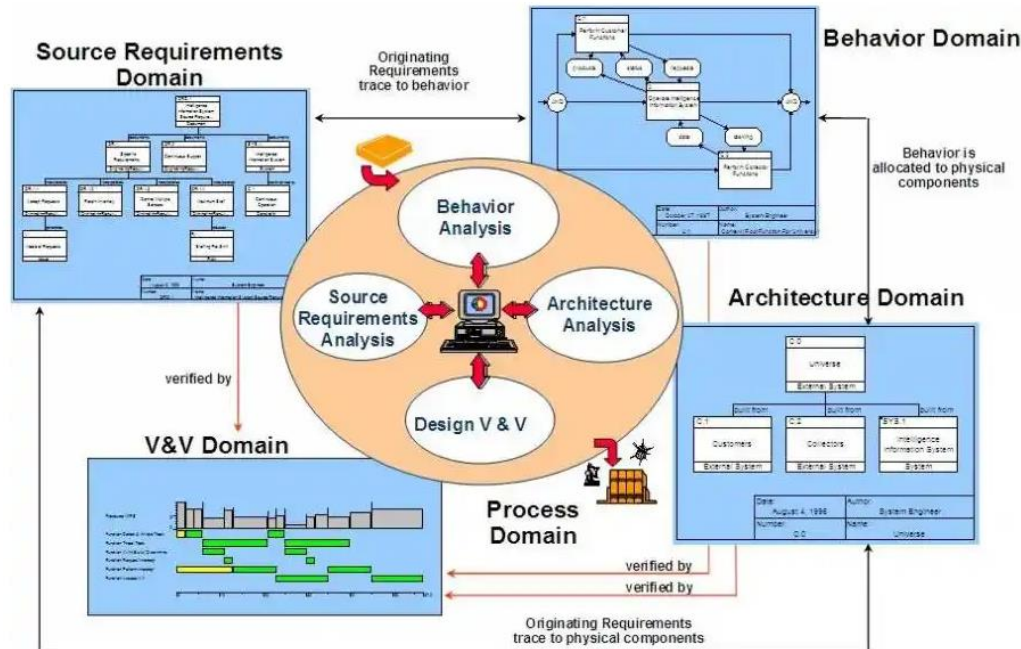


Figure 38 Vitech CORE 4 Domains [95]

Onion Model

Similarly, to the four steps above, the onion model is primarily a catchy name for iterative decomposition. Refining the top-down approach of JCIDS, this approach addresses behavioral then structural requirements at each level, resolving them before moving down to the next level. This approach is recursive, increasing the fidelity of the model at each level.

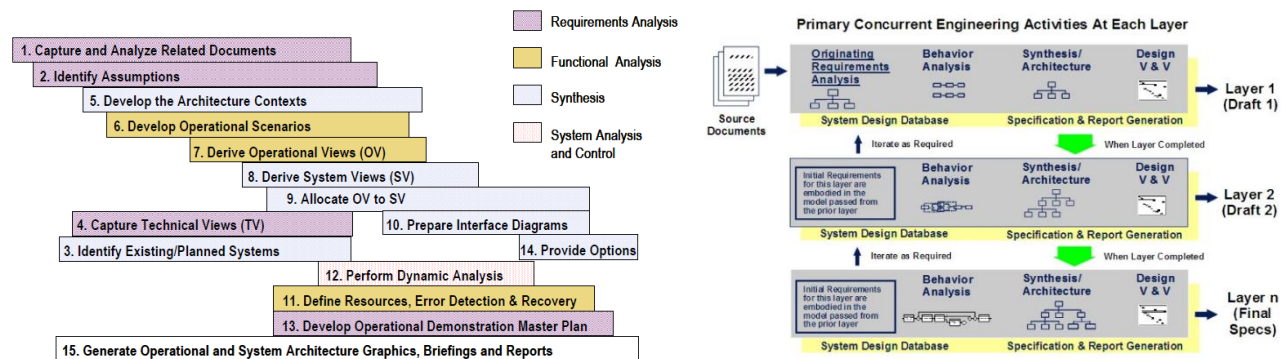


Figure 39 Vitech C4ISR Model Decomposition Process (Left) [96] and Recursive Layer Process (Right) [79]

Figure 39 shows the macro and microscopic views of the process, where each step from idea to final design options uses the recursive process on the right to add fidelity to each of the left-

hand objectives. The rationale to stepping through each layer is the ability to identify and correct errors as they occur before they can propagate [91].

This process can be done concurrently within an iteration, where the scope is small enough that all four domains can be considered concurrently, and reach a solution before moving to the next layer of the model [91]. This helps reduce the complexity dealt with at any one point, adhering to one of the core tenants mentioned previously, “the evil of complexity”, even if it only does so within a design sprint [97].

4. JPL State Analysis

The JPL State Analysis approach is designed to outline the behavior of a system as a function of lower-level behaviors wrapped into moment-by-moment higher level states, and focusing on the transition between states [79], [81]. This generates a product reminiscent of a control system, and traditionally maps requirements to behaviors, similar to the IBM methodologies (originally software, in the mission engineering domain it would be people) [78], [79]. The structure of the process is captured below in Figure 40:

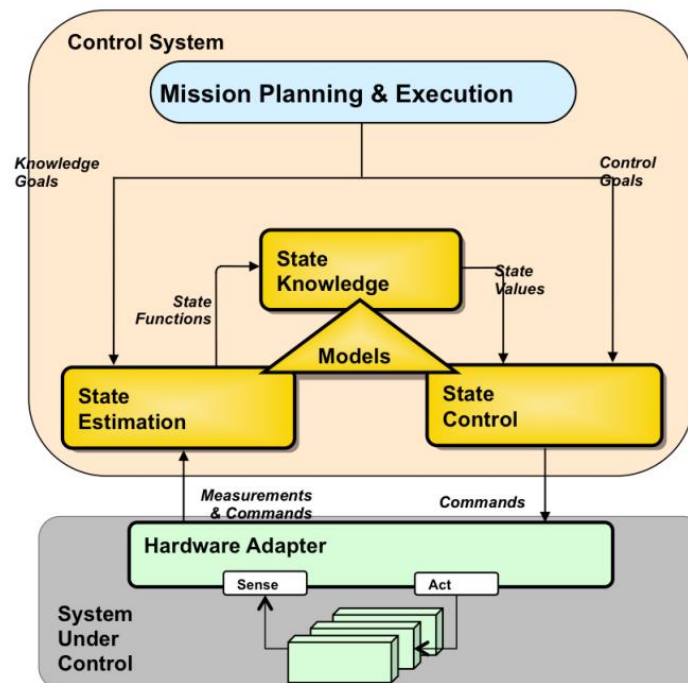


Figure 40 JPL State Analysis Structure [98]

The primary use of state analysis for mission engineering is encapsulated in the larger colored box, where the system under control will be touched on as part of the system model from outside of a state machine. For mission engineering, the primary uses of this methodology are:

- State machine creation process for top level state machines at mission and top of system level.
- Use of stereotypes in conjunction with generalized super types for widely applied and enforced inheritance, as well as for use with metachaining discussed later under Hypermodeling.

The tenants described in this state analysis approach will guide the construction and analysis of state machine represented behavior within the mission to link mission objectives, operator intent and processes, and behavioral interfaces. The tenants to produce a mission level state machine are:

- Differentiating system under control from control system. (i.e., specific sensor in a specific environment and a general behavior model)
- Use of stereotypes for needed class definitions.

System Under Control vs Control System

Within the JPL methodology is a distinction between the system being controlled, and the controlling system [99]. The system being controlled is comprised of estimators and hardware adaptors communicating via state variables. The paper's example is a flow sensor (estimator) providing flow rate (measurement) to the control software(controller), which then issues commands (functional state variable) to valves (hardware adapter) controlling flow of the fluid [98]. Note the distinction between the controlling entity and the input entity. Mission engineering also makes this distinction. Within mission engineering, there is a similar distinction between the controlling entity and the input entity. For example, command and control (C2) systems have with input sensors (estimators) that control offensive actions (hardware adapter). Rephrasing the previous example, a radar (estimator) providing air tracks (state variable) to the C2 post (controller), which then issues commands (state variable) to surface to air batteries (hardware adapter) directing the engagement. This distinction will become important when it comes to codifying roles and responsibilities within a kill chain, and outlines the structural relationships with a state machine, as the state variables will function as the state transition triggers.

The other benefit to having the control systems broken out is that it provides a logical entry point for user intent, and reporting to them, something currently underserved [100]. As human-machine teaming matures this will become more important to explore, as the human becomes further removed from the action [101], [102]. Detailed study of manned/unmanned teaming, training aids, and the interface with the outlined mission engineering methodology is beyond the scope of this paper.

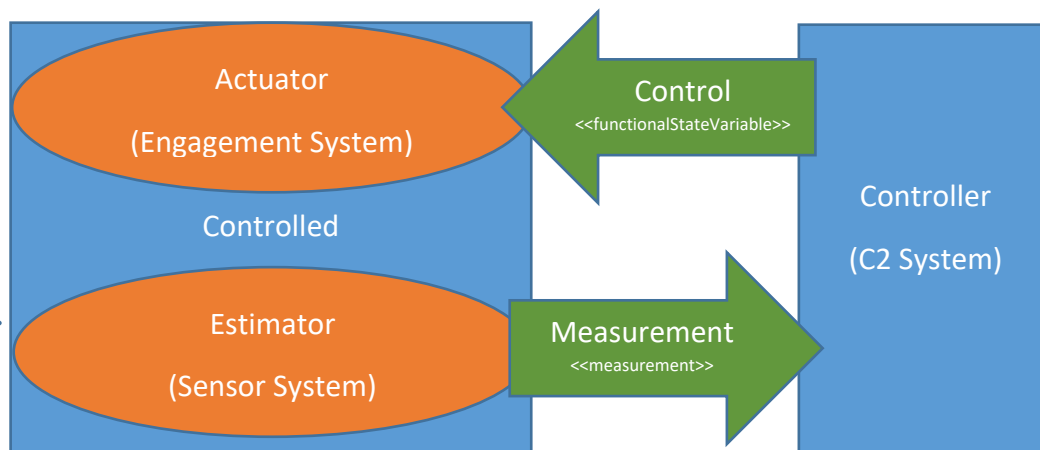


Figure 41 JPL Information Flow Structure [98]

Usage of Stereotypes vs Supertypes for Element Typing

In SysML there are two ways to inherit properties. Super/Sub has already been explored in Inheritance and Reuse section, which works for traceability and clarity. However, such approach begins to run into issues when a single supertype becomes very widely applied [98]. As will be explored in the next chapter, stereotypes will be used for top level classes that are intended to be widely applicable. The remaining architecture will be constructed using the inheritance rules already outlined. Stereotypes are more limited in terms of property conveying inheritance than supertypes as stereotypes do not carry part or value properties, only generic properties. Stereotypes, as a result, will be used primarily for navigation and tagging, rather than information-based inheritance.

The primary stereotypes to be used here are <<functionalStateVariables>> and <<measurement>> stereotypes. These type <<valueProperties>> in SysML to denote how that <<valueProperty>> is used. An estimator, sensor in this use case, under control provides a <<measurement>> based on its estimation of what it's sensing to its controller. The controller, the command and control system in this use case, makes a determination based off that <<measurement>>, and may issue a command in the form of a <<functionalStateVariables>>

back to the estimator. That <<functionalStateVariables>> affects the state of the estimator under control. There are other stereotypes in the JPL profile that could be beneficial to use, such as typing blocks as an <<estimator>> or <<controller>>, however the use case in this paper is not complex enough to warrant such a deviation from standard SysML. This method is one of the primary methods to review when adapting the methodology here for the customers use case.

This usage is consistent with the implementation of architecture frameworks (DoDAF, MoDAF, UPDM, UAF) within Cameo systems modeler, where the implementation of language extension is done via stereotypes, as well as within the interface modeling discussed later this section [103]. This does suggest that portions of the methodology will be dependent upon the customer, as the elements available and the viewpoints required will depend on the architecture framework mandated.

5. System of System Interface Modeling Method

One of the primary points of failure within systems engineering are the interfaces. Correct and consistent modeling is key in making interfaces work. When designing interfaces there are several considerations to remember: [103]

- Required Interface Components
- Interfaces, like systems, have logical, conceptual, and physical level of abstraction.

Required Interface Components

Interfaces include the connection points on the interacting elements, the data items that are exchanged, the constraints and/or rules that govern the exchange, and the medium for the exchange (i.e., link). This means that Interfaces are multilayered, particularly communications interfaces, including both physical effects (i.e., atmosphere) and software protocol behaviors and limitations (i.e., throughput). Each interface must address three aspects: [103]

- Physical ends
- Connection flow
- Connection material

Interface Abstraction

Logical interfaces support logical information flows at levels higher than physical levels. This could be passing a “track” from a radar to a C2 system, where the physical implementation of that “track” is a specific message type over a specific frequency transiting through the air. Receiving a “track” is what is needed for the functional behavior decomposition at the C2 system, and it would over specify the behavior to have a switch statement for each possible message implementation that would largely trigger the same behavior.

Using the various layers of abstraction, it is possible to start at the top and drill down, as with system decomposition within OOSEM or behavioral decomposition within functional analysis. While delving down through the layers of abstraction it is also possible to improve the level of detail of the interface as well. This stands for both black box and white box implementations of the systems and their interfaces [103]. This will factor in to <<pattern>> block definition discussed later.

6. Hypermodeling and SAIC Profile

Expressing a focus on the model behind the diagrams, Hypermodeling is intended to reduce the number of views needed to construct a model by leveraging some of the more esoteric details of the SysML language, reducing overall complexity and time to model [104]. As this is in keeping with the goal of simplification, the attributes that will be gleaned from the methodology are:

- Restrictions on requirements relationships to other element types
- Opaque Behaviors, and item flows as core tenants
- Metachain navigation and use of tables/matrices
- Utility of a validation profile for enforcement of consistent modeling, which is used by Hypermodeling [105]
- Surrogate models

Requirement Relationship Rest

As previously stated in the SysML language definition section, there are a surplus of nuanced relationships available. As with any large number of often similar options, the nuance can instead lead to complication without compensating utility. One of these cases is the choice between <<allocate>> and <<satisfied>> linking requirements to other element types. <<allocate>> is a “looser” relationship, as a functional requirement “may be constrained so that it can only be satisfied by a SysML behavior such as an activity, state machine, or interaction.” [67] This extends to other relationships as well such as the <<realization>> in place of <<generalization>> when a looser inheritance is required, <<verify>> specifically to behaviors built to conduct V&V (beyond the scope here) through returning a Boolean pass/fail result [104]. <<trace>> is used between a requirement and the generating artifact, while <<derive>> and <<refine>> are used between requirements and use cases [104].

Opaque Behaviors, and Item Flows

The crucial behavior is an opaque behavior, and its parametric diagram counterpart the constraint block [67]. These elements facilitate the use of text-based coding and function as

concise calculations and interface locations to exterior simulation engines such as MATLAB, as defined in Figure 36.

Hypermodeling also advocates for the use of item flows to denote how things are connected, and what flows across that connection. Item flows have already been defined, but the argument for inclusion posed by this methodology is that an item flow is used across multiple diagrams (connectors on *ibds*, object flows on *acts*, messages on *seqs*, and transitions on *stms*). This multiple use allows for synchronization of diagrams during execution [104].

Metachain Navigation and Use of Tables/Matrices

Metachains are a term defined by the creators of Cameo to describe the relationships, behind the scenes, of model elements. Metachains can be extended to link two otherwise disparate model elements through a relative navigational path (rather than a dynamic one) that can be displayed on tables or matrices to quickly show large amounts of relationships that are not conducive to being on a regular diagram together (too many lines and elements) [80]. While these relationships are brittle over multiple hops (as all relative navigation paths are), adherence to a model structure and element usage, supported by a validation profile, allows for the rapid creation of automatically updated summary views that are particularly useful when having to report a model to a customer.

SAIC Validation Profile

Noted in other works besides Hypermodeling, the adoption of a validation profile to add structure to the SysML model in a corollary manner as a compiler to other programming languages such as Cameos error checking is currently minimal [104]–[106]. Hypermodeling uses a profile from SAIC. Though it is built for Magic Draw/Cameo and Rhapsody, it is tool specific. Additionally, the profile is based off pure SysML, so is not currently compatible with UAF/UPDM [107]. It is releasable to the public from the SAIC website [107]. While SAICs validation profile is not the only validation profile available, depending on your usability and access constraints, adding error checking and automated syntax enforcement on top of the tools existing compiler will assist in model consistency and interoperability across a team.

Evaluating SAIC modeling profile more closely, as the same team is instrumental in both, it does impose restrictions on model elements in addition to the recommended approaches. This highlights a challenge with adopting a COTS validation profile, that it assumes a methodology. As an example, the SAIC validation profile will flag any reference property as a warning because the relation, an association relation, is determined to be a weaker version of the composition relation without adding value [107]. There are places, with association blocks, that automatically generate reference relations that will be flagged. Reference relations are used in this case because there is a difference in ownership between composition and association/aggregation. This example demonstrates the intrinsically linked nature of methodology and validation profile, and how the lack of a standardized methodology (the

impetuous of this thesis) impairs the adoption of a better validation profile. This is one of the primary trades that a team will have to make with their project, which is a different paradigm to the other options posited here in the thesis, which can be taken or left individually. Following that, the SAIC validation profile is not going to be adopted for the example mission in this paper due to conflicts it has with some of the methods being highlighted, and advocating for simple and total adoption of the SAIC validation profile and associated methodology is reductionist in the extreme.

Surrogate Models

One of the primary tenants of mission engineering is that the systems of interest are platforms as operationally deployed. This means that there should be system level models underpinning each entity at the mission level. These models are both complex and often originating from exterior organizations, and OEM from a governmental perspective. For reasons of processing efficiency, uniformity, or protection of IP, having a surrogate model of a form controlled by the mission engineer, yet filled out by the system engineer is useful. For the modeling efforts here, it also provides a clean lower level of abstraction boundary [107].

7. Weilkiens SysMod Process

SysMod process outlines a functional reference architecture supported by functional groupings. It links the two by alternating functional requirement-implementation, derived requirement, and component implementation; however functional decomposition has already been discussed in the sections discussing Harmony and CORE from Vitech. The methodology also makes use of allocation matrices, which has been discussed as part of metachaining and its uses with Hypermodeling.

As a point of future consideration, SysMod implements functional groups, which are aggregations of other units/groups, an idea that marries up well with the wargaming and operational sides of mission engineering. However, through that linkage, it should be noted that static, rigidly defined functional groups are of limited utility being expressly defined, due to the task-oriented nature of modern units. In the theme of limiting the number of elements needed, mission engineering models as analyzed here will directly link the order of battle to the platform level. In this context, a platform is the deployed entity, and systems are on that physical entity, as a means of easy distinction. However, this point above all others warrants reevaluation in the future as mission engineering portfolios grow and patterns worth formally codifying in a block and static structure may emerge. However, at this time, that information is not available.

Overall, the SysMod methodology concurs with the other functional decomposition methods outlined previously. The elements that are relevant to mission have already been discussed.

2.3.4. MBSE Methodology Summary

The previous section detailed a number of methodologies that have been put forward to address different priorities, and have discussed the merits of each in terms of applicability to mission engineering. While there are other approaches and methodologies out there, this selection represents a significant sampling of systems modeling space. In sum:

Table 9 Summary of Reviewed MBSE Methodologies and Key Takeaways

Methodology	Key Takeaways
INCOSE Object-Oriented Systems Engineering Method (OOSEM)	• Top-down approach for structural modeling as it aligns to JCIDS • Practices for encapsulation, inheritance, and reuse
IBM Rational Telelogic Harmony-SE, IBM Rational Unified Process for Systems Engineering (RUP-SE)	• Functional decomposition for behavioral modeling (activities tied to <i>ibds</i>, syntactical level) • Levels of abstraction • Intra-requirements relationships
CORE Vitech MBSE	• Overall constructive framework (breaking simulation out of the structural SysML model) • Four step process for functional decomposition (larger scoped functional decomposition than Harmony) • Onion model and concurrent processes for modeling within each layer
JPL State Analysis (SA)	• State machine creation process for top level state machines at mission and top of system level • Use of generalized supertypes in conjunction with stereotypes
Hypermodeling and SAIC validation profile	• Restrictions on requirements relationships to other element types • Utility of a validation profile for enforcement of consistent modeling • Opaque Behaviors and item flows as core tenants • Metachain navigation and use of tables/matrices • System-level model surrogates
Weilkiens Systems Modeling Process	• Possible future use of functional grouping

2.3.5. Presentation Views – Architecture Frameworks

Before delving into the details of how the modeling methodology will be synthesized into a cohesive approach, it is important to outline how the information modeled will be presented to

the end user. Systems modeling currently supports mission level analysis, even if that analysis is traditionally based, and as such there are provisions to fit a SysML model to the traditional diagrams.

The intent behind architecture views is, as DoDAF defines it, “the overarching, comprehensive framework and conceptual model enabling the development of architectures to facilitate DoD managers at all levels to make key decisions more effectively through organized information sharing across Department, JCA, Component, and Program boundaries.” [42] This is accomplished by codifying the majority of views in which different stakeholders and users would be interested. Codifying is accomplished by a common name, definition, and format that leads to a dizzying number of views. For DoDAF 2.02 there are 51 viewpoints. These viewpoints are the natural result of a lengthy evolution process where users further specialize their needed information of a complex system. Figure 42 shows a condensed history of this evolution:

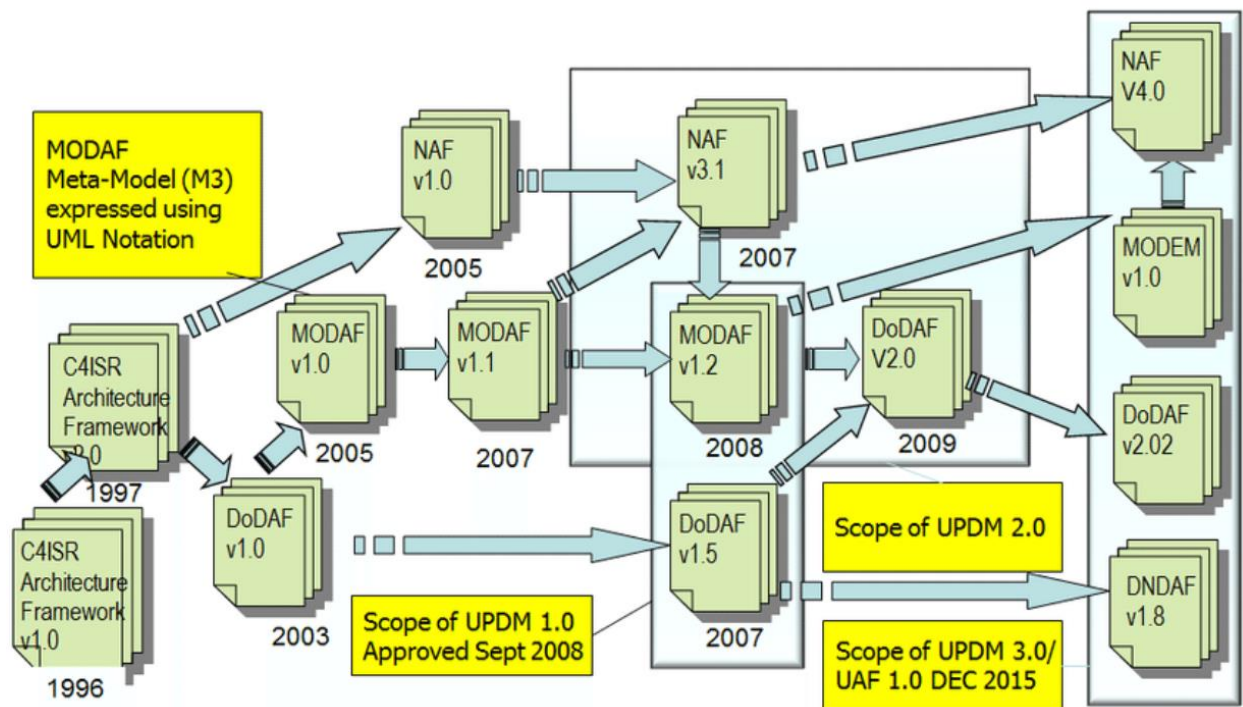


Figure 42 History of the Interrelated Architecture Frameworks [108]

As can be seen in the diagram above, a single source has expanded into four main branches, DoDAF, MoDAF, DNDAF, and NAF. DoDAF is the architecture from the United States, MODAF is the equivalent from the United Kingdom, DNDAF is from Canada, and NAF is for NATO. As all four share a common ancestor, there is significant amounts of shared heritage. As such, it logically follows that there would now be an effort to reunify the diverging frameworks, first manifesting as UPDM. This effort started as Unified Profile for DoDAF and MODAF (UPDM) by the Object Management Group (same group that built UML and the SysML extension), and was

echoed by the NATO Architecture Capability Team in 2012. [108] Since then, UPDM has evolved into the Unified Architecture Framework to account for other national frameworks and a less defense-oriented terminology. It is still lead by OMG [108].

This shift has been formally recognized by the US DoD CIO in 2012, with the intent that UAF will replace DoDAF in DoD. However, the lack of further public documents or discussion on the topic of transition, even in the roadmap specified 2016 adoption, casts doubt as to the ubiquity of adoption. [109] This highlights a key challenge with unifying the now disparate architecture frameworks, that is, habits can be difficult to change, particularly when the new method is more complex than the old. To highlight this point, Figure 43 outlines the viewpoints found in UAF.

 UAF UNIFIED ARCHITECTURE FRAMEWORK	Motivation Mv	Taxonomy Tx	Structure Sr	Connectivity Cn	Processes Pr	States St	Sequences Sq	Information If	Parameters Pm	Constraints Ct	Roadmap Rm	Traceability Tr
Architecture Management Am	Architecture Principles Am-Mv	Architecture Extensions Am-Tx	Architecture Views Am-Sr	Architectural References Am-Cn	Architecture Development Method Am-Pr	-	-	Dictionary Am-If	Architecture Parameters Am-Pm	Architecture Constraints Am-Ct	Architecture Roadmap Am-Rm	Architecture Traceability Am-Tr
Summary & Overview Sm-Ov												
Strategic St	Strategic Motivation St-Mv	Strategic Taxonomy St-Tx	Strategic Structure St-Sr	Strategic Connectivity St-Cn	Strategic Processes St-Pr	Strategic States St-St	-	Strategic Information St-If	Environment En-Pm and Measurements Me-Pm and Risks Rk-Pm	Strategic Constraints St-Ct	Strategic Roadmaps: Deployment, Phasing St-Rm-D, -P	Strategic Traceability St-Tr
Operational Op	Requirements Rq-Mv	Operational Taxonomy Op-Tx	Operational Structure Op-Sr	Operational Connectivity Op-Cn	Operational Processes Op-Pr	Operational States Op-St	Operational Sequences Op-Sq	Operational Information Model Op-If		Operational Constraints Op-Ct	-	Operational Traceability Op-Tr
Services Sv		Services Taxonomy Sv-Tx	Services Structure Sv-Sr	Services Connectivity Sv-Cn	Services Processes Sv-Pr	Services States Sv-St	Services Sequences Sv-Sq			Services Constraints Sv-Ct	Services Roadmap Sv-Rm	Services Traceability Sv-Tr
Personnel Ps		Personnel Taxonomy Ps-Tx	Personnel Structure Ps-Sr	Personnel Connectivity Ps-Cn	Personnel Processes Ps-Pr	Personnel States Ps-St	Personnel Sequences Ps-Sq	Resources Information Model Rs-If		Competence, Drivers, Performance Ps-Ct-C, -D, -P	Availability, Evolution, Forecast PS-Rm-A, -E, -F	Personnel Traceability Ps-Tr
Resources Rs		Resources Taxonomy Rs-Tx	Resources Structure Rs-Sr	Resources Connectivity Rs-Cn	Resources Processes Rs-Pr	Resources States Rs-St	Resources Sequences Rs-Sq			Resources Constraints Rs-Ct	Resources Roadmaps: Evolution, Forecast Rs-Rm-E, -F	Resources Traceability Rs-Tr
Security Sc	Security Controls Sc-Mv	Security Taxonomy Sc-Tx	Security Structure Sc-Sr	Security Connectivity Sc-Cn	Security Processes Sc-Pr	-	-	-		Security Constraints Sc-Ct	-	Security Traceability Sc-Tr
Projects PJ	-	Projects Taxonomy PJ-Tx	Projects Structure PJ-Sr	Projects Connectivity PJ-Cn	Projects Processes PJ-Pr	-	-			-	Projects Roadmap PJ-Rm	Projects Traceability PJ-Tr
Standards Sd	-	Standards Taxonomy Sd-Tx	Standards Structure Sd-Sr	-	-	-	-			-	Standards Roadmap Sd-Rm	Standards Traceability Sd-Tr
Actual Resources Ar	-	-	Actual Resources Structure Ar-Sr	Actual Resources Connectivity Ar-Cn	Simulation			-	-	Parametric Execution/ Evaluation	-	-

Figure 43 UAF Viewpoint Matrix for v 1.2 [110], [111]

There are 88 views in Figure 43, an increase by 37 over the number of DoDAF views. This means that there are a number of entirely new viewpoints within UAF that have no direct analog to DoDAF, and that even the rows and columns do not correlate neatly, as shown below in Figure 44:

 UAF UNIVERSITY ARCHITECTURAL FOUNDATION	Motivation Mv	Taxonomy Tx	Structure Sr	Connectivity Cn	Processes Pr	States St	Sequences Sq	Information If	Parameters Pm	Constraints Ct	Roadmap Rm	Traceability Tr
Architecture Management Am	N/A	N/A	N/A	AV-1	N/A	N/A	-	AV-2	N/A	N/A	N/A	N/A
	Summary (Sm) AV-1											
Strategic St	CV-1	CV-2	CV-2	CV-4	CV-1 CV-2	N/A	-	DIV-1	Pm SvcV-7 SV-7 En-Pm (N/A) Mle-Pm CV-6 CV-3 SvcV-7 SV-7 Rk-Pm SV-7 Ps-Pm SV-9	CV-4	CV-3 CV-5	N/A
Operational Op	N/A	OV-1 OV-2	OV-2	OV-2 OV-3	OV-5a OV-5b	OV-6b	OV-6c	DIV-1		OV-6a	-	CV-6
Services Sv		SvcV-1 SvcV-2	SvcV-1 SvcV-2 SvcV-3b	SvcV-3b SvcV-6	SvcV-4	SvcV-10b	SvcV-10c			SvcV-10a	SvcV-8 SvcV-9	CV-7 SvcV-5
Personnel Ps		OV-4	Ov-4 SV-6	SV-6 Ov-2	SV-4	SV-10b	SV-10c	DIV-2		OV-4 SV-10a SV-7	SV-8 PV-2	SV-5a SV-5b
Resources Rs		SV-1 SV-2	SV-1 SV-2 SV-4	SV-1 SV-2 SV-6 SV-3	SV-4	SV-10b	SV-10c			SV-10a	SV-8 SV-9	SV-5a SV-5b
Security Sc		N/A	N/A	OV-2 Sc-Sr	N/A	N/A	-			-	N/A	-
Projects Pj	-	PV-1 SV-1	PV-1	N/A	SV-4	-	-	-		-	PV-2	PV-3
Standards Sd	-	StdV-1	StdV-1	-	-	-	-	-		-	StdV-2	StdV-1
Actual Resources Ar	-	N/A	OV-4 SV-1 SV-2	OV-4	OV-4			-	-	Not a View (simulation)	-	-

Figure 44 DoDAF Viewpoint Mapping to UAF 1.2 [111], [112]

Regardless of the language used to describe the views or the underlying profile, the relations between those views are a complex web. The UAF profile is built off of the UAF Metamodel that details those relations, described below in Figure 45

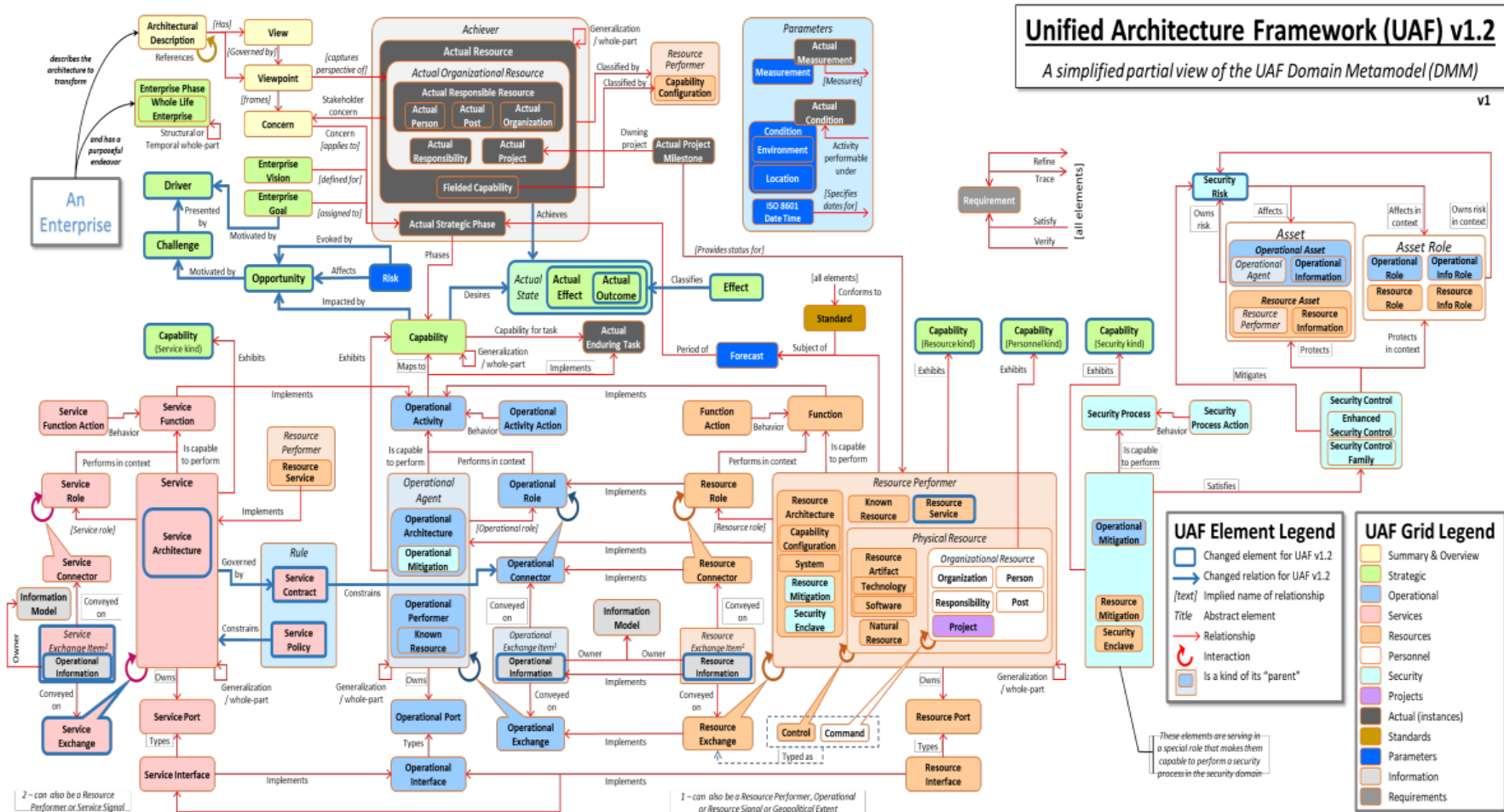


Figure 45 UAF 1.2 Metamodel [111]

The purpose of showing the complexity of the UAF metamodel is to illustrate that the implementation and detailed review of the architecture framework is too complex to cover in a simple review. While this introduction will be sufficient foundation for the example mission later on, any intended user should review the specification standards for their customers directed architecture framework.

Despite the difficulty inherent in implementing such uniformity across such a wide group of people and established languages, there is merit to having a common language. This was the impetus for SysML and UML as well. Given that UAF, or DoDAF, or MODAF represent viewpoints that are used to communicate out the model, strict adherence to them is not absolute during the design process. This approach is similar to report writing, where artifacts are produced for design milestones that are not working documents. Thus, the exact architecture framework adopted will be customer driven and project dependent.

UAF Implementation

UAF, like other architecture frameworks, is language agnostic. The exact implementation of that packaging is therefore up to the tool provider, which in this case is No Magic, the developers of Cameo. The implementation of UAF within Cameo is detailed within Appendix B: No Magic Implementation of UAF Viewpoints in Cameo – Reference Guide.

2.4. Literature Review Summary

The past chapter has covered an introduction of systems engineering, SysML, various implementations of SysML for MBSE, and, lastly, the overlays/views that can be imposed on top of the underlying model to implement traditional viewpoints. This is a breadth of information that is, as many training pipelines, a study period of months and not pages. However, for the purposes of a brief introduction, there are a few main points to reiterate:

- Purpose of the SysML language within MBSE
- Comparison of Systems Engineering and Mission Engineering
- Common themes from Systems Engineering methodologies
- Viewpoint overlays (UAF, DoDAF, UPDM, etc.)

SysML is the graphical programming language used most commonly to implement MBSE. It captures requirements, structure and behavior for a system or system of systems, and links them together in a construct that document-based engineering can't match. This linked construct functions as a basis for higher level, traditional, modeling and simulation, including at the mission and campaign level, which is the usage for mission engineering.

While each of the methodologies reviewed have strengths and weaknesses there are a couple trends that appear in each of them:

- Form follows function
- Element reuse
- Tailored levels of fidelity (and associated complexity)
- Recursive modeling techniques

Whether the primacy is behavioral or structural components, the approach is driven by the end goals of what the model is supporting. While this deduction is intuitive, it highlights that both approaches work, and that neither is “right”, leading to the conclusion that they can be combined and intermixed without compromising either approach as the intent is the same. This leaves only syntactical challenges to implementation, which are easier to surmount than fundamental incompatibilities.

Element reuse is critical to the mantra of avoiding complexity. Constituting inheritance, encapsulation, and referencing at primarily the element level, this approach can be extended to entire models as collections of elements, which themselves are simply aggregations of other elements. This is the foundation for federated modeling which is instrumental in configuration management and practical implementation of modeling across a wide variety of groups and industries, such as those producing systems for the DoD which mission engineering must address. Consequently, federated models, based on the pervasive practice of reuse, will appear early on in the mission engineering methodology fully synthesized in the next chapter.

Tailored levels of fidelity are, similar to element reuse, centered on avoiding complexity. Early and thorough identification of what questions the model will answer (the function of the model discussed in point 1) are critical to building a correctly-sized model. This does mean that any methodology will be a suggestion at most, and that each customer will require some amount of tailoring. The fact that each reviewed methodology recognizes this realistic limitation is one of the reasons that they can be deconstructed and cherry picked as has been done here. Between early identification of form and proper implementation of reuse, the methodology here will support an appropriate level of fidelity tailored to the mission engineering domain.

Recursive modeling techniques implementing the prior points is another attempt at limiting complexity. The difficulty of modeling, and of teaching it, would be much greater if each level of abstraction dictated a different modeling approach. A recursive, or largely recursive, approach allows for reuse of methodology, aiding in model consistency, and building proficiency/speed. As often as possible, recursive modeling will be used in the mission engineering modeling methodology here.

However, no modeling method or engineering practice is worthwhile if the results cannot be communicated to the customer. Similar to the tailored levels of fidelity, each customer may have a different internal process, to which the products of modeling must conform. This trend is why there is an architecture framework for the US, UK, Canada, Australia, NATO,

combinations of them, and still more in industry. While there are efforts to standardize (UPDM, and UAF) there is no one size fits all solution yet, and the architecture framework needed will be driven by the customer. Under that paradigm, the assumptions made here is that UAF, with its planned implementation within the DoD, is a logical architecture framework to use for the example mission later in the paper.

It is clear that there are several trends, both towards MBSE in general and within MBSE, that are applicable to mission engineering and are worth adopting here. This chapter's brief introduction lays the groundwork that will be formally synthesized next chapter into a possible mission engineering methodology.

3. MBSE in support of Mission Engineering

This chapter builds upon the literature review chapters to accomplish two primary aims; 1. Formal linkage of Mission Engineering and MBSE, and 2. Synthesis of the identified methodological excerpts into a cohesive methodology through application to an example mission. The formal linkage of Mission Engineering and MBSE is the foundation for the applicability of the MBSE grounded methodology to the Mission Engineering use case through comparison of the tasks and goals of the two disciplines. The example mission will serve as a proof of concept that Mission Engineering level analysis and modeling can be accomplished using methods pioneered for MBSE.

3.1. MBSE Related to Mission Engineering

This section details the similarities and synergistic nature of Mission Engineering and MBSE by comparing processes and survey results provided by SERC.

Thus far the paper has explored Mission Engineering and MBSE, and while there have been some clear indications of the synergy between them, it is now time to formalize the connection. Both disciplines follow the same general process:

1. Identify need.
2. Elicit and refine requirements.
3. Determine functional behavior to satisfy those requirements.
4. Determine structure needed to support the functional behaviors.
5. Identify and control interfaces and interactions between structural components to ensure interoperability.

MBSE Related to Mission Engineering
MBSE in support of Mission Engineering

6. Simulate/design around external factors and ensure that the design still meets the requirements outlined in step 2.

This process is generalized language from both literature review sections, and is intended to apply iteratively as the scope and abstraction of the functional behavior and supporting structure decreases. That scope and abstraction is the key axis here.

Mission Engineering focuses on missions where the first level of functional behavior and structure is not an individual system, or sub system, but a collection of systems typically grouped into a unit that spans multiple domains intended to achieve effects greater than the sum of its parts. Warfare has been integrated this way formally since the introduction of combined arms doctrine [113], [114]. Thus, the scope and abstraction of Mission Engineering starts at the level of national strategy (itself derived from non-technical political and economic sources) and drills down to the technical systems level. These systems are often the maneuver unit, such as an individual surface combatant.

Systems engineering springs from the need to manage interdisciplinary complexity of modern systems, making it a fundamentally bottom to top approach. As systems interoperated more, a new discipline called system of systems was created to deal with this new complexity.

While both of these points are review, it is apparent there is significant overlap between mission engineering and system of systems engineering (itself a subset of systems engineering). This overlap forms what is essentially a contiguous discipline along the scope/abstraction axis mentioned above from systems engineering up through mission engineering, as shown in Figure 46.

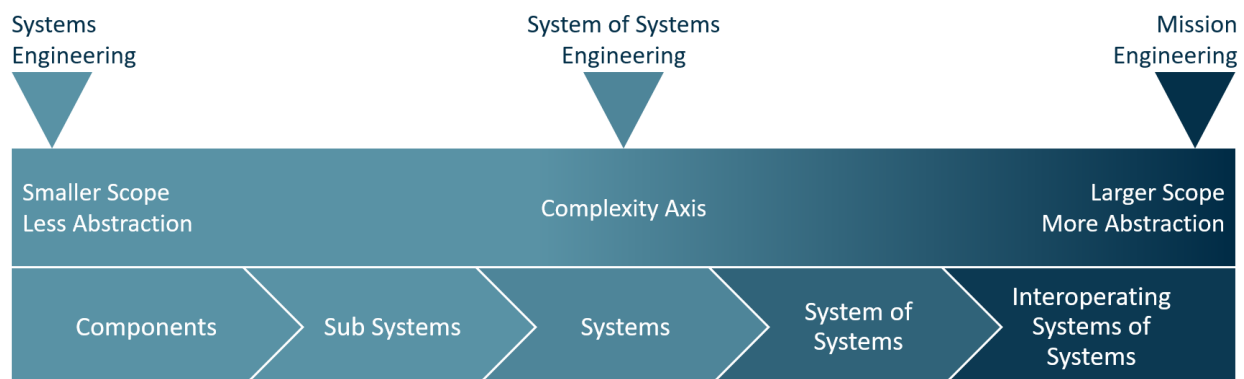


Figure 46 Systems Complexity Axis Continuum

3.1.1. Abstraction Linkage from Systems Engineering to Mission Engineering

Within the discussion of how Systems Engineering relates to Mission Engineering, the contiguous axis in Figure 46 warrants further discussion. The key linkage on that axis is the level

MBSE Related to Mission Engineering
MBSE in support of Mission Engineering

of abstraction. Abstraction has been mentioned often many times through this document, yet to this point the importance and foundational background have not been sufficiently laid out to support the discussion. Synthesizing the points about NDS being high levels of abstraction from the previous section, the RUP SE Abstraction section, and the System of System Interface Modeling Method Interface Abstraction section, a pattern begins to emerge. The level of abstraction dictates the scope of analysis, type of analysis and systems analyzed. It dictates the tooling, and the customers supported. While this line of thought lays out why, and in what ways correctly identifying the levels of abstraction, it does not specify abstraction bounds for Mission Engineering, and how it relates to Systems Engineering.

Using the definitions in RUP SE Abstraction, Mission Engineering exists at the Context and Analysis levels, where the mission itself is taken as the system. The systems are not implemented and hardware or software configurations within a class are not realized. The differentiation between the two levels, context and analysis, is a level of specificity similar to the one outlined in System of System Interface Modeling Method Interface Abstraction, where the mission Context specifies a “sensor” that outputs “tracks”. The analysis level would specify what that sensor is and how it operates (e.g. “sensor = active radar”), what format the tracks are in (e.g. “tracks = Link Message type”), and over what interface (e.g. “wireless RF at 915 MHz”). There is not the additional RUP SE Abstraction level of “design” where the system is realized because that is not needed for first pass mission level analysis. Bounding the levels of abstraction used also allows for the use of “surrogate blocks” described in the Hypermodeling and SAIC Profile Surrogate Models section. By defining the surrogate block through inherited features, and bounding the lower levels of abstraction, there is a controllable level of abstraction usable for mission engineering that can be directly linked to the supporting system model as shown in the Surrogate Models section. This approach is central to the methodology put forth later in this chapter, and is only possible through use and understanding of the abstraction relationship between systems and mission models.

3.1.2. Survey Data

The claim that Mission Engineering is an abstracted extension of Systems Engineering (via System of Systems Engineering) is supported by a Systems Engineering Research Center (SERC) survey funded by the Deputy Assistant Secretary of Defense for Systems Engineering (DASD (SE)) in 2018 [115]. The survey drew a sample of primarily defense industry and government personnel, and, while the sample size is small, the results fall in line with the arguments laid out in this paper asserting that Mission Engineering and Systems Engineering are closely linked.

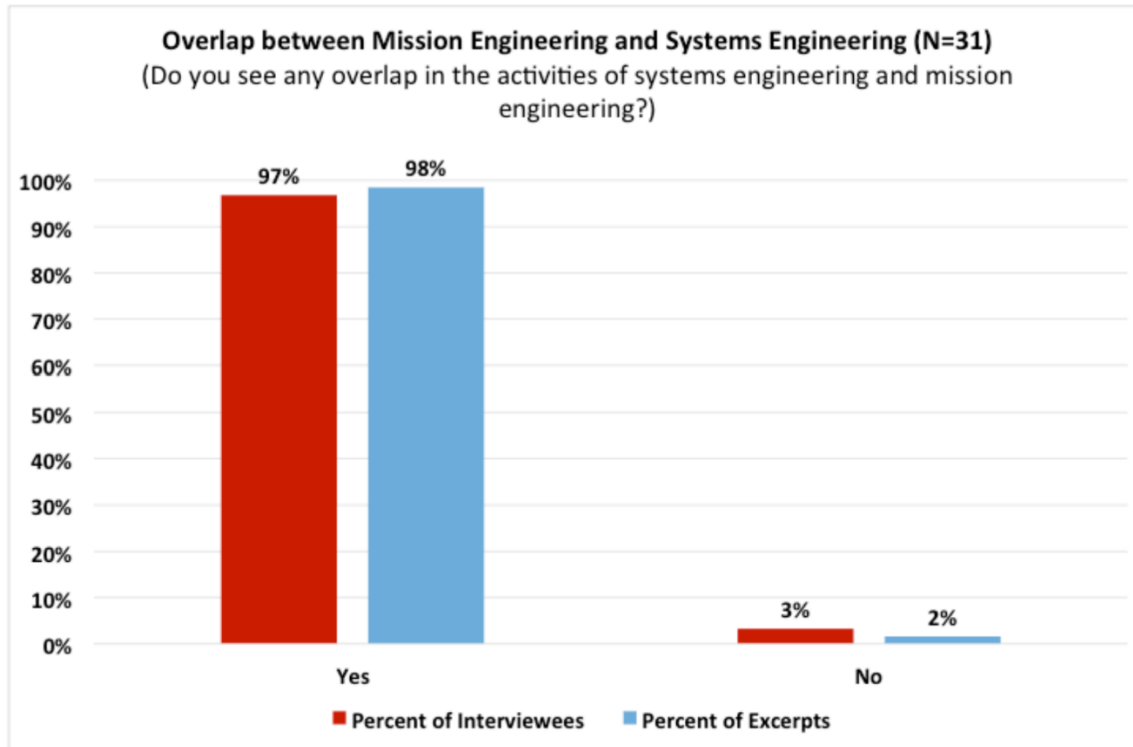


Figure 47 Mission Engineering, Systems Engineering Overlap [115]

While the study is overwhelmingly supportive of this overlap, there is some deviation in how that overlap is best described. With half assigning the similarity to Mission Engineering and System of Systems Engineering, with another thirty percent ascribing Mission Engineering directly to Systems Engineering.

MBSE Related to Mission Engineering
MBSE in support of Mission Engineering

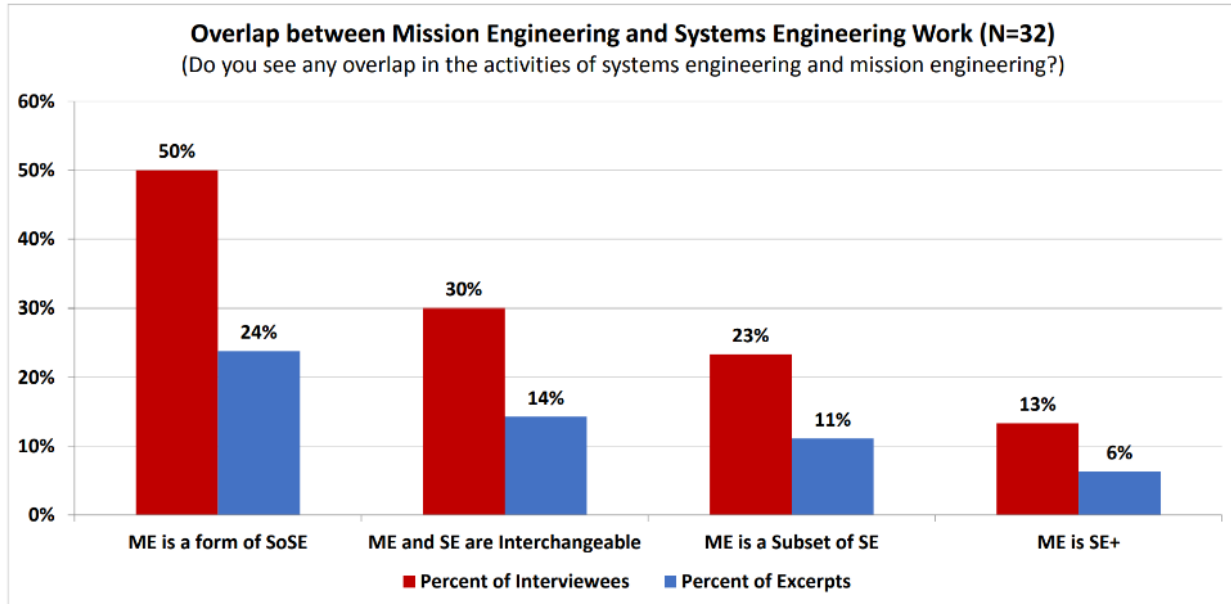


Figure 48 SERC Overlap of Mission Engineering and Systems Engineering [115]

Though these questions appear to be able to select multiple responses, the study agrees with the logic that Mission Engineering and Systems Engineering are closely related. However, as good a foundation as this overlap is, it has not yet established that the disciplines overlap on a technical and executable level. The best means of determining if the level of detail explored in the MBSE literature review portion of this paper is applicable to Mission Engineering is a review of the tasks. In Figure 49, Systems Engineering and System of Systems Engineering, established to be interchangeable with Mission Engineering, are contrasted.

	Systems Engineering (SE)	Systems of Systems Engineering (SoSE)
Management and Oversight		
Stakeholder Involvement	Clear set of stakeholders	Multiple levels of stakeholders with mixed and possibly competing interests
Governance	Aligned management and funding	Added levels of complexity due to management and funding for both SoS and systems; SoS does not have control over all constituent systems
Operational Focus (Goals)		
Operational Focus	Designed and developed to meet common objectives	Called upon to meet new SoS objectives using systems whose objectives may or may not align with the SoS objectives
Implementation		
Acquisition/Development	Aligned to established acquisition and development process	Cross multiple system lifecycles across asynchronous and development efforts, involving legacy systems, developmental systems, and technology insertion
Process	Well-established	Learning and adaptation
Test and Evaluation (T&E)	Test and evaluation of the system is possible. System requirements drive the system T&E and use Measures of Performance (MoP)	Testing is more challenging due to systems' asynchronous life cycles of component systems and the complexity of all the parts. At the SoS level, Measures of Effectiveness are needed, which are difficult to define, as well as MoPs
Engineering and Design Considerations		
Boundaries and Interfaces	System of Interest (SOI) is defined by focusing on boundaries and interfaces	The dynamic and reconfigurable nature of SoS mean that boundaries are interfaces may change. Also, component systems may belong to more than one SoS and have variable availability
Performance and Behavior	Performance of the system to meet performance objectives	Performance across the SoS that satisfies SoS capability objectives while balancing needs of the constituent systems
Metrics	Derivation from requirements is straightforward	Difficult to define, agree, and quantify due to independent management of component systems

Figure 49 Comparison between Systems and Systems of Systems as Applied to Systems Engineering [115]

Review of the table in Figure 49 provides one key takeaway: Mission Engineering is like Systems Engineering, only more complex. This complexity is a result of increased number of systems, a wider pool of stakeholders, and the introduction of a geographic map layer and movement across that map. This conclusion is supported by a cursory review of configuration management of a system vs a deployed unit. A system is managed by engineering change proposals and can be expected to remain together during operation. This allows for a comparatively high level of control over the interfaces and internal structure, which can be made largely static from a design perspective. A deployed unit constitutes multiple systems and platforms, which can be task organized at an ad hoc level as the mission (analogous to design

MBSE Related to Mission Engineering
MBSE in support of Mission Engineering

specifications) change at a much more rapid pace than a system engineering design specification normally would [114].

This contrast is one of design cycle duration, scope and abstraction, rather than a fundamental difference in objective. As such, the fundamentals of Systems Engineering can be applied to Mission Engineering, so long as they are adopted under the lens of the increased dynamic nature of Mission Engineering.

3.2. Modeling Methodology Applied to Mission Engineering Use Case

This section steps through the application of the MBSE informed methodology for Mission Engineering identified as part of the literature review. This section serves as the technical example further proving the case made in section 3.1. It covers structure, behavior, simulation, and a trade study using Cameo and MATLAB. It is not a step by step instruction manual, however there is sufficient detail for a proficient modeler to have little issue replicating the work.

The clearest way to demonstrate how the key points that have been discussed thus far fit together is through an example model. While the model itself will be attached, there are a number of highlights that are worth pointing out specifically, and how they relate to the points in Table 9.

3.2.1. Mission Description

An Integrated Air Defense (IAD) system is deployed to an austere location to protect high value assets in the rear of the operational zone. This system includes a pair of networked sensors offset from the command node to improve survivability, communicating over a wireless network. There is an inbound hostile aircraft that must be detected by the search radar and the tracking radar prior to entering its weapons engagement zone of 200km. Because the base is austere, fuel consumption and associated logistics are a key concern, limiting the fuel consumption to 110kg of fuel per hour. As multiple of these sites are required to provide sufficient coverage, the operational cost shall not exceed \$2000 per day. The goal is to determine if the systems specified can conduct the mission within the given constraints.

There are a number of assumptions that go into the Mission Engineering process derived from the mission description. The first is that the mission description and setting are in input to the engineering process. That is the block of text from which all else flows. As the mission description is also generated for this effort, it is worth briefly exploring three of the assumptions made in its creation, as those assumptions are inherited by the Mission Engineering process.

- The first is that the sensors being used are active radar type sensors. While passive RF and EO/IR sensors exist, the first and most ubiquitous sensor in use is active radar.
- The second assumption is that the 200km threat weapons engagement zone is a static variable, rather than a function of system design and employment. The 200km value assumes certain environmental conditions, as well as a specified release altitude.
- The third is on fuel consumption, where the consumption rate is a static value, rather than a dynamic value changing depending on the load.

There are more assumptions in every mission description, however the key takeaway is that those assumptions are not made by the mission engineers, but by the operational customer. These assumptions are critical for properly scoping the mission into something that can be captured by the mission model.

3.2.2. Model structure

Having established what question the model is answering, the supporting structure needs to be defined. This structure has a number of items required to link the system source of truth models to the mission analytics/simulation. These items are:

- Model structure detailing the package layout for the model
- The modeling process built off of the model structure

3.2.2.1. The Reference Modeling Process

The Reference Modeling Process is not required for a model in practice, but in this case, this is the location for all the elements and diagrams, such as those in Figure 50.

3.2.2.2. Library

The Library package is the location for all the building blocks of the mission model. It is a shared library and should be common to the modeling effort as a whole, and centrally managed to ensure consistency.

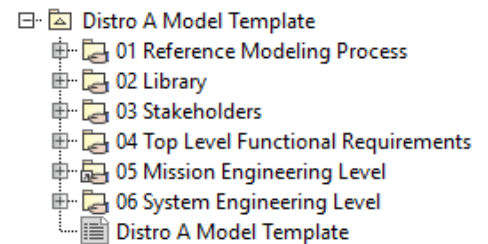


Figure 50 Top Level Model Structure

The library will contain previously completed surrogate systems aligned to the needs of the mission models simulation framework. It will also contain common use components include signals, for both information conveyance and control of the system under control (JPL State Analysis), constraints for math and calculations, interfaces with their appropriate associations, and unique value types (units such as miles per hour that may not be natively included in the SI based ISO-80000 profile). These elements are commonly controlled due to their reusability.

The template package is the repository for local model method documentation, for use as reference.

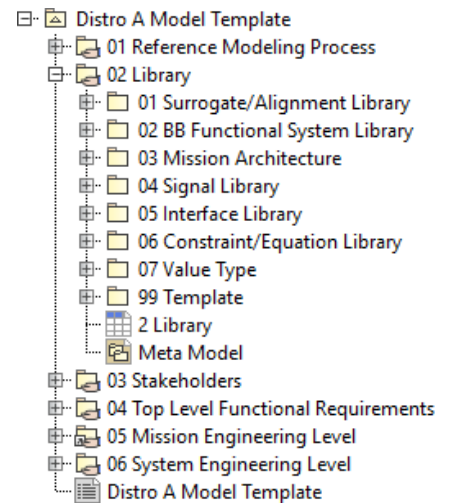


Figure 51 Model Library Package Breakout

3.2.2.3. Stakeholders

The Stakeholder Package shown in Figure 52 is where supporting documentation is stored. The primary use of information in this model is as background to the project and ensuring common understanding between the modelers/engineers and the customers. OV-1 and similar diagrams would be stored here as well.

3.2.2.4. Top Level Functional Requirements

The requirements for the mission are captured in the Top Level Functional Requirements section. Beginning with the JCIDS defined top level Key Performance Parameters (KPPs), the mission will be broken down into supporting requirements, functional and performance requirements, and eventually into measures of performance. This process maintains traceability across the entire spectrum of system – mission engineering, and will mirror the structure of the mission.

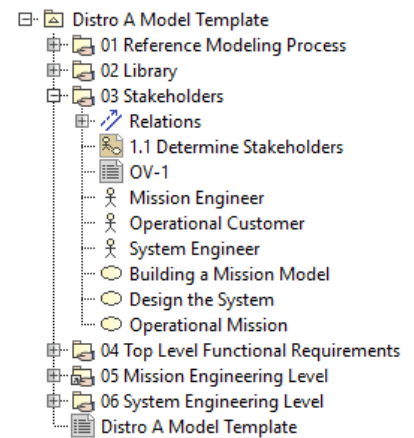


Figure 52 Stakeholder Package Breakout

Different types of requirements denote applicability to different model elements (i.e. functional requirements pertain to functional roles and behavior). Use of stereotypes to denote relationships in this way is native to SysML, but also falls in line with the JPL State Analysis key takeaways.

3.2.2.5. Mission Engineering Level

The Mission Engineering level is where most of the actual mission engineering will take place. Elements pulled from the library (surrogates, interfaces, constraints, etc.) are aggregated, connected together, and linked to the relevant requirements.

Within this package are several smart packages that *act* as pointers to the original elements in the Library package (Hypermodeling and SAIC Profile). Inclusion of these smart packages, and the use of metachains to fill them, allow for easy reference within the Mission Engineering Level package, simultaneously providing the convenience of a self-contained model structure organized by usage, with the benefits to configuration management of centralized control organized by function. While the time savings for each individual reference is small, there are enough reach backs that, as the model grows in size, not having to search through the model yields significant ease of life.

Also in this package are the Sub Missions and Mission Variants packages. These packages are included for completeness, however their main purpose is to support multiple concurrent missions (beyond the scope of this effort) as well as supporting trade studies for the mission (only covered briefly as the literature review produced nothing novel on this topic).

There are two matrices on this level, these matrices allow for quick assignment of roles, and a summary table of what systems are participating in what roles within the mission similar to forms of documents already in use as part of capability gap analysis.

The context block named IAD Mission is the element that represents the actual mission structure, and will be the central product at the end of the modeling process (discussed in the next section). Note that the <<system context>> stereotype is applied to this block as the mission is not a single system of the same nature of a radar or C2 node, nor is it purely <<external>> to the systems. As such, the <<system context>> stereotype is applicable and follows the use of stereotypes to define elements native to SysML and extended by the JPL State Analysis.

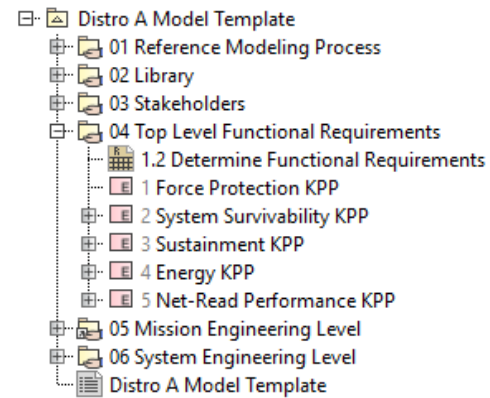


Figure 53 Top Level Functional Requirements Breakout

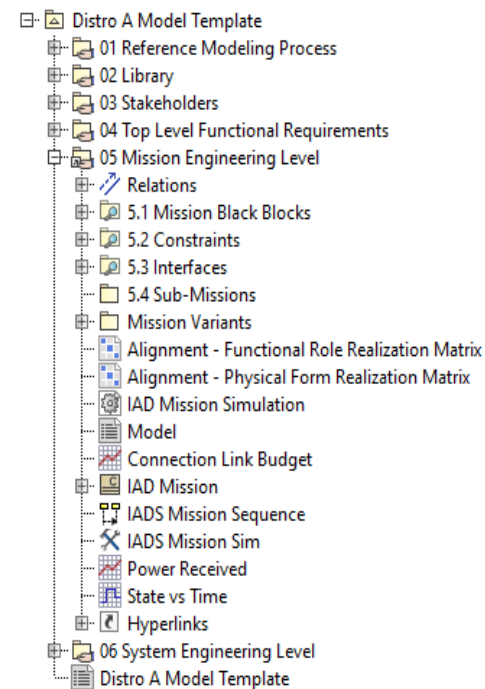


Figure 54 Mission Engineering Level Package Breakout

Lastly, there are a number of elements and diagrams left in the package that pertain to simulation. These represent everything from the mission sequence, to the simulation configuration, and to the graphs produced when simulated.

3.2.2.6. System Engineering Level

There are two sources that a system model can come from to support Mission Engineering; government and industry. Depending on which source the information stems from, and the sensitivity of the system model, the model may or may not be available to the mission engineer in its entirety. As such, the surrogate block contained under the Library package is the lowest element on the complexity axis used by Mission Engineering as it is the only one reliably present. However, if a system model is available (pending classification and propriety) , then there is use in doing a “project usage” to bring it into this package as a read-only reference. In this example, the search radar, aircraft, and C2 nodes are available while the tracking radar is not. Accurate portrayal of a system within its surrogate is critical to the accuracy of the mission, and having the source model as reference can improve that accuracy and understanding of the mission engineer. However mission engineering can be conducted without the model provided there is sufficient communication between the Mission Engineering team and the System Engineering team.

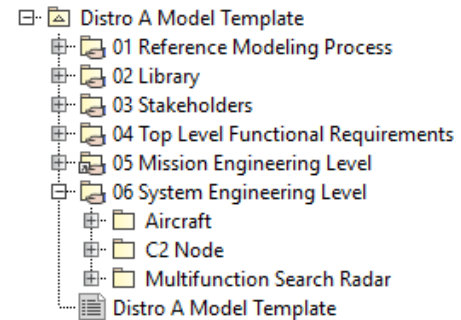


Figure 55 System Engineering Level Breakout

3.2.2.7. Distro A Model Template

Having a landing page/README for a model is particularly useful for mission engineering where customers or management may have a cursory understanding of SysML but are not yet proficient in it. Having a documented, legible means of navigating the model without knowing the package structure can be helpful.

3.2.3. Model Process

Having established what the model is, it is time to walk through how a Mission Engineering model is built. The first step is to identify the processes to be used in the modeling project. The example here will be conducted in SysML, using both functional (CORE, IBM, JPL SA) and structural (OOSEM) decompositions. There is no use of UPDM, UAF, or DoDAF in order to keep the example more general. As each profile adds additional stereotypes and elements, while renaming or removing others, their use could be counterproductive to establishing a commonly understood example.

3.2.3.1. Stakeholder Identification and Process Overview

Once the language and overarching process are identified, the next step is to identify the stakeholders for the mission. For this example the participants are a generic System and Mission Engineer working with a generic Operational Customer.

The Operational Customer typically brings the mission description to be solved, and the existing legacy options to possibly solve the problem with, and/or the functional role that a new system to be developed would play. This new system is known as a system of interest (SOI). The System Engineer provides authoritative sources for the legacy systems or possible solution SOIs. The Mission Engineer is responsible for knowing what mission analytics tools are available and applicable to the scenario based off of the questions the Operational Customer is asking. The model being built under this example is the interface between the System Engineer and the mission simulation framework, and is therefore owned and built by the Mission Engineer to answer the Operational Customer's question.

Once the stakeholders are identified, they can be brought together and requirements elucidated. In this case, the customer has provided the previously discussed example mission, and is wanting to explore the trade space of an integrated air defense mission, specifically the detection portion of the kill chain. They are interested in the relationship between detection and targeting distance, and the effects on fuel consumption and cost, to support a larger analysis of the logistical viability of the notional air defense site. As always, they are also interested in identification of any capability gaps in the mission architecture. The model is, in effect, a transfer function of form:

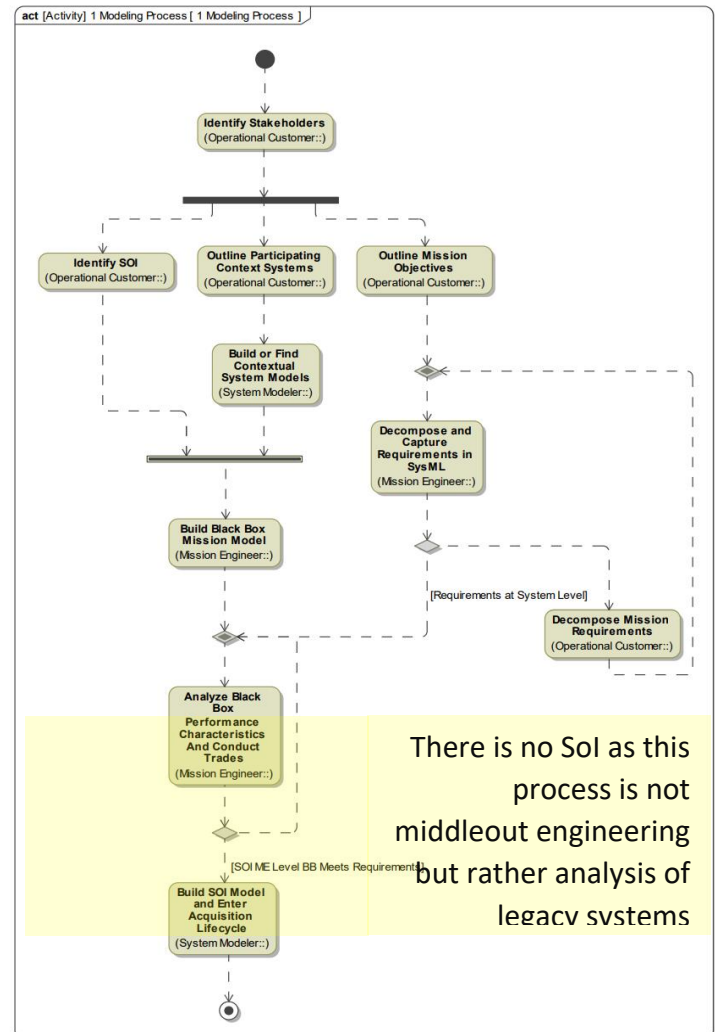


Figure 56 Mission Engineering Process

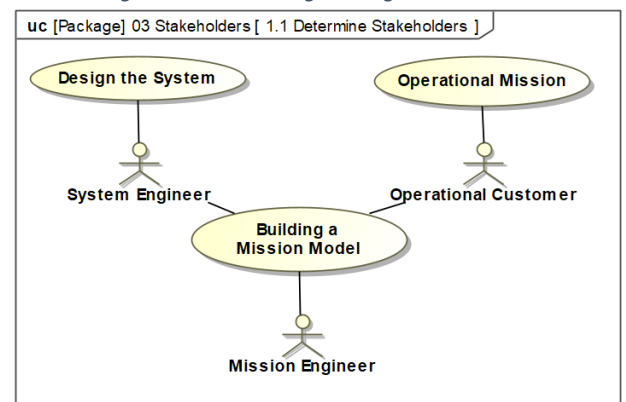


Figure 57 Generic Mission Engineering Use Case

Modeling Methodology Applied to Mission Engineering Use Case
MBSE in support of Mission Engineering

$$MassFlow_{fuel} = f(Power_{radar}, Power_{communications}, EngagementRange_{threat})$$

3.2.3.2. Mission Narrative and OV-1 Generation

There is no system of interest in this example as it is an operational analysis as opposed to middle-out engineering. Additionally the scope is relatively restricted, however, as will be seen, there is still a significant amount of detail needed to calculate even a single iteration of the above example. At this juncture there is enough information to generate a rudimentary OV-1:

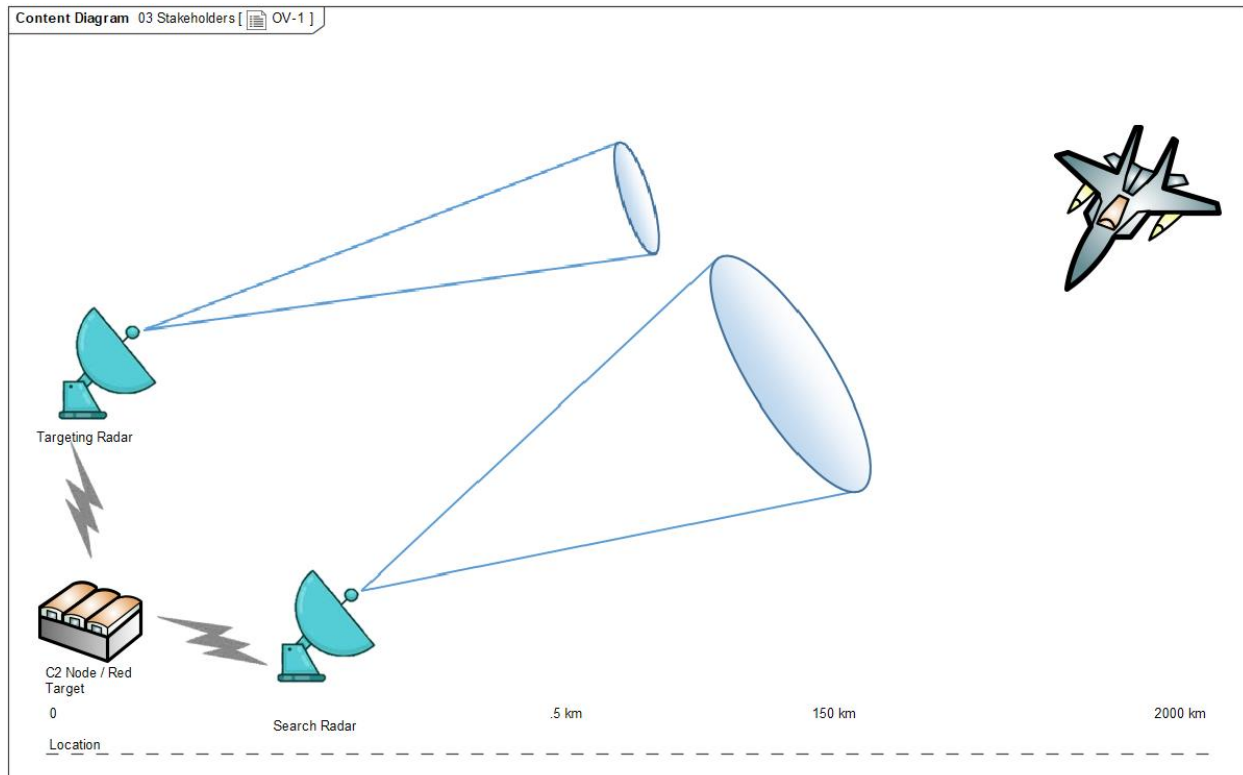


Figure 58 IAD Mission OV-1

The mission narrative progresses as follows:

1. The threat aircraft flying towards the target (not a communication, so not shown in Figure 59)
2. The threat aircraft is detected by the search radar (not a communication, so not shown in Figure 59)
3. Search radar relays that detection to the C2 node which is at that moment in standby waiting on and alert (1. *Target Detected* and 2. *Targeting Information*)
4. The C2 node notifies the targeting radar and enters the track from search radar into its internal data base. (3. *Target Detected* and 5. *Enter Track into database*)
5. Targeting radar is queued by the search radar via the C2 node locks on to the threat aircraft. (4. *Targeting Information*)

6. Targeting radar returns improved targeting solution to the C2 node (6. *Targeting Information*)
7. This sets the stage for follow on engagement by some effector against the threat aircraft. (Effector beyond scope of model, , so not shown in Figure 59)

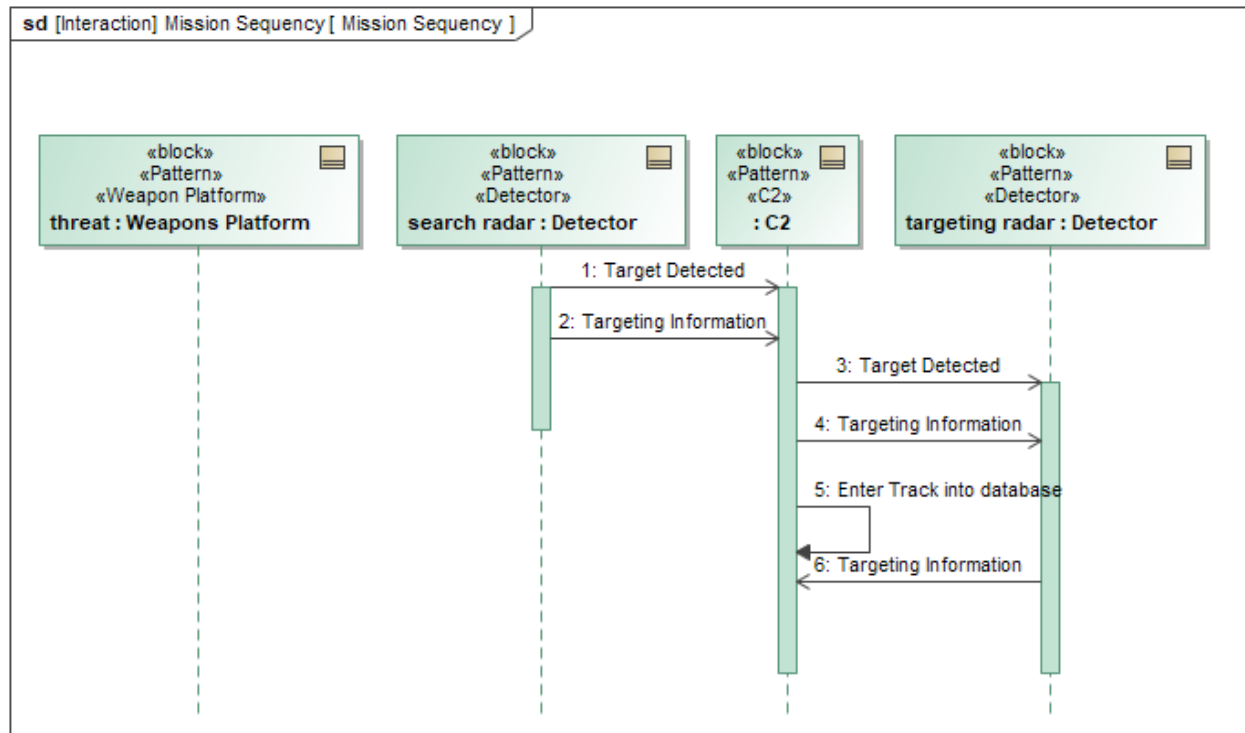


Figure 59 Mission Message Sequence

There are a few items of particular note with this last diagram: the lack of a message between weapons platform and either radar, the <<Pattern>> stereotype, in this diagram is not contained within the mission package (see previous section on model structure).

Taking each point in turn, this diagram shows intended communication paths. The interaction between a detector and the threat is not likely an intentional communication, and as such is a physics based interaction. In this modeling methodology there is a distinction between designed communications and physics based interactions. Designed communications are intended (but not guaranteed) and convey information, are time based/sequential, and are represented through signals sent by an activity (Figure 67). Physics based interactions are taking place regardless of the sequence or state of either object, represent interactions that are unavoidable, and are modeled using constraints on parametrics/internal block diagrams (Figure 69).

The <<Pattern>> stereotype denotes that this is neither a physical or logical system, but rather a class of system. The <<Pattern>> stereotype will be inherited by surrogates of systems that claim to be of that class through a generalization relationship. (JPL State Analysis & INCOSE

Object-Oriented Systems Engineering Method (OOSEM)) This inheritance ensures that there is consistency among surrogates and makes it possible to control those interfaces in support of mission simulation.

Lastly, this diagram is contained within the stakeholder package, as it is generated as part of the mission definition process. Additionally, it plays no part in the eventual simulation of the mission because it is a top down behavioral model. A mission has no controlling entity that ensures that each mission participant behaves in accordance with an overarching plan. Instead, a mission is an aggregation of systems, with their own behaviors, acting in concert. There are structural interactions, but those interactions are of a physics based variety, making them unsuitable to behavioral modeling.

3.2.3.3. Mission Requirements Capture

With the objectives identified (evaluate performance of specific kill chain in terms of fuel consumption, cost, and performance) they need to be captured within the model.

As identified in the literature review, the requirements begin with the JCIDS top level KPPs and drill down. For this mission, the Force Protection KPP is not utilized as there are no personnel

#	Name	Text	Source	Satisfied By	Verify Method
1	 1 Force Protection KPP	"The Force Protection (FP) KPP (one of the four mandatory KPPs) is intended to ensure protection of occupants, users, or other personnel who may be adversely affected by the system or threats to the system. Although the FP KPP may include many of the same attributes as those that contribute to System Survivability, the intent of the FP KPP is to address protection of the system operator or other personnel against kinetic and non-kinetic fires, CBRN, and environmental effects, rather than protection of the system itself and its capabilities."	JCIDS		
2	  2 System Survivability KPP	"The System Survivability (SS) KPP (one of the four mandatory KPPs) is intended to promote the development of critical warfighter capability that can survive kinetic (i.e., traditional, non-traditional, and CBRN) and non-kinetic (cyber and electromagnetic spectrum (EMS)) threats across domains and applicable environment including space. The SS KPP supports three system-oriented objectives: prevention, mitigation in tactically relevant time, and recovery from threats and fires. Simultaneously, the SS KPP maintains survivability from diverse environmental factors (i.e., heat, dust, moisture, EMS-congested areas, etc.). The SS KPP promotes the analysis of probable threats and advances a SoS approach that leverages in-depth, overlapped, or redundant defensive capabilities through networked, autonomous, or semi-autonomous systems to ensure mission completion, despite the loss or degradation of individual components."	JCIDS		
3	  2.1 Prosecution Range	Threats shall be prosecuted prior to the treats weapon release range			
4	 2.1.1 Detection Range	Detection range shall outrange threats weapon engagement range (detectionOutrangesEngagement shall be <u>greater than 0</u> / true)		 detectionOutrangesEngagement : Boolean	Analysis
5	  3 Sustainment KPP	Sustainment KPP (Mandatory KPP). The Sustainment KPP is intended to ensure an adequate quantity of the capability solution will be ready for tasking to support operational missions."	JCIDS		
6	 3.1 Operating Costs	Installation operating costs shall be <u>less than 2000</u> dollar per hour		 hourlyCostBLUEFOR : cost[dollar per hour]	Analysis
7	  4 Energy KPP	"Energy KPP (Mandatory KPP). The Energy KPP is intended to ensure combat capability of the force by balancing the energy performance of systems and the provisioning of energy to sustain systems/forces required by the operational commander under applicable threat environments."	JCIDS		
8	  4.1 Fuel Consumption	Fuel will be transported over unimproved roads by convoys			
9	 4.1.1 Fuel Consumption	Generators will consume <u>less than 110</u> kg of fuel per hour		 hourlyFuelConsumption : mass flow rate [kilogram per hour]	Analysis
10	  5 Net-Ready Performance KPP	"The net-ready performance attribute ensures interoperability between individually developed and fielded capability solutions. "	JCIDS		
11	 5.1 Compatible Network Interface	System Must Communicate with Rest of the Kill Chain			Inspection
12	 5.3 Bit Error Rate	Bit Error Rate (BER) <u>less than or equal to 1E-6</u>		 bitErrorRate : Real = 1.0E-6  bitErrorRate : Real = 1.0E-6  bitErrorRate : Real = 1.0E-6	Analysis
13	 5.4 Link Margin	Link Margin must be <u>greater than 0</u>		 linkMargindBSenderToReceiver  linkMargindBReceiverToSender	Analysis

Figure 60 Requirements Captured in Model

Modeling Methodology Applied to Mission Engineering Use Case
MBSE in support of Mission Engineering

modeled or considered. System survivability reduces down to the need to prosecute the threat before it can engage either the targeted C2 node, or the sensors themselves. Because the only part of the kill chain within scope in this example is the detection section, (composed of find, fix, and track) the only requirement is that the threat is detected beyond its engagement range. In order to have a requirement color code the satisfying value property, it must be written with an equation able to be pulled from it. The keywords are greater/less than. While doing this also links any value hard coded into the requirement, such as the cost or fuel consumption, it is necessary to create a dummy variable (“detectionOutrangesEngagement”) for when the requirement is a relation between two targets as the SS KPP is for this example. Of note, during this section of model creation the “satisfied by” column will be empty as that portion of the model has not been built at this stage. The Energy and Sustainment KPPs are self-explanatory, while the Net-Ready capability stands out as the only KPP that is satisfied by value properties that are not owned by the mission or its references. These value properties are owned by the interface between the systems, which will be explained during the interface implementation section.

The main point to draw from Figure 60 is that only performance requirements are directly satisfied by a value property. Functional requirements such as the compatible network interface are more difficult to ensure compliance, as they are not able to be fulfilled by a constraint and require manual verification, but should be drawn to operations or interfaces.

The table also specifies the method of validation for a requirement. While lower abstraction systems could be tested in isolation and have a live test as a verification method, the systems involved at a mission level generally prohibit that method of testing. That leaves analysis as the primary method of verifying that a requirement has been met, and the accuracy of that analysis is dependent upon the technical rigor of the tools used. The primary reason that this column is detailed here is that, for a new system, the analysis at the mission level is contingent upon the lower level design. In middle out engineering, where a system is being designed or a mission role, these mission level requirements should decompose into system level requirements to maintain traceability throughout the design process at both the mission and systems level.

While MBSE has its benefits, and it has been established within this paper and others that a model is more useful with lower CM effort than document based engineering efforts, it is important to understand some of the limitations. While subjective, as mission and systems engineers expectation management is a critical component of initial discussions with non-technical customers.

3.2.3.4. Mission System Definition - Structure

Now that the objectives have been captured as model requirements, and the system of interest identified (none in this case, all legacy systems), the next step is to identify the participating systems. Pulling from the OV-1, the mission has four participants (threat aircraft, two search

and tracking radars, and a C2 node), however there is one piece of support equipment critical to the Energy KPP analysis: a generator to power the radars. While not a functional part of the kill chain, and therefore not on the OV-1 it would appear in a Service Viewpoint diagram. Knowing this information allows construction of the first mission diagram; the mission structure *bdd* (Figure 61).

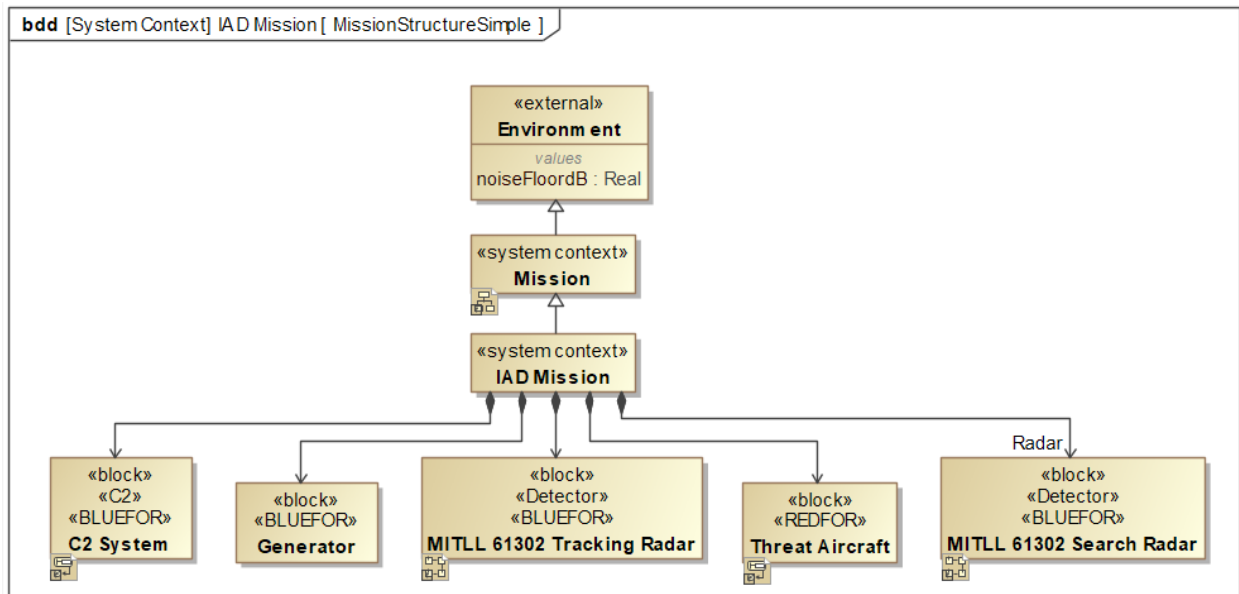


Figure 61 Mission Structure *bdd*

This *bdd* shows the “IAD Mission” is a sub type of “Mission” which itself is a subtype of “Environment”. This allows for some specialization of mission types as the mission model repository grows. It also follows the premise of inheritance outlined in OOSEM. The diagram also shows something that may be contradictory with some of the language thus far in the paper, where the blocks are *directed composition* as opposed to *directed aggregation* that is more syntactically accurate. While both are types of associations, they create different types of properties. Additionally, a composition relationship suggests a level of ownership that is too stringent and brittle for a mission where systems are more likely to be task organized and fluid. Acknowledging that aggregation is the more correct relationship, it is not used for a simple reason: composition relations simulate more simply. As will be discussed further in the simulation section, there is a difference in simulating a part property (composition) compared with a reference property (aggregation or association). Part properties automatically create an instance of the block that types the part property as part of the initialization process. A reference property requires manual specification of an instance of the block that types the property. Therefore, as a matter of convenience, and with acknowledgement that directed aggregation is the more technically correct linguistic implementation, this example model will use directed composition instead. Such is the invariable clash of technical with practical.

However, this concession is not without literary basis, as the SAIC validation profile prohibits reference properties entirely, flagging them when discovered. (Hypermodeling and SAIC Profile) Though for reasons discussed later with connector typing, this wholesale rejection of reference properties is a problematically heavy handed approach, one of the reasons that particular profile has not been adopted as part of this methodology.

Semantics aside, the construction of the “Mission Structure” *bdd* relies on the assumption that the necessary systems have surrogates already created. While this is a foundational assumption of this modeling methodology, it is worth a brief overview. Developed independently by both the author for mission engineering and SAICs profile for system of systems modeling, both approaches rely on a unique mapping of the system block to the surrogate block. SAIC uses a realization relation between elements (such as ports, System of System Interface Modeling Method) and the mission engineering approach takes it one step further and uses a unique *ibd* on the system model to link value properties on the system model that may be buried on parts of that system model and converting or combining them as needed to fulfill the values on the surrogate block. Behavior is similarly adapted, with the developers working with the mission engineer to apply methods to the requisite operations (inherited from the functional black box patterns). This process yields a surrogate block that is linked to the supporting system model, able to be updated as the system is updated and refined. The fundamental architecture to construct and fill a surrogate block is detailed in Figure 62, in a very simplified manner. There is an additional option if a significant quantity of information is required to facilitate interaction with the simulation framework, it is possible to add another layer of inheritance from a block containing that information so that it does not clutter the surrogate shown to the OEM/Systems Modelers as shown in Figure 64. This lower-level surrogate + sim block would be the one executed or instanced as required. As the simulation framework in this example is simply MATLAB integrated with Cameo, this layer is not needed.

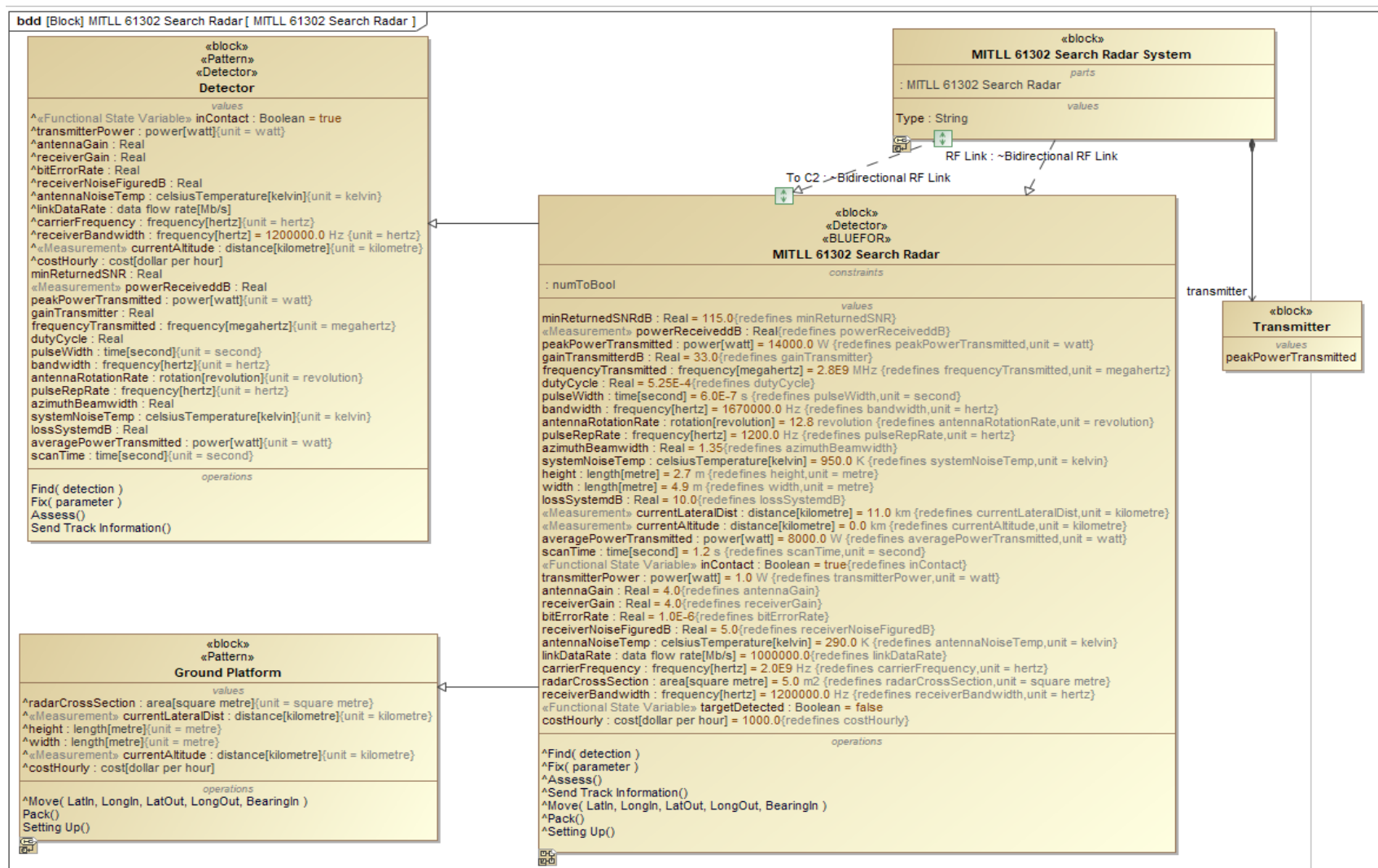


Figure 62 Surrogate Block Creation and Traceability

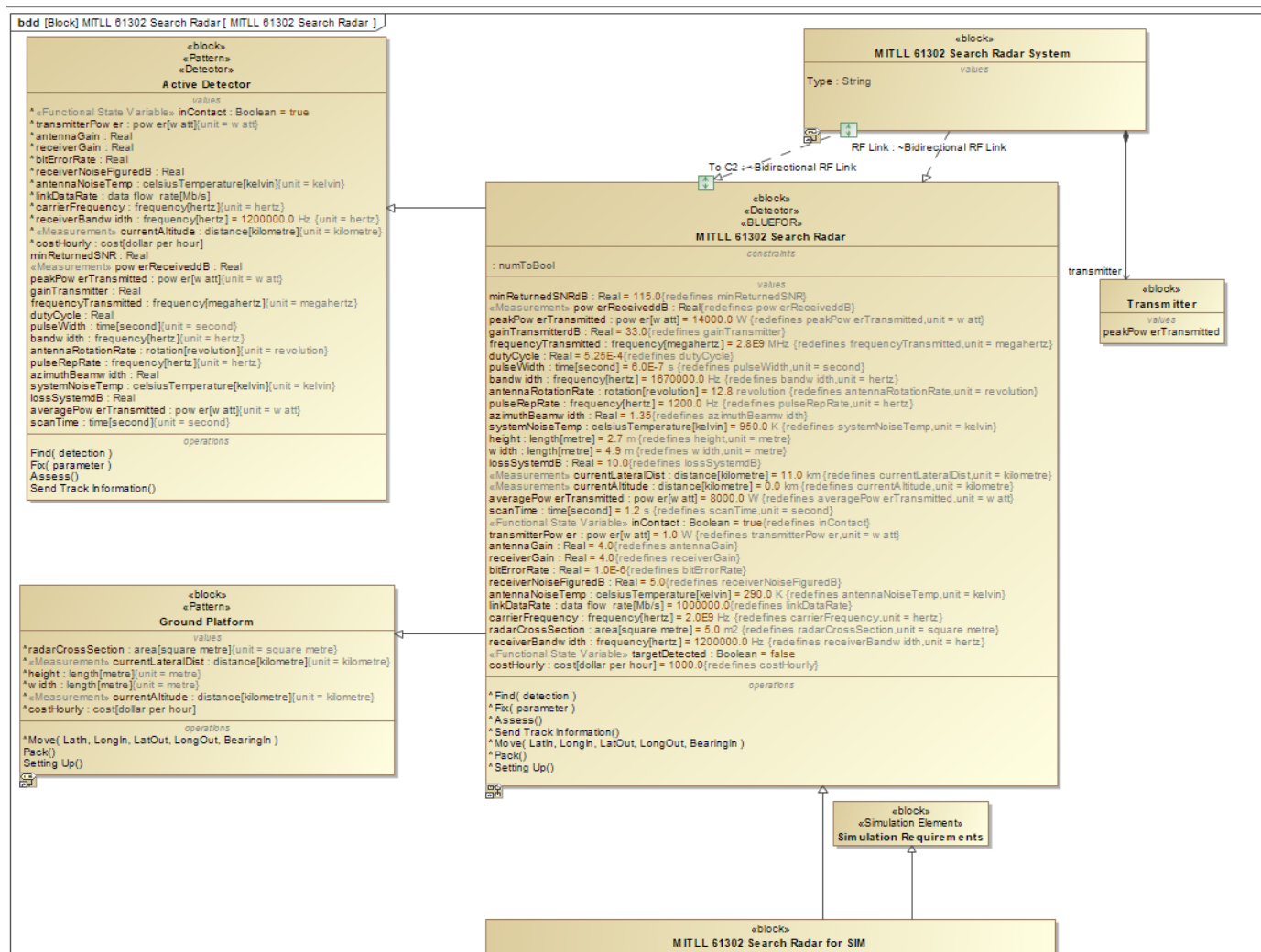


Figure 63 Surrogate Block Definition with Additional Simulation Information

Modeling Methodology Applied to Mission Engineering Use Case
MBSE in support of Mission Engineering

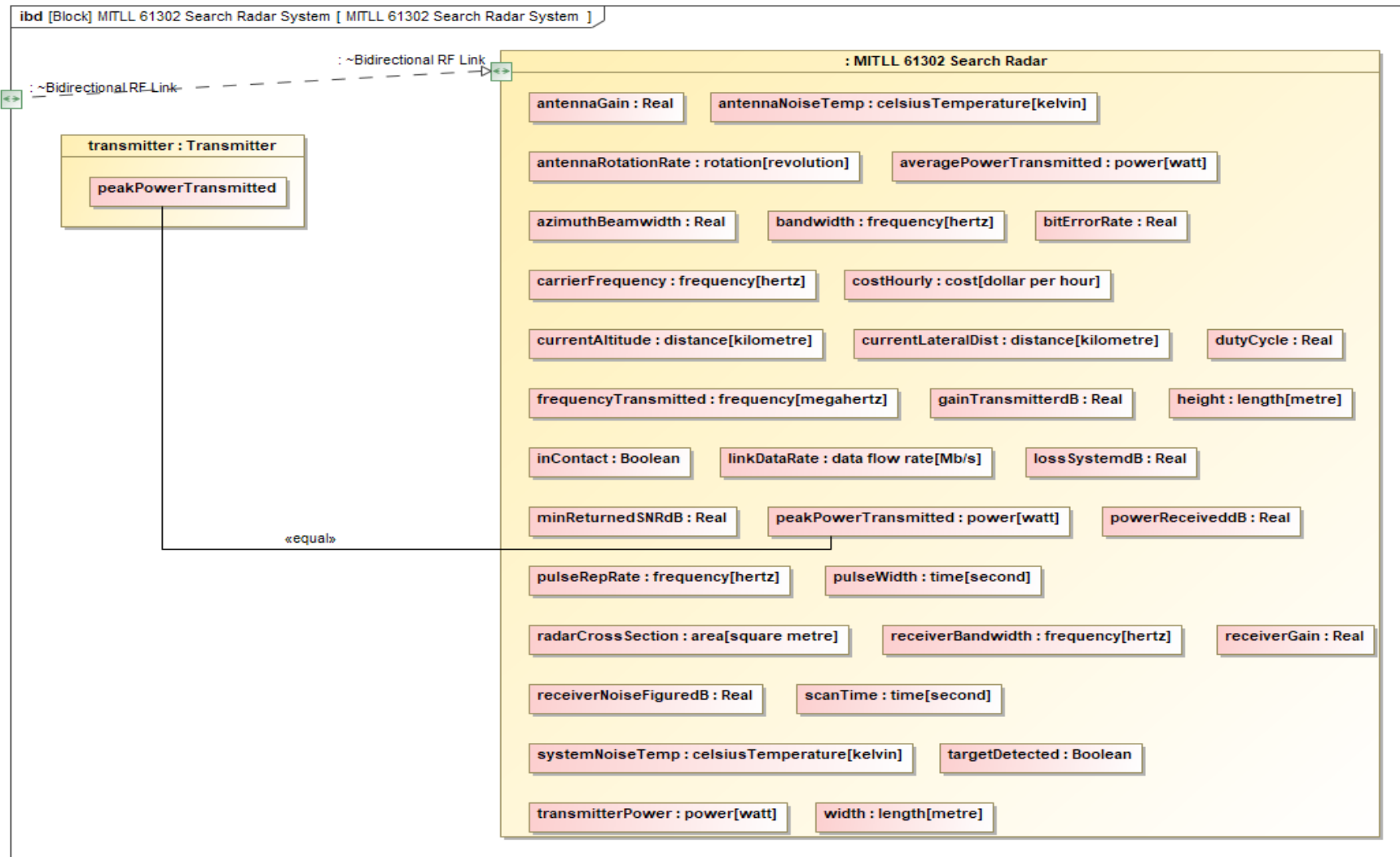


Figure 64 Surrogate Block Value Property Traceability

3.2.3.5. Mission System Definition - Behavior

Concurrently with the definition of the surrogate models, structural definition is the definition of behavior. At this stage, it is known what interfaces and information will be available to the surrogate, and there is reference information from the system model on how it should behave. It is not feasible to simply pull system level behavior as it may rely on structural depth or variables that are either no longer present or changed. As such, new behaviors intended to encapsulate the unique behavior of the system are created out of whole cloth. Those behaviors are not abstractions of the system model, but simplified and repackaged to support simulation. One important consideration at this juncture is on how the simulation framework will enact behavior. It could range from direct implementation of cameo behaviors to not using Cameo behavior at all and being restricted to native behavior types, all as a function of the maturity and details of implementation of the simulation framework [120]. This variability, as with all references to the simulation framework within this paper, precludes specific implementation by the methodology described here. Instead, the assumption will be that there are no external behaviors and behaviors need to be constructed within Cameo and will be run exclusively therein.

Having established that behavior will exist solely in Cameo, there are a couple of recommendations that can be pulled from the literature review: use of opaque actions, signals, and state machines. As mentioned previously, there is no overarching behavior for the mission. While easier to implement, an overarching mission level behavior would present an unrealistic picture of the mission and could lead to false conclusions. Instead, behavior is started simultaneously and run concurrently based only on interactions of those separate, concurrent, behaviors. While a mission level diagram can be useful to explain the intended mission flow, the concepts on that diagram are abstractions of operations conducted/realized by the platforms of the mission. This is why emphasis is placed on the platform behavior rather than centrally controlled. Therefore, as a rule, each platform surrogate is controlled by a state machine that determines what set of activities are operating at any one time, such as the example blow (Figure 65):

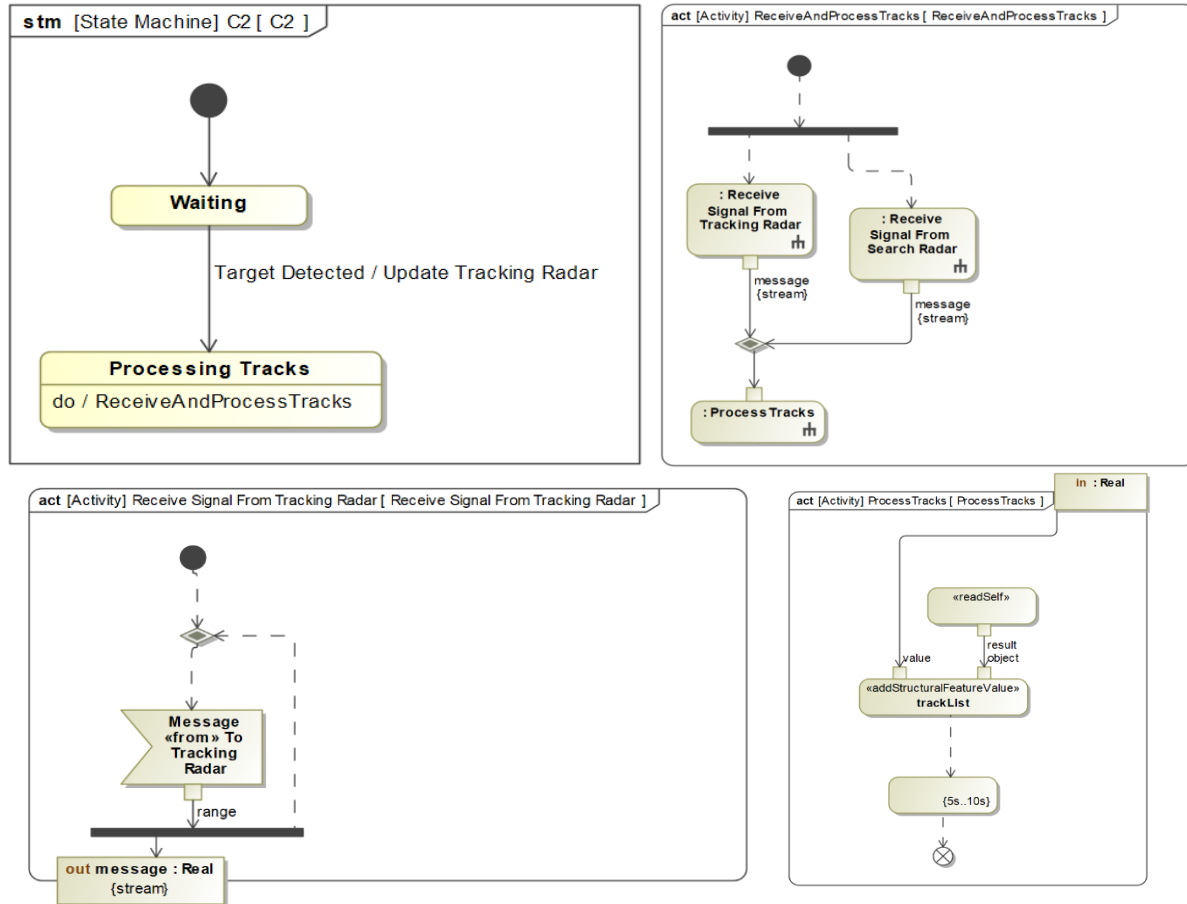


Figure 65 C2 Behavior State Machine (Top Left), Top Level Activity (Top Right), Message Reception (Bottom Left), and Track Storage Management (Bottom Right)

There are two items of note in these diagrams, the use of opaque actions and looping. As discussed in the Hypermodeling and SAIC Profile section, opaque actions contain executable code, a counterpart for constraints on *ibd/par* diagrams. In Figure 65 the only opaque action shown is the timer in the *Track Storage Management* section of the bottom right. Their defining feature is the ability to execute text based code, and represent a key attribute in capturing detailed systems behavior in a self-contained, concise format. In the diagram above, the processing of incoming tracks (which is presumably a detailed executable behavior within the system model) is reduced down to a duration constraint. While this is a simple example, the ability to inject text based code allows for effective abstraction of more complicated behavior.

Looping can be done in two ways: within the activity itself as shown above, or as a transition to self within the state machine, as is done in the “threat aircraft” state machine below:

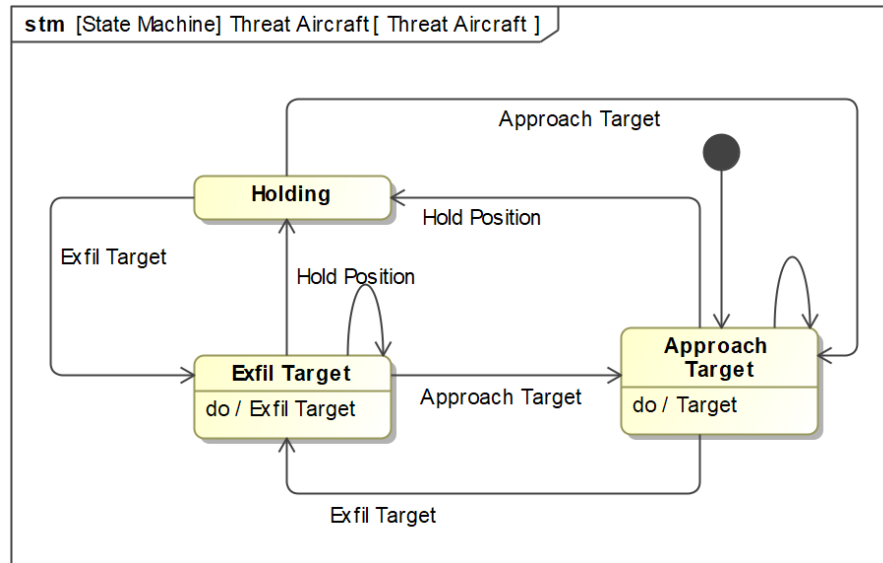


Figure 66 Threat Aircraft State Machine

The two options have one key difference, a transition to self-state change has the effect of clearing out the variables not saved to a value property, whereas looping within the activity never exits. Looping, as shown in Figure 65, is facilitated by use of the {stream} property on the activity pin, which allows for buffering of values on the pin, and will kick the token to the recipient without the activity itself finishing. Given that the activity is internally looping, it will not finish until a state change occurs. This also helps to ensure that the “accept event” action is receptive more of the time, as there are less actions to step through before it is reopened, decreasing the likelihood that a signal is not consumed in simulation.

While the transitions used on the previous two state machines are signal based, there are use cases where the JPL State Analysis use of functional state variables is useful, primarily where there needs to be persistent axis to a an inequality that may be constituted of different systems for different missions. What this means in practice is that the “search radar” state machine below in Figure 67 needs to transition from having no detections to having detections when the power received exceeds the sensitivity threshold. This change could be a direct inequality within the trigger, but this transition is significant enough that it will be used in multiple places and could be queried by external blocks, such as the “C2 Node”. The “C2 Node” may require that information in order to inform its own commands to the system or other systems as it functions as the “control system” to the rest of the kill chain (which are the “systems under control”).

The last point of behavioral modeling to address, before considering the surrogate model complete is decision making facilitated through usage of an association block, by way of the typed proxy port. While this does require the system to know whether or not the message can make it to the recipient prior to sending (that the “inContact” <<functionalStateVariable>> is based off of calculations made by the parametrics on the association, JPL State Analysis) it is a viable way to implement a conditional communications linkage.

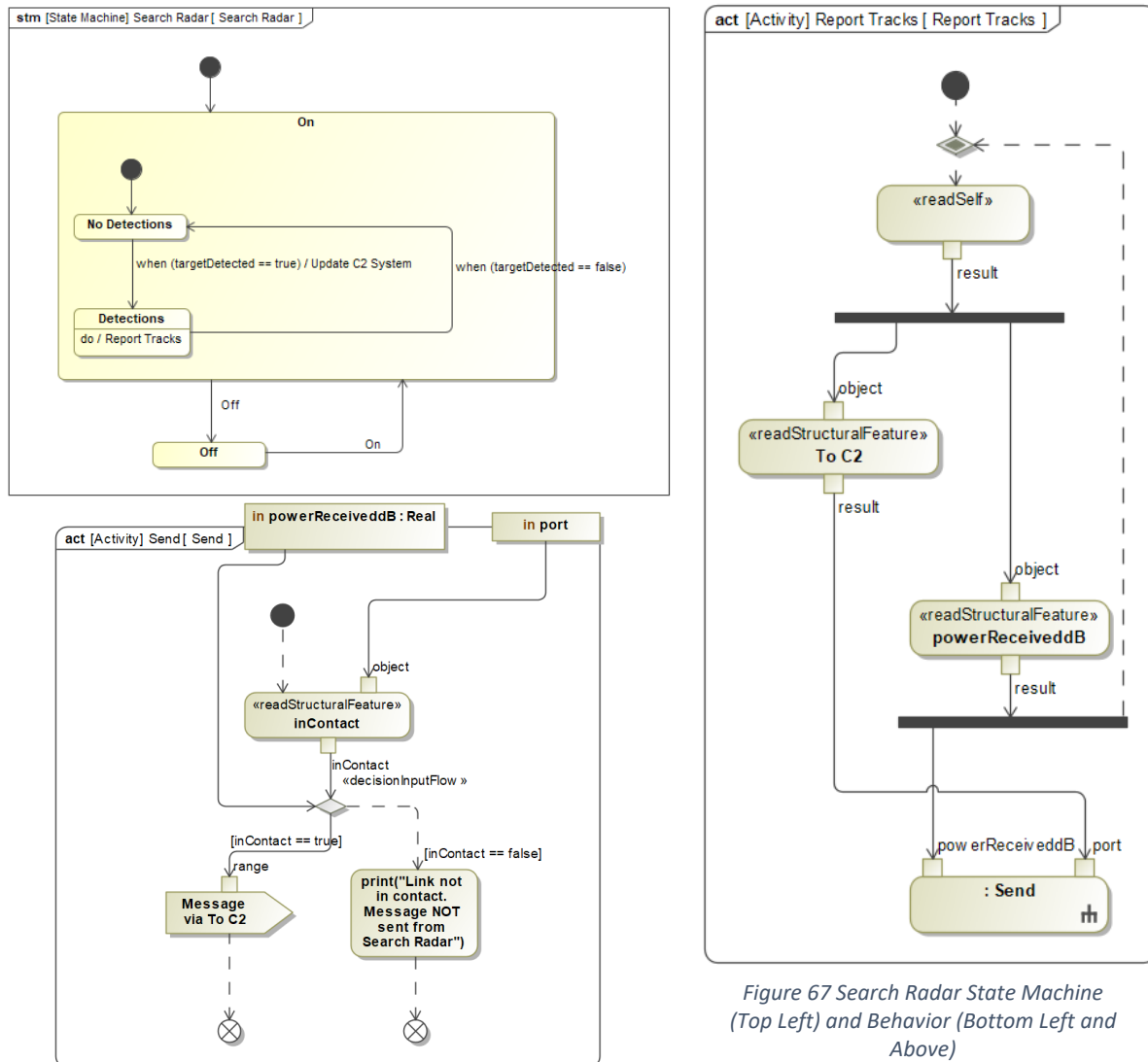


Figure 67 Search Radar State Machine (Top Left) and Behavior (Bottom Left and Above)

3.2.3.6. Note on ALH and dot notation

Within Cameo there is the Action Helper Language (ALH) and limited support of dot notation. It is possible to pass a block into a constraint parameter and access its contents through a call of *PartPropertyName.valuePropertyName*, ALH has a call of *ALH.getValue(PartPropertyName,*

“valuePropertyName”). Values can be set in a similar manner. The issue is that, if either the part property’s name, or the value property’s name changes, the linkage is broken. These calls appear to work on a sort of string compare, and do not have a means of updating all named usages of something such as found in an IDE like MATLABs. This is not a problem for anything that is typed by property, such as the *readStructuralFeature actions* on an activity diagram. While more concise, dot and ALH open the modeler up to significant potential problems, as the compiler in Cameo’s simulation window is (subjectively) less than helpful in identifying some problems. For these reasons, neither method is recommended unless the modeler is confident that the properties will not be changed in the future. This note is not a critique of No Magics implementation of SysML in Cameo, rather a cautionary note befitting an example bordering on tutorial, and a rationale why the approach was not adopted as part of the surrogate block approach in this example.

3.2.3.7. Mission Architecture and Parametrics

With the surrogate models built, and linked with the mission they participate in, the mission architecture itself is ready to be constructed (having progressed to the “Build Black Box Mission Model” step on Figure 56). The purpose of this section is three fold: establish the paths for signals to traverse, create the specific *connectors/interfaces* to enable that communication, and implement the math that supports analysis.

Creating the path for signals is straightforward, and does not deviate from how connectors are created on an *ibd* between typed proxy ports. The step of specifying the connector type through an association block with owned ends allows for the use of *parametrics* assigned to that association block. The utility of this for mission engineering is that communications are not assured. Whether it is the result of distance or enemy action, having logic within the model that can manipulate whether a message gets through is beneficial, particularly if it hooks into a higher fidelity simulation engine. To accomplish this with the same interface on both ends, an

association block with owned ends needs to be created in a self-referencing manner as shown below in Figure 68.

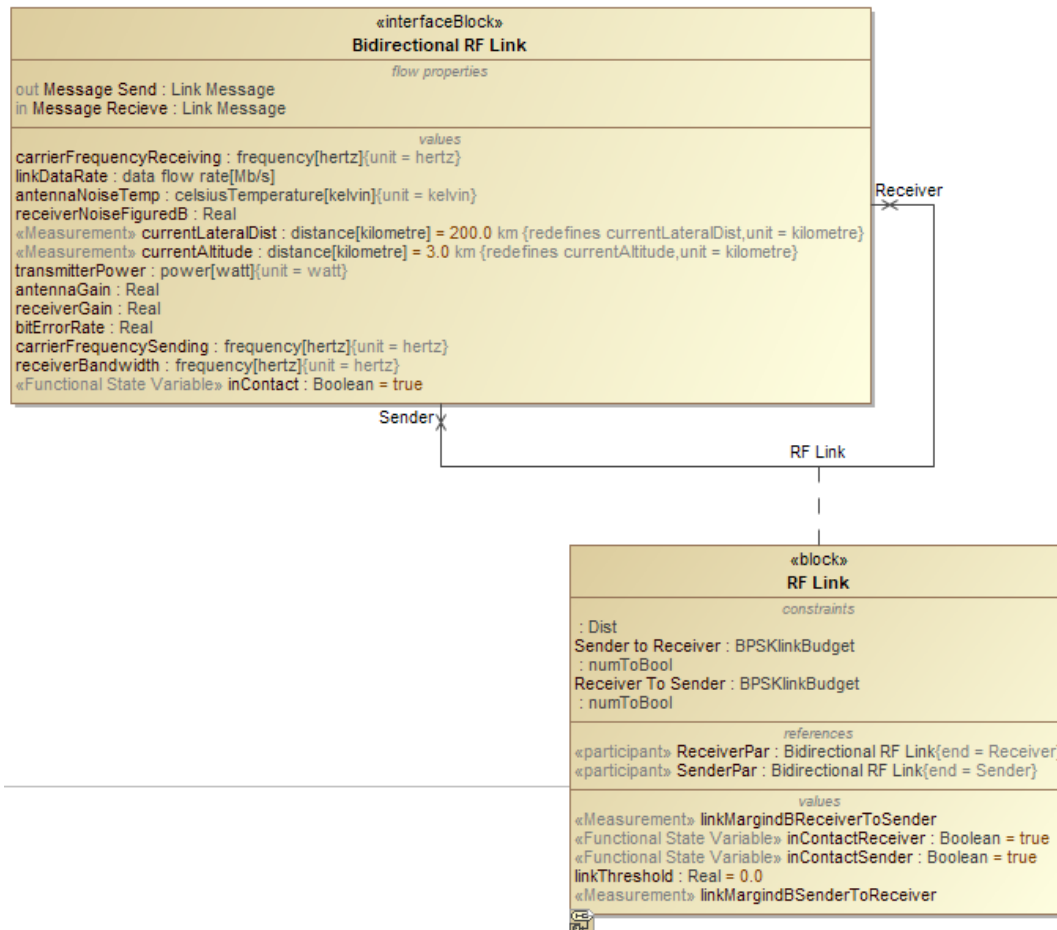


Figure 68 Self Referential Association Interface

This structure provides the three aspects of an interface identified in System of System Interface Modeling Method:

- Physical ends – *interface block*
- Connection flow – *flow properties* on the interface block
- Connection material – *association block* (defined by the parametrics stored there)

When typing a connector between two usages of the ports typed by the owning interface block, parametrics contained within the association block will be executed. These parametrics are built using participant properties, typed by reference properties of the two ends of the connector. Mentioned earlier, this use of reference properties, generated automatically during the creation of the association block, conflicts with the SAIC validation profile. Adoption of the validation profile would require avoidance of association blocks or acceptance of some errors. Beyond providing calculations on the connector, typing one with an association block will

generate a connector property, which is critical to accessing information calculated in the association for use in satisfying requirements and other constraints. Using all of these approaches yields a structural *ibd* that looks like the following:

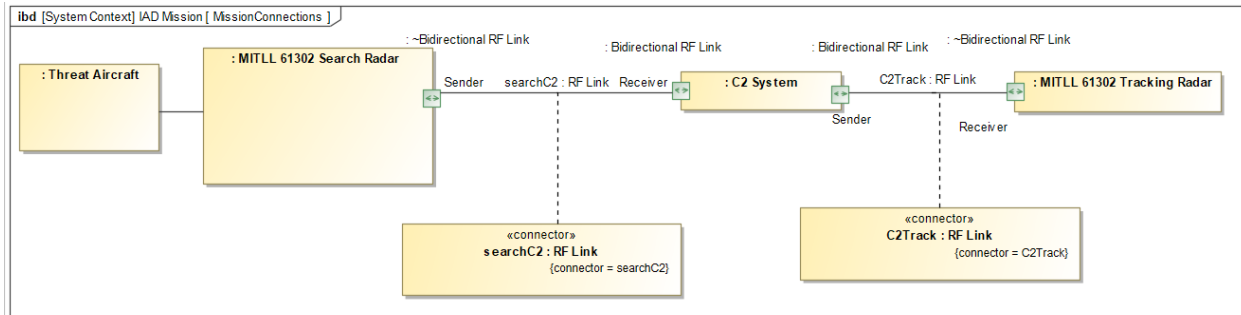


Figure 69 Mission ibd

This simple diagram shows the possible routes that communications can take. Of note is the lack of interfaces on the connection between the “threat aircraft” and the “search radar”, as the relation here is one of physics rather than intentional communications. As the surrogate blocks are black boxes without parts, there is no antenna for the connector to reach into the “search radar” part to, leaving only direct connection of the part properties. This diagram also contains instances of both physics based unavoidable interaction, between the “Threat Aircraft” and the “Search Radar”, as well as the physics based component of comms modeling. This latter component working in tandem with that described in Figure 67.

The simplicity of the *ibd*, also belies the visual complexity of the underlying math, as shown below:



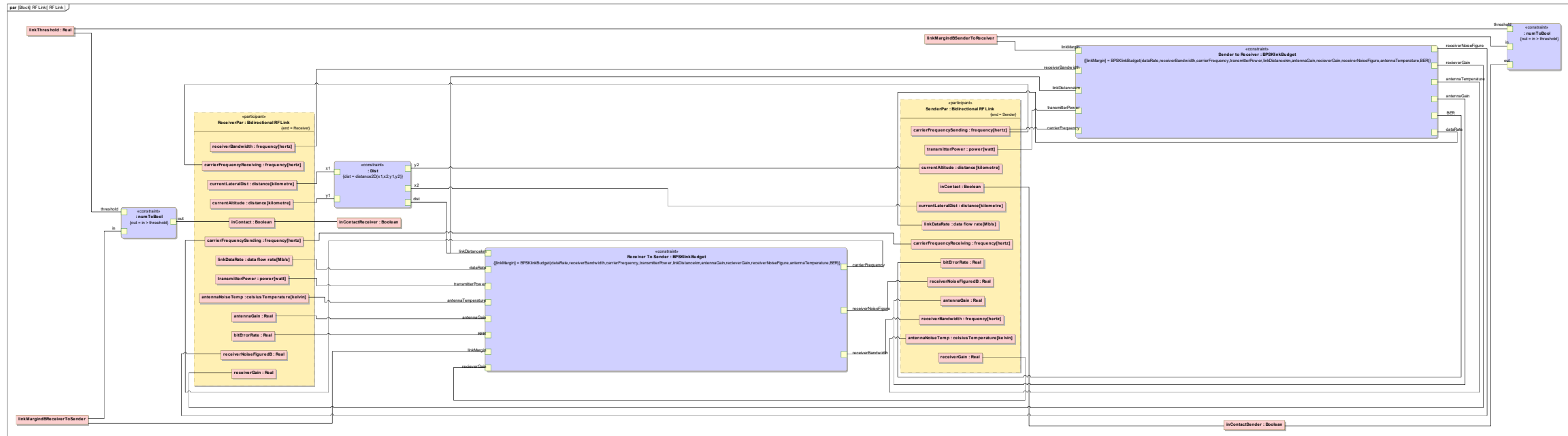
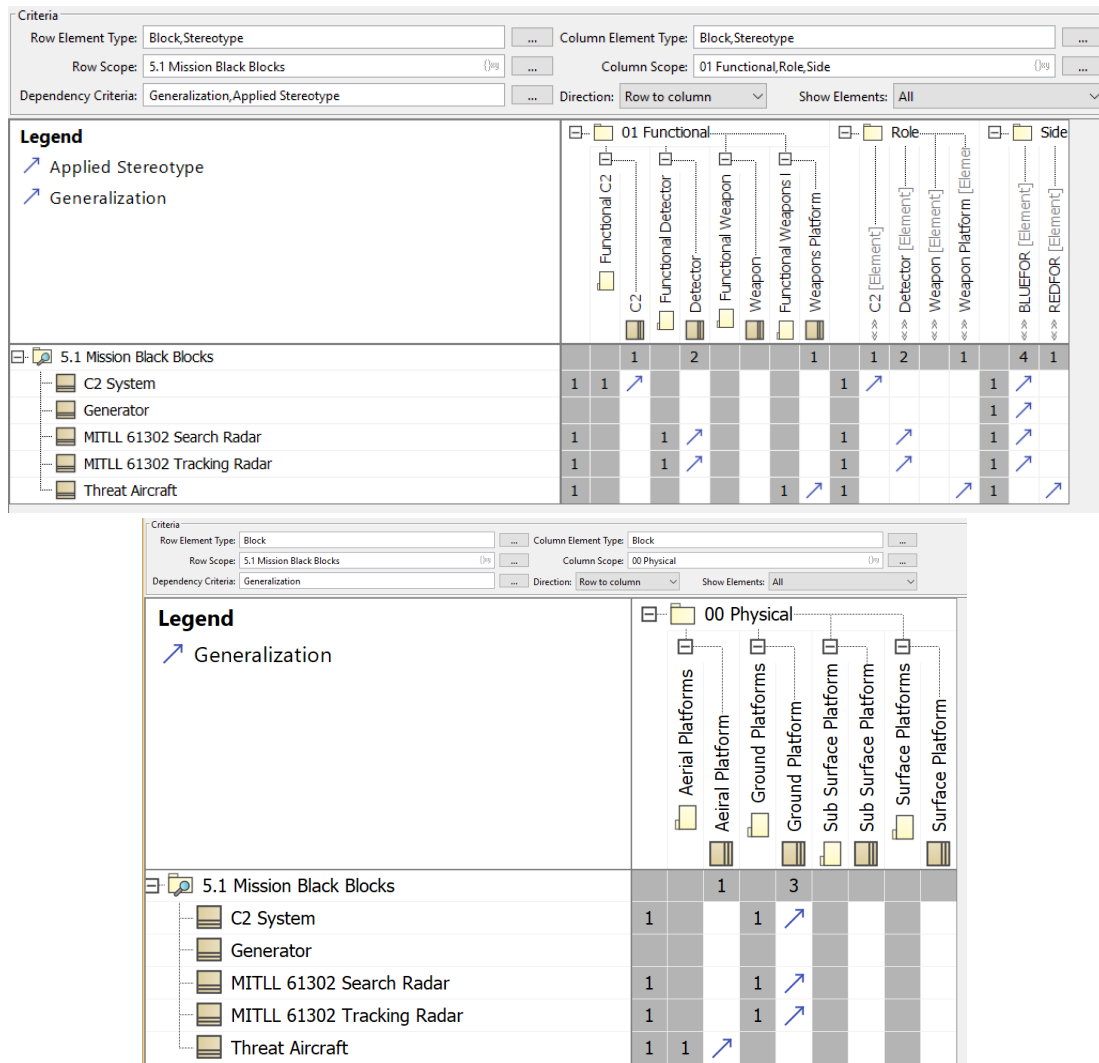


Figure 71 Connector Parametric Diagram

While there is nothing fundamentally novel about the usage of parametrics on the “IAD Mission” block, and only marginally so on the association block “RF Link”, the main takeaway is that the calculations are all being done outside of Cameo. While integration with MATLAB is native to Cameo (with admin privileges), this is in keeping with the external calculations provided by the mission engineering simulation framework, though the details of that framework are beyond the scope and sensitivity of this paper. The main impetus of this external simulation, as discussed later, is to leverage existing tools, and bypass one of Cameo’s limitations with respects to variable scope. It is prohibitively difficult and error prone to pass large quantities of variables between variables at the same level of abstraction, requiring redundant variables (which doesn’t scale well) or usage of methods that do not automatically update as variable names change.

At this stage, it is also necessary to verify that all functional roles of the mission are being fulfilled, and to summarize the systems being used. This can be accomplished through the use of matrices as shown below (Hypermodeling and SAIC Profile):

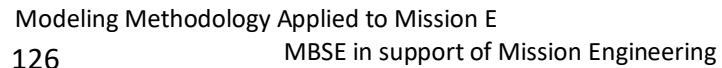


While the functional role will be auto populated based off of the definition of the surrogate block, there are three categories of assignment occurring in the Functional Role matrix. In the “Functional” section, a generalization to respective pattern block will be auto populated based off of the definition of the surrogate block. The “Role” section is an application of stereotype for the role that the surrogate is fulfilling in this particular mission. The “Role” must be a subset of “Functional” as a system cannot perform a role in a mission that it does not have the capability to perform. The last package “Side” simply denotes to which side the system belongs during that mission. The application of stereotypes in this manner allows for implementation of legends, and query tables useful on missions more complex than the example model. They are

not included here as there was nothing novel found in the literature review on the use of querying beyond the advocacy of their use in the Hypermodeling and SAIC Profile. Additionally the requisite background on metachaining and navigation within the model is complex enough to prohibit any useful brief summary in keeping with the intent of this already not so brief paper.

3.2.3.8. Requirements Satisfaction

The only addition to the requirements section detailed on 106 is the restriction on requirements relations to the <<satisfy>> relation as suggested within Hypermodeling and SAIC Profile. For the example here, with all the value properties filled out and linked with the requirements, the “IAD Mission” *bdd* looks as below (note not all inherited values have to be redefined, only those required for the simulation)



3.2.3.9. Simulation Results

With the model effectively complete, it is possible to start answering the questions that the customer originally proposed. As a review, the requirements gleaned from the question are available in Figure 60 above and are:

- Fuel consumption less than 100 kg per hour.
- Cost to operate less than 2000 dollars per hour.
- Communications function (Link Margin > 0) with a Bit error Rate less than or equal to $1E-6$.
- Detection range exceeds the Threats Weapon Engagement range.
- Identify any operational bottle necks in execution of the kill chain.

While operating in conjunction with a simulation framework will logically modify the exact simulation and analysis process, Cameo itself has rudimentary analysis capabilities to augment its simulation capabilities. There are a number of built in charts available on the *simulation configuration diagram*, and the creation of *simulation config* blocks allow for preset simulation configurations and outputs that are easily cycled through. Because simulation of this fidelity did not appear in the literature review, there is little to augment what would otherwise be considered a basic analysis. Therefore, only a brief review of the simulation setup is warranted. The simulation produced a state vs time chart, a time based plot of search radar power received, and a mission sequence diagram capturing everything that happened within the simulation. This configuration is captured in the *Simulation Configuration Diagram* below:

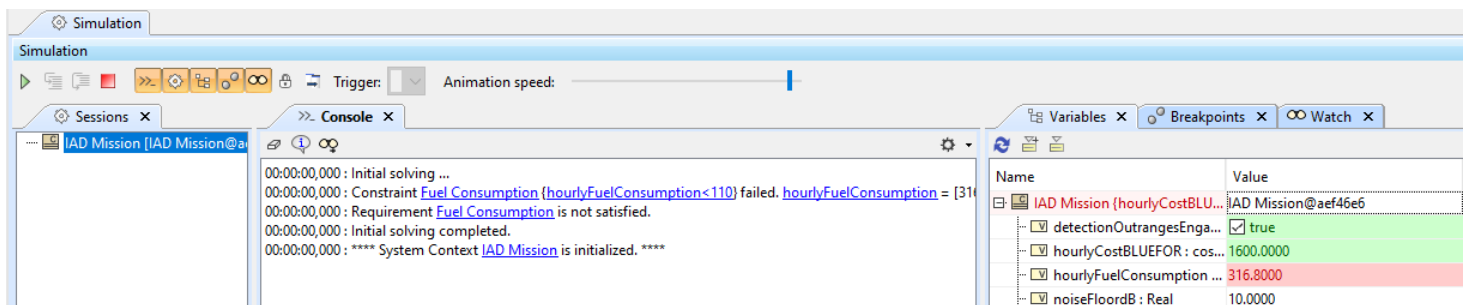


Figure 74 Initial Solving for IAD Mission Simulation

This simulation configuration yields the results shown in Figure 75 to Figure 79.

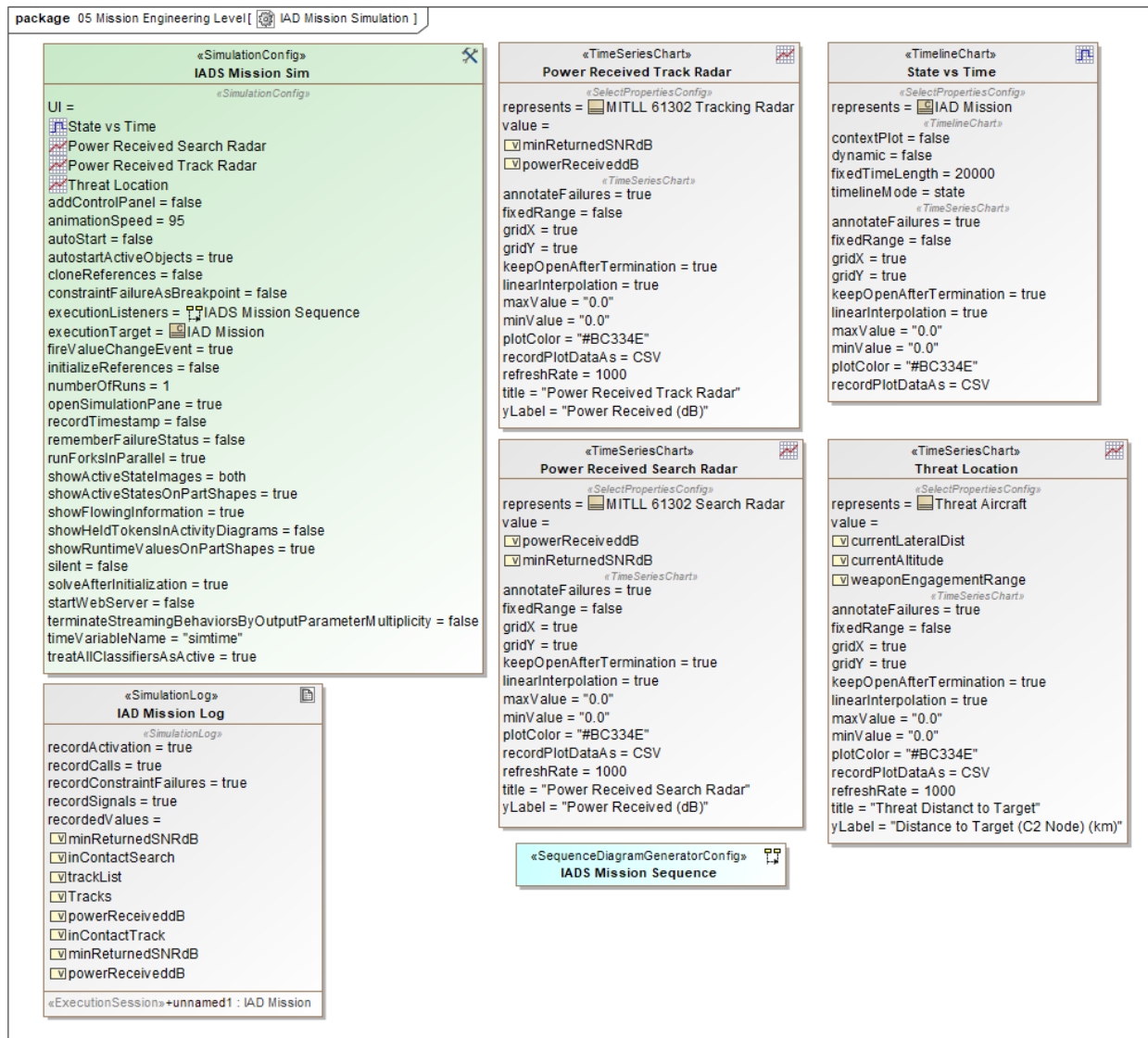


Figure 75 IAD Mission Simulation Configuration Diagram

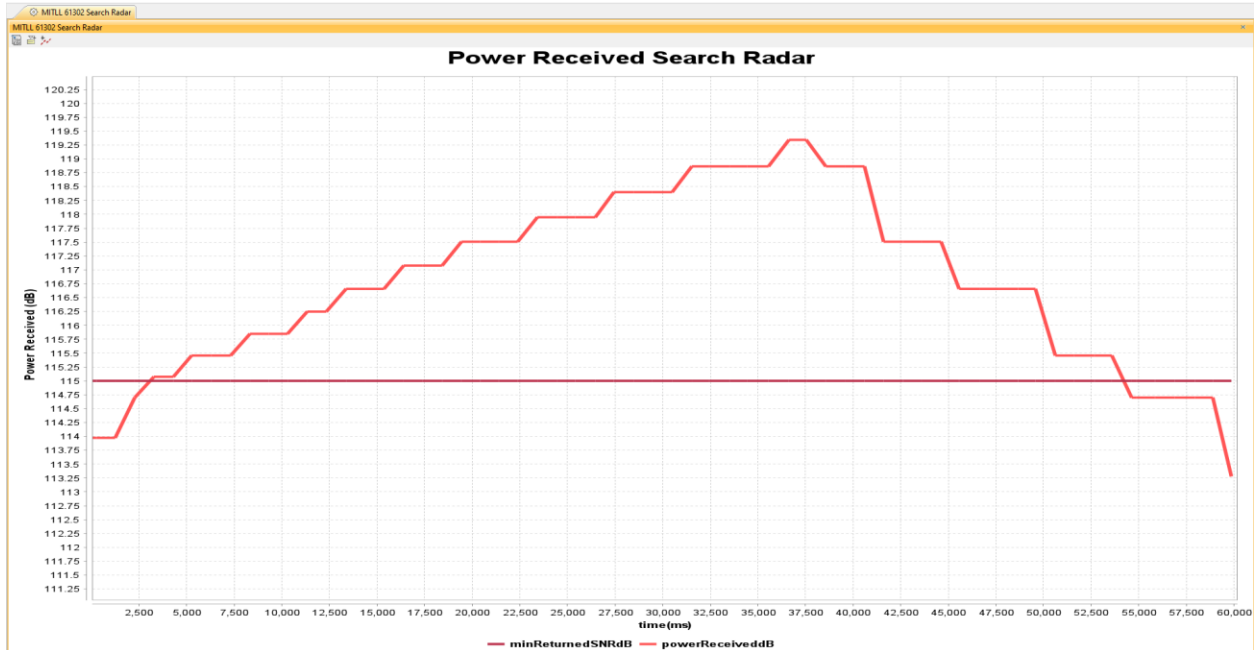


Figure 76 Power Received by Search Radar of Mission for IAD Mission Simulation

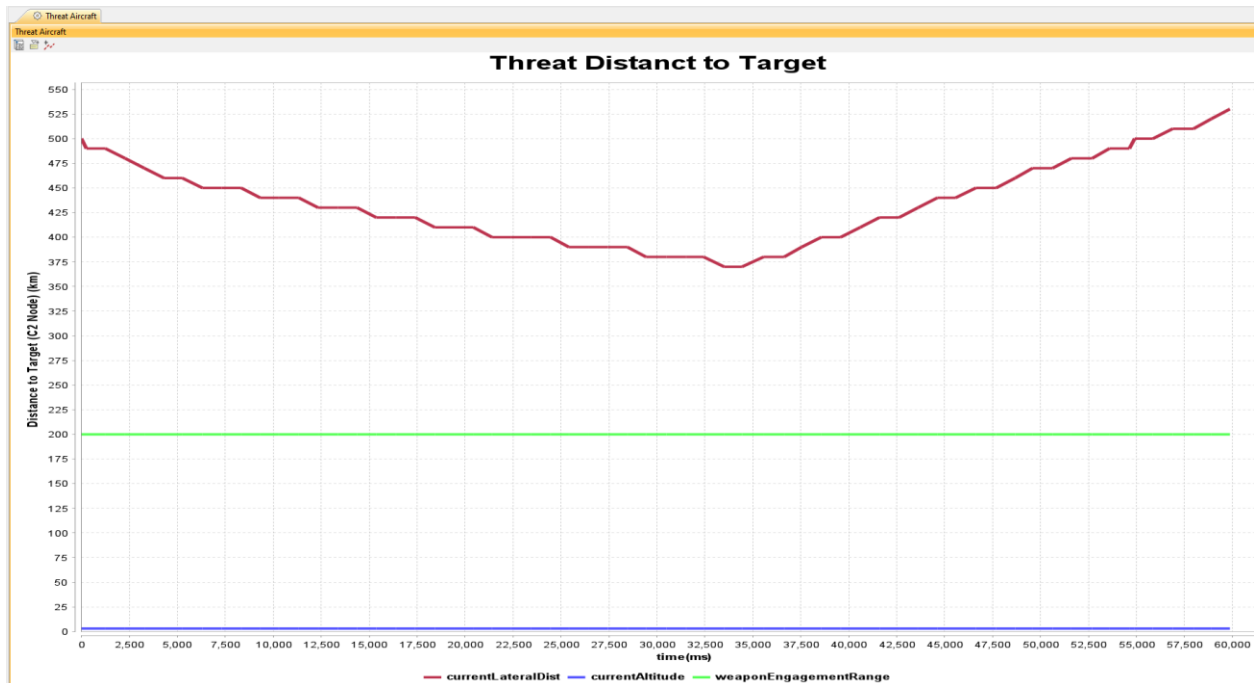


Figure 77 Threat Distance to Target and Engagement Range for IAD Mission Simulation

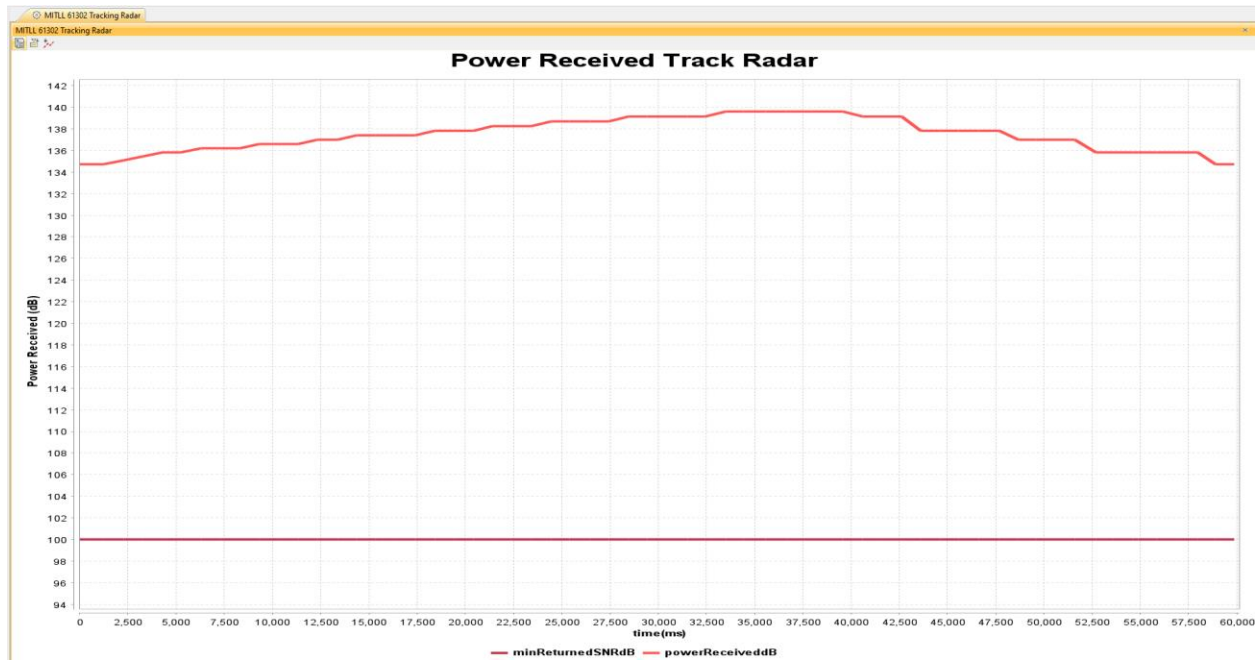


Figure 78 Power Received by Track Radar for IAD Mission Simulation

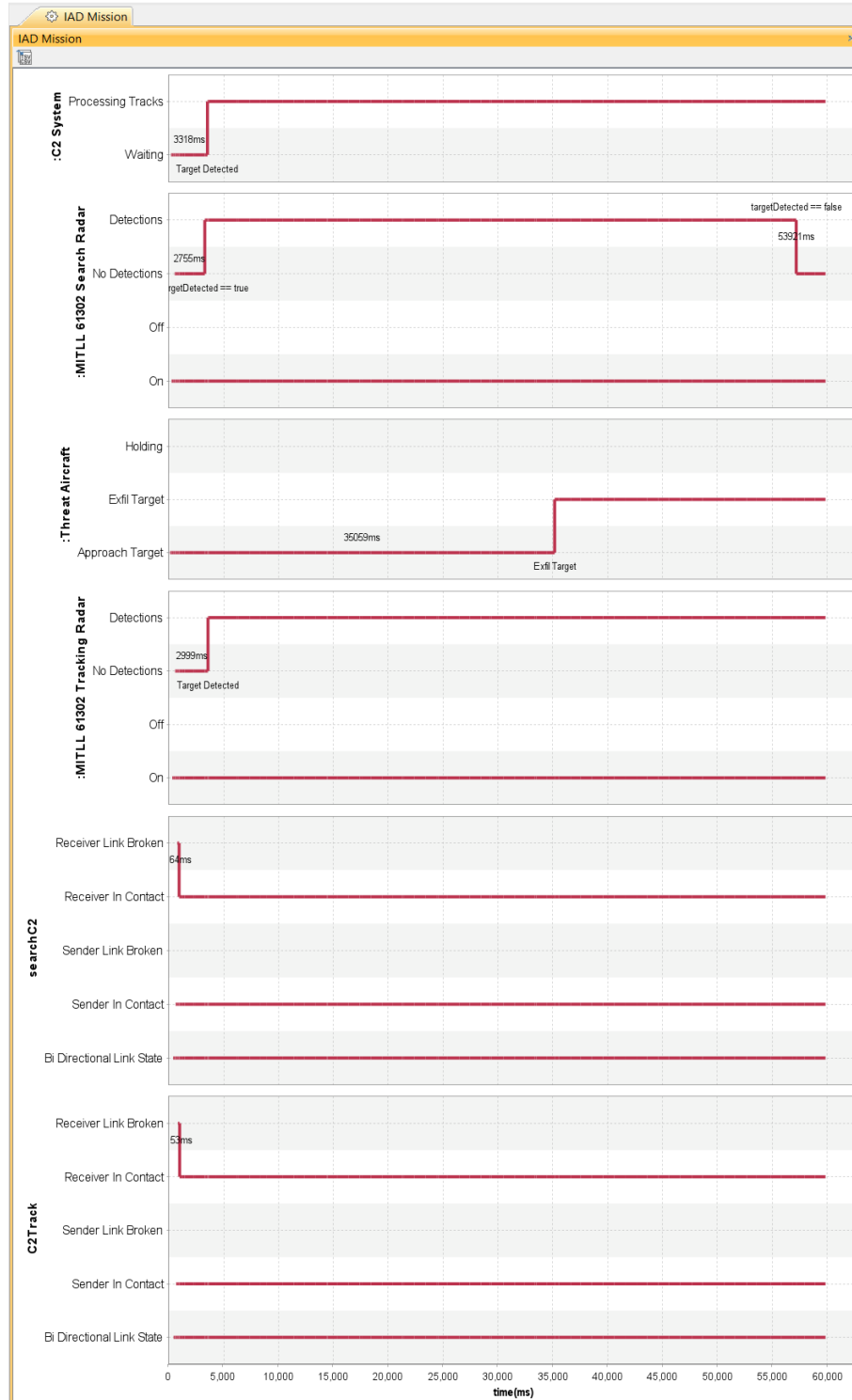
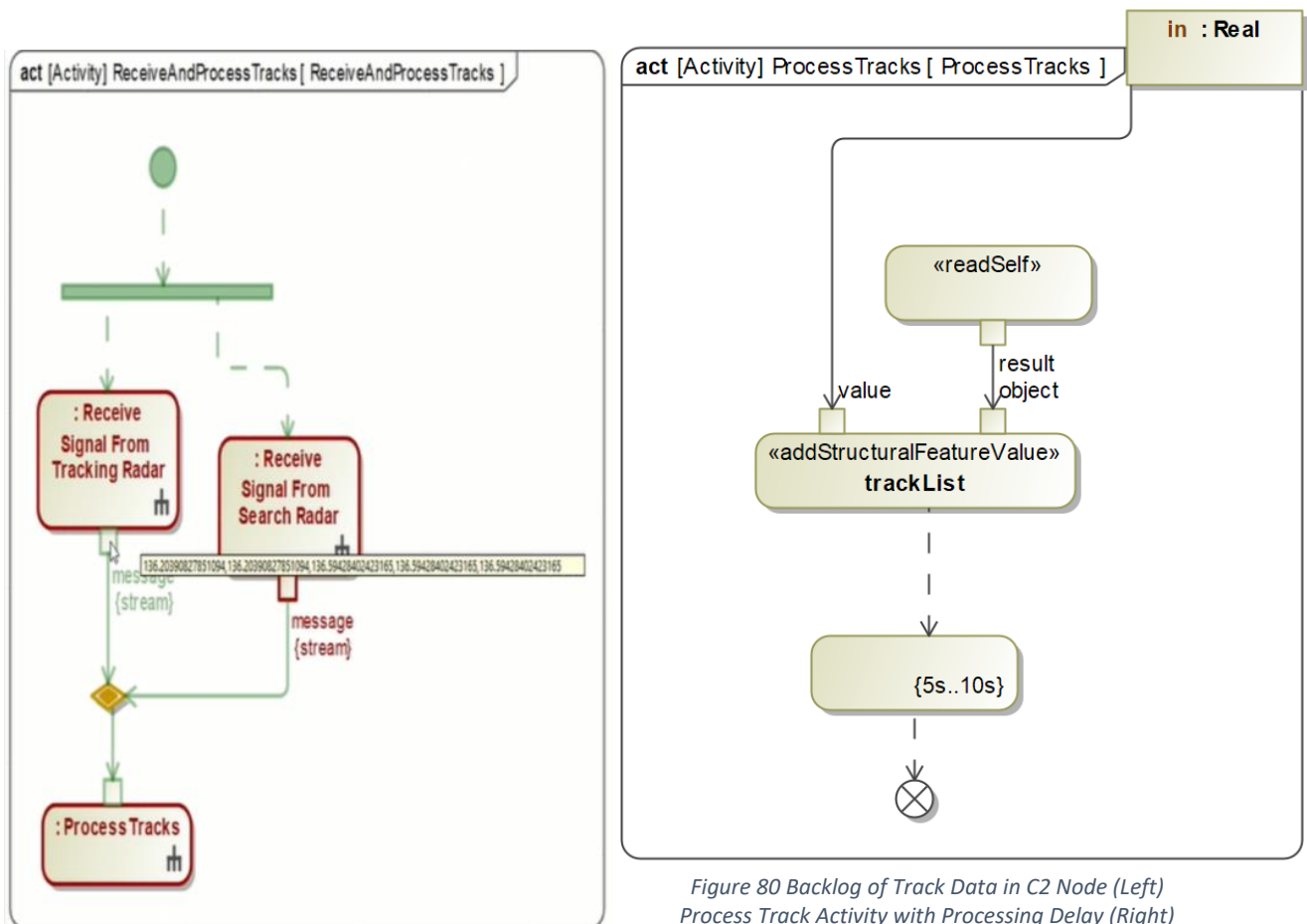


Figure 79 State Summary for IAD Mission Simulation

3.2.3.10. Simulation Results Analysis

In aggregate the figures answer all the questions initially asked. Approaching it on a per question basis, the results are as follows:

- Fuel consumption less than 110 kg per hour.
 - Figure 74 shows that the fuel consumed is 316 kg per hour, meaning that the requirement is failed for the configuration.
 - This fuel consumption does include the tracking radar as being powered on all the time. If the system had a nominally 0 fuel consumption standby state and short enough boot time such that it could be left off until a track is discovered the fuel consumption drops to 115 kg per hour. While still exceeding the desired consumption rate, 115 kg per hour is a more feasible issue to remedy through a more efficient generator, slightly lower operating power of the search radar, or acceptance of a small increase to logistical burden. Decreasing the power of the search radar is an acceptable trade for the given threat set as a sufficient decrease of the search radar average power output to 7600 W, still provides a detection at 460 km vs 470 km. Both values well exceed the 200 km weapons employment range of the notional threat. The main assumption is that the tracking radar has a spool up time fast enough to keep the system relevant. While this analysis could be conducted, it is not currently modeled in this analysis.
- Cost to operate less than 2000 dollars per hour.
 - Figure 74 shows that the cost requirement is met with an hourly cost of 1600.
- Communications function (Link Margin > 0) with a Bit error Rate less than or equal to $1E-6$.
 - Both conditions are met for the duration (once initialized) across all systems as can be seen in Figure 79.
- Detection range exceeds the Threats Weapon Engagement range.
 - Comparing the graphs of Figure 76 and Figure 77 it is shown that the target is detected well before it reaches the weapons engagement range. Detected at about the 470 km mark at 2800 ms.
 - This condition is still met even with the reduced power of the search radar.
- Identify any operational bottle necks in execution of the kill chain.
 - While not a direct simulation result, track processing is a bottleneck, as can be seen in the backlog of Figure 80.



- This bottle neck results in significant delays in processing tracks for any follow on usage, such as data fusion or weapons engagement. The delay, while a simple constraint, represents the aggregate process of both hardware computational time and human operator interaction. It is a random variable within the range of 5-10 seconds to emulate the variability inherent to any human interaction. This opaque action is a prime example of a location heavily dependent upon a detailed understanding of the system model. Without that fidelity, all that this analysis reveals is that the processing time is too slow to keep up with the rate of information being passed in from both the search and tracking radars, and further analysis is required to better identify the cause of the delay and any risk mitigation strategies or solutions that can be implemented.

3.2.4. Application of Results

There are two main types of results discussed in the previous section: operational and technical. These results are feeders for follow on analysis that answer questions at very different levels, at both the systems level and the operational level.

Traditional wargame and mission analysis results yield probabilities. That a radar's performance is captured with a P_{detect} . This probability is typically location and condition agnostic, with significant assumptions made about both the sensor and the assumed target. These assumptions are problematic for system designers as it introduces assumptions at the most fundamental level of the design, which will therefore propagate throughout to the end product. Mission Engineering provides an operational level analysis with a systems engineering level of fidelity, though the cost is additional iterations or the need for a Monte Carlo style analysis to offset the decrease in generality afforded by a P_{detect} . The data created here can be analyzed for parameter sensitivity and guide system requirements.

For the operational stakeholder the questions are capability based, rather than "what system parameters are dominant design variables". The capability based questions focus more along the lines of "are there issues with the tactics techniques and procedures employed?" Or "is an IAD battery at this location the best option, or should a combat air patrol be used instead?" The answers to these types of questions are still improved by a higher fidelity simulation environment as it reduces technical inaccuracies from the assumptions inherent in P_{detect} , however the improvement is not as immediately stark. The benefit comes from when the systems built from the capability gap analysis.

While both applications of Mission Engineering results are beyond the direct scope of Mission Engineering, it is necessary for any data generation process to understand how that data could be used. Ensure that it is accurately and ethically employed, and modify the generation process to do so. Mission Engineering enables both types of analysis to be more accurate, and more importantly, technically connected in an integrated process.

3.2.5. Example Model Summary

Overall, the simulation and results built off the model, while rudimentary, demonstrate that it is possible to answer relevant questions pertaining to survivability, performance, supportability, and sustainability, in a simple model while elucidating emergent relations, such as survivability of an air search radar being bound by the local road conditions supporting the site. Identifying these emergent behaviors is one of the most powerful abilities of conducting analysis in this way. Additionally, these elements created through this process are now usable in other missions, shortening the time to product of future analyses.

Additionally, by use of MATLAB as a surrogate external simulation engine, and the surrogate models, the concept of abstracted approximations of entire systems tying to an exterior simulation framework has been tested and found satisfactory. While there are always more stringent and complex tests to be run, for a foundational methodology to be built on and tailored, this simple example is sufficient to exemplify the novel aspects discussed here, primarily:

- Proving out that mission modeling and analytics can be conducted based on model based systems engineering methodologies identified in the literature review.
- Self-referencing association blocks with embedded parametrics and behavior.
- System behavior based on the state of a connector.
- The platform abstraction derived from the simulation framework toolset to support definition of the surrogate model.

3.3. Future Work

This methodology and example represent only one of the many possible implementations of this foundational methodology. It is the intent that the example work laid here is built on and tailored for individual customers rather than blindly adopted by providers. As highlighted by the drastic differences in MBSE methodologies reviewed in the Current MBSE Methodologies section, MBSE as a discipline is wide ranging. Such breadth makes a single uniform approach to every use case impractical. While this methodology is tailored towards Mission Engineering, it is proved out only in a specific Integrated Air Defense (IAD) mission, and will inevitably need tailoring. To illustrate this point, below are several areas that this approach would have to build upon:

- Surrogate block definition
 - The form of the surrogate block is dictated by the simulation framework. As the simulation framework is not defined here, the exact contents of the surrogate block will need to be identified by the user, based on their mission set and the tools available to them at that time.
 - Details of interaction with the simulation framework. While operations at the <<pattern>> level are intended to provide hooks for future integration, the details of how the model interacts with an external simulator is tool specific, and beyond the scope of this thesis due to sensitivity concerns, and the tradeoff that any solution arrived at would be narrowly applicable.
 - Additionally, if multiple customers are to be supported, a hierarchy of surrogate blocks, tailored to the customer may be used, in support of a changing simulation framework and for purposes of run time optimization and simplicity. This concept is shown below in Figure 81:

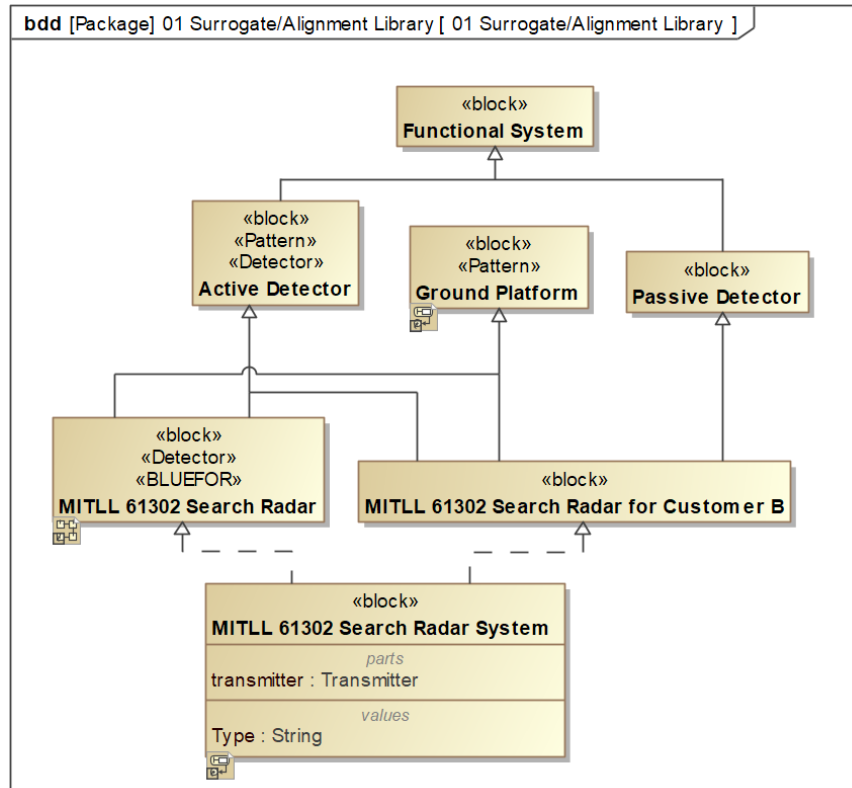


Figure 81 Multiple Surrogate Blocks of the Same System for Different Customers

As there was only one customer in the example mission, and the requirements and values arbitrary due to sensitivity considerations, this concept was not explored further in the example.

- One way ports. The association blocks are currently self-referencing, which works for bidirectional communications between like interfaces, such as a TCP/IP. However for transmissions in the blind, or one way protocols like UDP, two different interfaces may be considered to prevent a surplus of redundant communications on one end. However this poses ownership concerns, as to which end contains the association block. A notional solution would be inheritance, where a sender and a receiver are both subtypes of a common interface, which itself contains the association block. However, this concept was not explored in the example because communications needed to be bidirectional as sensors feed information to the C2 system and the C2 system needs a path to exercise control over the sensors, in accordance with the control system/system under control concept identified from the JPL State Analysis.

These examples show how this methodology, like many others, is still evolving. However, it would be beneficial to examine, in more detail, a significant evolution of the method that has occurred since the start of authoring this paper.

3.3.1. Example of Growth

A consistent theme that has been reiterated throughout the methodology outlined in this paper is that any methodology will have room for improvement to better suit a specific use case. An example of this has already happened during the interim between finalization of the methodology and conclusion of writing this paper, which will be outlined here as an example of how the methodology continuously grows and adapts.

One of the aspects of the methodology described here was the use of association blocks and connector properties (Figure 68 and Figure 69). This method was originally developed to support parametrics between two entities, rather than behavior. Having the *association block* type a *connector* was sufficient for that use case. There was no need to retain or create a *connector property*, though it was created and displayed as a visual representation of the *connector's* type, as the represented network component is crucial to modern networked kill chains. However, extension of this approach to behavior lead to the decision making behavior of Figure 67, highlighted in an excerpt below as Figure 82:

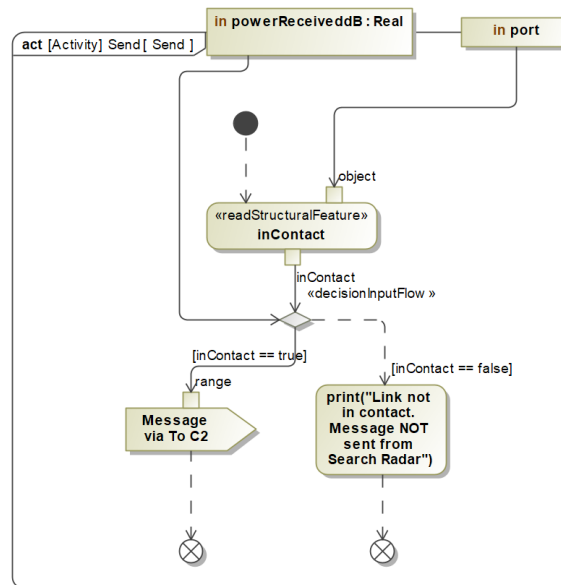


Figure 82 Send Behavior on the Sending System

While functional enough for simulation, it imposed knowledge on the sending platform that the sending platform would not know in reality. This discrepancy from reality is sub-optimal. The eventual solution was to better utilize the connector property by interposing it between the sender and receiver, as shown in the simplified development model excerpt below:

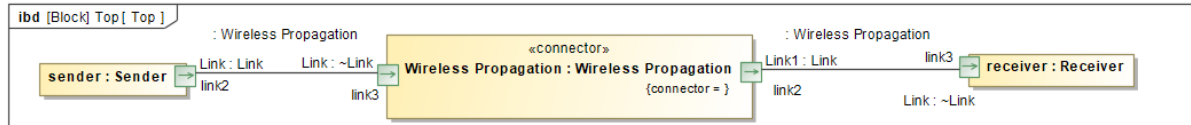


Figure 83 Improved Communications Behavior

This method does require that the *association block*, in addition to typing both connectors, has *proxy ports* typed by the owning *interface* as shown below:

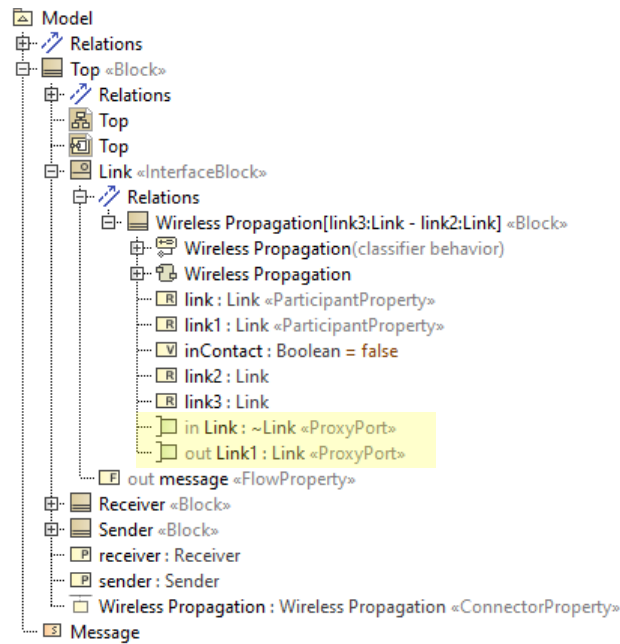


Figure 84 Improved Communications Model Structure

The additional elements, both within the *association block* and on the *ibd*, is offset by the ability to adjudicate message conveyance independent of both the sending and receiving platforms. This prevents any unique modifications to the platforms behavior, and more accurately compartmentalizes information while retaining all the advantages of the *association block* connector typing discussed in the previous section. This yields an *ibd* as in Figure 85:

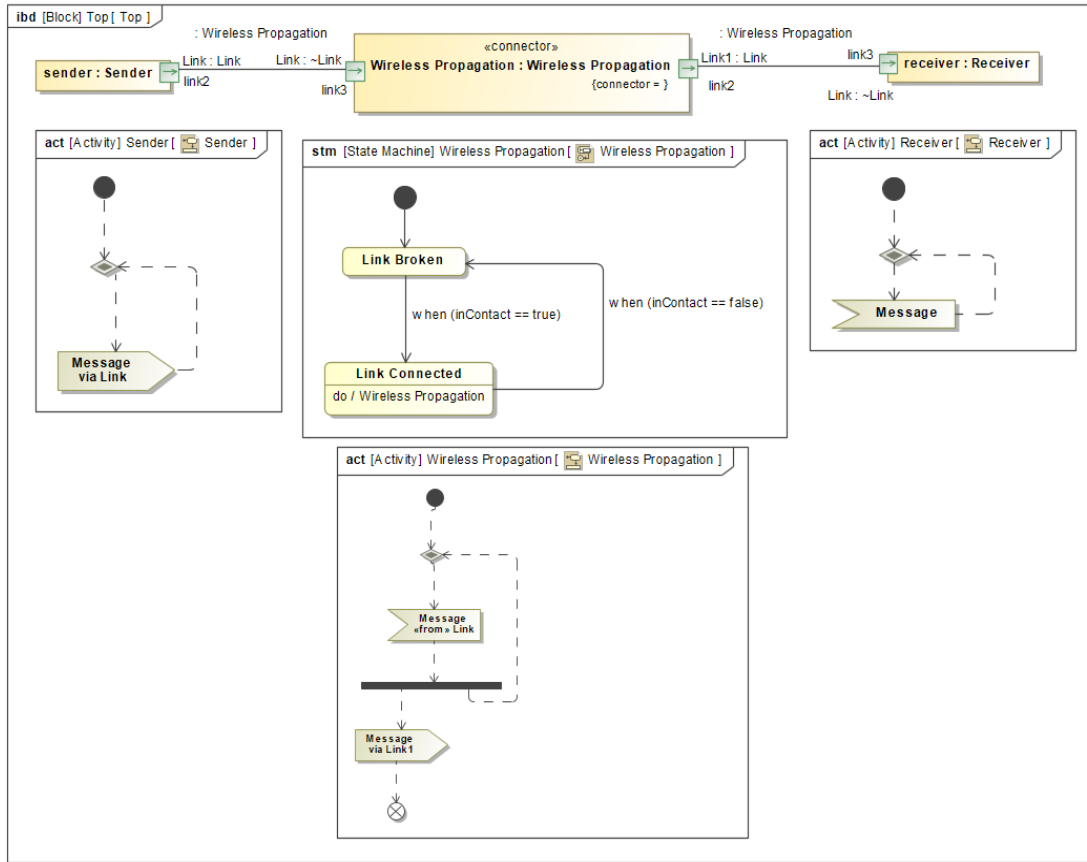


Figure 85 Dashboard View of an Association Block Interaction

The state of the connector is still determined by information drawn from the sender and receiver setting *functionalStateVariables*. The addition is a *stm* controlled *act* that simply relays the message when the state is active, and does not consume the sent signal token when inactive. This has the additional benefit of leveraging native Cameo error checking that will flag that the sender generated and sent a token that was not consumed, making error checking slightly easier. The improvement discussed in this section is one directional, as it was developed while working to solve the last bullet point on page 135, which regarded the assumed bi-directionality of communications in the example mission. While this method does realize that improvement in addition to the compartmentalization of information, there is still the vulnerability that the communications *connector* could be outrun by the sending system, and drop packets as a result of simulation rather than intentionally.

As stated, this section is left standalone within the future works section to highlight that there will always be room for improvement and optimization. This concept is related to a debate which has not yet been directly discussed in this paper: interoperable models vs universally adopted standard methodology. The argument for interoperable models lies on the assumption that it is not practicable to ensure common compliance with a universal methodology, and

emphasis should therefore be on a thin interface layer between models to facilitate interoperability. On the other hand, standardized methodology advocates argue that there is no better way to ensure model interoperability than a common style guide that is adhered to by all modelers. While it is reductive to state that this paper supports interoperable models rather than an enforced standardized models as a matter of practicable implementation because of this example of change and growth, this sort of continual evolution makes a static standard model at the current maturity of the discipline an unrealistic goal. An existing parallel can be drawn with text based code style guides. Both have a functional language defined and standardized, but there is no universal industry standard style guide that ensures out of the box interoperability. This is a result of differences in goals, expertise level, institutional knowledge, and modeling environment. All these factors prohibit a standardized modeling methods adoption out of the gate, leaving the emphasis on controlling an interface layer between two different models otherwise not interoperable (the genesis of the surrogate model developed in parallel by both SAIC and here at NSWC Crane).

3.3.2. Summary of Future Growth

The preceding sections outline just a few of the items that came up during the development of the example model. Although they were not included in this document for the reasons documented, they highlight the utility of such an example model to support a literary analysis. Working through the example also communicates the foundational aspect of the methodology posited here, better than any words can, giving a sense of how the conclusions arrived at here are intended to be used: as a jumping off point, not a turnkey solution for Model Based Mission Engineering.

Beyond improvements to the methodology itself there are a number of immediate applications of high fidelity simulations informed by detailed systems models. The two that will be examined here are 1. Synthetic data generation to support development of AI, and 2. Mission level multi-domain optimization and analysis (MDAO).

One of the primary challenges to training AI is the quantity of data required to get good results. There is already difficulty and expense in gathering sufficient real world data for robust decision making for accessible topics, a problem exacerbated further with the sensitive nature of wargaming's topics. Generating synthetic data is a means to mitigate this quantity issue, however synthetic data is inherently less "real" than real world data. Synthetic data generated using the methods described in this paper could be an improvement over existing sources of synthetic data. It will still suffer from inaccuracy inherent to the nature of synthetic data, which is driven by the accuracy of the generating model. However, for training data sets of less critical applications, or as design tools, the ML informed tools trained off synthetic data here could prove useful. Particularly considering the design perspective, where these tools would be used anyway and many of the risks are therefore already accepted. This method, by linking mission

level models directly with authoritative system models, stands to reduce the inaccuracy of the generation function by reducing the levels of assumptions.

Along the lines of the design perspective, linking design level MDAO tools with mission level analysis and metrics is a useful extension of this workflow. While there are already tools such as AGI Model Center that integrate MBSE models with CAD level tools, there has yet been connections to mission level analysis proved out in literature. The next step beyond the methodology discussion, such as here, is the creation of a similarly integrated performance simulation framework for Mission Engineering and wargaming. Such a framework would stand directly on the foundation laid out here, interfacing where the example mission leveraged MATLAB.

The improvements and applications discussed here are only a small subset of potential routes. They are proffered in accordance with the foundational nature of the methodology defined here. The primary driver of future work, and modifications to this methodology, should be customer focused, while maintaining the basic tenants discussed here.

4. Summary and Conclusions

This thesis has posited that a methodology to support Model Based Mission Engineering can be derived from advances in Model Based Systems Engineering (MBSE). In support of that hypothesis, a literature review of current Mission Engineering/Wargaming guidance and MBSE methodologies was conducted. The Mission Engineering and wargaming review yielded the following conclusions:

- Future missions will be cross-domain, increasing the complexity and reliance on technology outlined in the Technical Mission Priorities section.
- The fundamental analysis questions relate to cost, schedule, and performance, and will remain relevant under that more complex paradigm. (Determining KPPs)
- Supporting those fundamental analysis questions are five key performance parameters (Determining KPPs):
 - Force protection – centers on safeguarding the people
 - System survivability – focuses on increasing the resiliency of the system itself
 - Sustainment – concentrates on ensuring that a system is repairable and maintainable at the appropriate level
 - Energy – recognizes that modern warfare runs off of fuel, and accounts for its usage and supply
 - Net-ready performance – no system operates on its own, making interoperability absolutely key on a joint battlefield that is decided during design and construction

- There is already some precedent for Systems Engineering supporting Mission Analysis as shown in.

With the foundation that there is already a link between Mission and Systems Engineering, the focus shifts to determining applicability of existing MBSE Methodologies. Through this review process several key concepts were identified:

- The Systems Modeling Language (SysML) is the dominant language for MBSE.
- There is no industry standard methodological approach to MBSE due to the flexibility of the language and breadth of efforts supported. This flexibility and breadth demonstrate that application to Mission Engineering is not a misapplication of the language, as the language is not overly specialized on any particular use case.
- There are already cases of Mission considerations in MBSE models; [76], [98], and [116] however, they remain heavily system-focused. This concept further supports the applicability of MBSE methods to Mission Engineering.
- Review of the methodologies themselves yielded:
 - Overarching process for MBSE that supports Mission Engineering Objectives. (IBM Rational Telelogic Harmony , Onion Model)
 - Functional decomposition supported by object-oriented structure to reflect the configuration of mission functions supported by platforms. (INCOSE Object-Oriented Systems Engineering Method (OOSEM), IBM Rational Telelogic Harmony , JPL State Analysis)
 - Support of native analysis with external simulation (independently arrived at conclusion, CORE from Vitech) through both constraint blocks and opaque actions (Hypermodeling and SAIC Profile)
 - Surrogate blocks to represent complex system models in a single black box (independently arrived at conclusion, Hypermodeling and SAIC Profile, System of System Interface Modeling Method)
 - Classes of value properties pertaining to their usage (JPL State Analysis)
 - Use of metachains navigation for quality-of-life improvements and customer queried presentations (Hypermodeling and SAIC Profile)
 - The utility and limitations of additional validation profiles (Hypermodeling and SAIC Profile)

With both of these sets of concepts in mind, the subsequent chapter, MBSE in support of Mission Engineering, concludes that the two concepts are closely related through the term System of Systems Engineering. The primary differentiation is one of perspective and emphasis, whether it is on the mission functions and analysis or on system performance and behavior. Both views can use the same tools and methodologies, merely differing on which parts are

higher vs lower fidelity. Furthermore, the argument is supported that the two can be plotted along the same systems complexity axis, spanning component level to mission level. (Figure 46)

Having formally established the linkage between Mission Engineering and Systems Engineering, the existing MBSE methodologies as relevant to Mission Engineering, and the objectives of Mission Engineering and Mission analysis, the next step was application. Modeling the example mission allowed for highlighting of specific methodological items and their interoperability, as well as the connective tissue that fills in the gaps between said highlights. The review would have been incomplete without that enabling connective tissue, and it allowed proving that realistic operational questions can be answered by the proposed methodology.

Through the literature reviews, synthesis, and proof of concept application, it is determined with high confidence that advances in Model Based Systems Engineering can benefit the Mission Engineering use case as a subset of System of Systems Engineering. This conclusion provides a much greater breadth of development and information for Mission Engineers to draw from when optimizing and developing their capability in support of the warfighter.

Page Intentionally Left Blank

5. Appendix A: Bibliography

- [1] “F-35 Joint Strike Fighter: Cost Growth and Schedule Delays Continue | U.S. GAO.” <https://www.gao.gov/products/gao-22-105128> (accessed Aug. 18, 2022).
- [2] “Two Cost Studies Pertaining to the F-15 Aircraft Program | U.S. GAO.” <https://www.gao.gov/products/b-168664-0> (accessed Aug. 18, 2022).
- [3] “F-35 JOINT STRIKE FIGHTER Cost Growth and Schedule Delays Continue Report to Congressional Committees United States Government Accountability Office,” 2022.
- [4] “Navy Declares Initial Operational Capability for F-35C Joint Strike Fighter - USNI News.” <https://news.usni.org/2019/02/28/navy-declares-initial-operational-capability-for-f-35c-joint-strike-fighter> (accessed Aug. 18, 2022).
- [5] “Model-based systems engineering: Revolution or evolution?,” 2011.
- [6] “(PDF) Evolution of Model-Based System Engineering Methodologies for the Design of Space Systems in the Advanced Stages of the Project (Phases B-C).” https://www.researchgate.net/publication/280238984_Evolution_of_Model-Based_System_Engineering_Methodologies_for_the_Design_of_Space_Systems_in_the_Advanced_Stages_of_the_Project_Phases_B-C (accessed Aug. 18, 2022).
- [7] *Necessary DoD Range Capabilities to Ensure Operational Superiority of U.S. Defense Systems*. Washington, D.C.: National Academies Press, 2021. doi: 10.17226/26181.
- [8] P. D. Clive, J. A. Johnson, M. J. Moss, J. M. Zeh, B. M. Birkmire, and D. D. Hodson, “Advanced Framework for Simulation, Integration and Modeling (AFSIM)”.
- [9] Odusda and Tsse, “Systems Engineering Guide for Systems of Systems, V 1.0,” 2008.
- [10] “Engineering of Defense Systems Guidebook,” 2022, Accessed: Apr. 17, 2022. [Online]. Available: <https://ac.cto.mil/engineering>
- [11] USD (AT&L), “DoD Instruction 5000.02, January 7, 2015,” 2015.
- [12] R. J. Torok and C. A. Whitcomb, “NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA THESIS USING MODEL-BASED SYSTEMS ENGINEERING METHODS TO CAPTURE A DEPARTMENT OF DEFENSE ACQUISITION LIFE CYCLE,” 2021.
- [13] D. Walden, G. Roedler, K. Forsberg, R. Douglas Hamelin, and T. Shortell, “INCOSE Systems Engineering Handbook 4e 2015 07,” 2015.
- [14] H. Sillitto *et al.*, “Systems Engineering and System Definitions,” 2019.

- [15] "Final Report of the Model Based Engineering (MBE) Subcommittee NDIA Systems Engineering Division M&S Committee," 2011.
- [16] L. Delligatti, "Business Case for Model-Based Systems Engineering (MBSE) with SysML Metrics from an Aerospace Industry Case Study".
- [17] G. Bennett, "The Identification of Metrics to Support a Model Based Systems Engineering Approach to Requirements Engineering and Architectural Design".
- [18] Dod, "DoD Electromagnetic Spectrum Superiority Strategy 2020," 2020.
- [19] "DoD CIO Says Spectrum May Become Warfighting Domain - Breaking Defense." <https://breakingdefense.com/2015/12/dod-cio-says-spectrum-may-become-warfighting-domain/> (accessed Aug. 18, 2022).
- [20] United States Joint Staff Joint Force Development J7, "Cross Domain Synergy in Joint Operations," 2016.
- [21] Office of the Under Secretary of Defense for Research and Engineering, "Mission Engineering Guide," Washington, DC, 2020. Accessed: Jan. 27, 2021. [Online]. Available: <https://ac.cto.mil/engineering>
- [22] M. Blackburn, "Transforming Systems Engineering through Model-Centric Engineering TR-111," 2018.
- [23] "SERC-SR-2020-001-Benchmarking-the-Benefits-and-Current-Maturity-of-MBSE-3-2020".
- [24] M. B. Caffrey, "On Wargaming How Wargames Have Shaped History and How They May Shape the Future," 2019.
- [25] C. and D. C. UK Ministry of Defense - Development, *Wargaming Handbook Development, Concepts and Doctrine Centre*. Wiltshire: Ministry of Defence Shrivenham, 2017.
- [26] M. John and M. Brennan, "Assessment of SIMBAT for Naval Operations," 2006.
- [27] M. Kofman, K. Migacheva, B. Nichiporuk, A. Radin, O. Tkacheva, and J. Oberholtzer, "Lessons from Russia's Operations in Crimea and Eastern Ukraine," 2014, Accessed: Apr. 03, 2022. [Online]. Available: www.rand.org/giving/contribute
- [28] "Azerbaijan and Armenia: The Nagorno-Karabakh Conflict," 2021, Accessed: Apr. 03, 2022. [Online]. Available: <https://crsreports.congress.gov>
- [29] K. Lakkaraju, J. Reinhardt, J. Letchford, J. Whetzel, B. L. Goldblum, and A. W. Reddie, "Experimental Wargames to Address the Complexity-Scarcity Gap," May 2020. doi: 10.22360/SpringSim.2020.ANSS.010.

- [30] "Simulator Types - AcqNotes." <https://acqnotes.com/acqnote/tasks/simulator-types> (accessed Jun. 01, 2022).
- [31] "NSWC Crane spearheads nearly a decade of Electronic Warfare Live Virtual Constructive testing for Navy > Naval Sea Systems Command > News." <https://www.navsea.navy.mil/Media/News/Article/2312378/nswc-crane-spearheads-nearly-a-decade-of-electronic-warfare-live-virtual-constr/> (accessed Jun. 09, 2022).
- [32] S. Gallant and C. Gaughan, "SYSTEMS ENGINEERING FOR DISTRIBUTED LIVE, VIRTUAL, AND CONSTRUCTIVE (LVC) SIMULATION".
- [33] "UAS-NAS Live Virtual Constructive Distributed Environment (LVC) Software Design Description," 2015.
- [34] C. L. Haase, "AFIT Scholar Theses and Dissertations Student Graduate Works Tailoring the Statistical Experimental Design Process for LVC Experiments," 2011, Accessed: Jun. 09, 2022. [Online]. Available: <https://scholar.afit.edu/etd/1494>
- [35] E. H. Page, "Modeling and Simulation, Experimentation, and Wargaming-Assessing a Common Landscape".
- [36] R. Richbourg Robert Lutz, "Live Virtual Constructive Architecture Roadmap (LVCAR) Comparative Analysis of the Architectures," 2008.
- [37] C. Meirina, G. M. Levchuk, S. Ruan, K. R. Pattipati, and R. L. Popp, "Normative framework and computational models for simulating and assessing command and control processes," *Simulation Modelling Practice and Theory*, vol. 14, no. 4, pp. 454–479, May 2006, doi: 10.1016/j.simpat.2005.09.005.
- [38] P.-G. Taranti, R. Choren, C. Lucena, M. Luck, and P. McBurney, "INTEROPERATING WARGAME SIMULATIONS," in *XVIII Simpósio de Pesquisa Operacional & Logística da Marinha*, Aug. 2016, pp. 751–764. doi: 10.5151/marine-spolm2015-140939.
- [39] Rajarishi Sinha, Vei-Chung Liang, C. J. J Paredis, and P. K. Khosla Mem ASME, "Modeling and Simulation Methods for Design of Engineering Systems," *ASME*, vol. 1, pp. 84–91, 2001, doi: 10.1115/1.1344877.
- [40] Office of Secretary of Defense, "Defense Acquisition Guidebook," 2013, Accessed: Jun. 01, 2022. [Online]. Available: <https://dag.dau.mil>
- [41] Office of the Under Secretary of Defense for Acquisition and Sustainment, "DOD INSTRUCTION 5000.85 MAJOR CAPABILITY ACQUISITION," 2021. Accessed: Apr. 17, 2022. [Online]. Available: <https://www.esd.whs.mil/DD/>.

- [42] J-8, "MANUAL FOR THE OPERATION OF THE JOINT CAPABILITIES INTEGRATION AND DEVELOPMENT SYSTEM," 2018.
- [43] "USD(R&E) Top 10 Technology Focus Areas | AiDA." <https://aida.mitre.org/top-10-technology-areas/> (accessed Mar. 29, 2022).
- [44] "2018 National Defense Strategy of The United States of America," 2018. <https://dod.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf> (accessed Mar. 29, 2022).
- [45] "modernization-priorities – DoD Research & Engineering, OUSD(R&E)." <https://www.cto.mil/modernization-priorities/> (accessed Mar. 29, 2022).
- [46] "Autonomous Systems Artificial Intelligence and Safeguards. (Conference) | OSTI.GOV." <https://www.osti.gov/biblio/1561151> (accessed Jun. 01, 2022).
- [47] Smith and Chris HR, "CHARTER OF THE JOINT REQUIREMENTS OVERSIGHT COUNCIL AND IMPLEMENTATION OF THE JOINT CAPABILITIES INTEGRATION AND DEVELOPMENT SYSTEM UNCLASSIFIED UNCLASSIFIED," 2021.
- [48] "STATE OF THE DISCIPLINE 2019 Current and future state of Systems Engineering."
- [49] "nasa_systems_engineering_handbook_0".
- [50] M. D. Watson, B. L. Mesmer, P. A. Farrington, and G. C. Marshall, "Engineering Elegant Systems: Theory of Systems Engineering," 2020. [Online]. Available: <http://www.sti.nasa.gov>
- [51] "SYSTEMS ENGINEERING FUNDAMENTALS SUPPLEMENTARY TEXT PREPARED BY THE DEFENSE ACQUISITION UNIVERSITY PRESS FORT BELVOIR, VIRGINIA 22060-5565," 2001.
- [52] "SE Brainbook Home." <https://www.dau.edu/tools/se-brainbook> (accessed Jun. 12, 2022).
- [53] D. Seal, D. Farr, and J. Hatakeyama, "The System Engineering Vee-is it Still Relevant in the Digital Age?," 2018, Accessed: Jun. 12, 2022. [Online]. Available: <https://ops.fhwa.dot.gov/publications/seitsguide/section3.htm>
- [54] W. K. Vaneman, "The system of systems engineering and integration «vee» model," *10th Annual International Systems Conference, SysCon 2016 - Proceedings*, Jun. 2016, doi: 10.1109/SYSCON.2016.7490599.
- [55] "Space Missile Command Systems Engineering Primer & Handbook -Concepts, Processes, and Techniques," 2005.
- [56] "SYSTEMS ENGINEERING FUNDAMENTALS SUPPLEMENTARY TEXT PREPARED BY THE DEFENSE ACQUISITION UNIVERSITY PRESS FORT BELVOIR, VIRGINIA 22060-5565," 2001.

- [57] N. Mohammed, A. Munassar, and A. Govardhan, "A Comparison Between Five Models Of Software Engineering," *IJCSI International Journal of Computer Science Issues*, vol. 7, no. 5, 2010, [Online]. Available: www.IJCSI.org
- [58] D. F. Beam, "SYSTEMS ENGINEERING AND INTEGRATION AS A FOUNDATION FOR MISSION ENGINEERING," NPS, MONTEREY, 2015.
- [59] P. Beery and E. Paulo, "Application of Model-Based Systems Engineering Concepts to Support Mission Engineering," *Systems*, vol. 7, no. 3, p. 44, 2019, doi: 10.3390/systems7030044.
- [60] E. Paulo and P. Beery, "Application of Model-Based Systems Engineering Concepts to Support Mission Engineering," *Calhoun: The NPS Institutional Archive*, 2019, doi: 10.3390/systems7030044.
- [61] M. D. Griffin, "HOW DO WE FIX SYSTEM ENGINEERING?"
- [62] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," 2007.
- [63] C. Gedo, "Model Based Systems Engineering and Systems Modeling Language DoDAF Plenary," 2012.
- [64] M. Philomena, P. " Zimmerman, and N. M&s, "MBSE in the Department of Defense," 2015.
- [65] Lenny Delligatti, *SysML Distilled*. Crawfordsville, Indiana: Pearson Education Inc, 2014.
- [66] P. White, "Welcome to SysML, the Language of MBSE," 2019.
- [67] "SysML Specifications - Current Version: OMG SysML 1.6," 2019. Accessed: Jun. 15, 2022. [Online]. Available: <https://sysml.org/sysml-specs/>
- [68] Omg, "An OMG Systems Modeling Language TM Publication OMG Systems Modeling Language (OMG SysML™)," 2019. [Online]. Available: <https://www.omg.org/spec/SysML/20181001/sysmldi.xmi>
- [69] D. C. Sales and L. B. Becker, "Systematic literature review of system engineering design methods," in *Brazilian Symposium on Computing System Engineering, SBESC*, Jul. 2018, vol. 2018-November, pp. 213–218. doi: 10.1109/SBESC.2018.00040.
- [70] INCOSE, "New Standards for System of Systems Engineering (SoS)." [Online]. Available: <https://www.iso.org/store.html>
- [71] S. Wolny, A. Mazak, C. Carpella, V. Geist, and M. Wimmer, "Thirteen years of SysML: a systematic mapping study," *Software and Systems Modeling*, vol. 19, no. 1, pp. 111–169, Jan. 2020, doi: 10.1007/s10270-019-00735-y.

- [72] S. et al Friedenthal, *A Practical Guide to SysML : The Systems Modeling Language*, 3rd ed. Elsevier Science & Technology, 2014. Accessed: Jun. 15, 2022. [Online]. Available: <https://ebookcentral.proquest.com/lib/cranfield/reader.action?docID=1823236>
- [73] "Air Force Doctrine Publication (AFDP) 3-05 Special Operations," 2020.
- [74] Z. Scott and D. Long, "One Model, Many Interests, Many Views," 2017. Accessed: Jun. 26, 2022. [Online]. Available: https://vitechcorp-webdownload.s3.amazonaws.com/white_papers/OneModel.pdf
- [75] L. Delligatti, "Elements of definition."
- [76] P. Pearce and S. Friedenthal, "A Practical Approach For Modelling Submarine Subsystem Architecture In SysML," 2013.
- [77] D. A. Long and Zane. Scott, *A primer for model-based systems engineering*. Vitech Co, 2011.
- [78] P. T. Beery, "A model-based systems engineering methodology for employing architecture in system analysis: developing simulation models using systems modeling language products to link architecture and analysis," 2016. [Online]. Available: <http://hdl.handle.net/10945/49363>
- [79] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," *Jet Propulsion*, 2008.
- [80] Michael J. Vinarcik, "Hypermodeling: A Profile for Teaching SysML Modeling American Society for Engineering Education." Accessed: Jan. 28, 2021. [Online]. Available: <https://www.asee.org/public/conferences/140/papers/26050/view>
- [81] B. H. Rao, K. Padmaja, and P. Gurulingam, "A Brief View of Model Based Systems Engineering Methodologies," *International Journal of Engineering Trends and Technology (IJETT)*, Accessed: Jun. 23, 2022. [Online]. Available: <http://www.ijettjournal.org>
- [82] J. Moschler, "NAVAIR Systems Engineering Transformation".
- [83] "AFSIM: The Air Force Research Laboratory's Approach to Making M&S Ubiquitous in the Weapon System Concept Development Process - CSIAC." <https://csiac.org/articles/afsim-the-air-force-research-laboratorys-approach-to-making-ms-ubiquitous-in-the-weapon-system-concept-development-process/> (accessed Jun. 23, 2022).
- [84] S. Friedenthal, A. Meilich, and H. Lykins, "Adapting UML for an object oriented systems engineering method (OOSEM) MBSE and SysML View project conceptual modeling View project Adapting UML for an Object Oriented Systems Engineering Method (OOSEM)," 2000, doi: 10.1002/j.2334-5837.2000.tb00416.x.
- [85] P. Pearce and M. Hause, "ISO-15288, OOSEM and Model-Based Submarine Design".

- [86] S. Friedenthal, L. Izumi, and A. Meilich, "Object-oriented systems engineering method (OOSEM) applied to joint force projection (JFP), a lockheed martin integrating concept (LMIC)," *17th Annual International Symposium of the International Council on Systems Engineering, INCOSE 2007 - Systems Engineering: Key to Intelligent Enterprises*, vol. 1, pp. 176–196, 2007, doi: 10.1002/J.2334-5837.2007.TB02961.X.
- [87] "Object-Oriented SE Method." <https://www.incose.org/incose-member-resources/working-groups/transformational/object-oriented-se-method> (accessed Jun. 23, 2022).
- [88] A. E. Trujillo, O. L. de Weck, and A. M. Madni, "An mbse approach supporting technical inheritance and design reuse decisions," 2020. doi: 10.2514/6.2020-4167.
- [89] H.-P. Hofman, "Deskbook Release 4.1," 2013.
- [90] S. R. Childers and J. E. Long, "A CONCURRENT METHODOLOGY FOR THE SYSTEM ENGINEERING DESIGN PROCESS," *INCOSE International Symposium*, vol. 4, no. 1, pp. 226–231, Aug. 1994, doi: 10.1002/J.2334-5837.1994.TB01709.X.
- [91] S. R. Childers and J. E. Long, "A CONCURRENT METHODOLOGY FOR THE SYSTEM ENGINEERING DESIGN PROCESS," *INCOSE International Symposium*, vol. 4, no. 1, pp. 226–231, Aug. 1994, doi: 10.1002/j.2334-5837.1994.tb01709.x.
- [92] "Systems Thinking: Schemas, Metamodels, and MBSE – Community.Vitechcorp.com." <https://community.vitechcorp.com/systems-thinking-schemas-metamodels-and-mbse/> (accessed Jun. 26, 2022).
- [93] P. Gaignic, T. Vosgien, M. Jankovic, V. Tuloup, J. Berquet, and N. Troussier, "Complex system simulation: Proposition of a MBSE framework for design-analysis integration," 2013. doi: 10.1016/j.procs.2013.01.007.
- [94] S. Mittal, U. Durak, and T. Ören, *Simulation Foundations, Methods and Applications Guide to Simulation-Based Disciplines Advancing Our Computational Future*. Springer, 2017. Accessed: Mar. 02, 2021. [Online]. Available: <http://www.springer.com/series/10128>
- [95] "vitech mbse domains," *OMG MBSE Wiki*, Oct. 30, 2010. https://www.omgwiki.org/MBSE/lib/exe/detail.php?id=mbse%3Avitechmbse&media=mbse:vitech_mbse_domains.jpg (accessed Jun. 29, 2022).
- [96] S. H. Dam and J. D. Willis, "Defensible C4ISR Architectures Require the Application of Good System Engineering Practice," 2002, Accessed: Jun. 29, 2022. [Online]. Available: www.spec-1.com
- [97] J. Holt and S. Perry, *SysML for Systems Engineering*, 3rd ed. London: The Institution of Engineering and Technology, 2018. Accessed: Jun. 15, 2022. [Online]. Available: <https://app.knovel.com/web/view/khtml/show.v/rcid:kpSMLSEA0L/cid:kt011W2KU1/viewerTy>

pe:khtml/root_slug:sysml-systems-engineering/url_slug:introduction-sysml-systems?&b-toc-cid=kpSMLSEA0L&b-toc-root-slug=sysml-systems-engineering&b-toc-title=SysML%20for%20Systems%20Engineering%20-%20A%20Model-Based%20Approach%20%283rd%20Edition%29&b-toc-url-slug=part-i-introduction&kpromoter=marc&page=12&view=collapsed&zoom=1

- [98] D. A. Wagner, M. B. Bennett, R. Karban, N. Rouquette, S. Jenkins, and M. Ingham, "An ontology for state analysis: Formalizing the mapping to SysML," *IEEE Aerospace Conference Proceedings*, 2012, doi: 10.1109/AERO.2012.6187335.
- [99] M. Bennett, D. Dvorak, J. Hutcherson, M. Ingham, R. Rasmussen, and D. Wagner, "An architectural pattern for goal-based control," *IEEE Aerospace Conference Proceedings*, 2008, doi: 10.1109/AERO.2008.4526594.
- [100] W. Eric Marsh and W. Eric, "An Initial Methodology For The Definition And Implementation Of An Initial Methodology For The Definition And Implementation Of Unmanned Aerial Vehicle Agent Behaviors Unmanned Aerial Vehicle Agent Behaviors Repository Citation Repository Citation", Accessed: Jun. 29, 2022. [Online]. Available: https://corescholar.libraries.wright.edu/etd_all://corescholar.libraries.wright.edu/etd_all/85
- [101] T. R. Hernandez, "AFIT Scholar AFIT Scholar Theses and Dissertations Student Graduate Works Modeling Situation Awareness in a UAV Scenario Using SysML Modeling Situation Awareness in a UAV Scenario Using SysML", Accessed: Jun. 29, 2022. [Online]. Available: <https://scholar.afit.edu/etd/5039>
- [102] A. J. Spriggs and S. E. Hooper, "Fusion: A Framework for Human Interaction with Flexible-Adaptive Automation Across Multiple Unmanned Systems," pp. 464–469, 2015, Accessed: Jun. 29, 2022. [Online]. Available: https://corescholar.libraries.wright.edu/isap_2015https://corescholar.libraries.wright.edu/isap_2015/28
- [103] P. M. Shames, M. A. Sarrel, and S. A. Friedenthal, "Modeling systems-of-systems interfaces with SysML," 2016, doi: 10.2514/6.2016-2500.
- [104] M. Vinarcik, "The NeMO Orbiter: A Demonstration Hypermodel," 2018. Accessed: Jun. 29, 2022. [Online]. Available: https://www.academia.edu/37211953/The_NeMO_Orbiter_A_Demonstration_Hypermodel
- [105] R. A. Noguchi, "Recommended Best Practices based on MBSE Pilot Projects," Jun. 2019.
- [106] "(PDF) A Mars Octet: Lessons Learned from Federating Eight Student Models in a SysML Class | Michael Vinarcik - Academia.edu." https://www.academia.edu/67778096/A_Mars_Octet_Lessons_Learned_from_Federating_Eight_Student_Models_in_a_SysML_Class (accessed Jun. 29, 2022).

- [107] "SAIC | Digital Engineering Validation Tool." <https://www.saic.com/digital-engineering-validation-tool> (accessed Jun. 29, 2022).
- [108] M. Hause, G. Bleakley, and A. Morkevicius, "Technology Update on the Unified Architecture Framework (UAF)," *INCOSE International Symposium*, vol. 26, no. 1, pp. 1145–1160, Jul. 2016, doi: 10.1002/J.2334-5837.2016.00217.X.
- [109] W. Okon, "DoD Architectures and Systems Engineering Integration," 2012.
- [110] M. C. Hause, "Elemental Links Transitioning UPDM to the Unified Architecture Framework," 2016.
- [111] Omg, "Enterprise Architecture Guide for UAF (Informative) Appendix C," 2021. [Online]. Available: <https://www.omg.org/spec/UAF/20211201/UAF.xmi>
- [112] "Unified Architecture Framework (UAF) Traceability (Informative) Appendix A," 2021. [Online]. Available: <https://www.omg.org/spec/UAF/20211201/UAF.xmi>
- [113] J. M. House and U. S. Army, "Toward Combined Arms Warfare:-A Survey of 20th~Century Tactics, Doctrine, and Organization Toward Combined Arms Warfare: A Survey of 20th-Century Tactics, Doctrine, and Organization," 1984.
- [114] Headquarters Department of the Army, "ADP 3-0 Operations," 2017. Accessed: Aug. 21, 2022. [Online]. Available: <https://armypubs.army.mil/>
- [115] "RT-171: Mission Engineering Competencies," 2018.
- [116] S. C. Spangelo *et al.*, "Model based systems engineering (MBSE) applied to Radio Aurora Explorer (RAX) CubeSat mission operational scenarios," Jan. 2013. doi: 10.1109/AERO.2013.6496894.
- [117] "Exponential Growth of System Complexity." <https://savi.avsi.aero/about-savi/savi-motivation/exponential-system-complexity/> (accessed Aug. 18, 2022).
- [118] Naval Surface Warfare Center Crane, "Interoperability of Digital Mission Engineering (DME) Software Suites" https://www.navsea.navy.mil/Portals/103/Documents/Exhibits/SAS2021/SAS2021-Interoperability_of_DM_NSWC_Crane.pdf, 2021
- [119] DASN RDTE, "United States Navy & Marine Corps Digital systems Engineering Transformation Strategy", <https://nps.edu/documents/112507827/0/2020+Dist+A+DON+Digital+Sys+Eng+Transformation+Strategy+2+Jun+2020.pdf>, 2020
- [120] Miller, J. "A State-Based Model for Validating Autonomous Munition Behaviors", <https://scholar.afit.edu/cgi/viewcontent.cgi?article=5954&context=etd>, 2021

6. Appendix B: No Magic Implementation of UAF Viewpoints in Cameo – Reference Guide

