STATE-BASED MODEL FOR VALIDATING AUTONOMOUS MUNITION BEHAVIORS

THESIS

Dalton J. Miller, Captain, USAF

AFIT-ENV-MS-21-M-245

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-21-M-245

STATE-BASED MODEL FOR VALIDATING AUTONOMOUS MUNITION BEHAVIORS

THESIS

Presented to the Faculty

Department of Systems Engineering and Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Systems Engineering

Dalton J. Miller, B.S.S.E

Captain, USAF

March 2021

AFIT-ENV-MS-21-M-245

STATE-BASED MODEL FOR VALIDATING AUTONOMOUS MUNITION BEHAVIORS

THESIS

Dalton J. Miller, B.S.S.E
Captain, USAF

Committee Membership:

David R. Jacques, Ph.D
Chair

John Colombi, Ph.D
Member

Jeremy Gray, M.S.
Member

Abstract

Autonomous munitions provide an opportunity for the Department of Defense (DoD) to remove the human in-the-loop from life threatening situations and extend the capabilities of operators in combat situations. However, it also presents a challenge for the DoD due to their inherent need for a high level of trust in the effectiveness and accuracy of the system. With the advent of model-based standards in autonomy and munitions, there is a need to implement these techniques toward an effective modeling and simulation (MS) capability. By leveraging modern MS tools such as Cameo Systems Modeler and the DoD's Advanced Framework for Simulation, Integration, and Modeling (AFSIM), this thesis proposes a framework for simulating complex autonomy architectures within high fidelity simulation environments. Building on this proposed framework, a state-based behavioral model was developed that captures a collaborative autonomous munition within the context of a Suppression of Enemy Air Defenses (SEAD) mission. Using model-based systems engineering best practices and integrated tool capabilities, the system model shows the ability for Cameo to host interactive, executable state machines and demonstrate autonomous decision making based on internal system and environmental cues in order to generate mission effectiveness performance measures.

Acknowledgements

First and foremost, I would like to thank my wife for all her steadfast love and support during this entire AFIT experience. I'm so blessed to have not only a wife, but a fellow student and officer, that encourages, commiserates, and helps no matter what the challenge.

I would also like to thank my advisor, Dr. David Jacques, who guided me during my research. Your instruction and direction was vital to the completion of this research.

Finally I wish to express my gratitude to Mr. Gregory Haun, who took the time to lend his vast expertise towards the modeling shown in this thesis.

Dalton J. Miller

Table of Contents

Page

List of Figures

List of Tables

List of Acronyms

Acronym		Page
ANT	Autonomy and Navigation Technology	1
AFIT	Air Force Institute of Technology	1
AFRL/RW	Air Force Research Lab, Munitions Directorate	1
MBSE	Model Based Systems Engineering	1
DoD	Department of Defense	1
AFRL/RW	Air Force Research Lab Munitions Directorate	2
LCCM	Low-Cost Cruise Missile	2
V&V	verification and validation	2
T&E	test and evaluation	2
WDE	Weapon Digital Enterprise	3
ASRA	Autonomous System Reference Architecture	4
FOR	Field of Regard	10
FTOs	False Target Objects	10
HAMR	Hybrid Architecture for Multiple Robots	12
OOP	Object Oriented Programming	13
INCOSE	International Council on Systems Engineering	13
SEAD	suppression of enemy air defenses	18
DoE	Design of Experiments	20
SAMs	surface-to-air missiles	41

STATE-BASED MODEL FOR VALIDATING AUTONOMOUS MUNITION BEHAVIORS

I. Introduction

The Air Force acquisitions community is trending toward integrating Digital Engineering within every facet of the systems engineering process. This, coupled with the increasing demand for autonomous systems, drives the need for a standard in digitally modeling and validating autonomy. The Autonomy and Navigation Technology (ANT) Center at the Air Force Institute of Technology (AFIT) has conducted research towards this end, and this thesis stands as a continuation of that effort. It also supports the work done by the WeaponONE office in the Air Force Research Lab, Munitions Directorate (AFRL/RW), whose current task involves building a reference architecture for advanced munitions, including those with autonomous behaviors. To serve these efforts this thesis will demonstrate the ability of Model Based Systems Engineering (MBSE) to capture and validate autonomous behaviors such as those found in autonomous munitions.

1.1 Background and Motivation

The Department of Defense (DoD) is increasingly turning to autonomy to imbue greater efficiency into its processes and missions [4]. One Air Force process that poses both an opportunity and challenge for augmentation by autonomy is the "kill chain". This six-stage process, also known as F2T2EA, includes the steps—Find, Fix, Track, Target, Engage, and Assess—all of which are currently executed by human operators using a diverse range of multi-domain weapon systems. Since this process is primarily

implemented in combat situations, it serves the safety of personnel to reduce the cognitive workload of the operator through the use of autonomy.

The Air Force seeks to accomplish this in part through autonomous munitions. These offer a true "fire and forget" capability where all coordination, execution, and assessment is completed by the munition post-launch. There are many variations on the concept of autonomous munitions, but the focus of this thesis will be on medium-range, low-cost missiles that can be deployed individually or in small swarms with inter-vehicle communication. These are often referred to as collaborative munitions. As a concept, these munitions have been in development since the early 2000s, but most recently the Air Force Research Lab Munitions Directorate (AFRL/RW) has been developing the Low-Cost Cruise Missile (LCCM) to fill this role. In order to not only approve, but trust the use of these munitions in an operational environment, a robust development process with significant simulation and testing is required.

One of the major hurdles, especially with lethal autonomous systems, is trust—both by the operator and the organizations utilizing them. A major factor in trusting autonomous systems is predictability, both in effectiveness and in reaction to a variety of external stimuli. [5] Operational environments are generally not a good time to find emergent behaviors in munitions. However, with advanced physics simulators, it is possible to generate digital "operational environments" in which to validate autonomous systems as well as begin to identify unexpected behaviors and capability gaps.

A software and systems engineering truism is "test early, test often", and is all but required in order to establish trust and approval for autonomous and collaborative munitions. Within the DoD these weapons must "go through rigorous hardware and software verification and validation (V&V) and realistic system developmental and operational test and evaluation (T&E)" in order to ensure that systems (among other

requirements):

1. "Function as anticipated in realistic operational environments against adaptive adversaries."

2. "Are sufficiently robust to minimize failures that could lead to unintended engagements or loss of control of the system to unauthorized parties." [6]

To support these requirements, it certainly benefits programs to incorporate validation activities as early in the design process as possible. This is now possible more than ever through digital engineering, which the DoD has made a priority to utilize in its systems engineering practices moving forward [7]. Digital engineering is defined in the 2018 DoD Digital Engineering Strategy as an "integrated digital approach that uses authoritative sources of system data and models as a continuum across disciplines to support lifecycle activities from concept through disposal." [7] To comply with this effort to apply digital engineering across the life cycle of most programs, AFRL/RW has begun the Weapon Digital Enterprise (WDE) initiative to apply modern systems engineering techniques to the development and testing of the LCCM as well as other munition programs.

The WDE program is attempting to use MBSE to apply more modularity and reusability to the LCCM, hypersonic weapons, and other munition programs. This is driving the ongoing development of a government reference architecture using Cameo Systems Modeler. While this effort helps the program manage the technical baseline, it also presents opportunities for further system analysis and validation. Aside from the physical architecture, programs must also be concerned with the system software, especially for autonomous systems programs. In MBSE, there are models that describe and define a system's behavior to better understand the context and physical system. Using modeling software such as Cameo, these models can be used to define

behavioral logic and produce notional system responses. For autonomous systems, these behavioral models can be representative of the intended system software. This presents an opportunity to not only manage the software technical baseline, but the potential to also evaluate these models in order to validate the software architecture before generating any code.

The AFIT ANT Center has made strides in this area by developing an Autonomous System Reference Architecture (ASRA), that abstracts the behaviors found in autonomous systems and creates a framework for logical flows. The ASRA has been recently applied to the multi-agent Wide Area Search problem and shows promise for application to autonomous, collaborative munitions. The architecture that has been developed has been tested using both simple point-mass simulations and software-in-the-loop, but still requires validation using a full physics-based simulation.

1.2 Problem Statement and Scope

Within this wide problem space, there is a notable gap in the executable modeling of autonomous behaviors. In order to achieve the applications described above for validation of autonomy models, there must be a standard for effectively defining appropriate SysML models that capture the operation and logic of an autonomous system and can be simulated within the model. This is not only necessary for validation of the model, but also for the system software. The WDE, while addressing the physical modeling of a munition, has expressed its need to convert the hard-coded logic of current munition behaviors into more generalized state machine models. To this end, a standardized framework must be defined and proposed for future modeling efforts.

Using strides made in the development of the ASRA, this thesis will attempt to define and model behaviors for the LCCM in a simple Suppression of Enemy Air

Defenses (SEAD) mission using Cameo in order to support the development of an object-oriented programming approach to the LCCM software. It will attempt to show that SysML behavior models can be used as a basis for virtual certification through advanced simulation and act as a framework for modeling and simulation of autonomous software. This research will help the Weapon Digital Enterprise program to understand how autonomous behaviors can be modeled for reuse and validation. This allows them to efficiently apply digital engineering and modularity across the life cycle of a weapon, including the autonomy algorithms found in smart munitions. Much of this work will be based on prior research focused on the Wide Area Search (WAS) munition problem and will apply findings and lessons learned to the LCCM case study.

1.2.1 Research Objectives and Questions

1.2.1.1 Research Objectives

1. Determine the effectiveness of the SysML state machine diagrams to act as an abstract model for the autonomous behaviors required for cooperative munitions. Demonstrate this using a surrogate for the LCCM munition as a case study.

2. Investigate the interoperability of SysML models in Cameo with simulation capabilities and develop a surrogate model appropriate for simulation of an autonomous and cooperative munition.

3. Analyze the effectiveness of SysML and Cameo as an integrated mission modeling tool and develop an executable model of a typical SEAD scenario.

4. Research the feasibility of high fidelity physics simulations, driven by an executable state machine model, to validate common autonomous behaviors in

typical DoD missions.

1.2.1.2 Research Questions

1. How and to what level of fidelity should autonomous behaviors be modeled in a reference architecture?

2. How well does the SysML state machine diagram capture autonomous behaviors?

3. How can Cameo Systems Modeler use executable diagrams to interface with external simulation engines?

4. Which cooperative behaviors must be identified and modeled to capture the LCCM autonomy software within the current model?

5. How can the model be effectively simulated to validate mission effectiveness?

6. What are valid measures to determine mission effectiveness for the concept munition?

1.3 Methodology

From an implementation standpoint, this research seeks to devise a technique to create a real-time simulation of an abstracted behavioral model. This method is comprised of three major components: an MBSE modeling tool, an advanced physics simulator, and a software coding suite. For this case study, the SEAD mission will be used to validate the mission effectiveness of behaviors and a notional LCCM-type munition. A state machine diagram will be built using Cameo to capture the desired behaviors of an autonomous munition in a SEAD mission, but abstracted of detailed hardware definition. This state machine will be constructed to execute

MATLAB-based scripts that can execute a simulation in AFSIM. The state machine will also (through scripts) be capable of receiving events and outputs from AFSIM. Data collected from the resulting simulations will be used to validate the behavior models and consequently the software algorithms they will produce. Further details on methodology will be explored in chapter three of this thesis.

1.4 Assumptions and Limitations

The broad problem addressed in this thesis is being tackled from numerous directions. This includes different aspects of MBSE, means of validation, and varying domains. For this thesis, since the scope of this thesis is limited to a particular case study, the results may be limited in broad applicability and may require integration with other digital SE techniques and processes. Further assumptions and limitations are listed below:

1. The executable model will be focused on behavior, so some physical aspects will be abstracted out of the system.

2. Cooperation between munitions will be modeled with only two weapons interacting.

3. Munitions will be modeled as a simple fixed-wing vehicle with flying capabilities similar to the Low-Cost Cruise Missile.

4. Sensor and warhead performance will be simulated using previously researched and established probabilistic models (more details in Chapter 2).

5. Physical and performance characteristics will be based on prior built AFSIM entities and defined primarily outside of the architecture.

6. Behaviors will be as modeled as close to previous research as is feasible for the scenario.

1.5 Preview

This chapter outlined the background, problem statement, research goals, and a brief look at the methodology and assumptions. Chapter two will review the general concepts behind the research, including: WAS and collaborative munitions, autonomous systems architectures, validating mission effectiveness for autonomous munitions, as well as behavior modeling and simulation. Chapter three will discuss the methodology employed to develop an executable behavior model for the LCCM with SEAD mission considerations, and how a functioning simulation was achieved. The results of this simulation will be presented in chapter four and discuss the suitability of the model and simulation framework. Finally, chapter five will deliver final conclusions, lessons learned, future work, and recommendations moving forward in this line of research.

II. Background and Literature Review

The following section will cover the core concepts and fundamentals necessary to guide the research completed in this thesis. It will introduce relevant details concerning MBSE, and its use in modeling autonomous systems. Previous work in the wide area search scenario and collaborative munitions will be explored as well as an introduction to advanced physics simulation engines viable for testing mission effectiveness.

2.1 Previous Work

2.1.1 Wide Area Search Munitions

The origin of this work can be traced to research done by Dr. David Jacques in the field of Wide Area Search munitions—specifically effectiveness analysis. The Wide Area Search (WAS) problem is primarily an implementation of munitions (or autonomous vehicles) that can perform search, detect, classify, and attack functions autonomously. There are several of these systems with similar missions that exist across the Air Force. Prior work on the subject has produced formulas that capture mission success based on probabilistic models of sensors, ordnance, and external events. These probabilities are associated with the sensor and warhead performance as well as the environmental factors that confound them. At the most general level Jacques described the probability of mission success by the following equation [8]:

$$P_M S = P_K * P_{TR} * P_{LOS} * P_E$$

where:

1. P_K is the probability the target is successfully killed provided an accurate target

report;

2. P_{TR} is the probability of an accurate target report given there is clear Line of Sight to the target;

3. P_{LOS} is the probability of a clear Line of Sight to the target given it is in the field of regard;

4. P_E is the probability that the target appears in the Field of Regard (FOR).

In his research, Jacques explored the factors informing each of these parameters. For instance, the probability for an accurate report is most significantly influenced by the capability of the system's sensors and the number of objects in the search area similar to the target, or False Target Objects (FTOs). Their performance is represented in terms of a confusion matrix similar to the one below, which shows the probabilities of accurate target detection and recognition [8]:

Table 1: Binary Confusion Matrix

	Target	FTO
Target Declaration	P_{TR}	$1 - P_{TR}$
No Declaration	$1 - P_{TR}$	P_{TR}

The concept of a confusion matrix can be extended to a variety of other outcomes such as the classification of the target. Extended out to a multi-target scenario, the probability of the target entering the FOR is a function of the area and number of targets and FTOs. Similarly, the multi-munition scenario further changes these values as well as the P_K, due to the possibility of cooperative attack. A series of research efforts have developed the WAS scenario and performed validation of these models.

2.1.2 Levels of Cooperation

Continuing work in this area further developed the levels of cooperation to determine the impact on mission effectiveness. Three levels of cooperation were introduced

in Dunkel's research—no cooperation, cooperative attack only, and cooperative classification and attack. Additionally, his work developed a value function to determine the utility in engaging previously identified targets. This function was built through success probabilities associated with each alternative course of action. By analyzing the levels of cooperation and optimizing the mission effectiveness formulas, the levels yielded varying levels of effectiveness with respect to the number of targets attacked and the false target attack rate. Dunkel's work concluded that cooperative classification and attack delivers the most ideal, if conservative, outcomes[9].

2.1.3 ASRA Implementation

Most recent research in this field includes the work done by Lts David King and Katie Cheney in their thesis titled "Development, Test, and Evaluation of Autonomous Unmanned Aerial Systems in a Simulated Wide Area Search Scenario". This effort was primarily concerned with a WAS implementation of the ASRA through a simple simulation. Although only using a UAS rather than a munition, the scenario they used closely resembled that of previous work. They again used different levels of cooperation to compare the performance of a WAS agent in detecting and confirming targets. These levels mirrored those from Dunkel but added a single agent case. Also, because the system in question was not a munition, there was no form of cooperative attack. The levels of cooperation are below: [10]:

1. Single Agent Case - One agent searches and confirms the entire area;
2. Basic Cooperation Case - Two agents search separate halves of the search area;
3. Extreme Cooperation Case - Two agents search the area and immediately respond to confirmation requests when a target is detected by the other;

4. Moderated Cooperation Case – The decision to respond to a confirmation request is determined based on a value function.

The ASRA architecture is an attempt to apply the MBSE concept of reference architectures to autonomous agents. One instantiation currently utilized by the ASRA and explored by King and Cheney is the Hybrid Architecture for Multiple Robots (HAMR). This is composed of 4 layers: the Controller, Sequencer, Deliberator, and Coordinator. This represents a blended approach to autonomy modeling discussed in the next section. In King/Cheney's thesis, this architecture was implemented in software using Python and executed through a point-mass simulation and software-in-the-loop of the WAS scenario. This allowed for a design of experiments analysis to inform the probabilities necessary for mission effectiveness analysis[11].

2.2 Autonomous Behavior Modeling

Autonomous systems behaviors can be defined and executed using a variety of methods. Among these are programming-based, learning-based, and model-based approaches. Programming-based behaviors are specifically defined responses designed and encoded by the programmer, whereas learning-based behaviors are developed from the system's experience. Model-based behaviors select actions based on an analytical model of the problem space.[12] Each of these behavior methods can be modeled descriptively in order to better define autonomous behavior software architectures.

The HAMR architecture, as modeled in the ASRA, describes a layered architecture that blends the behavioral methodologies described above—a combination that is necessary for robust autonomous systems. However, the issue facing autonomous systems engineers is the complexity resulting from this architecture when defining the controlling software. For this reason, the HAMR architecture, taking cues from

Object Oriented Programming (OOP), provides a means of decomposing complex autonomous behaviors into manageable and reusable models. This is accomplished through the delegation of the behaviors to the appropriate layers in the architecture. This approach allows the use of OOP-based modeling languages such as a UML and SysML to more easily describe and simulate the architecture such as the work shown in Cheney and King's research.[10]

2.3 Behavior Modeling Standards

DoD and industry standards have shifted towards widespread digital engineering of systems at every stage of the life cycle. As a result, MBSE has moved to the forefront of system development within DoD acquisitions. In the USAF, offices have been created to manage the transition from traditional document-based systems engineering to one that is primarily model-based. Most MBSE standards involve the use of SysML, which is based on the UML typically used in software engineering. In MBSE the state machine diagram is a powerful tool not only to describe the system behaviors but to also validate them using executable models.

When modeling system behaviors it's important to clarify the semantical definition of states. There are a number of disparate definitions for the clarification between standard terms such as phases, modes, and states and how they relate to system behaviors. In his article, Wasson surveys the range of academically accepted definitions and rigorously determines a standard for describing system behaviors—a standard which has been published by the International Council on Systems Engineering (INCOSE). This research is most interested in his conceptualization of states as well as how they contrast with system modes. A state is defined by Wasson as "an attribute used to characterize the current logistical status or performance-based condition of a system... or system components at the element, subsystem, etc. levels

of abstraction." [1] This definition shows different aspects of states for which Wasson delineates them into four types: system, operational, dynamic, and physical. System states are concerned with the logistical status of the whole system and for the purposes of this thesis the least relevant for a behavioral model. Operational states are most commonly in terms of the system's level of operational capability—on/off, operational/degraded/failed, etc.—and capture the status of any level abstraction within the system. Dynamic states are the most relevant to behavioral modeling as they describe relative action or change within the system, usually represented with action verbs. Finally, physical states refer to the physical configuration of the system, whether the actual presence of certain system hardware or the physical status of those components.

Wasson also contrasts states with the often confused concept of "modes", which he defines as primarily "a user-selectable option" that allows certain system capabilities in order "to achieve a specified set of mission objectives, outcomes, and levels of performance." [1] This essentially considers modes to be the language of the command and control of the system. The simple example in the article relates this to driving "modes" in a car (Park, Reverse, Neutral, Drive, etc.). Applying this definition to autonomous systems, however, requires further consideration. As the general goal for these systems is to remove the human from the loop, modes cannot be just considered user-selectable. The commanding software agent in an autonomous agent should be able to shift through modes without user input, although this does not preclude the option for users to pre-select or override these modes.

Using this nomenclature, a system can be represented at any given moment as a cross-section of all of its states and modes. Based on this, the system is able to perform certain actions or "atomic behaviors" that when combined with others can create a complex activity. For instance, a series of physical states and operational

states can describe the status of a component or subsystem; then, based on the mode of the system only certain dynamic states will be available for entry. These states, in turn, influence the activities performed by the system components and subsystems. This concept is shown visually below:

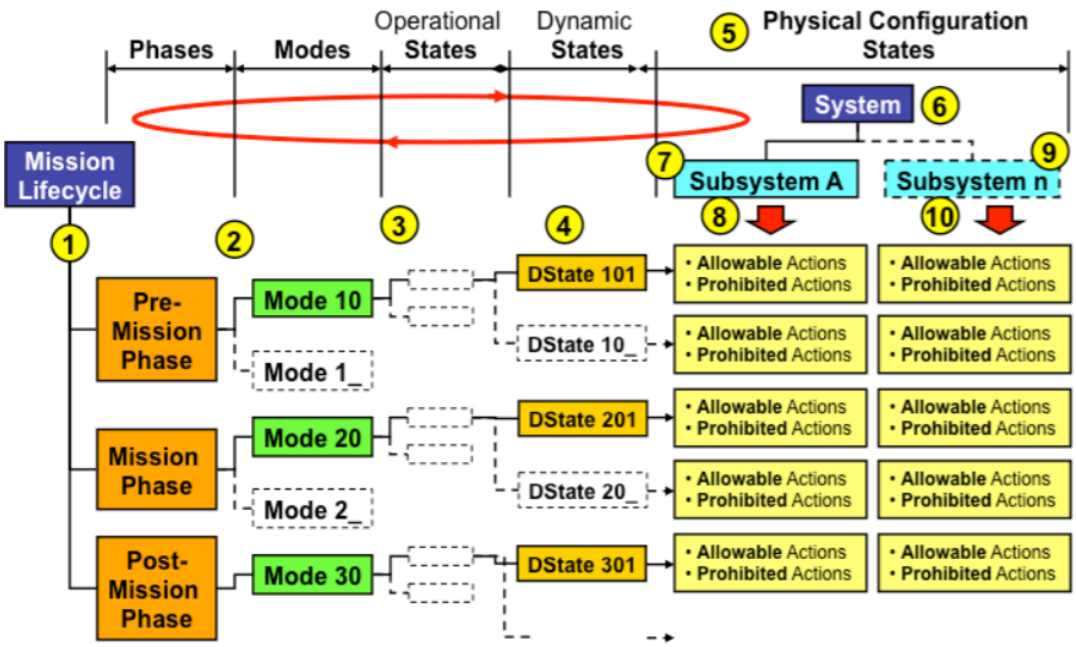

Figure 1: Wasson's Decomposition of States, Modes, and Phases [1]

2.4 Simulation/Testing Autonomy

To further increase the level of fidelity behind mission effectiveness analysis requires accurate system simulation to replace probabilistic models. There are a number of detailed physics simulation engines that are employed by the Air Force that are viable options. Most notable for this research is AFSIM, an Air Force sponsored and owned simulation environment that specializes in (but is not limited to) Integrated Air Defenses and Air-to-Ground scenario simulations. AFSIM relies on C++ based scripting to define a variety of platforms and missions. While it is meant as a standalone simulation environment it offers alternative methods of control outside of its

development environment.[2]

AFSIM is an object-oriented simulation engine, which pairs well with the modeling concepts described above. The primary class-level entity within AFSIM is the Platform. Platforms have unique "instantiations" (vehicles, buildings, ordnance, etc.) based on their attributes and parts. Parts make up the defining physical and behavioral characteristics of the platform and include primarily, a processor, mover, weapon, communications, and sensors.[2] This concept is captured in the figure below:

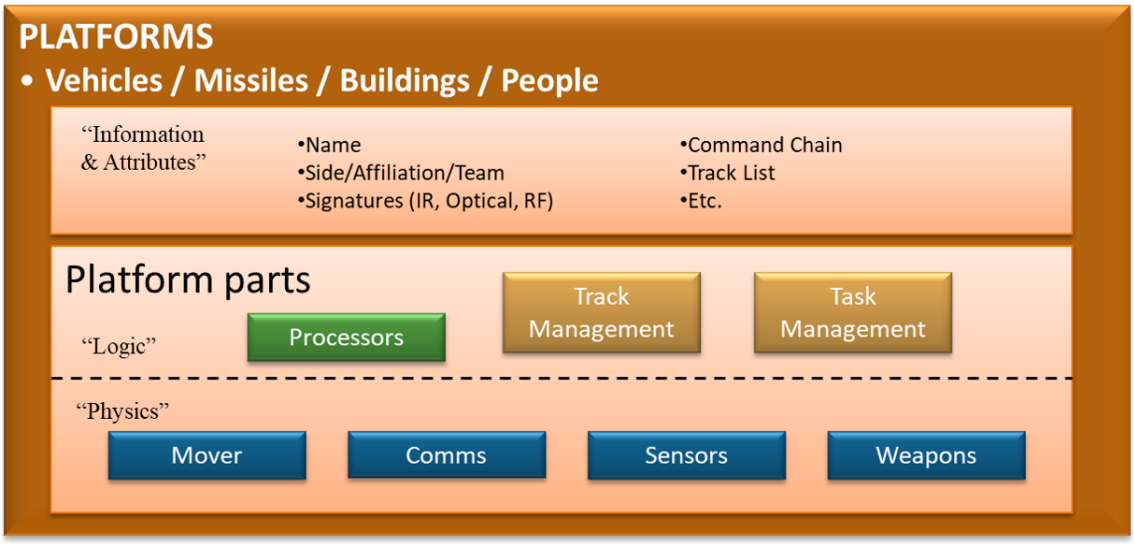

Figure 2: AFSIM Object-Oriented Structure [2]

2.5 Parallel Work

There is relevant ongoing work that can support or further the efforts made in this thesis, specifically with regards to the integration of Cameo with simulation environments. There are several efforts that establish a link between the two tools but typically rely on an intermediary software tool. In one such effort, current AFIT space systems students are developing CubeSat SysML models that are able to perform mission assurance and tradespace analysis through the STK simulation environment. The

framework for this simulation will be discussed in later chapters, but the technique leverages a parametric model in Cameo that informs a MATLAB/Simulink-based tool that is capable of running an STK simulation and processing the output for analysis.[3]

Similarly, there is ongoing work by Mr. Gregory Haun, a DoD Contractor with Cognatus, LLC, to create a fully simulate-able model within STK using a custom plug-in. This technique offers the ability for the model to completely control the behaviors of the simulated system, while STK's output informs the model. This work seems very promising and useful to the direction of this research.[13]

2.6 Summary

This chapter discussed prior and current work pertaining to the problem of evaluating the mission effectiveness of autonomous collaborative munitions. It also reviewed the concept of behavioral models and the semantics of states and modes. These concepts will inform the following research methodologies and applications for this thesis work.

III. Methodology

The overall effort for this research area is to design and integrate a system model with simulation capabilities in order to validate the model as well as assess system mission effectiveness during developmental efforts. This thesis is focused on the development of the executable SysML-based model that defines the system and mission context. This section discusses the rationale and methods utilized to develop the product of this research. It begins with an introduction to the contextual scenario in which the system will be utilized and discusses the framework in which the model is to be built and eventually integrated. This chapter will also introduce some of the novel modeling practices used to build the system behavioral model.

3.1 Scenario Design

The system model derived will be styled as a notional autonomous, collaborative cruise missile as presented in previous work in the field of wide-area search and collaborative munitions. The case study that will be used to demonstrate the capability is a representative mission for a wide area search munition: suppression of enemy air defenses (SEAD). This is an appropriate implementation of the system as a typical SEAD mission involves searching for an enemy threat, classifying the type of threat, and performing an attack.

The SEAD mission is primarily comprised of the munitions, enemy elements, and environmental actor elements. Each of these mission elements will be modeled to a sufficient level of detail to inform a simulation environment.

Due to the similarities between the SEAD and WAS scenarios, similar mission success parameters can be used. As detailed in chapter 2, the mission effectiveness of a WAS munition can be captured by a probabilistic model that is informed by a

number of parameters. These parameters can loosely be grouped into:

1. Number and Type of Targets and FTOs;
2. Size of the Search Area;
3. Area Coverage Rate of the Munition;
4. Capability of the Sensor;
5. Effectiveness of the Ordinance;
6. Effectiveness of Enemy Threats;
7. Environmental Factors.

While these areas can certainly be broken down further into more specific measures, these capture the range of factors that affect mission success. The means of measuring mission success should almost mirror the mission effectiveness formula discussed in the previous chapter. Results from simulation testing should yield the following mission effectiveness results:

1. Percentage of total targets detected;
2. Percentage of detected targets correctly classified;
3. Percentage of correctly classified targets attacked;
4. Percentage of attacked targets destroyed.

To attain these results, the model must track associated values that will be generated when the model is simulated. These values will be tracked on the individual systems within the scenarios as properties and evaluated once the simulation has ended.

3.2 Integration Framework

The design of the model is dependent on the framework for integration with a complex physics engine. There are multiple avenues for this integration depending on the type of analysis. Previous and current work has explored integrations for Design of Experiments (DoE) analysis, trade-space analysis, and mission assurance analysis; however, these integrations involve running the model simulation and physics simulation with minimal coupling. The generalized method can be seen in the top half of Figure 3:

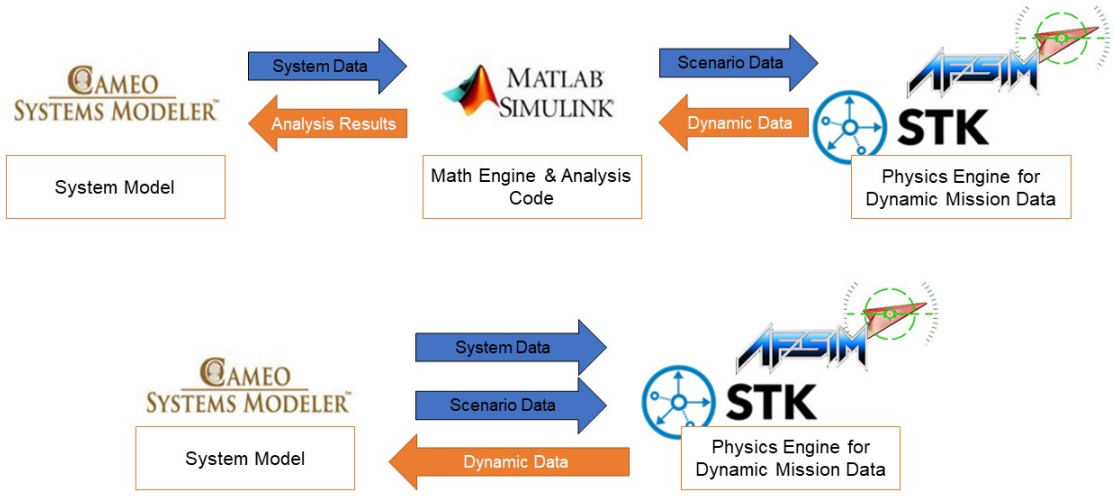

Figure 3: Cameo-Sim Integration Methods [3]

The method described here involves loose integration between multiple tools. While there is a complete feedback loop it relies on an intermediary software tool that is able to pull system values from the system model to inform a complete simulation. The simulation generates reports that are processed by the software tool and inform the system model. This method can offer benefits in the realm of mission

effectiveness analysis. It can run single or multiple scenarios depending on the type of analysis required. By running a large sample of simulated scenarios concurrently, it can more adequately determine probabilistic values based on DoE analysis of simulated operational environments. It also offers the ability to resolve simple behavioral models using the output of the simulated mission.

However, these methods are not currently viable for the validation of the autonomous behavioral model. For this reason, it is desirable for the system model to be able to interact with the simulation in real-time, and for the simulation to resolve the system behaviors commanded by the behavioral model. The benefit lies in the ability of Cameo to negotiate complex behavioral models that are necessary for autonomous agents. The top-level concept is shown in the bottom half of Figure 3. It involves a mostly direct linkage from Cameo to the simulation engine where the data flow is a steady feedback loop. A more detailed look at this type of framework is described in Figure 4:

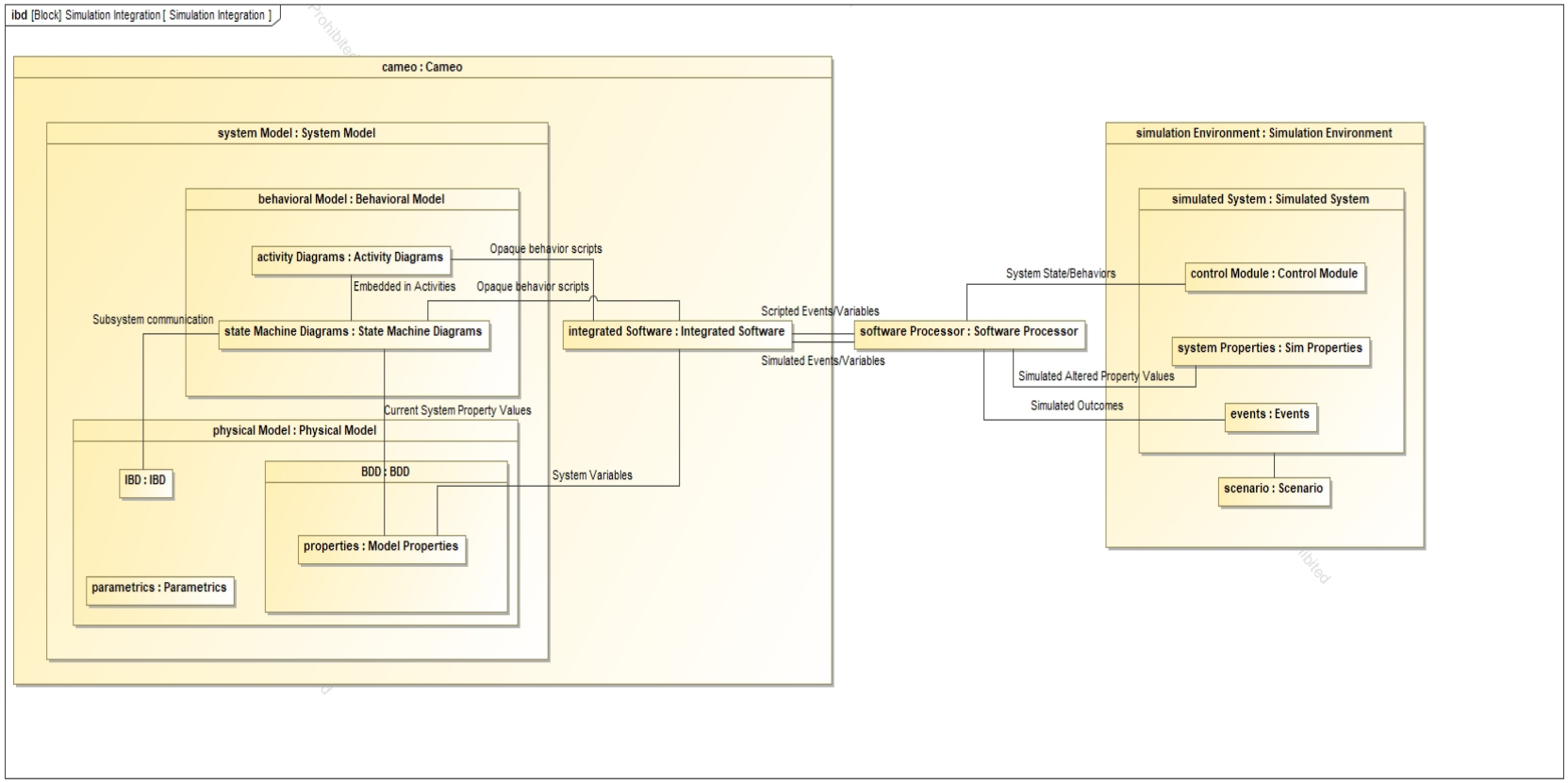

Figure 4: Proposed Cameo-Sim Integration Model

This figure shows the relationships between the program elements and the level of integration required to carry out a real-time behavior analysis. Within the system model in Cameo, the state machine and activity diagrams harness the embedded software capabilities of the program to inform the simulation engine. This may require, such as in the case of AFSIM, the use of a developed plug-in or processor to allow the data to flow. The information from the model will inform the relevant entities within AFSIM, such as a platform's processor part, via a control interface port.

This framework effectively treats the behavioral model as the autonomous software agent that commands the system. The behaviors are carried out by the system replicated in the simulation engine, which provides constant feedback to the model for decision making. It was from this standpoint that the model will be developed, in order to facilitate future integration.

3.3 System Modeling

In order to build a model to facilitate the integration proposed in the prior discussed framework, the proper tooling and syntax must be established. The use of UML and derivative, SysML has already been previously discussed in context with modeling and simulation. Due to its use within AFIT, WeaponONE, and other major programs within the Air Force, Cameo Systems Modeler will be used. It provides pre-existing integration tools to facilitate the framework proposed. The following sections will briefly discuss scenario-specific considerations as well as the SysML and UML concepts and diagrams used to model the system.

3.3.1 Physical System Design

For the purposes of this research, the system will be modeled down to the major subsystem level of the autonomous munition. These subsystems, while not detailed

to the component level, will be indicative of a wide range of smart munitions and UASs. Common subsystems among these include: the air vehicle platform, sensor suite, munition payload, software agent, communications system, and central data bus. The reason for not developing the model to the next level of fidelity is in line with the twofold nature of this thesis—major subsystem analysis should be sufficient for mission effectiveness analysis as well as be able to inform design decisions during the development process.

To increase the modularity and broad applicability of the model, an object-oriented approach was used to create the executable mission. This required the use of the UML notation of classes and instances. Likewise, common properties shared across multiple blocks were established using the principle of inheritance. This established overarching classes that contain the mission elements. This also mirrors how AFSIM manages its elements (e.g., platforms, movers, weapons, etc.).

For the physical modeling the following diagrams were used:

1. Block Definition Diagram (BDD)
2. Internal Block Diagram (IBD)
3. Parametric Diagram (PD)

To describe the properties and drive the execution of the model, systems and components are given value properties. These value properties are assigned to their associated blocks as they relate to the hierarchy. For example, all systems will have a location, so the property is defined at the highest class level. More specific properties, such as airspeed, are assigned at lower block levels. These properties are inherited through classes and resolved in the instances of the systems or components for a specific simulation.

3.3.2 System Behavior Design

In line with the concept of model-based autonomy discussed in the previous chapter, the system behaviors are modeled as layered state machine diagrams. Using Wasson's definitions of states and modes, the autonomous behaviors commanded by the software agent can be described as systems modes while subsystem behaviors can be described by traditional dynamic states. Similarly, some operational states could be defined hierarchically at the class level to provide common states among all the mission elements.

3.3.2.1 State Machines

In SysML, state machines are used to describe event-driven behaviors in a relatively flexible manner. State machines can be used from the component to the system level and can capture simple and complex behaviors. By describing state machines at all levels in the physical model, it is possible to have interactive behaviors that create the type of intersecting modes and states described by Wasson. While state machines are primarily designated for states, the autonomous system modes can be adequately modeled the same way. Modes in an autonomous system are self-driven and based primarily on events or internal planning. In this way, the system modes would be navigated similarly to a state machine.

Within each state, behaviors can be triggered by three means. Using the Entry, Do, and Exit behaviors, the state can execute predefined activities or code as the system enters, while it's in, or as it exits the state, respectively. In order to effectively execute a model-based simulation, these behaviors must be executed outside of the system model and Cameo. While Cameo is capable of some algorithmic computation, it cannot execute complex math, models, and simulation without using an external software suite. However, Cameo has the ability to host opaque behaviors, which are

embedded scripts or calls that run external code in a variety of languages (MATLAB, Java, Jython, etc.). Opaque behaviors can be used both within a state as an action or on the transition to trigger behavioral responses. This provides a powerful tool not only for simple simulation but also integration with complex physics engines.

3.3.2.2 Signaling State Changes

The benefit of using state machines is the ability to plan system behaviors based on internal cues and external events, so it is important to define the method of executing them. State changes can be accomplished in a variety of ways, but the primary methods used in this thesis are signals and change events. Signals provide a simple and repeatable means of executing state changes that can also be configured for manual execution of the model simulation. For this reason, they are used for simple internal system triggers that require little flexibility. That includes sending system updates for the current mode of the system.

Change events are more robust and flexible as they can be written with marginally complex code that waits for property changes or external cues. These events are necessary for external integration, as they will need to monitor simulation outputs or to monitor the value properties that change in response to those outputs. For this reason, all state machines that could directly interact with the simulation will use these events. They utilize externally defined coding languages to perform most of the logic on these events. In this thesis, MATLAB will be used as the primary language.

3.4 Summary

Chapter three detailed the methodology undertaken to build the system model. Using the proposed Cameo-AFSIM integration framework, the behavior model can be developed to include hooks for the external interfaces. Using standard MBSE

techniques combined with Cameo capabilities, the model detailed in the next section was designed.

IV. Results and Analysis

This chapter will discuss the resultant product of this thesis in the form of a state-based behavior model that represents a generalized autonomous munition with cooperative capabilities. The model provides all the structure, behaviors, and context to conduct a mission effectiveness simulation of the munition in a simple SEAD mission. The following sections will detail the physical structure and behavioral implementation to represent the munition as well as details on how the model can be utilized.

4.1 Munition Model

4.1.1 Physical Model

The modeling of this munition was meant to mirror the overall structure of the ASRA both due to the research already involved in its development as well as to provide an opportunity to further validate it in an operationally oriented scenario. This structure applied to a generic air-launched smart munition similar to the LCCM resulted in the system model below.

4.1.1.1 Munition Subsystems

As briefly stated in the previous chapter, the WAS munition is modeled with 6 major subsystems modeled in the BDD shown in Figure 5. Each of the subsystems is described below. Because this research is oriented toward the behavioral side, the physical aspect is only developed with sufficiency to facilitate the execution of the behavioral model.

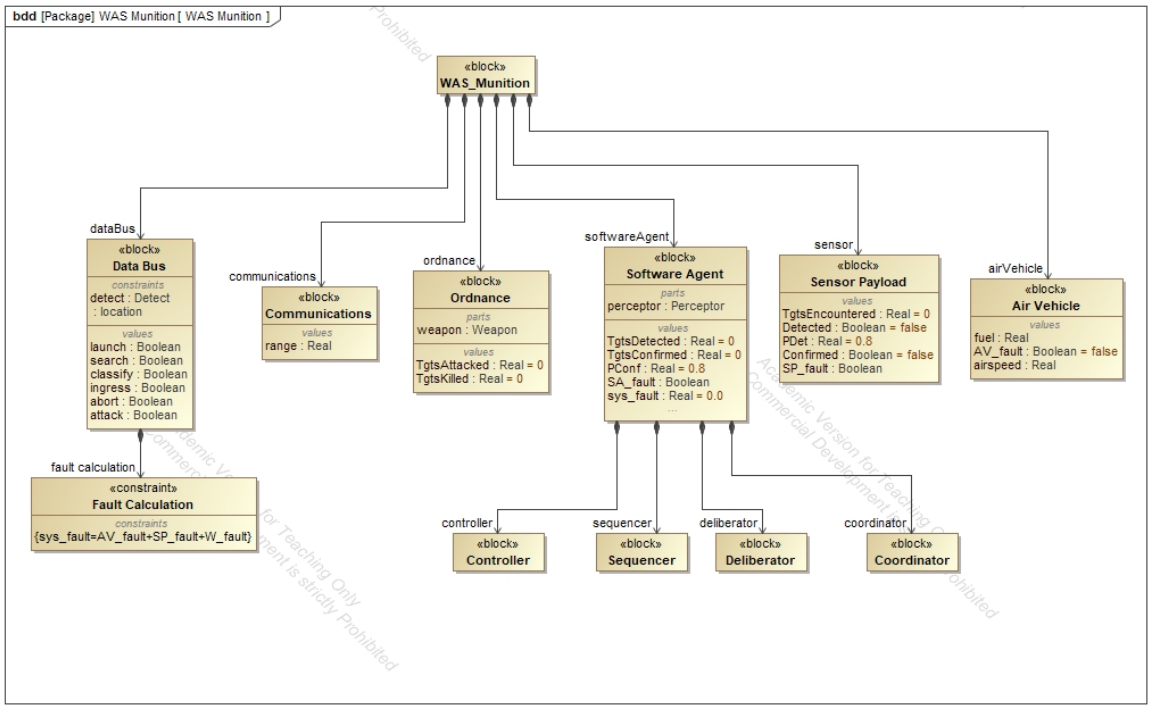

Figure 5: WAS Munition Block Definition Diagram

1. Software Agent: serves as the command-and-control node for the system;
2. Data Bus: data storage and through-put that receives and distributes all system data and statuses;
3. Air Vehicle: subsystem that encompasses all flight systems to include housing, control surfaces, actuators, flight sensors, and simple autopilot;
4. Sensor Payload: subsystem that includes only mission-specific sensors;
5. Communications: subsystem that includes only inter-munition (cooperative) communications;
6. Ordinance: subsystem that represents a mission-specific lethal payload;

Each of these subsystems is represented by a block and exists under the parent WAS Munition block. Their interconnections are shown in the IBD in Figure 6.

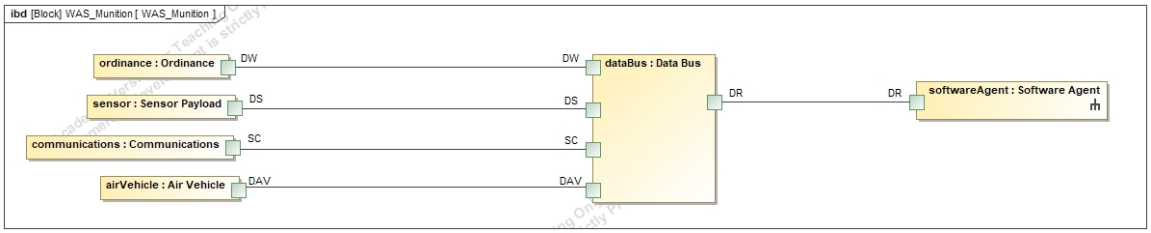

Figure 6: WAS Munition Internal Block Diagram

The software agent is modeled as a single entity that controls the entire system; however taking lessons learned from the previous ASRA implementation, it could also be modeled using the HAMR structure as shown in Figure 7.

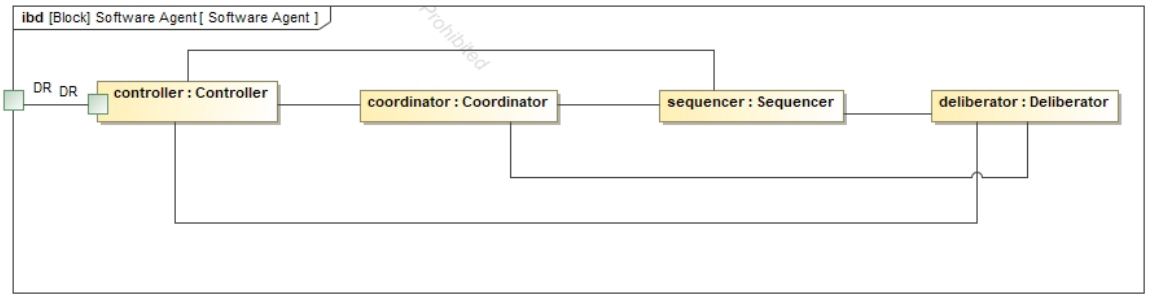

Figure 7: HAMR Implementation of the Software Agent

As shown in Figure 5, the data bus is modeled as the primary interface between all the subsystems and the software agent. This is accomplished within the model through the management of value properties. The exchange of values is accomplished in two ways: the software agent updates Boolean value properties associated with the system modes, and the subsystems update real value properties from the simulation using a parametric diagram. This flow of information informs the system modes and the subsystem dynamic states. This is explained further in the following section.

Each of the remaining subsystems (air vehicle, sensor payload, communications, ordinance) is abstracted at a high level to simplify this model and show that the model can be simulated early in the development process. As shown in Figure 8, they can be

broken down into further subsystems and components as design decisions are made, which can be supported by tradespace analysis using the concept of instances. An example of this is with the sensor payload where the chosen sensor could be primarily imaging based or signal based. This could inform certain states within the sensor state machine and alter how the system interacts with other mission elements.

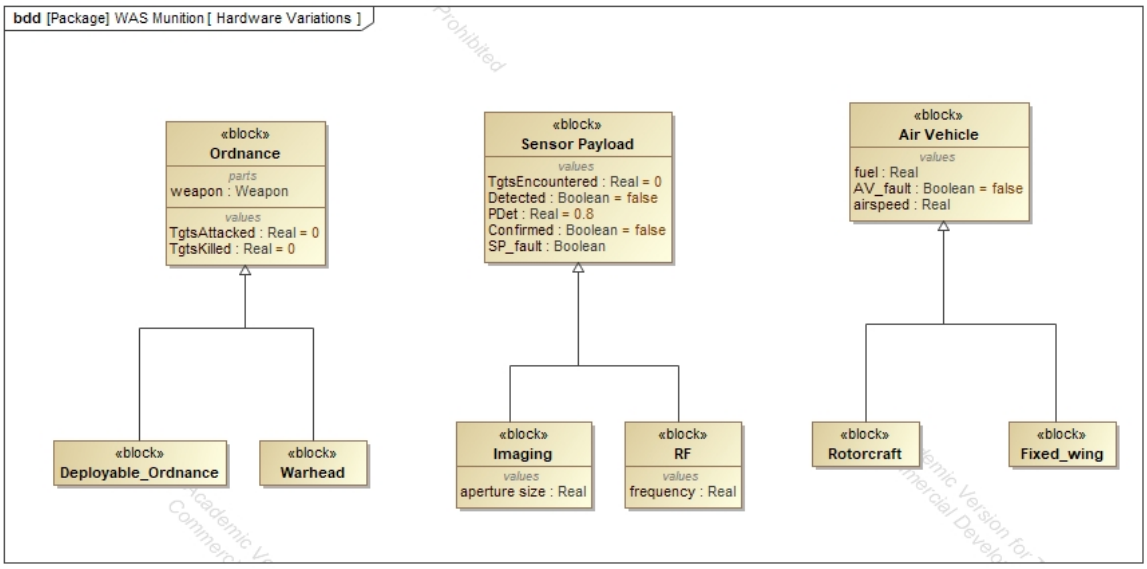

Figure 8: Hardware Variations for Munition Subsystems

4.1.1.2 System Properties

Elements of the physical system defined by blocks are given value properties to track quantifiable aspects of the system or subsystem. Using the object-oriented concept of inheritance, many of the common properties can be defined at a higher abstract level or class. This allows scenario elements to share common properties, which can be later defined when simulating instantiations of the systems. The principle can be shown in the class diagram in Figure 9. Some of the properties owned at the class level are location, detectability factor, survivability factor, system faults, etc. More specific properties are owned within subsystem blocks, such as air vehi-

cle specifications (range, fuel remaining, airspeed, altitude) and sensor specifications (aperture dimensions and probabilities for detection and classification).

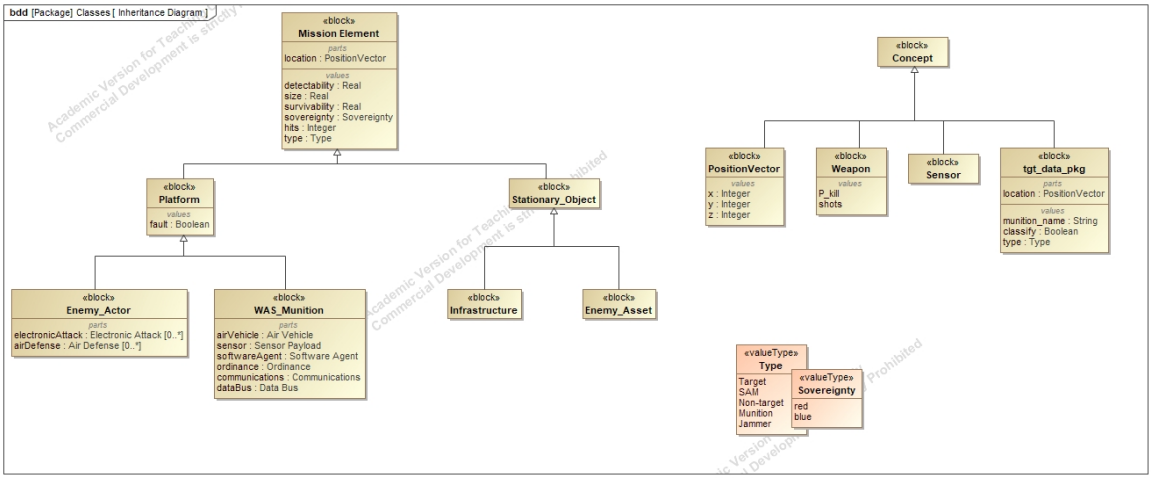

Figure 9: Hardware Variations for Munition Subsystems

Value properties are also used throughout the model to track events during execution. Some of these properties for the munition include: targets detected, targets classified, targets attacked, target type, etc. These, along with some of the dynamic properties of the system, will change as the model or simulation is executed. As shown in the following sections, this is how the model triggers behavior changes.

4.1.2 Behavioral Model

It is beneficial to describe the behavioral model from the perspective of Wasson's states and modes, specifically modes, dynamic states, and operational states. While most of the work has consisted of defining the dynamic states of the subsystems, this model demonstrates the ability to create a comprehensive look at the cross-section of behaviors active at any given time and the circumstances necessary to allow them.

4.1.2.1 Modes

The system modes are currently realized within the Software Agent subsystem block and are represented by the state machine diagram in Figure 10.

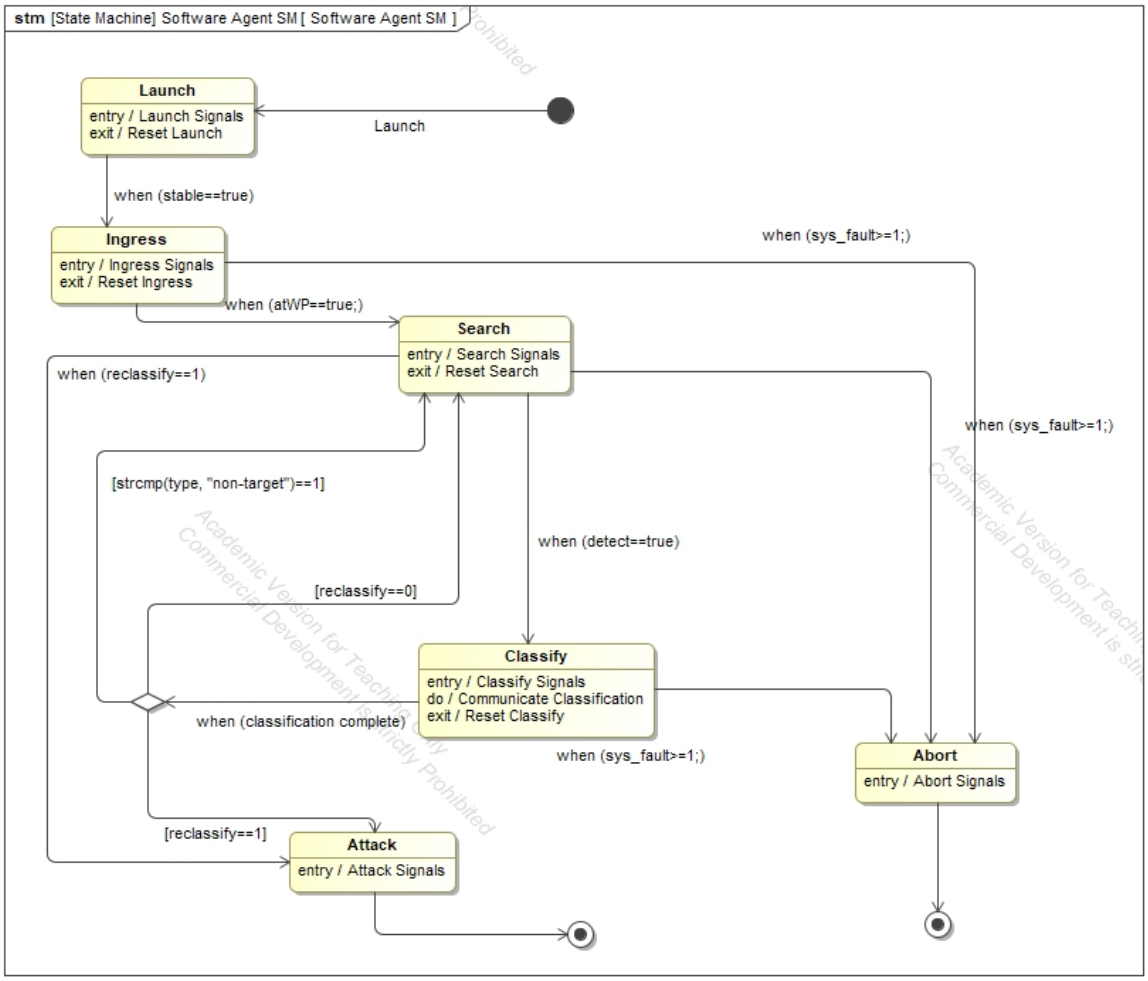

Figure 10: Software Agent State Machine

This state machine encapsulates the system-level modes experienced by the munition within the SEAD mission. The modes with brief descriptions are listed below.

1. Launch: The munition detaches from the housing unit and begins flight. This is triggered by a "Launch" command.

2. Ingress: The munition flies toward a pre-selected area of operation. This is triggered on completion of the air vehicle stabilizing.

3. Search: The munition enters a search pattern and utilizes sensors to attempt to detect a target. This is triggered once the munition arrives at the search area indicated by a waypoint.

4. Classify: After the munition detects a potential target, it flies to the area of interest and attempts to classify it as a true target. This is triggered by the sensor payload reporting a detected object.

5. Attack: Upon successful classification of the target the munition initiates a terminal attack attempt. This is triggered by 2 successful classification reports—one from itself, one from a collaborative munition (if available).

6. Abort: If the munition or its subsystems encounter a major fault it triggers self-destruct activities.

The transitions between modes are based primarily on change events that are monitoring value properties on the block. These are being updated by the subsystems through the data bus as previously described.

Each of these modes also triggers subsystem dynamic states as part of the entry activity. This is currently accomplished through activity diagrams and signals which trigger property updates in the data bus. Each mode has a Boolean property value associated which is set true on entry and false on exit. The subsystems monitor these properties to determine entry into associated dynamic states.

4.1.2.2 Dynamic States

The dynamic states are captured within the other munition subsystems, also as state machines. Each subsystem hosts a state machine model that describes the vari-

ous states of which the subsystem is capable. Each state machine for the subsystems is described and shown in the figures below.

1. Air Vehicle: The Air vehicle has the most complex behavior model due to the wide range of states required of the flight systems. Each of these states is dependent on the command of the software agent and logically flows through a typical SEAD mission. The Air Vehicle must be launched, fly to provided or derived waypoints, fly search patterns, conduct loiter or classification patterns, avoid obstacles, make attack runs, and potentially fly to a safe area for self-destruction.

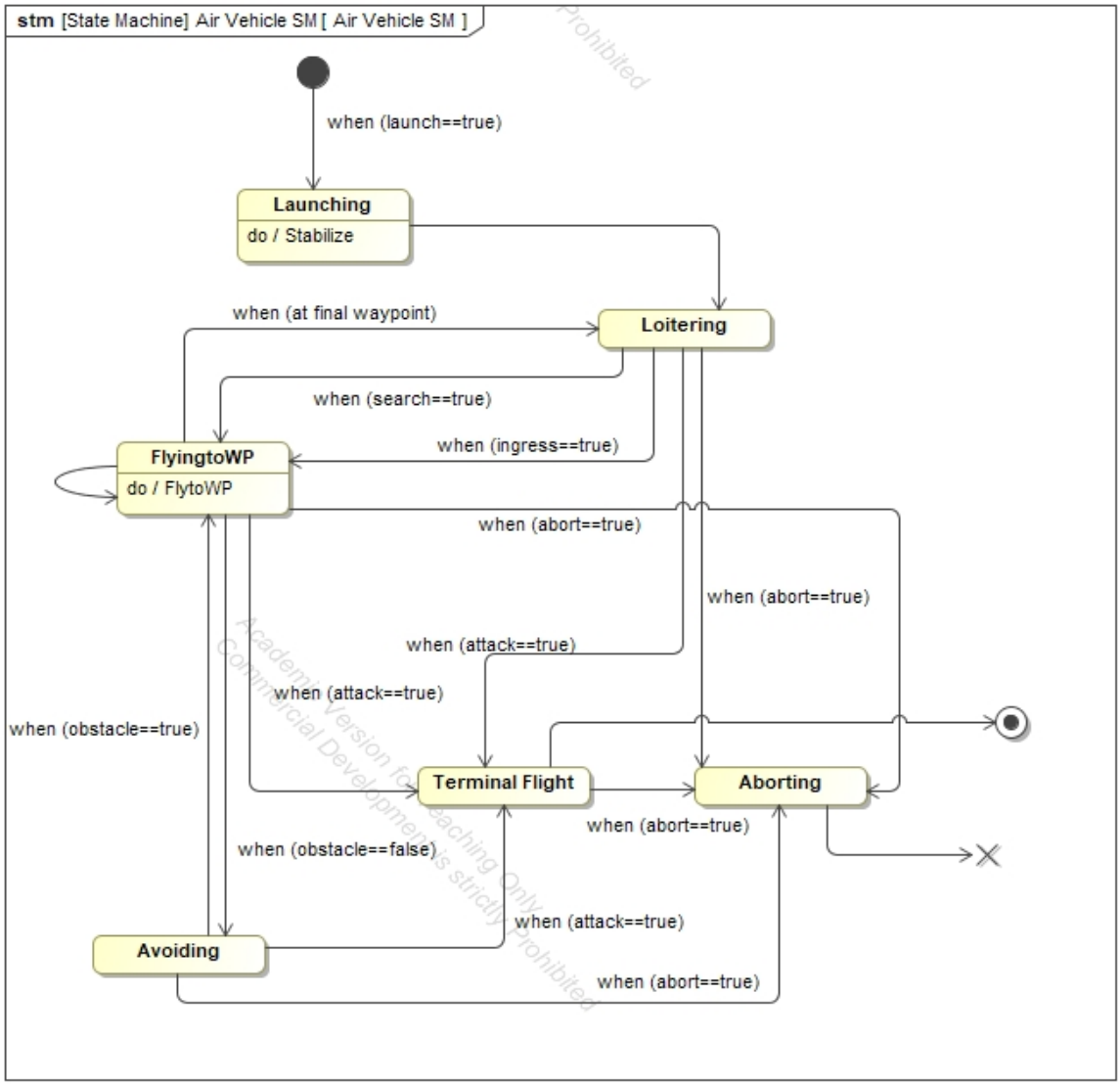

Figure 11: Air Vehicle Subsystem State Machine

2. Communications: The communications behavior model is comprised of two levels of state machines. The first determines the connectivity between the munition and the communications network node; the second deals with whether the subsystem is uploading or downloading information from the network. This is a simplified take in order to better execute the model.

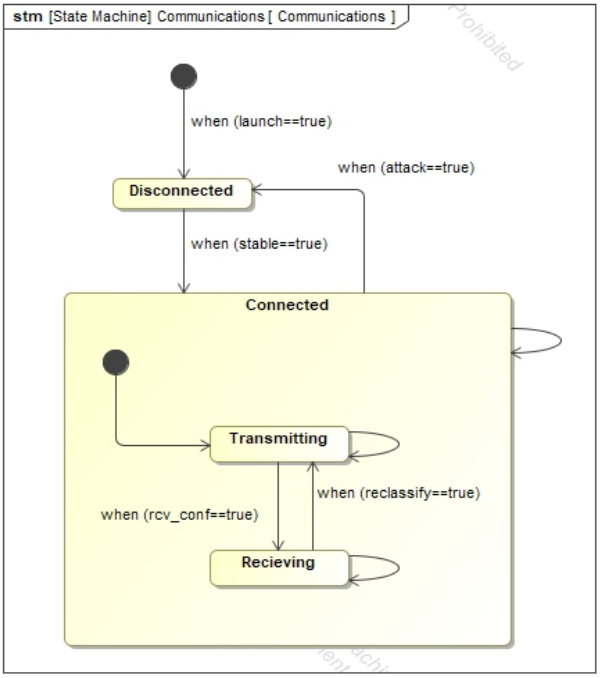

Figure 12: Communication Subsystem State Machine

3. Sensor Payload: The sensor payload, like the communications state machine, is layered. Because the munition must at all times be sensing for obstacles or attacks, the more deliberate states (searching, classifying, etc) are contained within a greater sensing state. This subsystem must also provide information to the ordnance during attacks so it includes fuzing and homing states.

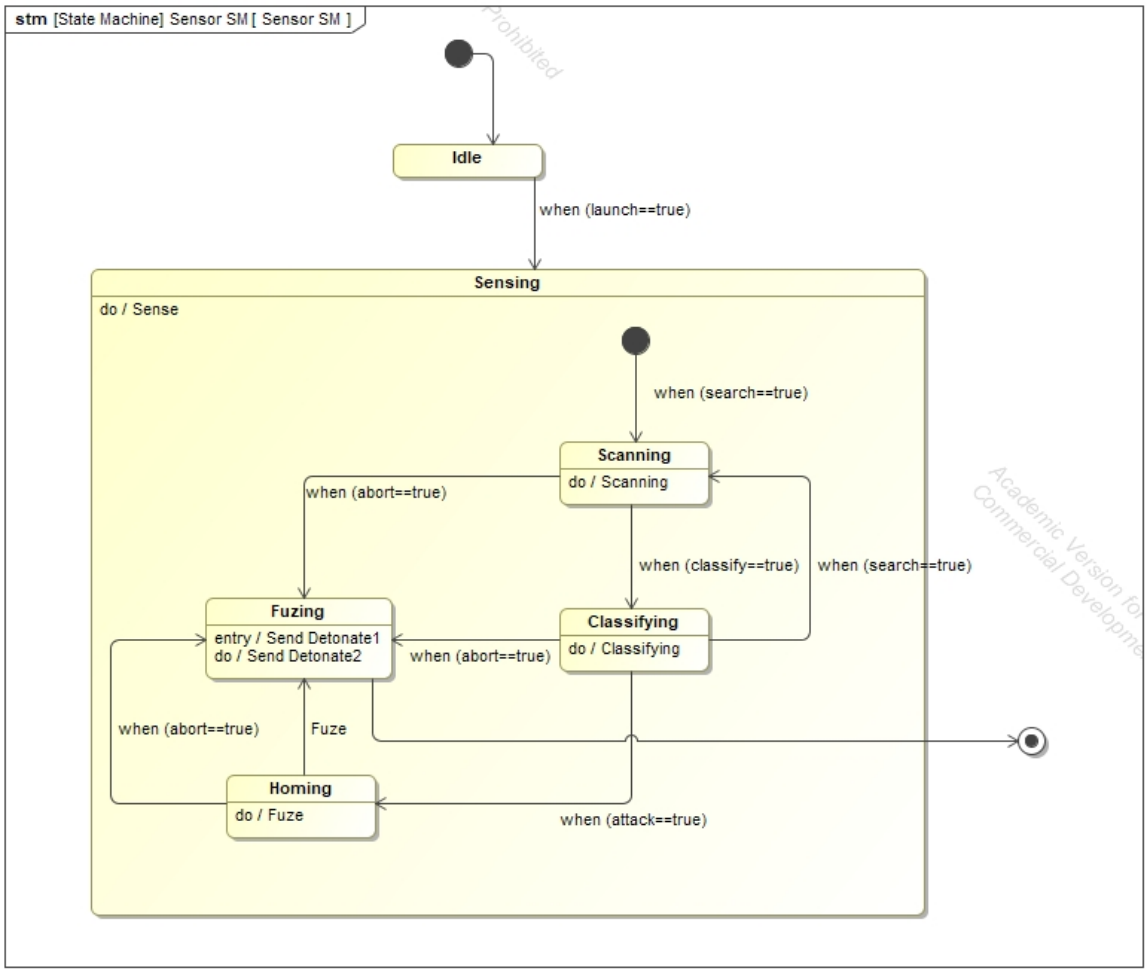

Figure 13: Sensor Payload Subsystem State Machine

4. Ordnance: The ordnance is the most simple state machine, as it does not require significant decision making or processing. It is updated by the software agent or sensor agent into one of the three states shown below.

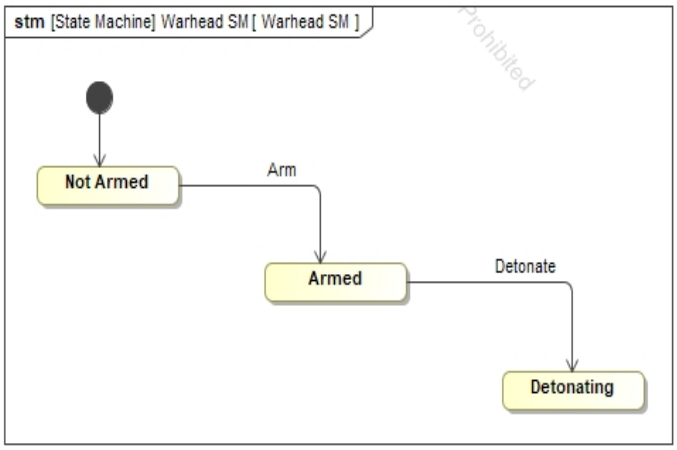

Figure 14: Ordnance Subsystem State Machine

4.1.2.3 Operational States

The Operational States are contained within the class hierarchy in order to capture common states among all systems or subsystems. The most notable operational states are contained in the Platform class block in the Inheritance diagram shown in Figure 13. This state machine is common among all systems relevant to the mission, some of which will be described below. This state machine describes all systems in the scenario as "Operating", "Nominal", or "Destroyed" dependent on whether the system has been successfully hit by an adversary's attack. Residing in these states will change the operational ability of the system, i.e. the system's performance may be degraded or stop altogether.

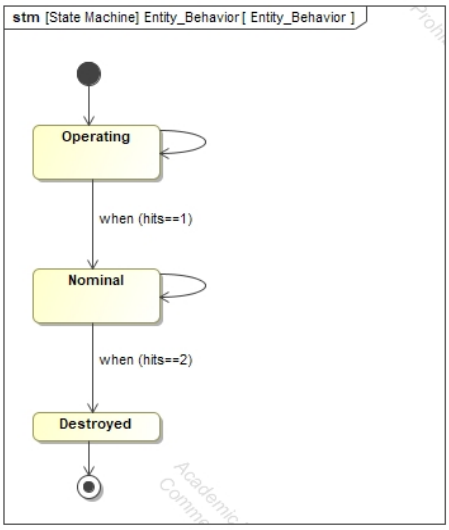

Figure 15: Operational State Behaviors at the Class-level

4.2 Cooperative Behaviors

For this implementation of a cooperative munition, a high cooperation behavior was chosen. This was modeled within the communications block in the munition. As discussed earlier, to limit the necessary modeling, a single munition was modeled with multiplicity that could allow for multiple instances for a simulation. To deconflict the potential issues with intercommunication between instances, all communication activities are done through a "communications network" block.

This method utilizes the behaviors captured in the communications subsystem, which dictates that the munitions effectively upload and download information in separate states. This ensures that within the model, a munition will not intercept its own transmission. The data sent to the network is in the form of a "target data package" that includes the target location and type as reported by the munition.

4.3 Adversary Model

In order to facilitate a full SEAD scenario for simulation, there must be enemy systems that are modeled. These systems should also have behavior models in order to better represent the operational environment. This model contains a simple enemy laydown consisting of surface-to-air missiles (SAMs), jammers, and assets. The physical model is composed in the BDD in Figure 16:

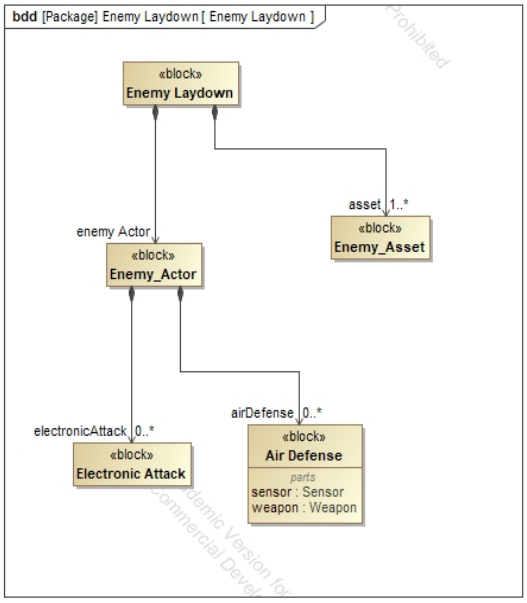

Figure 16: Adversary Model BDD

Each element can be described by its own state machine behavior diagram similar to the own shown in Figure 17. For this implementation, the enemy systems will be primarily defined within AFSIM, so only simple modeling is necessary. However, these behaviors can be interfaced with the munition model using the methods found in the following section.

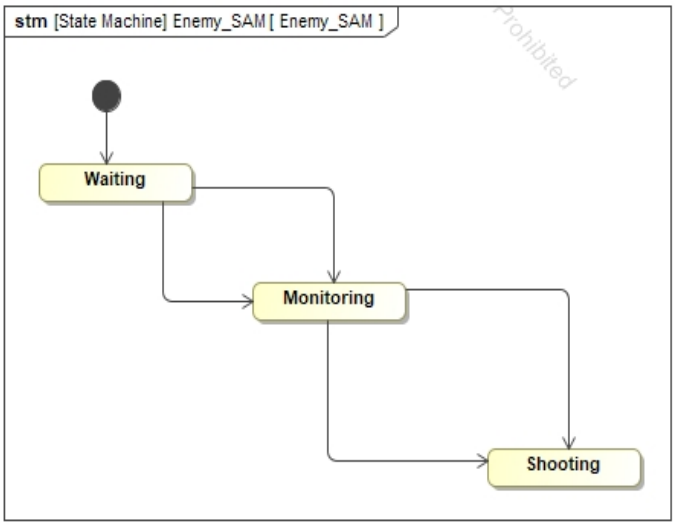

Figure 17: Simple SAM State Machine

4.4 Mission Level Model

The last step is creating an environment model in which these systems can interact. This involves a "SEAD Operational Environment" block which is composed of all the elements relevant to the mission—the munition, enemy laydown, and infrastructure. This is shown in Figure 18:

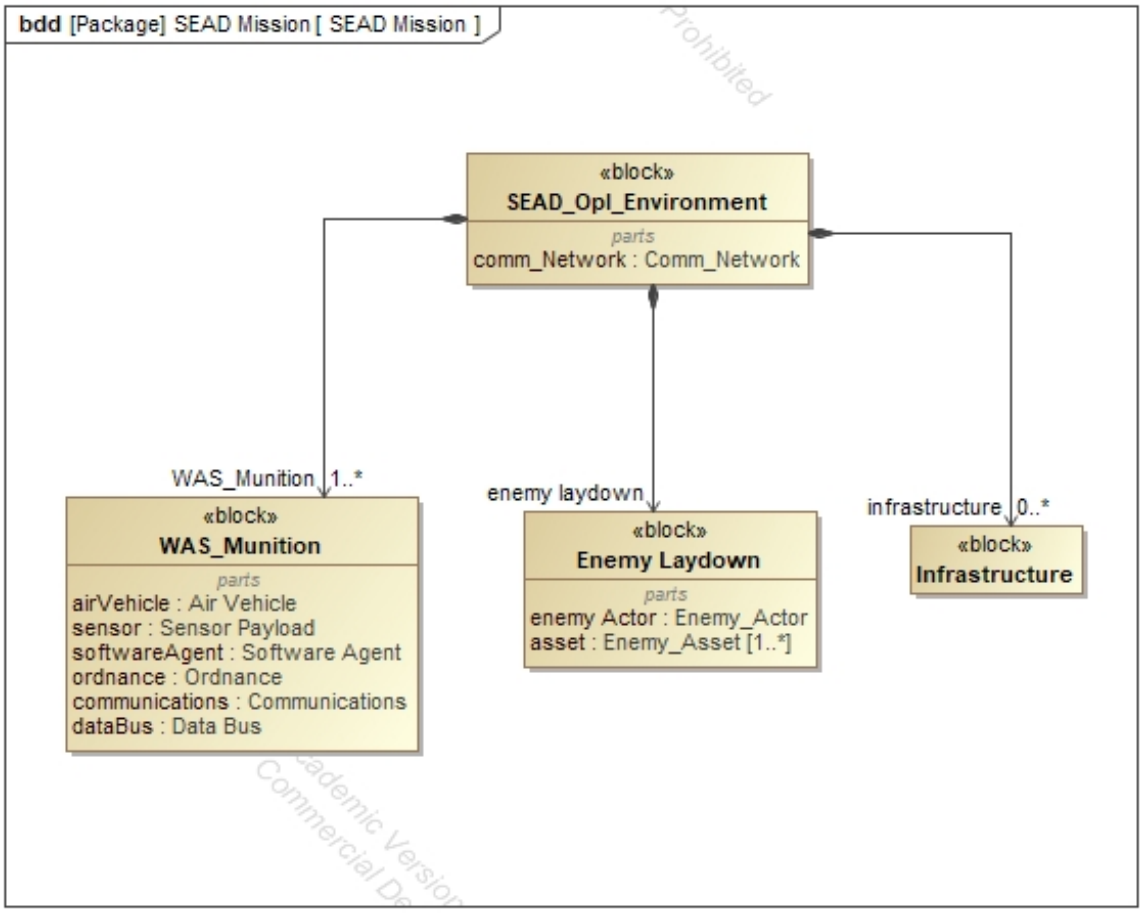

Figure 18: SEAD Operational Environment BDD

Again, an IBD is used to show all relevant interactions that could occur in the mission scenario. In the intended use case, an integrated simulation environment, these interactions represent how the elements will connect within the simulation engine. However, the connections can host signals that demonstrate the interactions purely within Cameo. For example, the enemy actors can project signals to the munition sensor block to represent a visual or electronic signature. Likewise, the ordnance block can deliver a "hit" signal to enemy actors when given the cue to attack. These relationships are shown in the diagram in Figure 19:

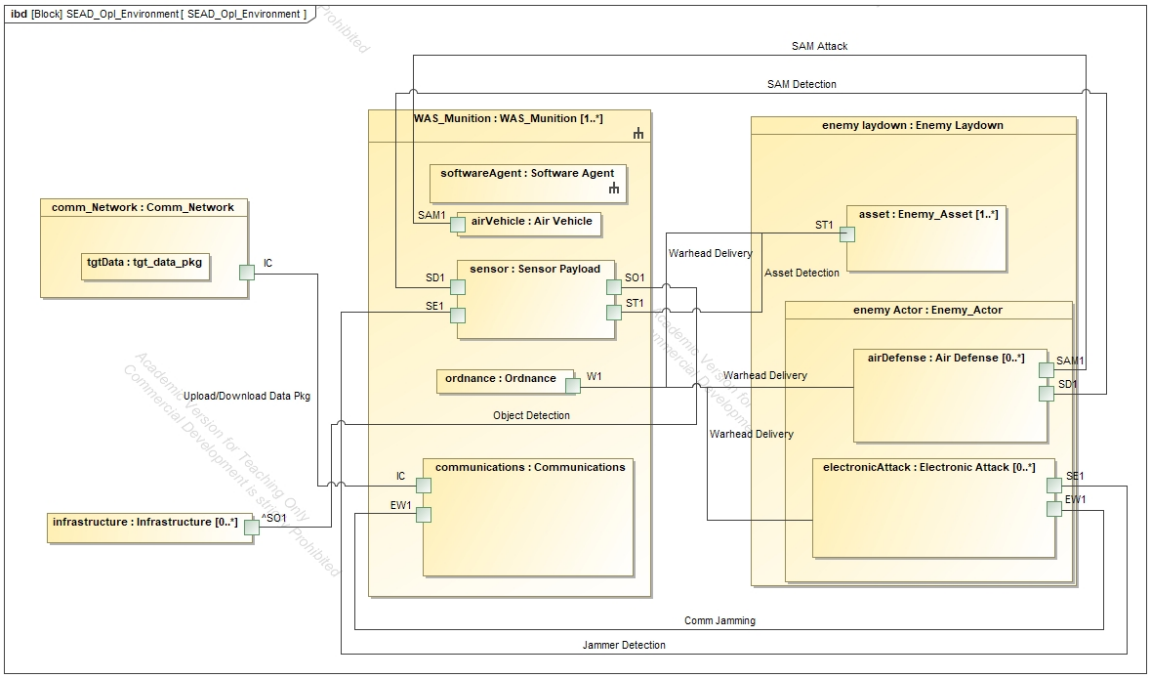

Figure 19: SEAD Operational Environment IBD

Once these relationships are established, further definition is required to prepare the mission for simulation. This is accomplished in the UML instance diagram. In this diagram, the simulation elements are created as instances and given relevant property values dependent on the scenario or physical system. In the Figure 20 example, the simulation involves two instances of the munition and an enemy asset, both of which are assigned starting locations. These instances will leverage all associated diagrams to conduct the simulation, allowing for significant reusability of the model.

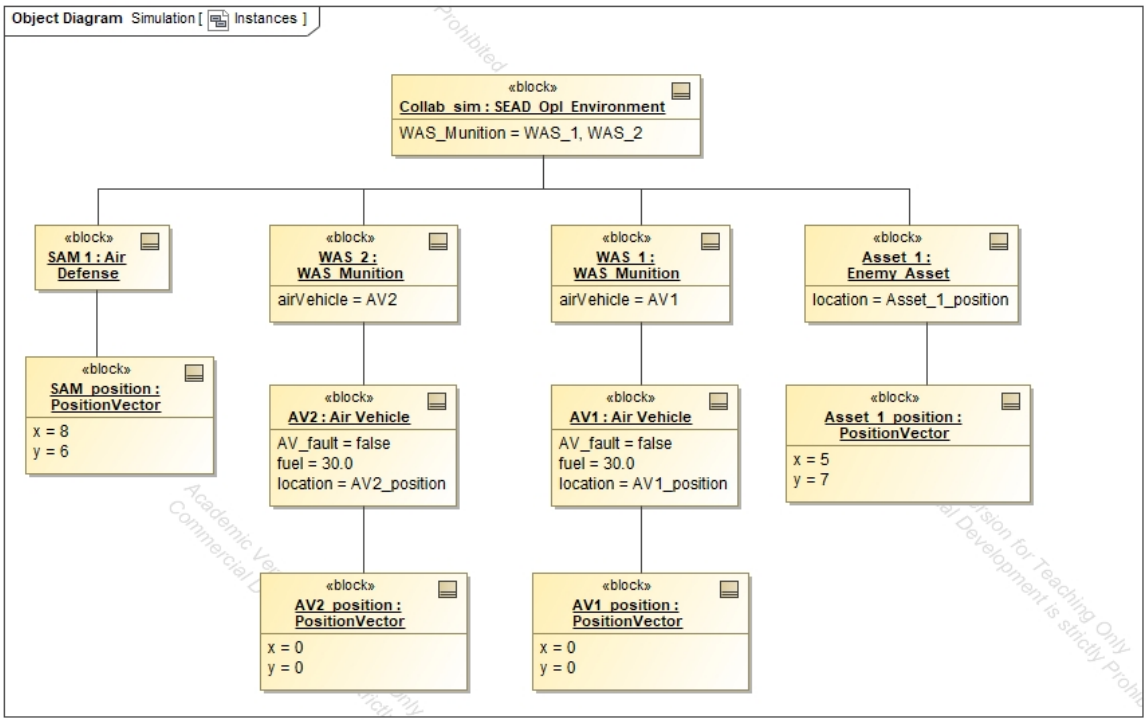

Figure 20: Instance Diagram

4.5 Simulation Potential

Previous sections have detailed the relevant aspects of the model, but it is worthwhile to discuss the application of the model as it has been developed. Due to the flexibility of the model, there are various ways to conduct mission effectiveness analysis. There are three major ways in which to apply this model: entirely within Cameo, Cameo with external algorithms or simple simulation, and Cameo to high fidelity simulation environments.

The first application entirely within Cameo requires the use of signals to execute the mode and state changes within the behavioral model. This can be implemented most simply with manually entered signals, however, with further modeling they can be partially automated. By embedding activity diagrams within the states, simple logical processes can be simulated to demonstrate the validity of the behaviors. While

this method is certainly possible and requires little software experience, it does require more modeling than is necessary with little to be gained from the effort. To effectively validate the model and autonomous behaviors as well as produce useful data, one of the methods below should be used.

The second application involves the use of code embedded into the behaviors that can be executed outside of Cameo in a separate software suite, such as MATLAB. Using opaque behaviors in the state activities, simple code can be written to simulate probabilistic draws and show how the system flows through behaviors based on the random outcomes. These probabilistic algorithms can be used to show how the behaviors satisfy mission effectiveness measures such as those developed by Dunkel and Jacques in prior WAS research. Similarly, the model can execute more complex scripts or functions that can represent a simple simulation similar to the point-mass simulation used in the Cheney and King research. By feeding the outcomes of these scripts back into Cameo—most simply, by altering existing property values—opaque behaviors residing on the state transitions can trigger state changes.

The final application involves developing a full scenario in a high-fidelity simulator such as STK or AFSIM and driving a simulation such as described in the previous chapter. That would provide the greatest validation of the model and the autonomous behaviors it represents. By using Cameo as effectively the autonomous agent in the loop, it can execute behaviors within a simulated operational environment. Similar to the previous method, opaque behaviors trigger both external code execution and state transitions, however, in this method, the simulation is not reliant on probability draws. A physics environment such as AFSIM will use modeled properties values of the munition to determine its effectiveness against adversarial entities.

4.6 Summary

This chapter detailed a behavior-based model of an autonomous, collaborative munition similar to the LCCM. Using the techniques outlined in chapter three, the model demonstrates methods of capturing autonomous modes and states into an executable model. This required a sufficiently detailed, object-oriented physical model, and a hierarchy of state machine diagrams. Using the inherent capabilities built into Cameo, these form a scenario that can be executed within the program or paired with a simulation engine. Using the measures identified previously and that exist in the model, these methods can help determine the validity of the model and eventually the munition behaviors.

V. Conclusions

The following section will conclude this thesis by summarizing the research efforts completed and discussing some of the lessons and limitations discovered during the process. It will briefly discuss opportunities for future research and address the closure of all research questions and objectives.

5.1 Contributions

The outcome of this thesis was a novel state-based behavior model of a generic autonomous, cooperative munition. This resulted from a demand for model-based analysis of autonomous systems and the marriage of two technological advancements: systems modeling tools and advanced physics engines. The research conducted in this thesis identified a framework for integrating these tools and the modeling practices necessary to prepare a systems model that captures autonomous behaviors.

The work done has partially addressed the main research objectives and questions outlined in Chapter One. This research has certainly explored the feasibility and mechanics of modeling autonomous munitions behaviors in Cameo and shown that it provides capabilities to validate the model both internally and through integration. While this thesis was not able to fully validate the effectiveness of the modeled behaviors or the tool itself, it created a basis of techniques and lessons learned for future modeling, integration, and validation.

5.2 Limitations and Lessons Learned

5.2.1 Limitations

Due to the author's research emphasis on model-based systems engineering and limited experience with AFSIM and software integration, the research was scoped to

cover efforts required to develop the system behavioral model. Further limitations were discovered in the use of AFSIM and Cameo Systems Modeler.

The primary limitation of the proposed framework and model is the current lack of integration between the tools of interest. The effort necessary to accomplish this requires significant software experience to develop a custom plugin for Cameo to execute its opaque behaviors in terms of AFSIM's external interface (XIO) capability. While there are currently efforts underway to directly connect Cameo and AFSIM (as described in chapter two), there is none currently available. In the interim, it is possible to connect the two programs via a third-party software tool, however, the effort required to tailor these for an executable behavior model is still significant.

Further limitations were found with regards to Cameo's ability to facilitate internal simulation. While the program is able to conduct simple behavioral simulations through the modeled diagram, it becomes very fragile with increasingly complex models. For this reason, the model is oriented specifically toward an external simulation engine. Efforts in this thesis to manage most of the system and subsystem interactions within Cameo were difficult and are ultimately unnecessary within the scope of the problem.

5.2.2 Lessons Learned

From a modeling perspective, this research demanded rigorous modeling practices to build a robust system model that is useful and navigable. After several iterations, it became clear for the need of a specific ontology with regards to the behavior model. This deals with the hierarchy of classes and the inheritance of various properties. By defining this early in the model development process, it helps organize and direct the development. The model created has its own ontology, but future models may require significant changes.

5.3 Areas of Future Work

Just as this research is a continuation of past work, it invites further study and development to implement the tools and procedures developed here. There are three major areas of future work that would lead to the optimal utilization of the advances made in this thesis. These areas are: Cameo integration with advanced physics engines, such as AFSIM; model implementation using the ASRA framework; and increasing the variability of the model by adding alternative hardware and mission models.

5.3.1 AFSIM Integration

As denoted previously, the direction for this thesis was intended for the integration of an executable behavior model and complex physics simulations. Due to the limitations presented above, this integration still needs to be developed. This will require the development of a mission scenario in AFSIM and determine the coupling necessary to link the two programs. It is recommended that future research focuses on the modeling techniques demonstrated in this thesis to embed code that can both execute behaviors in AFSIM as well as monitor outputs that drive property values in the model.

5.3.2 ASRA Integration

The model developed during this thesis took into consideration some of the architectural cues from the HAMR implementation of the ASRA. However, these designs were not integrated into the executable model. To further validate the ASRA concept for applicability to a DoD-relevant system such as the LCCM, future research could work to associated behaviors, through state machines, with the layers of the HAMR architecture. This would include state machines with opaque behaviors for

the Controller, Sequencer, Deliberator, and Coordinator.

5.3.3 Scenario and Hardware Variability

In the model presented here, only one mission application and hardware configuration of an autonomous, collaborative munition is demonstrated. Continuing work on the model could explore other relevant Air Force mission sets, such as surveillance, air interdiction, or air-to-air interception. Similarly, using the concepts found in this model, further work could be done to model and simulate alternative subsystems and components that can be used with autonomous air vehicles or munitions. This can include sensor payloads, ordnance, and flight systems, which allows both for reusability of the model and the opportunity for tradespace analysis.

5.4 Research Question Summary

Due to the limitations presented above, and the resultant scope of the thesis, the research objectives could not be completely addressed quantitatively. However, the foundational research was explored and steps were made toward resolving the problem space. From this perspective, it is useful to review the results as they pertain to the research questions.

How and to what level of fidelity should autonomous behaviors be modeled in a reference architecture? Based on the phase of development for the system, behaviors can be modeled at varying levels of fidelity. For this case study, the behaviors associated with subsystems were of interest and could be modeled for future simulation in AFSIM. However, it was also demonstrated with the enemy elements, that systems behaviors could be modeled at the system level only.

How well does the SysML state machine diagram capture autonomous behaviors? Using the Wasson definition of modes and states to delineate behaviors, autonomous

system behaviors can be described primarily using event-driven state machine diagrams. It provides a flexible means to create a hierarchy of behaviors to activate certain system actions.

How can Cameo Systems Modeler use executable diagrams to interface with external simulation engines? Based on a review of current efforts and tool capabilities, Cameo is able to fully interface with AFSIM provided an integration software package is developed. Once this is completed, scripts can be embedded into Cameo diagrams to execute the simulation.

Which cooperative behaviors must be identified and modeled to capture the LCCM autonomy software within the current model? This model made use of one of the simple cooperation schemes shown in previous work. The model requires a munition, when requested by another munition, to assist in the re-classification and subsequent attack of an identified target. The communications were modeled using a simple upload/download scheme, which made use of a notional "data network". This scheme stemmed from a limitation of the modeling tool and diagram, but is sufficient to demonstrate and eventually simulate the cooperative munitions behaviors.

How can the model be effectively simulated to validate mission effectiveness? By building a full mission scenario within the model and connecting the elements through external simulation—either in a simple math engine or full physics simulator—it is possible to track mission effectiveness within the model.

What are valid measures to determine mission effectiveness for the concept munition? While these could not be validated through simulation, it was determined that the following measures (in percentages) would be of interest: targets detected, targets classified, targets attacked, and targets destroyed.

5.5 Summary

The work conducted in this thesis has shown that behaviors can be adequately modeled and simulated using a combination of SysML and UML modeling techniques within the program Cameo Systems Modeler. The case study of collaborative autonomous munitions such as the Low-Cost Cruise Missile proved that state machines are powerful tools for examining system behaviors and judging mission effectiveness in an operationally relevant mission. It also provides a framework for future research to build a fully functioning model-based simulation using complex physics simulations. While this thesis builds on previous strides made in autonomy modeling and effectiveness analysis, it lays the framework for a full modeling and simulation capability to aid operational Air Force autonomous capabilities in proving system reliability and effectiveness.

Bibliography

1. Charles S. Wasson. System Phases, Modes, and States Solutions To Controversial Issues. *21st Annual International Symposium of the International Council on Systems Engineering, INCOSE 2011*, 1:278–292, 2011.

2. Peter Clive, Jefferey Johnson, Michael Moss, James Zeh, Brian Birkmire, and Douglas Hodson. Advanced Framework for Simulation, Integration, and Modeling (AFSIM). *Conference for Scientific Computing*, 2015.

3. Kyla Brown and Keith Dreyer. CubeSat Payload Analysis Using MBSE. Unpublished Article, 2021.

4. United States Air Force Office of the Chief Scientist. Autonomous Horizons: System Autonomy in the Air Force—A Path to the Future. Technical report, 2015.

5. Christina Rusnock, Michael Miller, and Jason Bindewald. Observations on Trust, Reliance, and Performance Measurement in Human-Automation Team Assessment. Technical report, 2017.

6. US Department of Defense. Directive 3000.09. (3000.09):1–15, 2012.

7. Office of the Deputy Assistance Secretary of Defense for Systems Engineering. Department of Defense Digital Engineering Strategy. Technical report, 2018.

8. David R. Jacques. MODELING CONSIDERATIONS FOR WIDE AREA SEARCH MUNITION EFFECTIVENESS ANALYSIS. In *Winter Simulation Conference*, 2002.

9. David R. Jacques and Robert Dunkel. Investigation of Cooperative Behavior in Autonomous Wide Area Search Munitions.

10. Katherine Cheney and David King. DEVELOPMENT, TEST, AND EVALUATION OF AUTONOMOUS UNMANNED AERIAL SYSTEMS INA SIMULATED WIDE AREA SEARCH SCENARIO: AN IMPLEMENTATION OF THE AUTONOMOUS SYSTEMS REFERENCE ARCHITECTURE. Master's thesis, Air Force Institute of Technology, 2019.

11. David King, David Jacques, Jeremy Gray, and Katherine Cheney. Design and Simulation of a Wide Area Search Mission: An Implementation of an Autonomous Systems Reference Architecture. In *Winter Simulation Conference*, 2020.

12. Hector Geffner. The model-based approach to autonomous behavior: A personal view. *Proceedings of the National Conference on Artificial Intelligence*, 3:1709–1712, 2010.

13. Gregory Haun. Ssdp enduring high level cell model-based system engineering & ufos simulation. Unpublished Presentation, 2020.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
25–03–2021	Master's Thesis	Sept 2019 — Mar 2021

4. TITLE AND SUBTITLE

State-Based Model for Validating Autonomous Munition Behaviors

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Miller, Dalton J., Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENV-MS-21-M-245

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

James Sumpter
AFRL/RW
Eglin AFB, FL 32542
COMM 850-882-3871
Email: james.sumpter.1@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Autonomous munitions provide an opportunity for the Department of Defense (DoD) to extend the capabilities of operators in combat situations. However, there is also a need for a high level of trust in the effectiveness and accuracy of these systems. With the advent of model-based standards in autonomy and munitions, there is a need to implement these techniques toward an effective modeling and simulation (MS) capability. By leveraging modern MS tools such as Cameo Systems Modeler and the DoD's Advanced Framework for Simulation, Integration, and Modeling (AFSIM), this thesis proposes a framework for simulating complex autonomy architectures within high fidelity simulation environments. Building on this proposed framework, a state-based behavioral model was developed that captures a collaborative autonomous munition within the context of a Suppression of Enemy Air Defenses (SEAD) mission. This system model shows the ability for Cameo to host interactive, executable state machines and demonstrate autonomous decision making based on internal system and environmental cues in order to generate mission effectiveness performance measures.

15. SUBJECT TERMS

Wide Area Search, Autonomy, Model Based Systems Engineering, State Machine

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. David Jacques, AFIT/ENV
U	U	U	UU	56	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3355, ext 3329; david.jacques@afit.edu