Networked Cooperative Autonomous Munitions Parallel Modeling Utilizing Model-Based Systems Engineering

THESIS

Christopher R. Reed, Capt, USAF

AFIT-ENV-MS-22-M-250

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-22-M-250

NETWORKED COOPERATIVE AUTONOMOUS MUNITIONS PARALLEL MODELING UTILIZING MODEL-BASED SYSTEMS ENGINEERING

THESIS

Presented to the Faculty

Department of Systems Engineering

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

Christopher R. Reed, B.S. Aeronautical Engineering

Capt, USAF

March 2022

AFIT-ENV-MS-22-M-250

NETWORKED COOPERATIVE AUTONOMOUS MUNITIONS PARALLEL MODELING UTILIZING MODEL-BASED SYSTEMS ENGINEERING

THESIS

Christopher R. Reed, B.S. Aeronautical Engineering
Capt, USAF

Committee Membership:

David Jacques, Ph.D.
Chair

Lt Col Jeremy Geiger, Ph.D.
Member

Lt Col Warren Connell, Ph.D.
Member

Abstract

Digital engineering and digital design are an emerging area of focus for the United States Air Force (USAF), especially for use in modern, complex systems. An example of a high-complexity system is a swarm of Networked Cooperative Autonomous Munitions (NCAM) that prioritize wide area search and multiple view target confirmation. First, this research discusses methods toward building behavioral models within a Model-Based Systems Engineering (MBSE) tool. Then, this research presents the parallel modeling effort of NCAM in two environments: the MBSE model in Cameo Systems Modeler, and a physics-based model in the Advanced Framework for Simulation, Integration, and Modeling (AFSIM). Each digital model in its environment provides distinct benefits to the stakeholders of the design process, so the models must present consistent and parallel information. Thus, this research also presents automated methods to translate design information between models. Overall, the pair of models working in concert build trust with decision making authorities through understanding of the autonomous processes through systems cognition and digital scenario simulation.

Contents

Page

List of Figures

NETWORKED COOPERATIVE AUTONOMOUS MUNITIONS PARALLEL MODELING UTILIZING MODEL-BASED SYSTEMS ENGINEERING

I. Introduction

1.1 General Issue

The United States military has fostered continuous and rapid evolution in air-to-ground attack capabilities throughout the history of heavier than air flight that began with the Wright Brothers' first flight in 1903. Initially, flight was limited in military application to reconnaissance and surveillance with the U.S. Army Signal Corps in 1909; however, the onset of World War I and later World War II created a boom in military aircraft technology and doctrine. Simple reconnaissance biplanes were replaced by jets that could go faster than the speed of sound by 1946. The United States saw the viability of this rapidly developing technology and created an independent Unites States Air Force (USAF) service in 1947. The air power momentum has continued all the way to present, where modern USAF aircraft can hide their radar signature and drop guided munitions with precision to put 5 bombs in the same hole on the ground!

A logical next capability to add to this incredible USAF portfolio is cooperative and autonomous munitions that utilize intercommunication to find, identify, and strike a target while assessing damage to the target. There are two key definitions from the Assistant Secretary of Defense For Research and Engineering (USD(R&E))for this capability:

- "Automation: The system functions with no/little human operator involvement;

however, the system performance is limited to the specific actions it has been designed to do. Typically these are well-defined tasks that have predetermined responses (i.e., simple rule-based responses).

- Autonomy: The system has a set of intelligence-based capabilities that allows it to respond to situations that were not pre-programmed or anticipated (i.e., decision-based responses) prior to system deployment. Autonomous systems have a degree of self-government and self-directed behavior (with the humans proxy for decisions)." [4]

Current guided munitions follow the automation definition very closely. A target is manually designated by laser or global position, then the munition performs the programmed actions to hit the designated location. In this case, control is closely held by the operator and the decision to fire on a target takes multiple human steps. These human steps give the operator a feeling of trust in the automation because there is a minimization of risk when the trigger is pulled; the munition uses its automation to hit the target more accurately than if the operator had used an unguided munition. When the next evolution to autonomy is discussed, there is a rational fear that the decisions a human normally controls would be instead made by the machine brain of the autonomous system. This distrust leads to hesitancy in deploying a weapon designed to destroy targets autonomously.

Understanding the behaviors associated with the system's autonomous decision making is an excellent way to build trust in autonomy. There are multiple ways to convey behavioral understanding to a human evaluator: first is to provide formal documentation that describes every aspect of the system, next is to create a digital model that represents the system structure and behavior in diagrams, another is to run simulations covering a broad spectrum of scenarios, and finally demonstrations can prove capabilities of the physical systems in test and evaluation. The documentation

method has been the standard for all DoD acquisitions back to the hand drawn schematic era of design. Recently, however, the DoD has expressed interest in using modeling and simulation to document and manage systems. One concept that has emerged is the digital twin, where every aspect of a system is modeled virtually to enable rapid prototyping of modifications and precise configuration control. [5] This digital twin focus also creates clear, navigable data for the physical structures and behaviors of the system it represents leading to sensible comprehension of the system.

1.2 Problem Statement

Model-Based Systems Engineering (MBSE) has rapidly been adopted into USAF Digital Engineering efforts for program and system structure modeling projects as shown by Reed [6]. However, behavioral MBSE modeling for complex systems is not commonly demonstrated within the same projects for the USAF. For autonomous systems, the algorithmic complexity and the emergent behavior that occurs when these otherwise autonomous systems collaborate makes it difficult to evaluate both the logical behavior and the performance impacts. The capability to model system behaviors is inherent to MBSE processes, but MBSE models typically lack the ability to provide detailed physics based models capable of providing performance evaluation of systems in operational scenarios. There are purpose built physics based simulation platforms like the Advanced Framework for Simulation, Integration, and Modeling (AFSIM) that exist for this latter purpose, but often they are disconnected from the definitional models within an MBSE tool [3]. A method and tools for tying together an MBSE behavior model of a complex system and a physics based simulation model of the same complex system is necessary. Ensuring consistent behavior between the pair of models will require an ability to transfer design data between the modeling platforms.

1.3 Research Objectives and Questions

The purpose of this research is to create a behavioral MBSE model of a complex, cooperative munition system and establish an automatic and repeatable method to transfer data from the MBSE model to AFSIM scenarios with capability to execute simulations of the same cooperative munition behavior. The MBSE model will be sufficient to verify the logical behavior of both an individual autonomous munition and multiples of the same munition in a cooperating concept. The AFSIM simulations will in turn provide feedback to the modeler for potential changes to the munition model to achieve enhanced performance.

Cooperative munition modeling research questions include:

- What are the strengths and weaknesses of SysML for behavior modeling?
- Which MBSE elements and/or properties are suitable for translation into AFSIM native language for scenarios?
- To what extent can a SysML digital model represent the behaviors of the cooperative munition used in AFSIM simulation?
- What automatic and repeatable methods can be utilized for data exchange between a SysML model and AFSIM scenario?

1.4 Methodology Summary

This research must first determine connection points and required variables for integration into AFSIM which will help define the logical interfaces of the MBSE system model for the cooperative munitions. These interfaces help to define the boundaries of the MBSE model for cooperative munitions and provide data points for integration back into the AFSIM scenario models. Critical areas to design and test

are: modeling the variables and underlying equations required for AFSIM entities; providing automatic export usability of both munition and scenario parameters from the MBSE model to AFSIM; and identifying modifiable areas of the MBSE model that will impact the simulations. Based on the assessment for connection points, the research will shift into creating an MBSE model that maintains the connection points while building out behaviors in parallel to the AFSIM model. Behaviors in the MBSE model will be evaluated against the AFSIM model as appropriate.

1.5 Assumptions and Limitations

This research is limited to virtual munition modeling and simulation. Furthermore, the cooperative munition concept defined for this research is notional; as such, the munition models will be populated with notional data.

1.6 Preview

Chapter II is a literature review of publications related to munition modeling, AFSIM integrations, autonomous UAS behavior modeling, and historic USAF applications of advanced munitions. Chapter III covers the design methodology of cooperative munition concepts and methods for automatic data transfer into AFSIM scenario simulations. Chapter IV discusses the completed Networked Cooperative Autonomous Munition (NCAM) MBSE model with behavioral analysis, automatic translation results, and comparisons between the parallel models Chapter V summarizes the significant findings of the research and recommends future topics of research.

II. Background and Related Work

2.1 Overview

Background information on the cooperative munitions concept, MBSE methodologies, autonomy architecture, AFSIM munition simulation, and previous work is presented in the literature review. The following information is relevant to understanding this research.

2.2 Cooperative Munitions Concept

The cooperative munition concept directly assists with USD(R&E) Autonomy Community of Interest Test and Evaluation, Verification and Validation (ATEVV) Gaps 1, 2, and 4. [4] For Gap 1, development of a representative autonomous system like the cooperative munition helps to define the needed, verifiable requirements for a more complex autonomous system though discovery of issues and performance specifications within AFSIM. For Gap 2, Allen developed and tested the UAS swarm agents in a modeled environment [7] and Combs further refined the testing to include formal test and evaluation of the total system [8]. The cooperative munition research further addresses Gap 2 through AFSIM scenario simulation testing of the modifiable MBSE munition design. Finally, this research addresses Gap 4 through new experience with autonomy that will need to operate in intermittently denied environments, i.e. temporary signal outages.

The overall concept of cooperative munitions is a fusion of autonomous system architecture with generic aerial munition architecture. The primary mission set for the cooperative munition could range from search and destroy to precision strike with autonomous Battle Damage Assessment (BDA). The closest analog to base the cooperative munition behaviors is in cooperative Unmanned Aerial Systems (UAS)

that have been researched in the past. Previous work on the Software-In-The-Loop (SITL) swarm agents developed by Allen [7] and refined by Combs [8] can be tailored to a more specific munition focus. However, a newer autonomy modeling method used by King and Cheney [2] with the Autonomous Systems Reference Architecture provides a better framework for future development.

A realistic use case for further automation in munitions is in BDA and immediate reattack decision making through the use of cooperative munitions. The Air Force Doctrine Publication (AFDP) 3-60 Targeting breaks BDA into six phases. Phase 1, 2, and 3 asses damage to the target at various levels. Phase 4 focuses on whether the weapon used functioned as intended. Then, Phase 5 is an Estimated Damage Assessment (EDA) to provide the commander of the strike with enough information to decide on Phase 6, Reattack Recommendation and Future Targeting.[9] These phases have clearly defined rule sets and objectives which provide a great opportunity for system automation as defined by USD(R&E).

2.3 MBSE Methodology

The International Council On Systems Engineering (INCOSE) defines MBSE as: "Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases." [INCOSE SE Vision 2020]. MBSE works by capturing structural and behavioral architecture within an interconnected model using a standardized language. The Object Management Group (OMG) has chaired the creation of a systems engineering specific modeling language called SysML which is an extension of a traditional software engineering language, Unified Modeling Language (UML)[10]. SysML enables engineers with diverse backgrounds to interpret various

diagrams in a uniform manner.

While SysML is a standard modeling language, MBSE as a practice includes more than a modeling language. OMG recognizes the additional need for a modeling methodology and a modeling tool. There are multiple tools offered to create SysML based diagrams, including: Sparx Systems Enterprise Architect, Cameo/No Magic Suites, and IBM Rational Rhapsody. Each tool offers unique capabilities which complicates the ability to share information across tools. Thus, it is important to standardize a tool at least at the working group level. For the purposes of this research, Cameo Systems Modeler will be used as the MBSE tool.

In terms of methodology, there is a near infinite number of options because the methodology is best equated to a group's modeling rules. A consistent methodology is important because models or elements created by each systems engineer working on a project need to be properly interfaced to join seamlessly. One standard modeling methodology presented by INCOSE is the Object Oriented Systems Engineering Method (OOSEM). A brief overview of OOSEM is presented in Figure 1. [1]

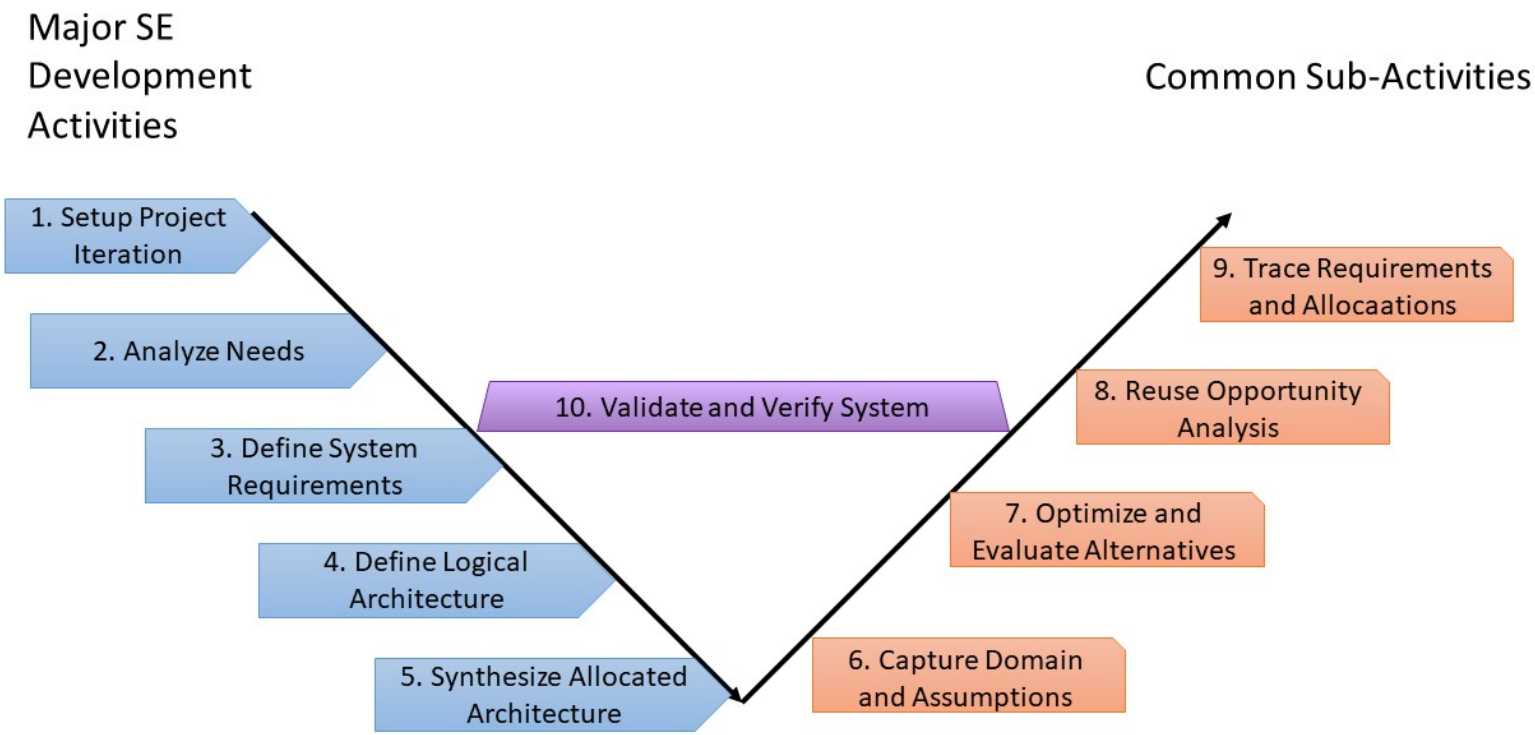

Figure 1. INCOSE Description of OOSEM derived from [1]

OOSEM is generally applicable to a system in development starting with requirements definition and use cases, then moving into a logical system composition, and finally building out the structure and behaviors. This methodology ensures the system model is thorough and consistent from project to project. OOSEM is an overall context for model generation, but does not address organization specific processes. Thus, organizations should spend time creating a methodology that is suitable for the projects in its portfolio. In the case of this research, a pair of reference architectures were used as a starting point for concept exploration and trade space analysis; therefore, a shorter OOSEM approach starting at Step 4 in Figure 1 will be used to build out the NCAM model.

2.4 Autonomous System Reference Architecture

The Autonomous Systems Reference Architecture (ASRA) is an AFIT-developed reference architecture with the purpose of giving researchers a uniform starting point toward development of autonomous system architectures within a system model. A standard layered autonomy architecture for an agent core would consist of three layers of abstraction for division of roles and functions: controller, sequencer, and deliberator. The controller acts as the signal bus that sends specific signals to actuators or subsystems to complete tasks. The sequencer acts as the to-do-list, which passes commands to the controller in a desired order to accomplish a behavior. The deliberator reads the system's status and decides which behavior should be engaged to best achieve the desired system objectives. ASRA translates an AFIT-developed Hybrid Architecture for Multiple Robots (HAMR) into SysML for use in Cameo Systems Modeler as shown in Figure 2. A key component of HAMR is a top, fourth, layer of abstraction called the coordinator which interacts with external systems to provide additional input to the deliberator. [11]

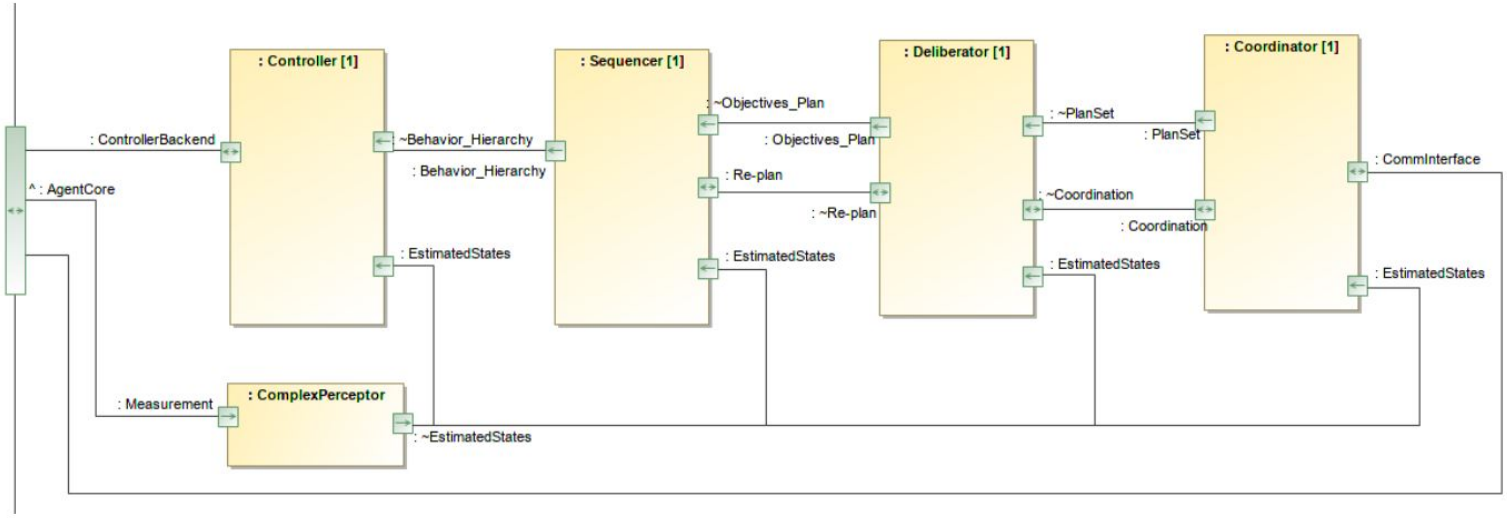

Figure 2. ASRA Model of HAMR for an Agent Core [2]

2.5 AFSIM Background

The Advanced Framework for Simulation, Integration, and Modeling (AFSIM) is a simulation tool developed by Air Force Research Lab (AFRL) which acts as a flexible platform to assist with research, development, analysis, and experimentation. Simulation scenarios within AFSIM can utilize common models, environments, and threats with specified behaviors in any domain from ground to air to space. Additionally, customization in the models and behaviors can be included within the scenarios by importing a text file.

Two colleagues, Hatzinger and Gertsman, have developed a custom AFSIM scenario for 14 NCAMS against different quantities and levels of targets given set munition input parameters.[3] The scenario involves a large area search with randomly placed targets where munitions have different behaviors dependent on identified target type based cooperation thresholds. If a NCAM encounters a target and classifies it as such, it becomes a leader munition and it will initiate an auction where it asks other NCAMs to provide a bid between 0 and 1 based on suitability to assist. A high bid value, close to 1, indicates the other NCAM is nearby and can provide assistance; while a low value means the NCAM is better off not breaking from search. The input value of cooperation threshold for each target type determines whether the lead munition will request another NCAM to assist. A threshold of 0 means always cooperate, where the highest bidder becomes a follower munition that maneuvers to the identified target and runs an independent classification. A cooperation threshold of 1 means no bids will win and the lead munition will either perform a solo attack or return to searching. These decisions are presented in a decision tree in Figure 3. The actions prescribed by the decision tree are repetitive but require different encounters to occur. The repetition gives an opportunity to save time in an OOSEM architecture by reusing activities as objects. Thus, a change in one activity is automatically

propagated to every place it is used, preventing obsolete design from lingering in the system.

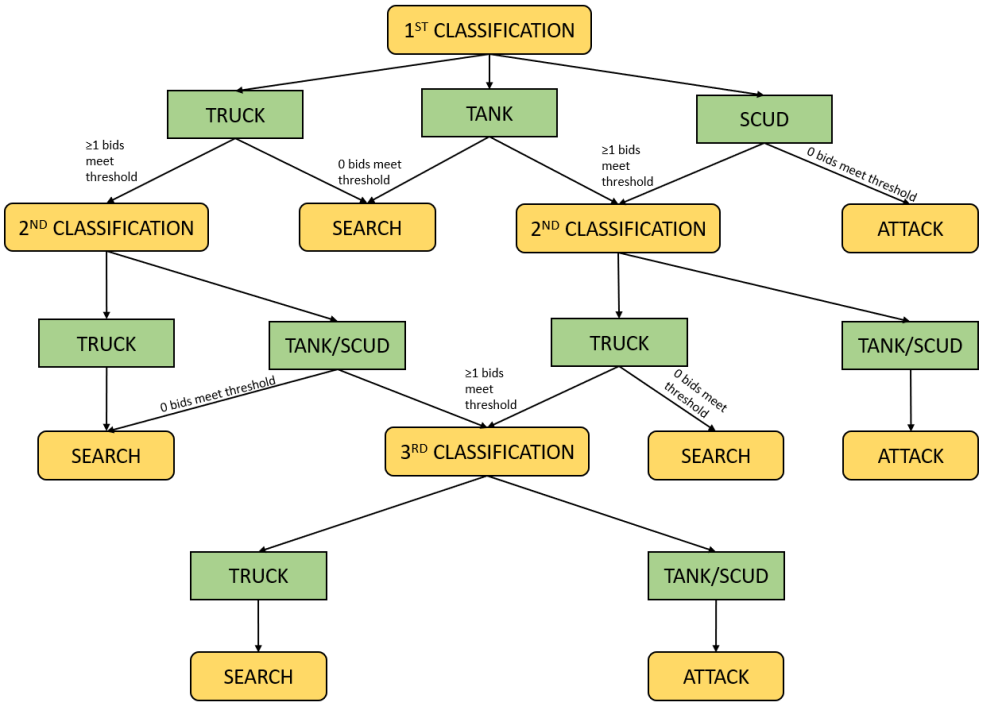

Figure 3. NCAM Target Identification Decision Tree [3]

Keeping true to AFSIM goals, Hatzinger and Gertsman designed their scenario with the intent to design an experiment with 40+ levels and a total of 40,000+ runs. Each run takes a moderately powerful computer less than 10 seconds to simulate with key inputs and output results recorded in a spreadsheet. While AFSIM provides a simulated environment to observe the behaviors of models in the scenario and log files to record every piece of data, comprehension of the AFSIM code is only aided by the coder's recorded comments.

2.6 Previous Cooperative Air Vehicle Work

There are two primary sources of previous work that help to frame the NCAM digital modeling effort. First, an MBSE model created by Jacques utilizing state machines with activities reacting to signal inputs. Second, the thesis created by Cheney and King covering an unmanned air vehicle wide area search scenario.

Jacques' model of a Wide Area Search (WAS) munition is structurally designed with a composition of high level subsystems presented in Figure 4. The air vehicle, warhead, sensor, and software agent act as the abstracted subsystems of the munition with an additional block for environment to provide inputs to the systems. The structure of the model is further built out with an Internal Block Diagram (IBD) for the WAS munition where each subsystem is logically connected with proxy ports, as seen in Fig 5. Additionally, the subsystems have value properties that influence the behaviors of the system in simulation. For example, PConf (the probability of the sensor to confirm a target) gives a threshold for a random number generated in the simulation to determine whether a target is recognized by the software agent. Other values, like Confirmed (number of targets confirmed by the software agent), are updated within the simulation by opaque behaviors to track the status of the munition and evaluate performance. Figure 5 also includes the state machine diagrams for each subsystem that create the framework of the system behaviors. The state machines determine which activities or actions each subsystem is allowed to accomplish at a given time. Each state additionally has one or more transitions with guard conditions of a specific signal to leave the state and enter another. These guard conditions are triggered within the simulation by the subsystem receiving the specified signal either from an internal action of the subsystem, an external subsystem sending the signal to the subsystem, or the user manually inputting the signal to the subsystem. Overall, the methods used by Jacques to create this reference model will be expanded upon

for the NCAM digital twin representation.

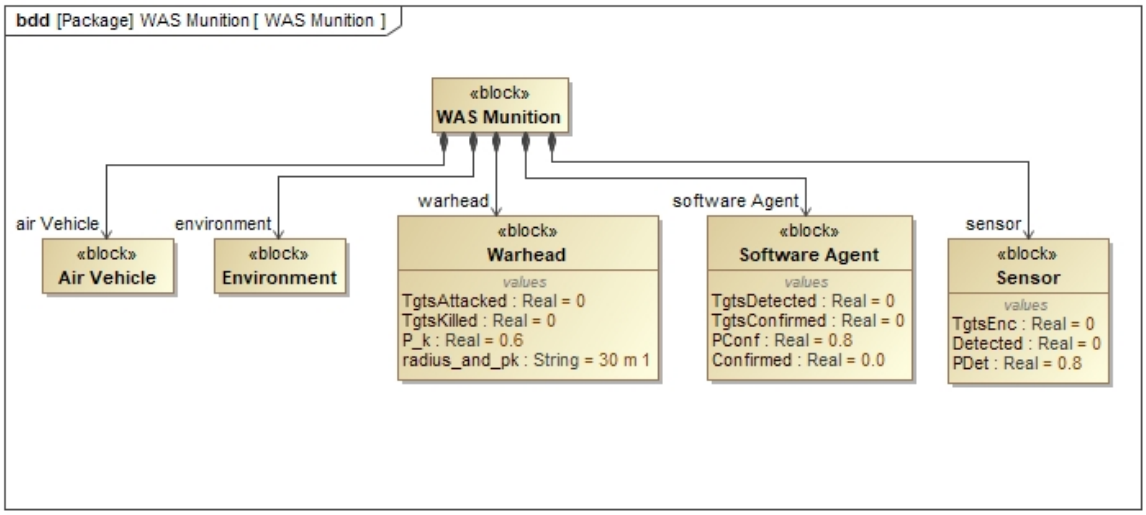

Figure 4. WAS Munition Composition

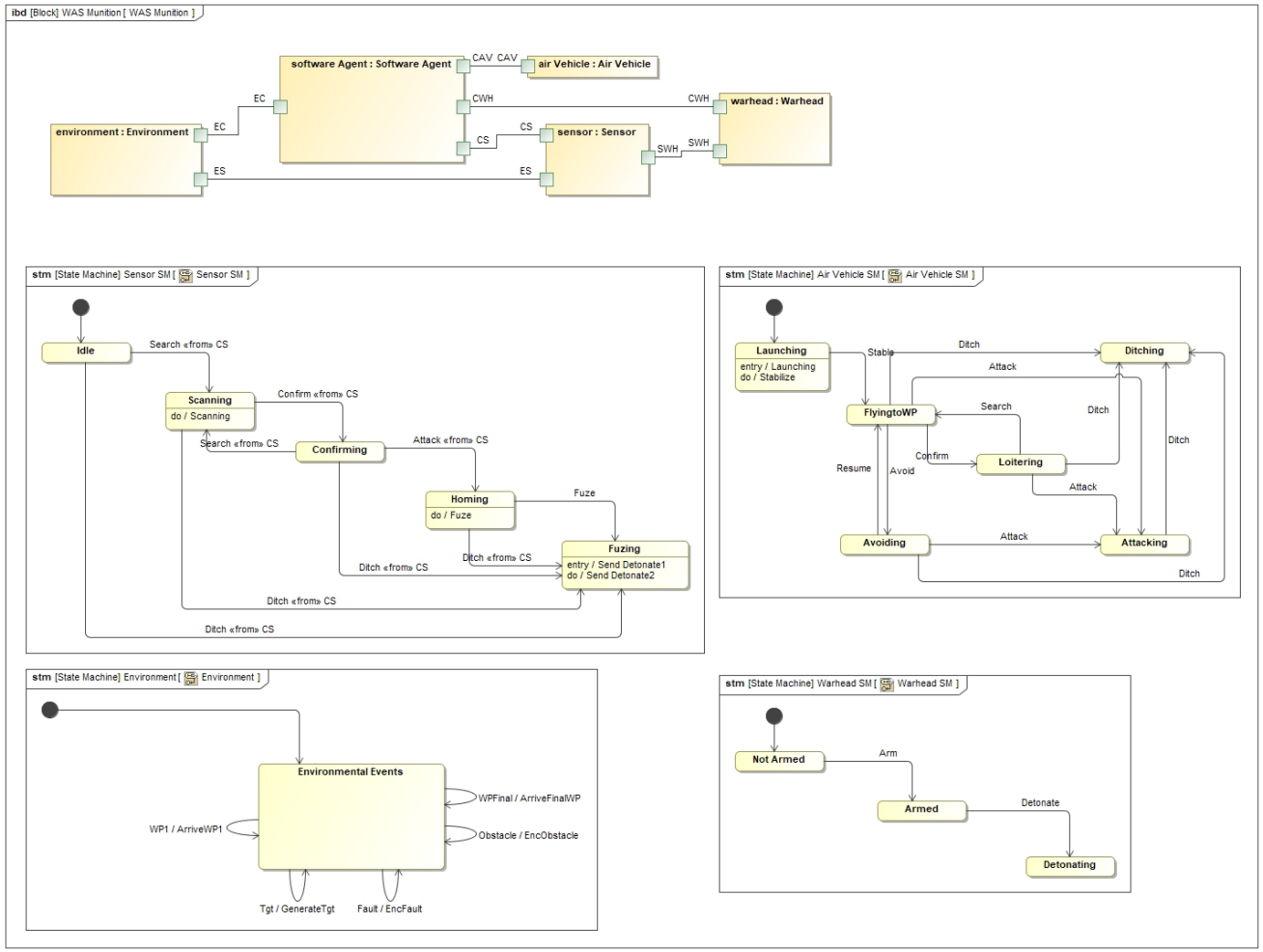

Figure 5. WAS Munition IBD

Cheney and King utilized the HAMR to design an autonomous WAS Unmanned Aerial System (UAS) in SysML which controls a physics based UAS simulation in Ardupilot SITL. Cheney Section 4.3 [2] describes the functions allocated to each module of the HAMR to enable the WAS scenario simulation. The controller was built to send commands based on selected behaviors to the desired simulation platform. The sequencer checks the status of the controller and sends prioritized behaviors based on the current objective plan provided by the deliberator. The deliberator tracks past encounters and current status of the UAS to determine which objective to provide to the sequencer. The coordinator was only used in multi-UAS scenarios to communicate with the other agents on splitting search areas and cooperatively confirming targets. The Cheney and King implementation of HAMR relates closely to the concept of cooperative munitions that need to perform functions similar to the WAS with multiple UAS.

III. Model Methodology

3.1 Overview

This methodology chapter discusses the processes used to create the NCAM MBSE model and the decisions made for direction of the model. First, the AFSIM physics based simulation model is assessed for points of connection to the MBSE digital twin model. Then, the MBSE methodology decision is discussed which leads to the logical and behavior model methodology. Finally, the automatic features of MBSE are discussed for relation back to AFSIM.

3.2 AFSIM Connections

The AFSIM model developed by Hatzinger in collaboration with Gertsman and Reed is presented in the thesis, Mission Effectiveness Analysis of Networked Cooperative Munitions using Modeling and Simulation [3]. Importantly, sections 3.1 and 3.2 of their thesis discuss the assumptions and limitations of the AFSIM model followed by a description of the NCAM scenario design. The MBSE model will assume identical scenario structure but with further simplification due to the lack of a physics based simulation engine within Cameo. The key simplifications to scope the systems engineering model are as follows.

1. Scenario actors and profiles closely match the AFSIM variations through use of MBSE value properties.

2. Actions normally triggered by physics interaction (e.g. waypoint arrival, target presentation, fuel exhaustion) are triggered by user inputs.

3. Actions that can be abstracted to probability (e.g. bid creation, sensor function) are automatically simulated by the MBSE activities.

4. The "modular command and script" files used in the AFSIM model are similar in structure and function to the object oriented components of the MBSE model.

The above simplifications are designed to make the MBSE model coherent from a systems engineering standpoint while maintaining the capability to run MBSE simulations that test functional or structural changes in the NCAM system before changing the physics-based AFSIM code. The key to finding a connection to the AFSIM model is in simplification 4. The modular code provides an excellent single point of entry to control the AFSIM scenario via pre-processor variables in an initial configuration file called *variables.txt* further discussed in Hatzinger Section 3.3.4 [3]. The singular control point enables an MBSE user to output a single text file with configuration variables for use in the AFSIM scenario.

3.3 MBSE and Autonomy Design Method

Before autonomy can be considered, the mission profile for the NCAM system should be described. There are 5 key phases of the NCAM mission profile. First, the munition is in standby awaiting the fire command. Then, when fired, the munition initializes the subsystems and begins the ingress to the target area. After entering the target area, the munition begins searching for pre-determined target profiles. From search, a munition can become a leader if it identifies a potential target, or a follower if it is the highest bidder over the threshold for a target another munition found. Finally, the munition will either engage in attack of a confirmed target or self-destruct in flight when out of fuel. Each of these phases could be a state within an MBSE state machine diagram which contain activities that describe the behavior of the state. Importantly, a state machine must be owned by a block, which is in turn part of a structural hierarchy. So, it is possible to utilize state machines as a method to bridge the structure of a system to the behaviors of the system, as seen

in the Jacques model in Figure 5. This report will use a similar method of block structure, state machines, and activities to create an interactive simulation of the NCAM mission profile to aid in system comprehension.

To add autonomy to this mission profile, an architecture for autonomy needs to be fit within the structure of the NCAM MBSE model that reacts to inputs with a degree of self-governance as described by USD(R&E) [4]. The HAMR discussed in Section 2.4 is an open architecture intended to be modified to fit the application in a system [11]. For NCAM, the key components for autonomous decision making will be the deliberator and coordinator. The deliberator acts as the decision making agent that determines which phase of the mission profile an NCAM should be in, while the coordinator acts as the communication agent that handles the input and output of signals between NCAMs to enable autonomous cooperation. As for the sequencer and controller, this report is focused on the systems engineering level of the NCAM model; so these components of HAMR can be abstracted into the autopilot for flight controls, and/or the deliberator to reduce complexity. A weakness of the Cameo simulation toolkit is the potential for signals to disappear from an active simulation for unknown reasons. Thus, reduced complexity with fewer signal pass-throughs in simulation is desired.

3.4 NCAM Logical Structure

The logical structure of the NCAM system will be consistent with the AFSIM implementation while tying together the blocks and state machines similar to those presented in Jacques' model in Figure 5. With a middle-out approach to the logical structure, there is the key central system of NCAM, consisting of multiple subsystems, interacting with other systems within a domain or mission profile. The subsystems of NCAM will be populated in a collaborative effort with the AFSIM model to cover

the actions needed in the physics based simulation. Essential subsystems include the sensor subsystem, ordinance package, autonomy subsystem, and air vehicle. Each subsystem will be further expanded with components as identified for functionality of the munition and its simulation. For example, the autonomy subsystem will need components for the autopilot and the custom HAMR software agent. In terms of other systems in the domain, there will at least need to be a target system with different properties for target type, and a friendly aircraft system to deploy the NCAMs. Blocks should closely resemble platforms in the AFSIM model and value properties should convey identical variables to the AFSIM configurations. As described in Hatzinger Section 3.2.1 [3], pre-defined AFSIM platform types are correlated with NCAM implementations, which are then translated into MBSE blocks.

3.5 NCAM Behavior Model

An essential bridge between the logical structure and the behavior model will be state machines. Each state machine must be directly owned by a block within the domain and may contain a collection of states that indicate the instantaneous status of the block, known as a state of existence. A state may contain entry, during, and/or exit activities that initiate on entry, while active, or on exit from the state. There must be a logical flow of connections between the initial state and final state of a state machine. Changes in state are triggered by receiving a specified external signal, producing a specified internal signal, or completing all activities within the state. State machines to at least the subsystem level will act as the structure for the behaviors of the NCAM. The critical state machines for autonomy will be the deliberator and coordinator as described in Section 3.3. The connections between the states in the deliberator will act as the process to move between phases of the mission profile.

The behaviors themselves will be modeled using activities composed of actions and signals. Actions have broad application within MBSE to include text descriptions of something happening, mathematical opaque functions, branching decision nodes, splitting or joining forks for parallel logic, and reading value properties of blocks to name a few. These actions will be utilized in an object-oriented fashion to create a parallel behavior model to the AFSIM physics based simulation. Hatzinger Section 3.2.1 [3] describes communications and data transfer through messaging as a key to the behaviors of an NCAM. The MBSE application of these messages are signals, and each message will be accurately translated in the MBSE behavior model. Additionally, the confusion matrix used by the sensor in AFSIM to determine a target's identity can be replicated in MBSE activities through probability based decision nodes.

3.6 MBSE Simulation of NCAM

As described in the MBSE design method, the simulation will focus on a user interactive version of the AFSIM physics-based simulation. The main goal of this parallel simulation is to visually observe the consequences of changes to the NCAM design by observing the behaviors of the munitions in reaction to user input. Thus, a user interface must be developed with controls to influence the simulation environment and drive behaviors. Some controls that will need to be included in the user interface are individual waypoint arrivals, target encounters, and NCAM launch commands. Each of these controls directly influence critical events in the mission profile while maintaining autonomous decision making in each NCAM. A main reason to include the controls is the lack of a physics-based simulation engine within the Cameo simulation toolkit, as locations, velocities, and other physical interactions would have to be manually coded into the MBSE environment. Another option would be to abstract each manual control interaction into probability based actions; however, this research

is focused on creating a parallel systems engineering model of the AFSIM model, so minimizing unnecessary abstraction is desired. Additionally, the MBSE model acts as a systems engineering tool to help understand and communicate the NCAM design, so the MBSE simulation should have different systems engineering objectives from the AFSIM physics based simulation.

3.7 Automatic Design Parameter Transfer from MBSE to AFSIM

Figure 6 depicts the process for generating a document in the MBSE tool, Cameo Systems Modeler. The document generator utilizes template files coded in Velocity Template Language which can read all elements or properties in the MBSE model, then write specified items in the output file. Thus, a portion of this research is dedicated to configuring a working template for use in automatic report generation for the *variables.txt* file used in the AFSIM simulation as described in Section 3.1 of this report. The key automation to this document generation method is that each current value property equating to a configuration variable in the AFSIM code is parsed into a pre-formatted text file that can be immediately used in the AFSIM simulation.

Hatzinger also developed an alternative automatic conversion tool utilizing MBSE simulation with Jython code to run a concurrent AFSIM simulation. This method is discussed in Hatzinger Section 3.5 [3]. Chapter 4 of this report will discuss the advantages of each approach to address research question 4.

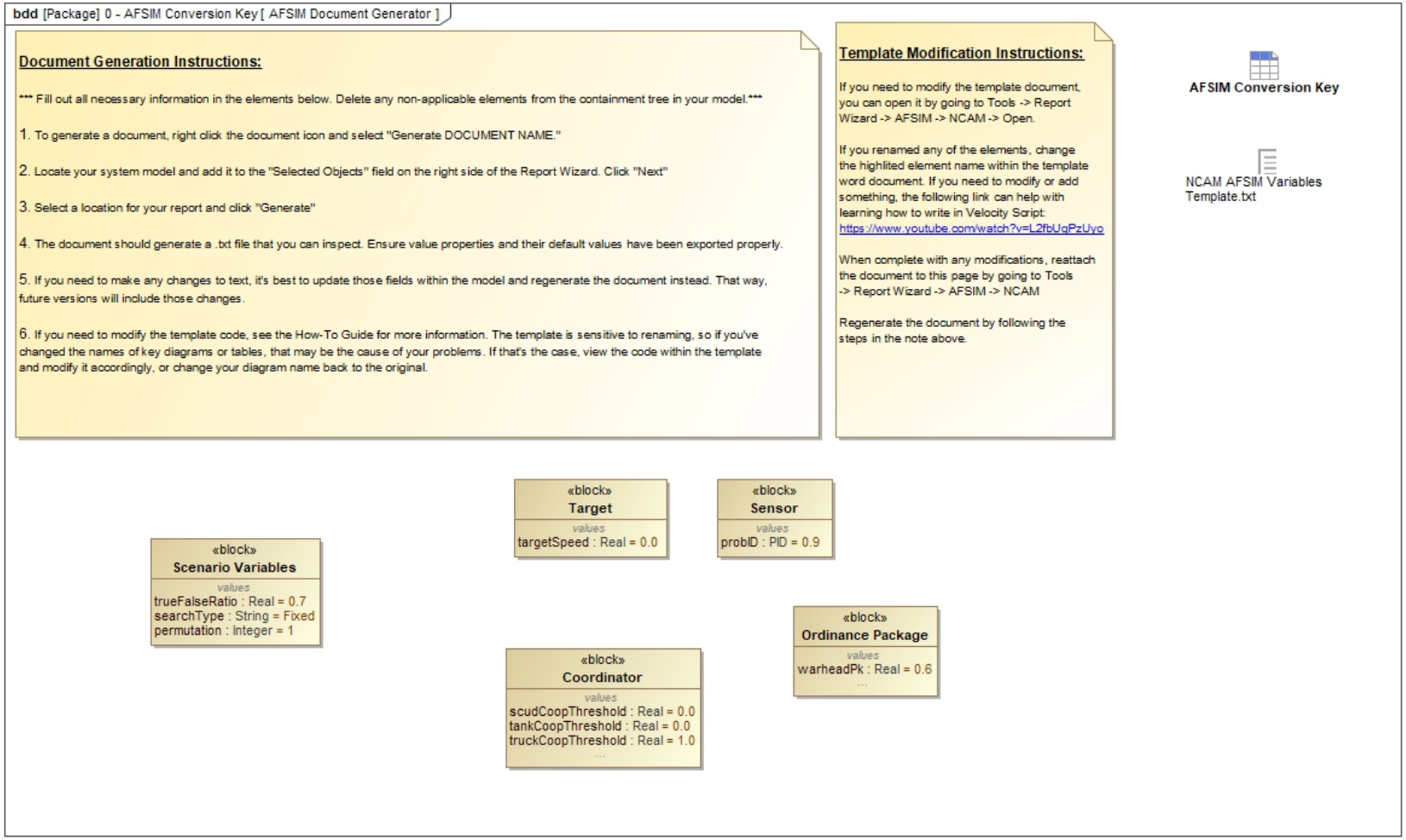

Figure 6. AFSIM Document Generation Process Diagram

IV. Analysis and Results

4.1 Overview

This chapter discusses the completed MBSE model of the NCAM system and compares the results to the physics based, parallel model in AFSIM. First, the MBSE NCAM model is described at a diagram/containment tree level. These elements are then compared to the AFSIM raw code to verify identical design. Next, the MBSE NCAM simulations are presented and compared to the behaviors of the AFSIM simulations. Elements of the MBSE model and simulation are then assessed for translation into the AFSIM environment, followed by results of the translations. Finally, an overall assessment is presented for the value of creating parallel models in separate platforms. For the purposes of this section, model elements will use the following text formatting: ***State***, "Activity", *signal/message*.

4.2 MBSE Diagrams vs AFSIM Code

The first step in designing the NCAM MBSE model was to create a logical structure that organizes the main actors in the mission profile within a hierarchy. This hierarchy is depicted in the NCAM Structure Block Definition Diagram (BDD) in Figure 7. The top level is the NCAM domain, which splits into three main systems. First is the NCAM system, comprised of an autonomy subsystem, air vehicle, sensor subsystem, and ordinance package. Each subsystem can be decomposed further into working components as necessary. For example, the autonomy subsystem is decomposed into multiple layers of components due to its complexity. The coordinator and deliberator are present from the HAMR and exist as the main parts of the software agent. The value properties within the NCAM system decomposition are critical for defining behaviors described in later sections, and act as a potential point of transla-

tion into AFSIM.

The other two systems within the NCAM domain are the environment and the launch aircraft. The environment currently only contains the NCAM target but is designed to be extensible if additional realism features like ground search area are desired. This report simplified the environment to only include the target because the physics based model inherently contains realism features desired for a physical environment in simulation. The launch aircraft is also designed for simplicity to start, with potential for future expansion to change the mission loadouts depending on mission vehicle. Each top level system performs very simple tasks to help in the future simulation, such as controlling the launch command for all munitions and presenting a type of target to an NCAM. The majority of complexity in both structure and behavior belongs to the NCAM system and in particular the software agent.

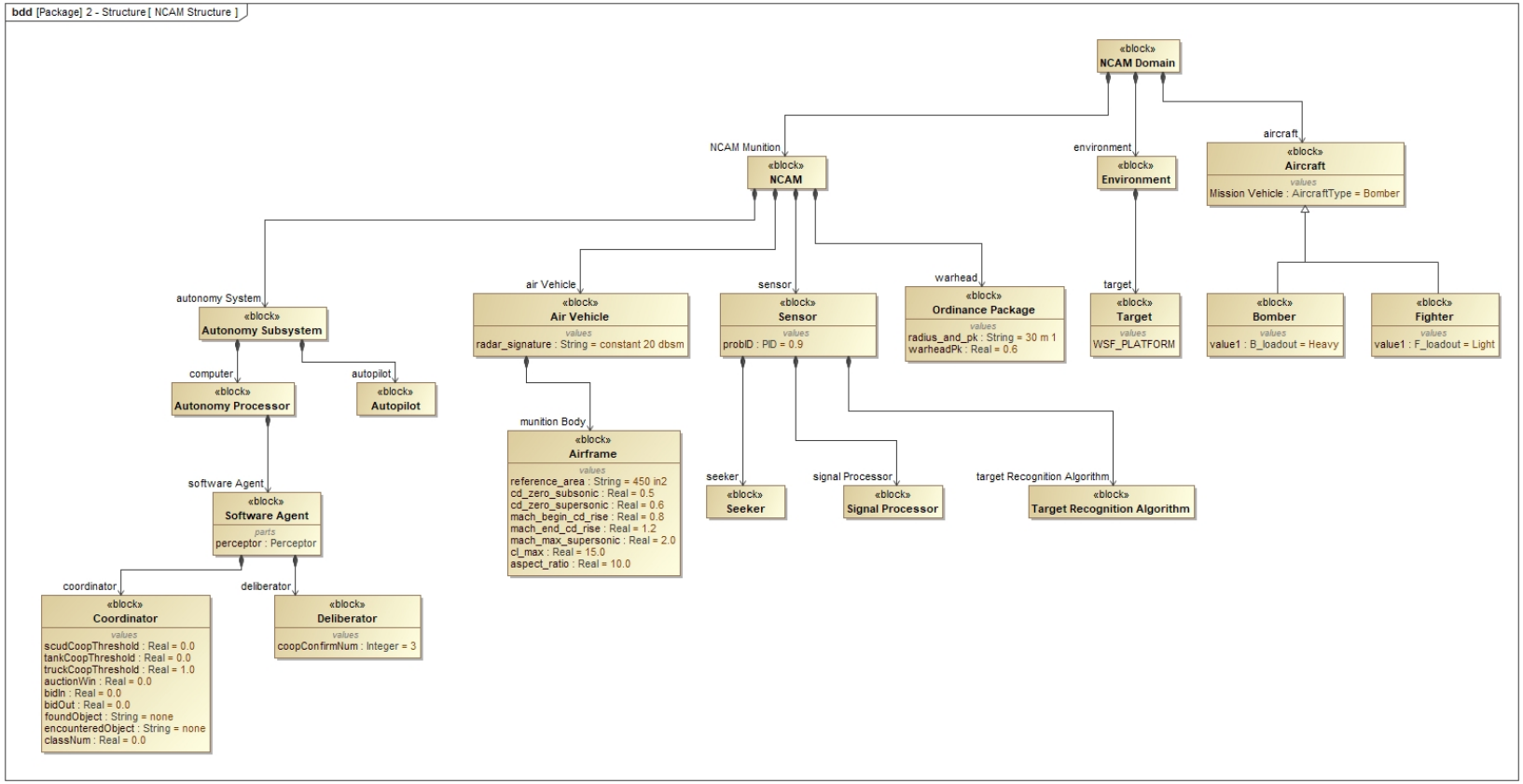

Figure 7. NCAM Logical Structure

The NCAM block from Figure 7 has an intricate internal structure presented in the NCAM Internal Block Diagram (IBD) in Figure 8. The IBD presents each of the components of the NCAM system in layers that match the decomposition from Figure 7 but includes additional information in the form or ports, connectors, and embedded state machine diagrams. The embedded state machine diagrams are individually included for reference in Appendix A. The ports in the IBD act as logical connection points between the components of the NCAM with connectors that enable signals to flow between the components. Three ports are located on the exterior of the diagram, which act as connection points for the NCAM to external systems; for instance, the aircraft system, or the ground target, or another NCAM. The reason these ports and signals are logical is due to the level of the systems engineering model for the NCAM. Realistically, the port connecting the coordinator to the exterior boundary of the NCAM represents a wireless communication system that would enable the NCAM to communicate in flight with other NCAMs. Similarly, the port connecting the autonomy system to the exterior boundary represents a hard point for physical data communication connection to the aircraft. However, the level of fidelity in this NCAM model is intended to communicate that a connection exists, rather than specifying the exact type of connection.

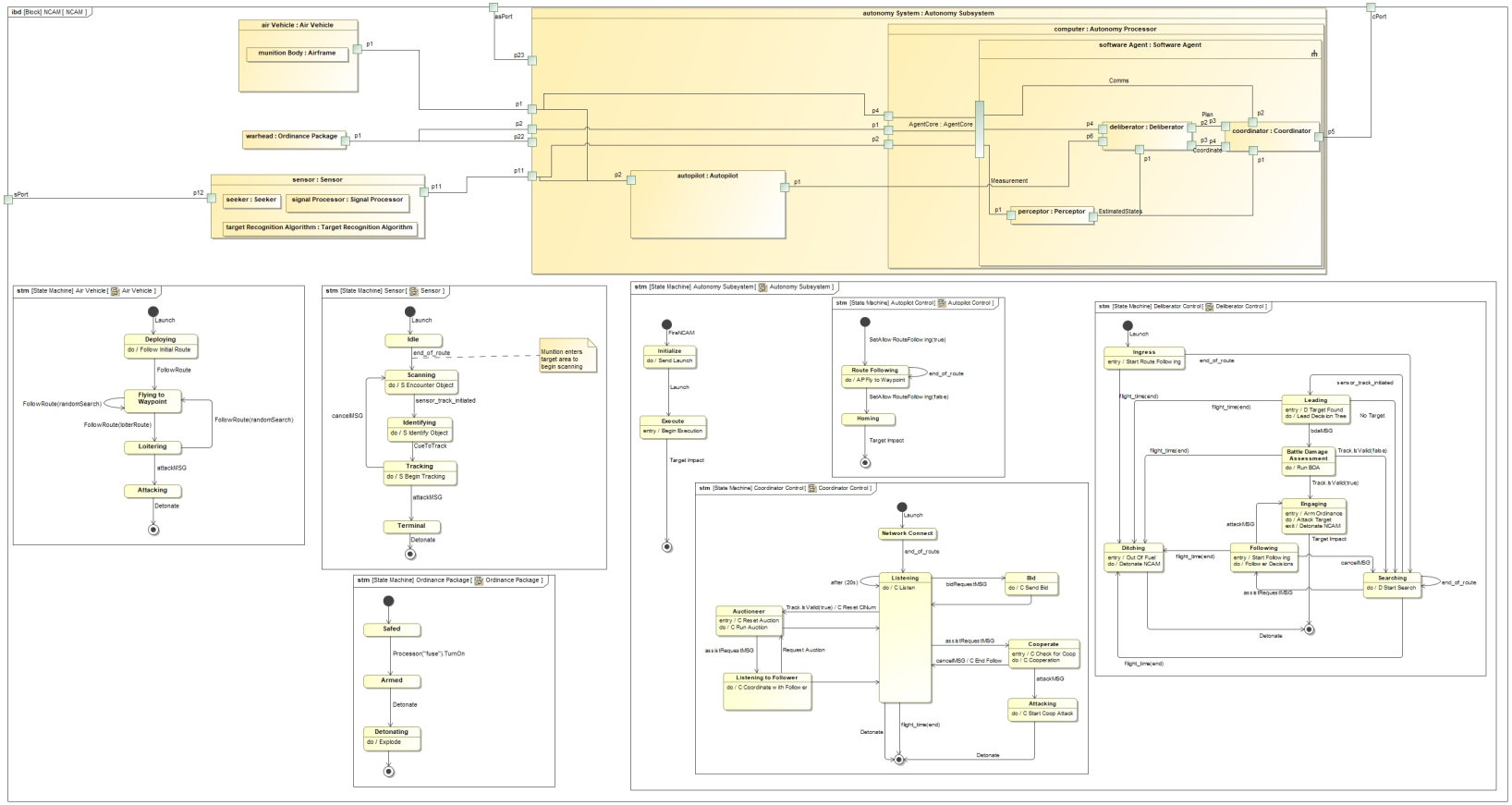

Figure 8. NCAM Internal Logical Structure

The state machine diagrams embedded into the NCAM IBD in Figure 8 represent a logical structure to the behavior of an NCAM. Each NCAM subsystem and autonomy component has a flow of states that determine what is happening in the system at any given time. For example, immediately after launching an NCAM, the air vehicle begins deploying aerodynamic surfaces and propulsion to direct itself toward the known target area. Meanwhile, the autonomy subsystem is simultaneously initializing each system, connecting to the other NCAMs in flight, and directing the autopilot to the first waypoint. The air vehicle knows where to go because the autopilot control has engaged route following and begins sending control signals to the air vehicle through the p2 port on the autopilot. Additionally, the sensor subsystem is in an idle state awaiting arrival at the first waypoint, indicating entry into the search area. All of these states and activities are occurring in harmony to give the NCAM a complex behavior profile.

4.2.1 Deliberator

The keys to the highly complex behavior expected with autonomy in the NCAM system are the deliberator and coordinator. This subsection presents the behaviors of the deliberator, while the following subsection describes the coordinator. The main function of the deliberator within the NCAM model is to plan and control the overarching behaviors of the NCAM throughout the mission. From Figure 8, the deliberator has five logical connections. In order, p1 is the connection to the perceptor which interprets the raw status data from each other subsystem and provides it to each agent core of the HAMR. Next, p2 is the connection to the coordinator where signals from the coordinator are received for planning. Port p3 is the connection to the coordinator where signals are sent from the deliberator for coordination. These ports are separate for easier tracking of signals; however, a single bi-directional port could

be used instead. Port p4 is the connection to the agent core data bus where signals are sent and received by all other subsystems, and p6 is an abstracted connection to the autopilot where flight sequencing and controlling occurs. Each of these connections carry information in the form of signals to other components within an NCAM.

The deliberator determines which signals to send through the use of activities in a state machine structure. The deliberator state machine diagram shown in Figure 9 is the same state machine embedded on the bottom right of Figure 8. Each state in Figure 9 contains a set of instructions for which actions to take and which signals to send in the form of either entry activities, during activities, or exit activities. The deliberator state ***Leading*** can only be attained by leaving the ***Searching*** state triggered by the deliberator receiving a *sensor_ track_initiated* signal created by the sensor subsystem and received on p4 of the deliberator. Upon entry into the ***Leading*** state, an activity is initiated called "D Target Found" which starts a series of command signals to the other components to begin loitering over the identified target. The activity with signal commands are presented in Figure 10, with a loiter signal sent out though p4 to reach the air vehicle subsystem, and a track signal sent out though p3 to the collaborator so it can begin requesting assistance from other NCAMs. After completing the "D Target Found" activity, the deliberator will then begin operating in the ***Leading*** state.

During the ***Leading*** state, the "D Lead Decision Tree" activity will begin running. This activity is shown in Figure 11 where the deliberator logic splits into three potential courses of action pending specific signal reception. These three paths equate to yellow "search" or "attack" action bubbles from the mission profile decision tree in Figure 3. Each logic path has actions to help explain what happens when a signal sent by the coordinator indicates the path to be taken. On the left, the target could not be confirmed so all munitions return to the search state. In the middle, the lead

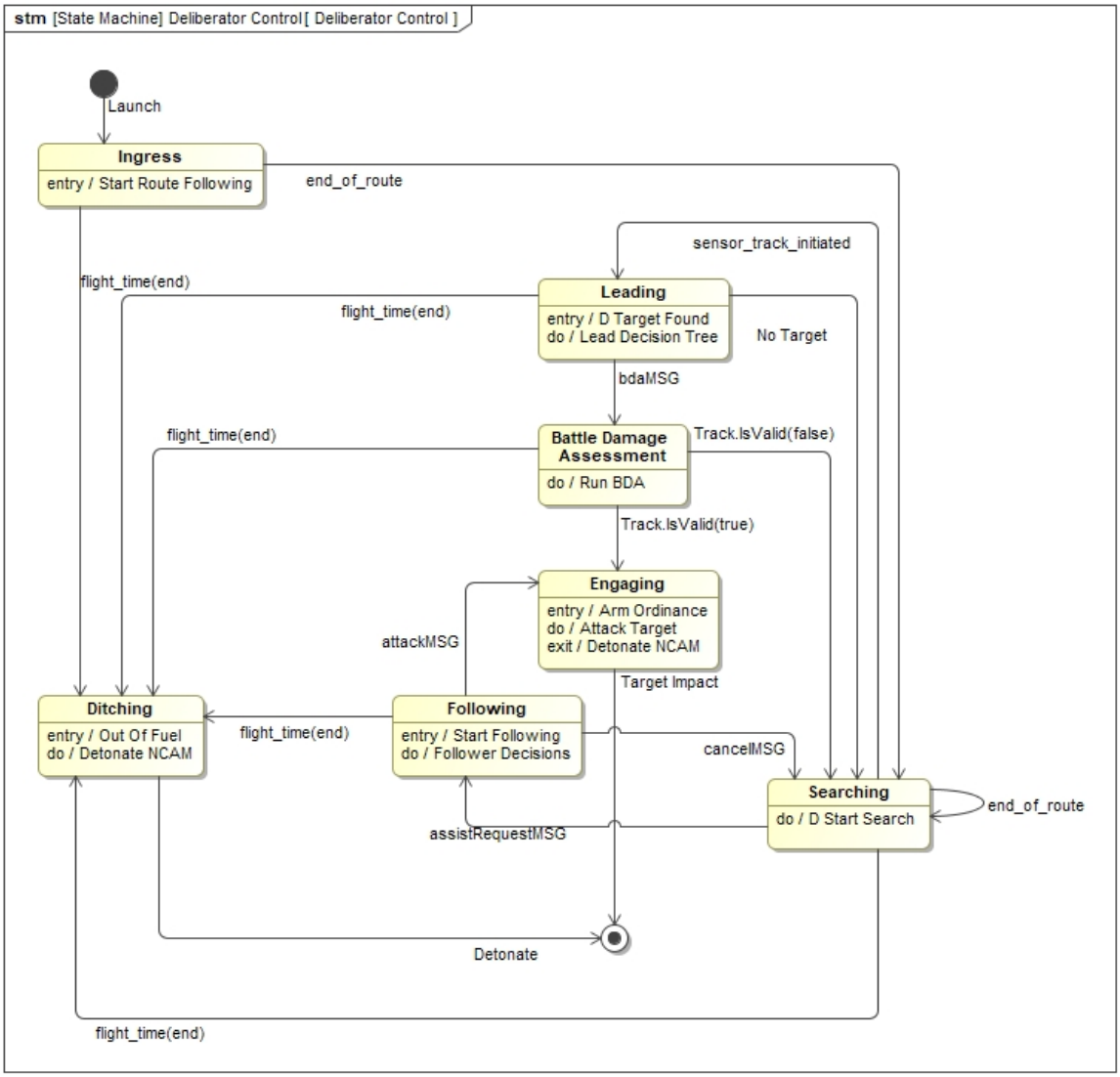

Figure 9. NCAM Deliberator State Machine Diagram

munition has confirmed the target is valid through help from another NCAM, thus it sends an attack message to the follower munition and sends a signal to itself to switch the deliberator state to begin battle damage assessment. On the right, no other munition assisted in identifying the target (the type of target and outcome is factored into the coordinator logic), so the lead munition begins an attack on the target.

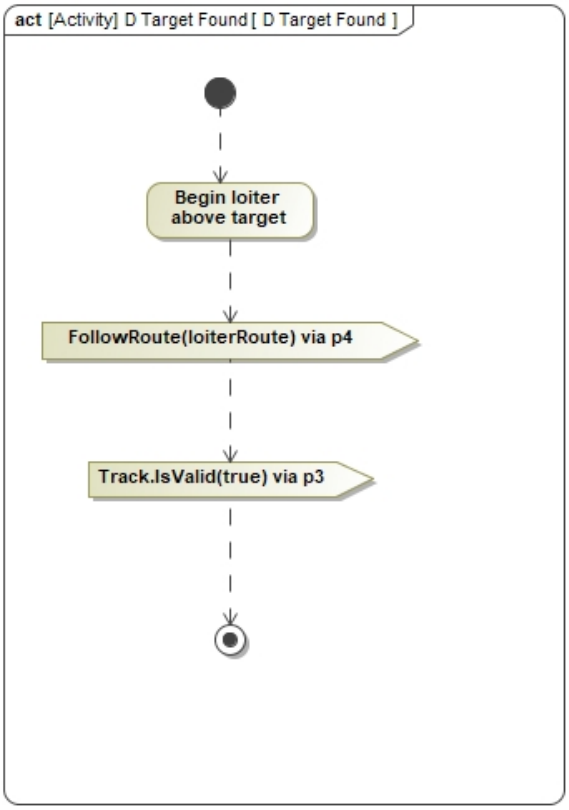

Figure 10. NCAM Lead Munition Identifies a Target Activity Diagram

The deliberator state machine presented in Figure 9 can be directly compared to the deliberator state machine developed by Hatzinger [3] which visually describes the AFSIM code for the deliberator. Hatzinger's deliberator state machine is presented in Figure 12 and contains many more states than the MBSE implemented deliberator. Most of the variations are due to the use of activities within the MBSE model and key states are present in both state machines. The ***Ingress, Searching, Leading, Following,*** and ***Ditching*** states are present in both diagrams and both diagrams start with a launch then end with either a ditch or attack. The main difference between the configurations is the location of the auction state/activity. Within the AFSIM code in Hazinger Appendix A.43 and A.45 [3], the deliberator runs the auction

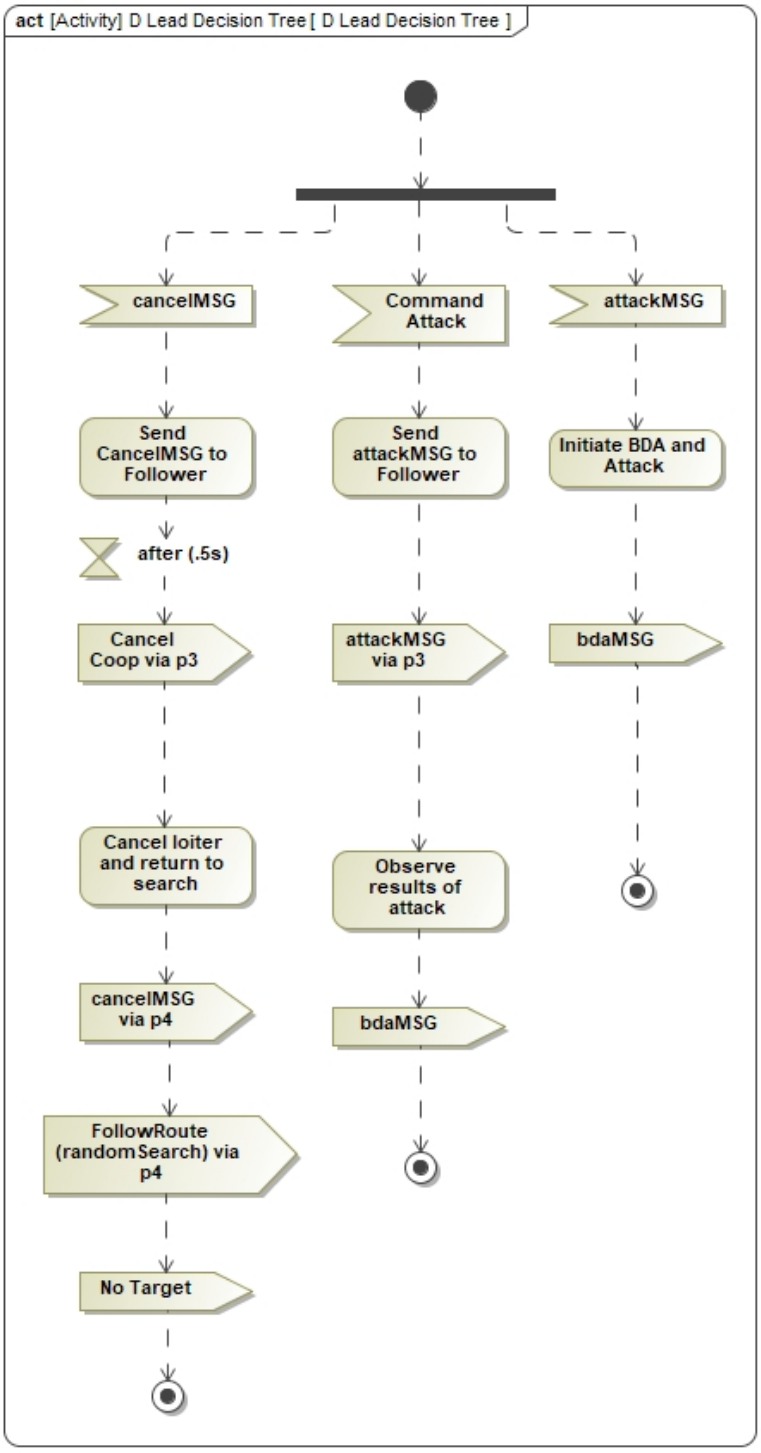

Figure 11. NCAM Lead Munition Decision Tree Activity Diagram

and requests then receives bid data from other NCAMs via the coordinator. In the MBSE model, the coordinator runs the auction as described in the following subsection, directly communicating with other NCAMs, then feeds auction outcomes internally to the deliberator. This difference in design was implemented due to the signal losses in the MBSE simulation as described in Section 3 of this report. While the deliberator design may be slightly different, the NCAM mission behaviors are essentially identical.

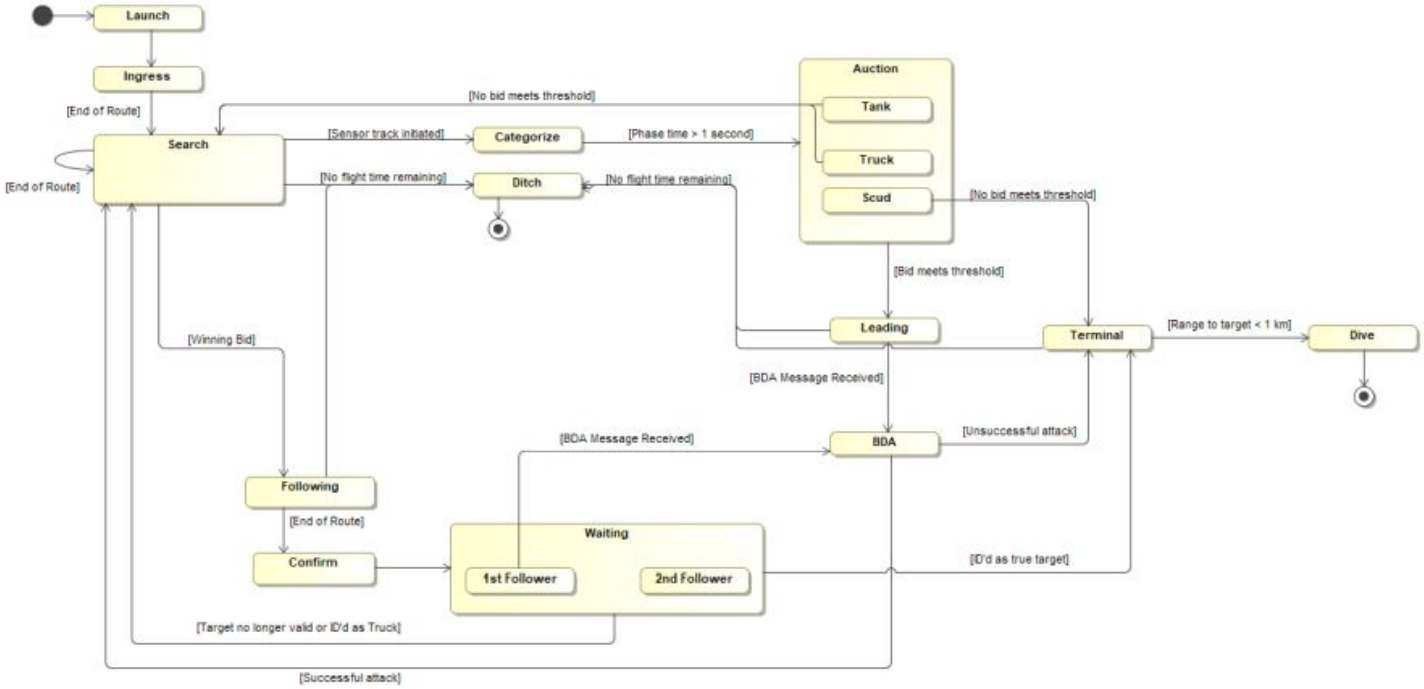

Figure 12. NCAM Deliberator State Machine Diagram for AFSIM Code developed by Hatzinger [3]

4.2.2 Coordinator

The coordinator is configured with five logical ports shown in Figure 8. P1 is connected to the perceptor, p2 is connected to the agent core bus for communication to other internal subsystems, p3 is connected to the deliberator for coordinator outputs used for planning, p4 is connected to the deliberator for deliberator outputs requesting coordination, and p5 is the external communications port that links to other NCAM coordinators. The signals that flow though these ports are controlled by activities within the coordinator control state machine diagram in Figure 13. These states are set up in a similar manner to the deliberator where activities are in sequence and states mostly change by specified signals. However, some state changes occur in the coordinator when all activities in the state are complete, like in the ***Bid*** state, where the coordinator switches back to listening after finishing the "C Send Bid" activity.

As mentioned in the previous subsection, the MBSE NCAM coordinator handles the auction process differently from the AFSIM model. However, both models start with the same request for assistance sent to all other NCAMs in range. An advantage of the AFSIM simulation environment is the ability to send messages to specific platforms by identification number after it is identified as the cooperative munition. This individual message function would require a coordinator with an individual identification system manually built into the MBSE model to function identically to AFSIM (this idea is further explained in Section 5.3 for future work), so simplifications were made in the MBSE NCAM model specifically for the auction process. The MBSE designed auction process is more of a signal blast where signals are always sent between all NCAM coordinators but only interpreted by munitions that are prepared to receive the signals. The key signal to potentially initiate cooperative behavior is the *bidRequestMSG* sent by another NCAM. So, the MBSE coordinator model needed some inherent safeguards to maintain a behavior profile identical to the

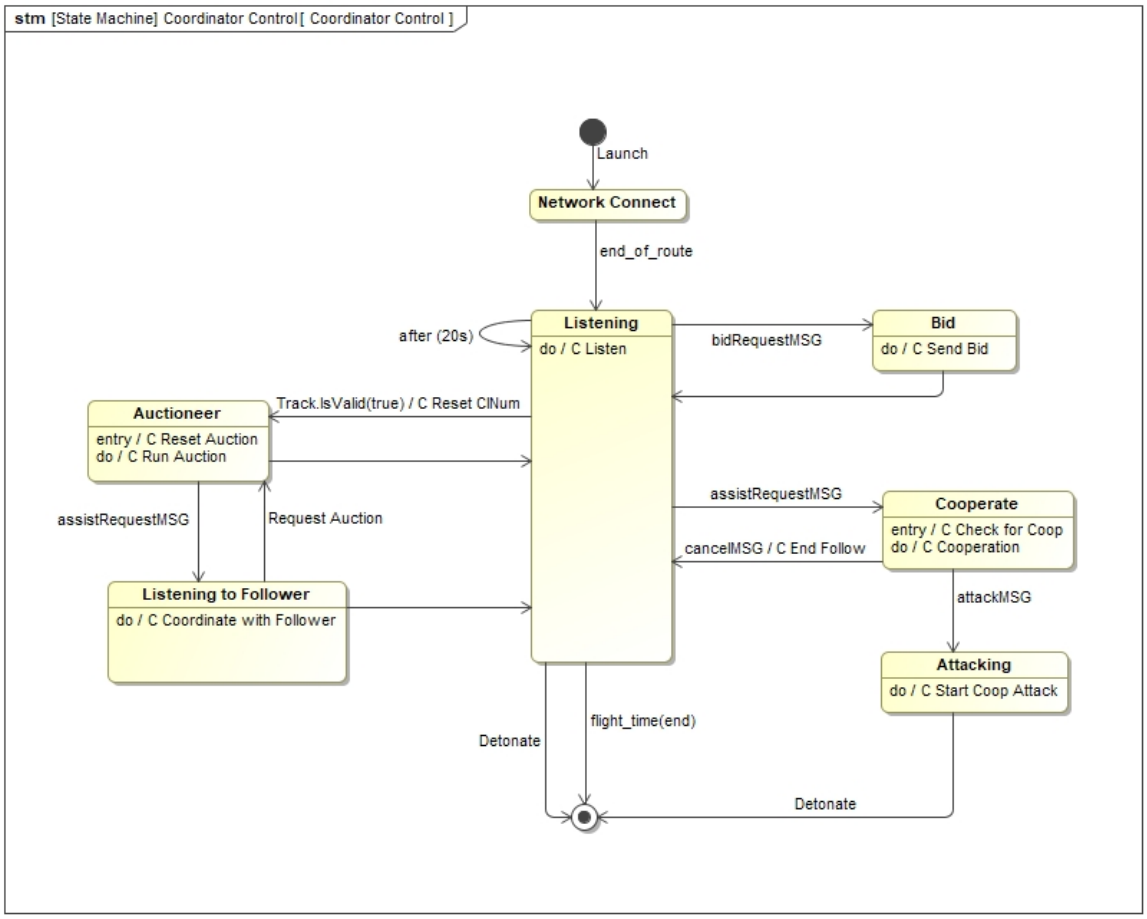

Figure 13. NCAM Coordinator State Machine Diagram

AFSIM model.

The reason the *bidRequestMSG* is so important is the combination of the "C Send Bid" activity shown in Figure 14 and the "C Check for Coop" activity shown in Figure 15. An NCAM that enters the ***Bid*** state by receiving the *bidRequestMSG* will create and store a random bid value "bidOut" between 0 and 1 that acts as a pseudo identification number. This random bid value is an abstraction of the bid values calculated in AFSIM based on physical location and fuel remaining as described in Hatzinger Section 3.2 [3]. After an auction is complete, the winning bid value is sent to all other NCAM coordinators where any coordinator in the ***Listening*** state

will move into the ***Cooperate*** state and immediately run the "C Check for Coop" activity. Within that activity, the winning bid value is compared to the bidOut that the coordinator internally stored. If the value matches, the coordinator knows it is supposed to be the follower and continues cooperating; if the value does not match, the coordinator cancels cooperating and immediately returns to the ***Listening*** state. Thus, a coordinator that does not send a bid cannot falsely win an auction and break the mission profile logic.

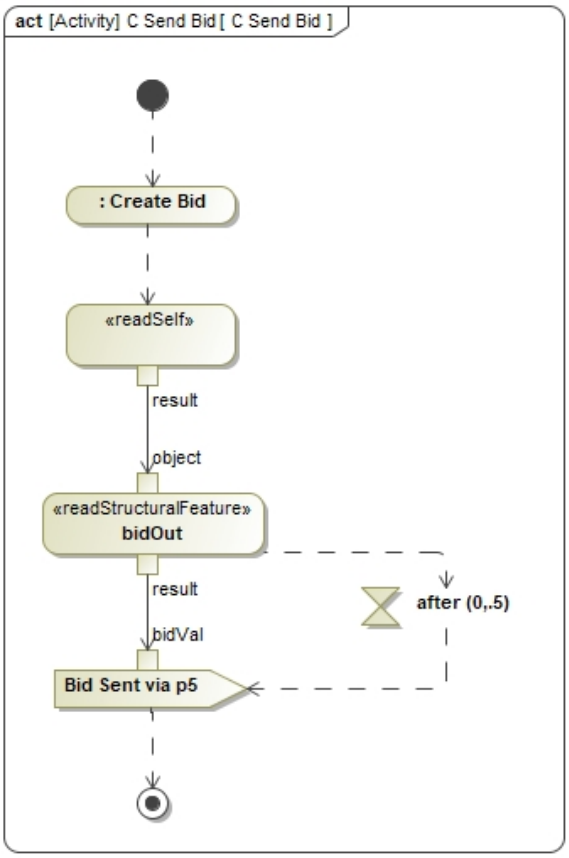

Figure 14. NCAM Follower Munition Send Bid Activity Diagram

The ***Listening*** state will run the "C Listen" activity while waiting for any of the specified signals to break into another state. The "C Listen" activity is shown in Figure 22 where the coordinator is continuously waiting for signals from the sensor

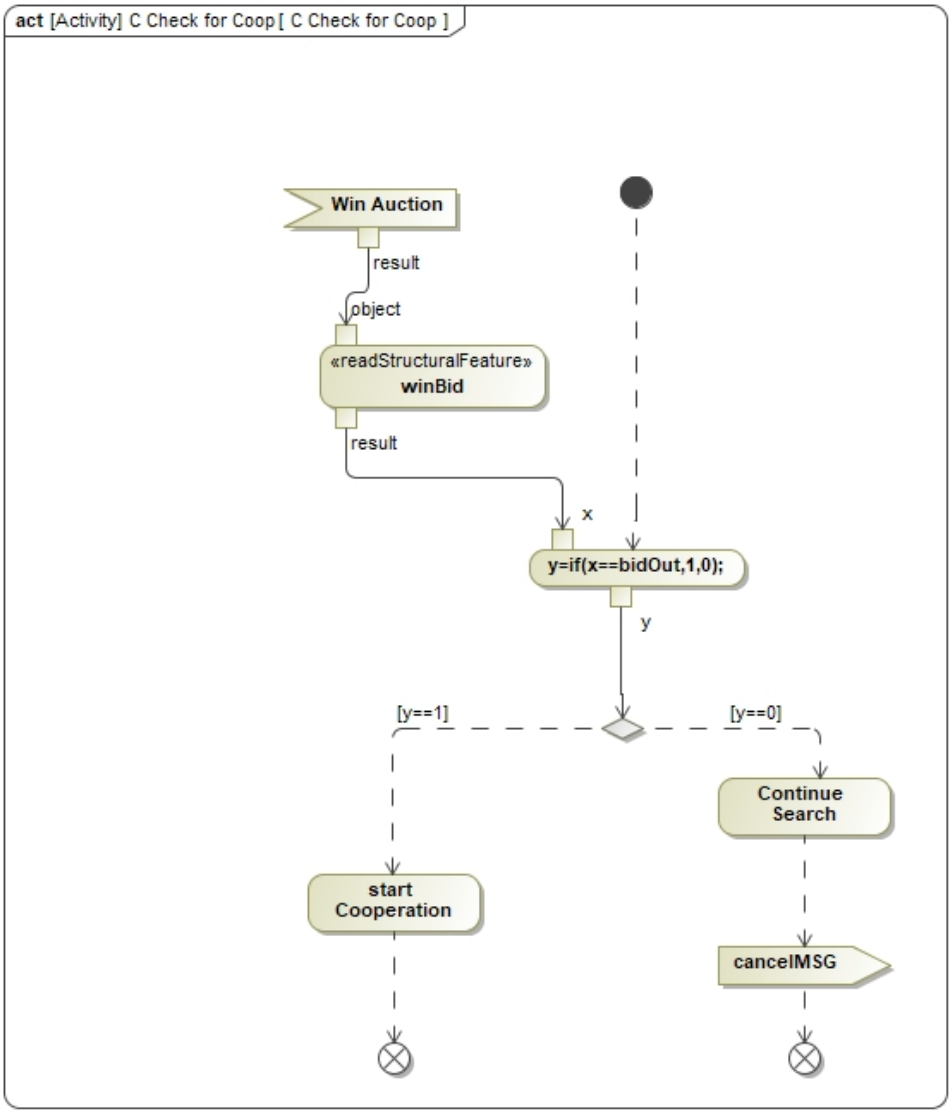

Figure 15. NCAM Follower Munition Check for Cooperation Activity Diagram

subsystem containing data for which target was encountered and identified. The data is then translated and stored as value properties for use in decisions in other states. Interestingly, this activity has no end point but the simulation logic has no issues due to the state machine structure; activities automatically end when the component is no longer in the host state.

The first coordinator safeguard is the ***Network Connect*** state. In this state, an

NCAM will not accept communication from other NCAMs in the MBSE simulation until it has reached the first waypoint and moves into the ***Listening*** state, indicating entry into the target area. The next safeguard is the split in coordinator states where lead munitions will enter the ***Auctioneer*** or ***Listening to Follower*** on the left of Figure 13 where it cannot change to the *Bid* state. Munitions that are followers will be in the ***Cooperate*** or ***Attacking*** state on the right of Figure 13 if they win an auction where, again, it cannot accept a *bidRequestMSG*. Essentially, the coordinator must be in the ***Listening*** state to accept the *bidRequestMSG* sent by a lead munition.

An auction is triggered within the coordinator while in the ***Listening*** state by a *Track.Is.Valid(true)* signal from the deliberator which means the munition has identified a potential target. The first action taken is the "C Reset CINum" activity on the connector between states, which simply resets the stored classification number value property in the coordinator to 1 in case the munition had previously been an auctioneer. Then, on entry to the ***Auctioneer*** state, the coordinator resets the auction value properties to defaults with the "C Reset Auction" activity. Next, the coordinator begins running the "C Run Auction" activity shown in Figure 16. The auction activity is loaded with many complex actions but can be simplified to four main sections. The activity starts with a logic fork to run a pair of parallel tasks. The top left of Figure 16 sends a bid request to all other NCAMs and starts a 10 second wait for responses. The other side of the fork on the top right of Figure 16 is a loop that reads each bid sent by other NCAMs, then compares received bid values to the stored "auctionWin" bid value (-1 is default so any bid received will win) and stores the greater bid value as "auctionWin". The middle section of Figure 3 is a check of the winning bid value against cooperation thresholds, and the bottom section of Figure 16 indicates the three potential outcomes of an auction. The outcomes of this check match the decision tree in Figure 3 where a winning bid that is lower than

the cooperation threshold results in a "y" value of 0, which could mean the NCAM reverts to searching or immediately attacks if the target was identified as a SCUD or Tank then Tank/SCUD as prescribed by the decision tree. If the winning bid is greater than the cooperation threshold, a "y" value of 1 is assigned which means the NCAM will send an assist request signal with the winning bid value attached. The assist request signal triggers a state change in the coordinator to the ***Listening to Follower*** state; otherwise, the coordinator will return to the ***Listening*** state.

The ***Listening to Follower*** state is a lead munition specialized coordinator state that executes the "C Coordinate with Follower" activity presented in Figure 19. The purpose of this activity is to inform the follower munition of which target type the lead munition encountered, check which classification number from the decision tree in Figure 3 the lead munition coordinator is completing, then read the stored target identification and listen for what the follower identifies the target as, then ultimately send command signals that follow the end results from the decision tree. The "C Coordinate with Follower" activity utilizes forks and joins to accomplish the various combinations of lead identified target and follower identified targets to determine the correct outcome to match the AFSIM NCAM simulation. In the case of a third classification, a slight difference in the MBSE model design from AFSIM is that both follower munitions will attack a Tank/SCUD simultaneously due to the lack of individual messaging in the MBSE model.

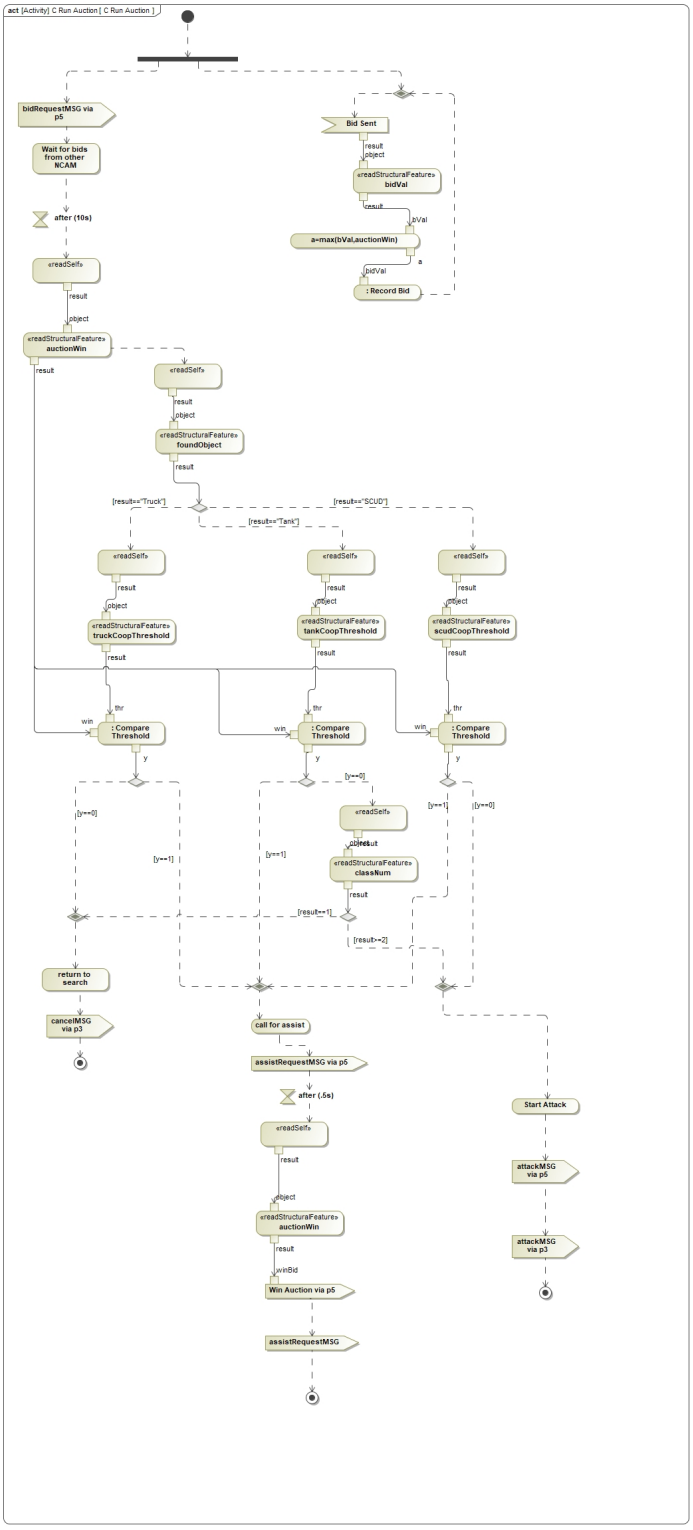

Figure 16. NCAM Lead Munition Auction Activity Diagram

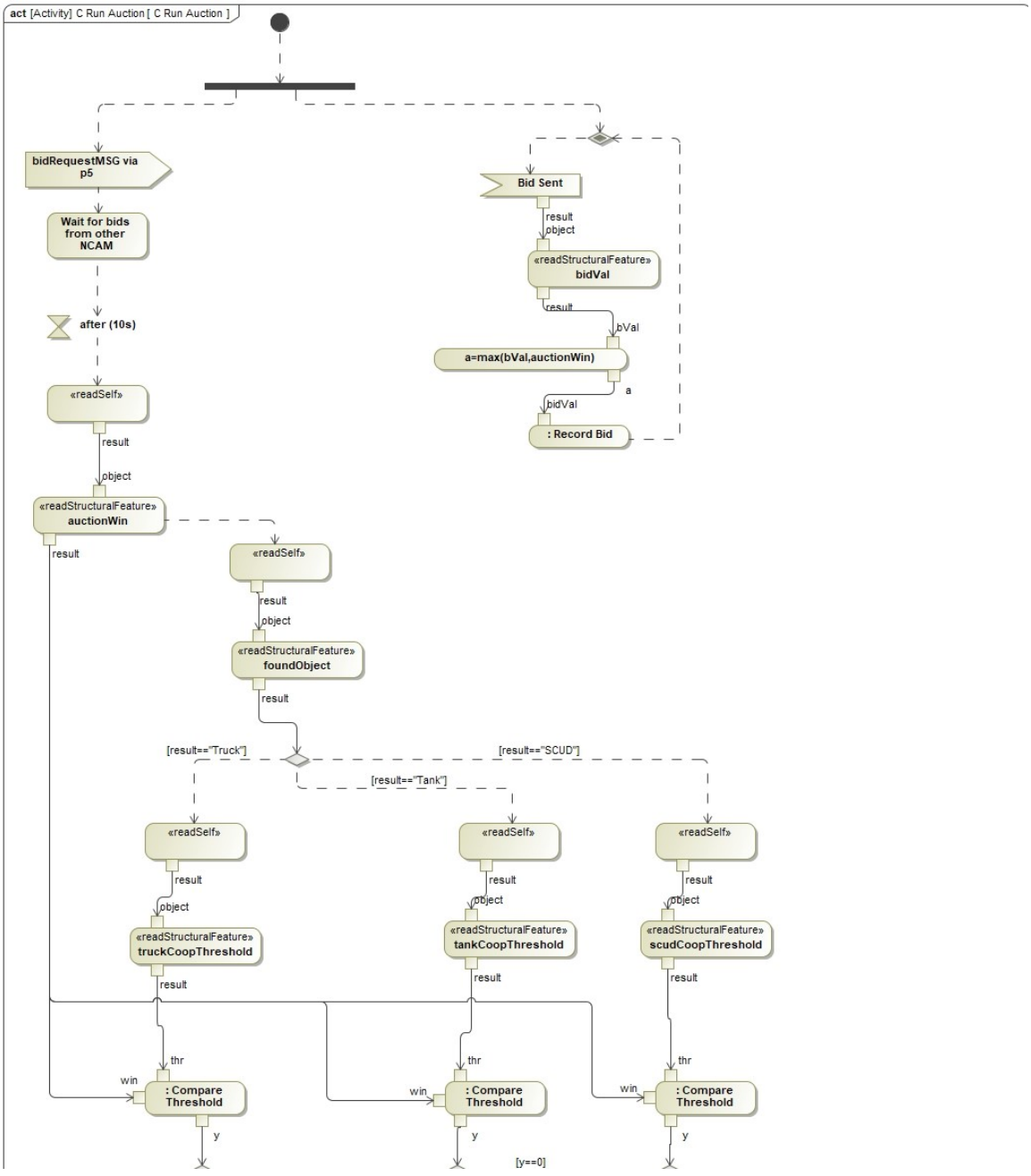

Figure 17. NCAM Lead Munition Auction Activity Diagram (upper)

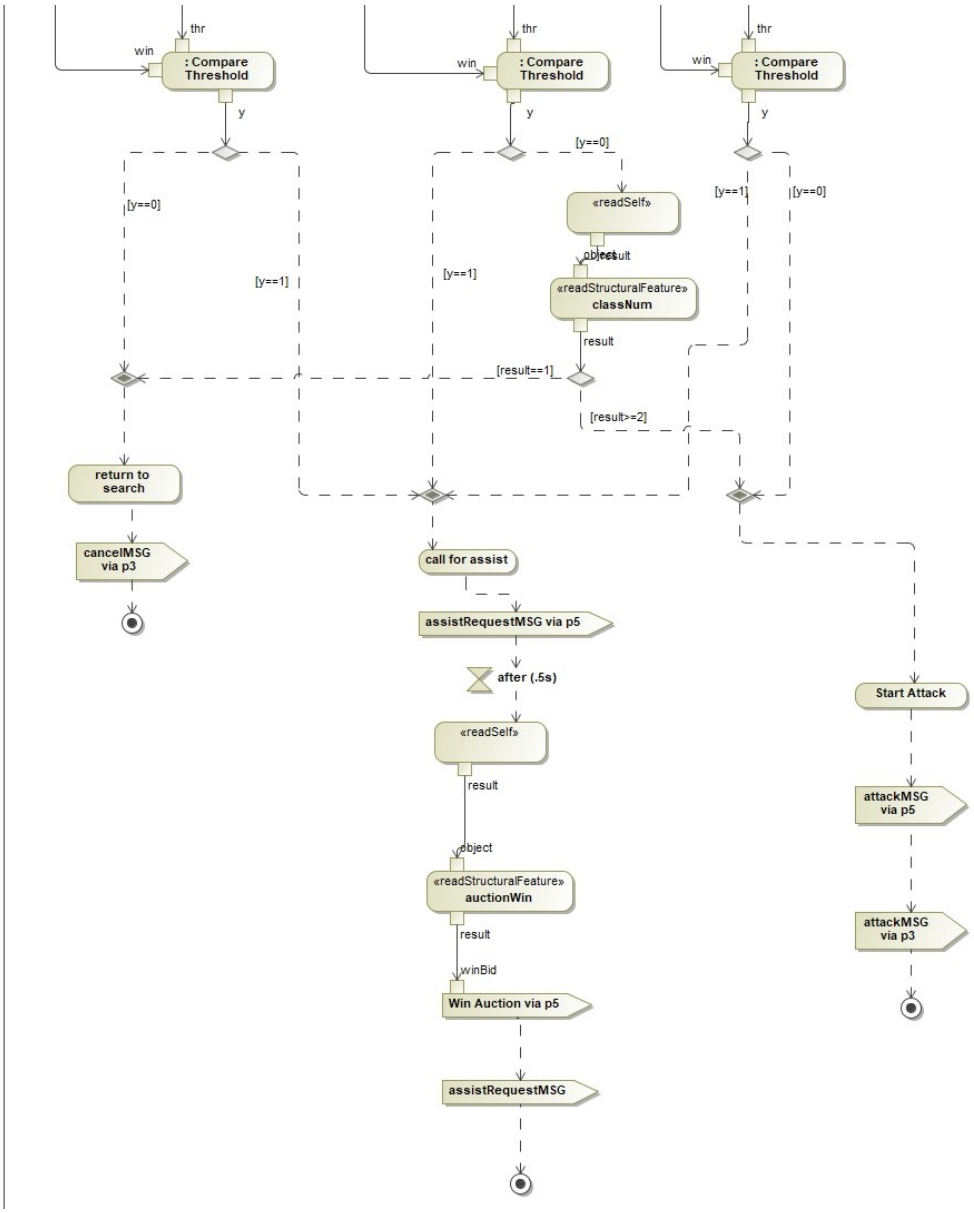

Figure 18. NCAM Lead Munition Auction Activity Diagram (lower)

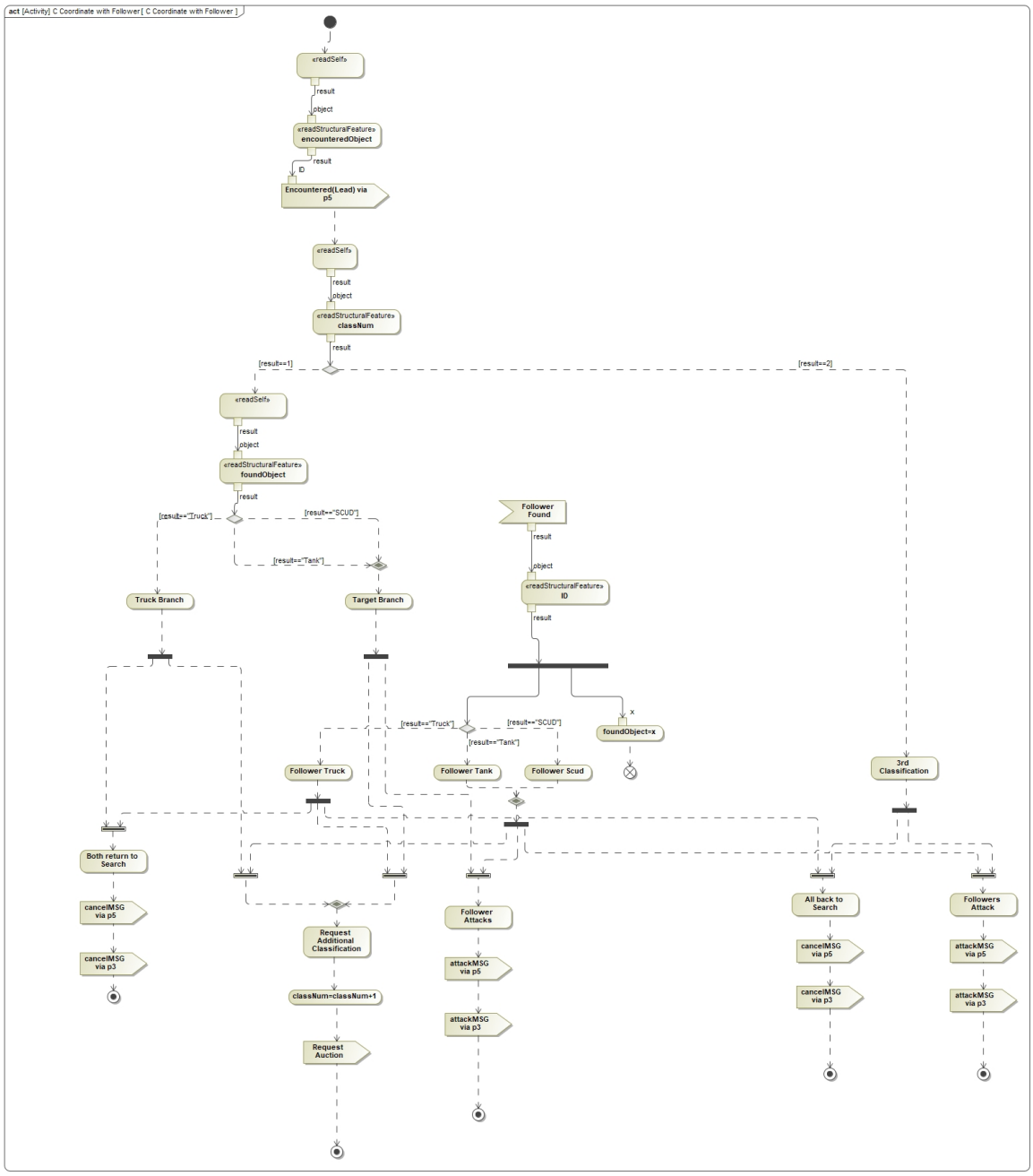

Figure 19. NCAM Lead Munition Coordinate with Follower Activity Diagram

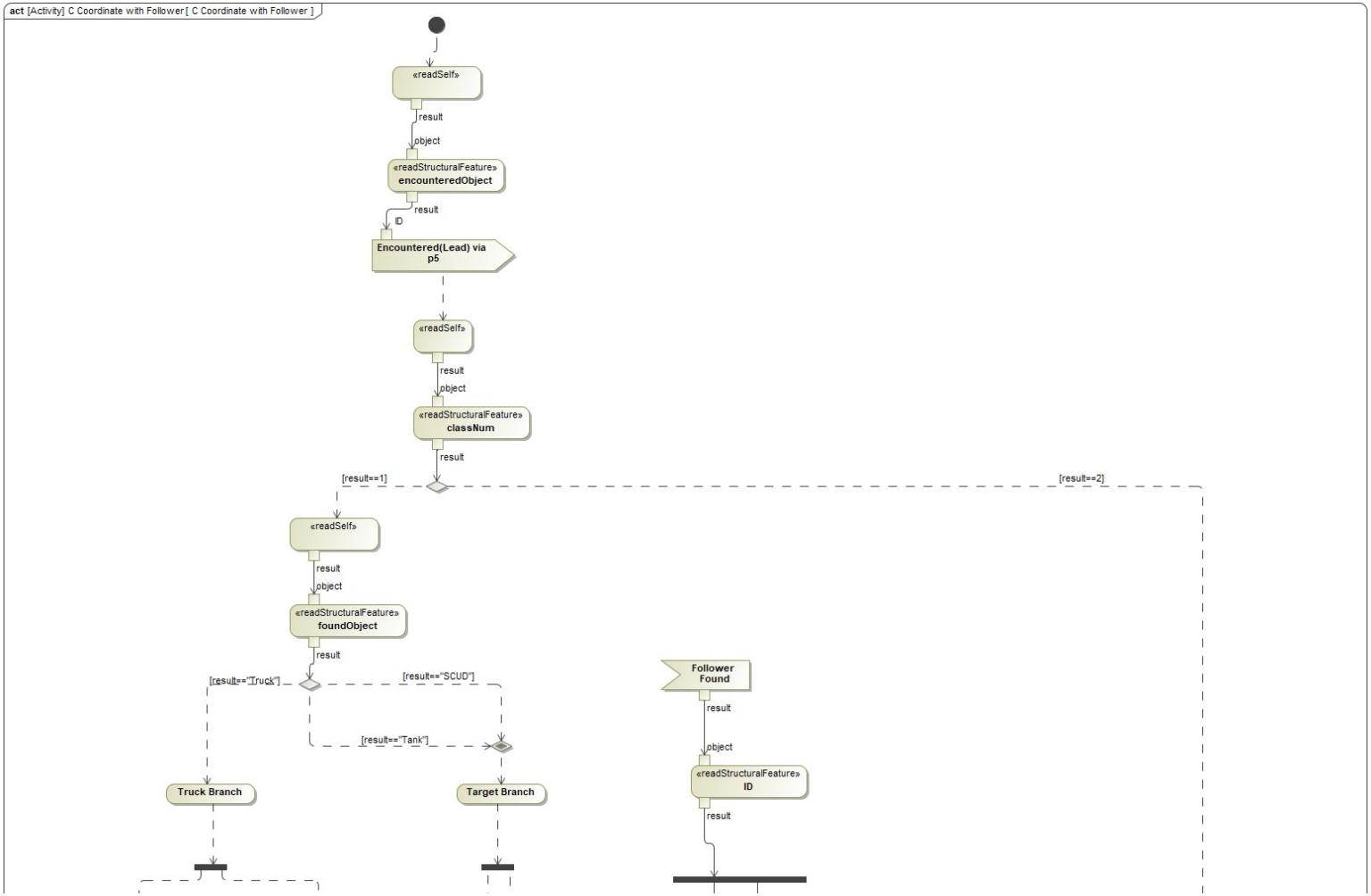

Figure 20. NCAM Lead Munition Coordinate with Follower Activity Diagram (upper)

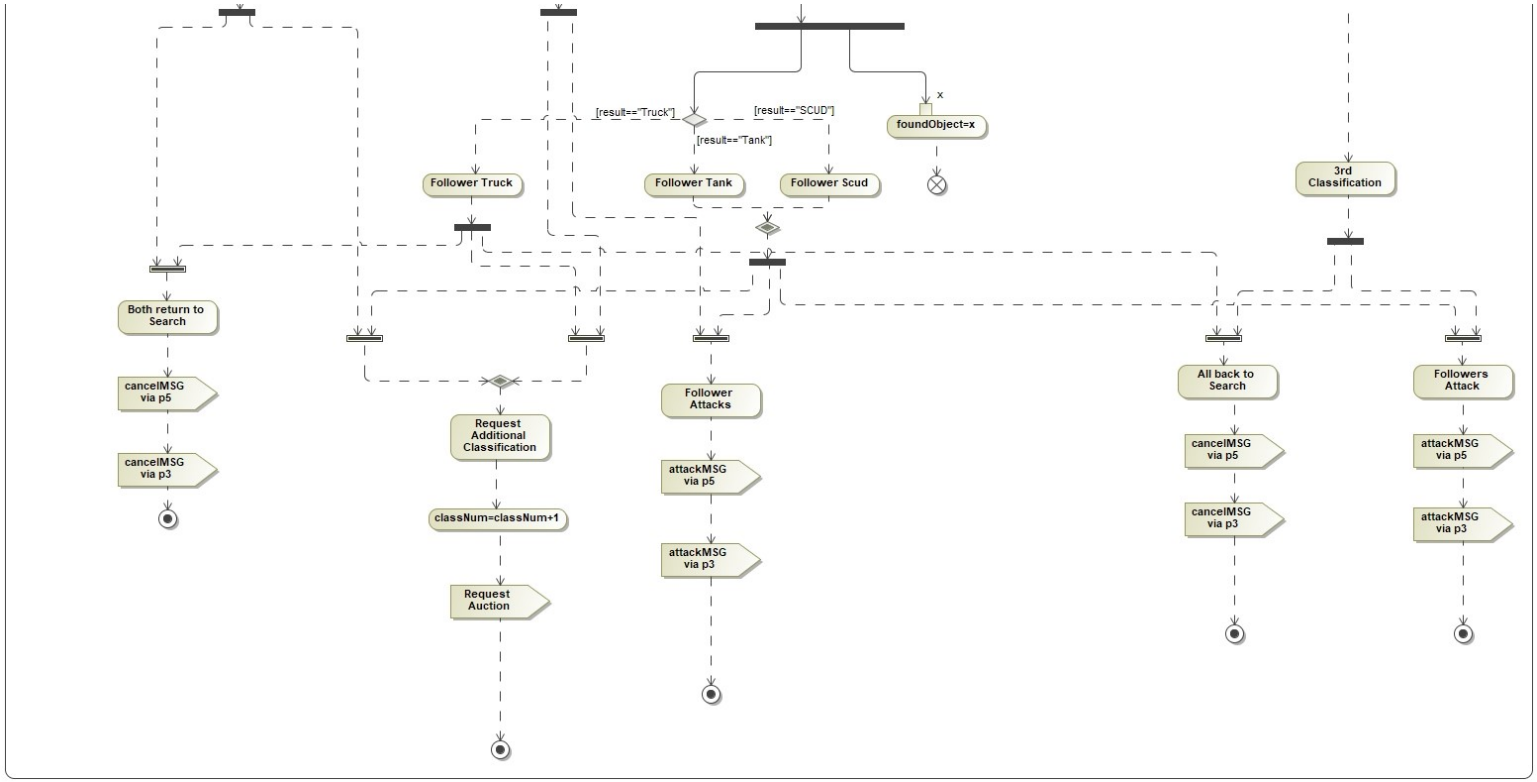

Figure 21. NCAM Lead Munition Coordinate with Follower Activity Diagram (lower)

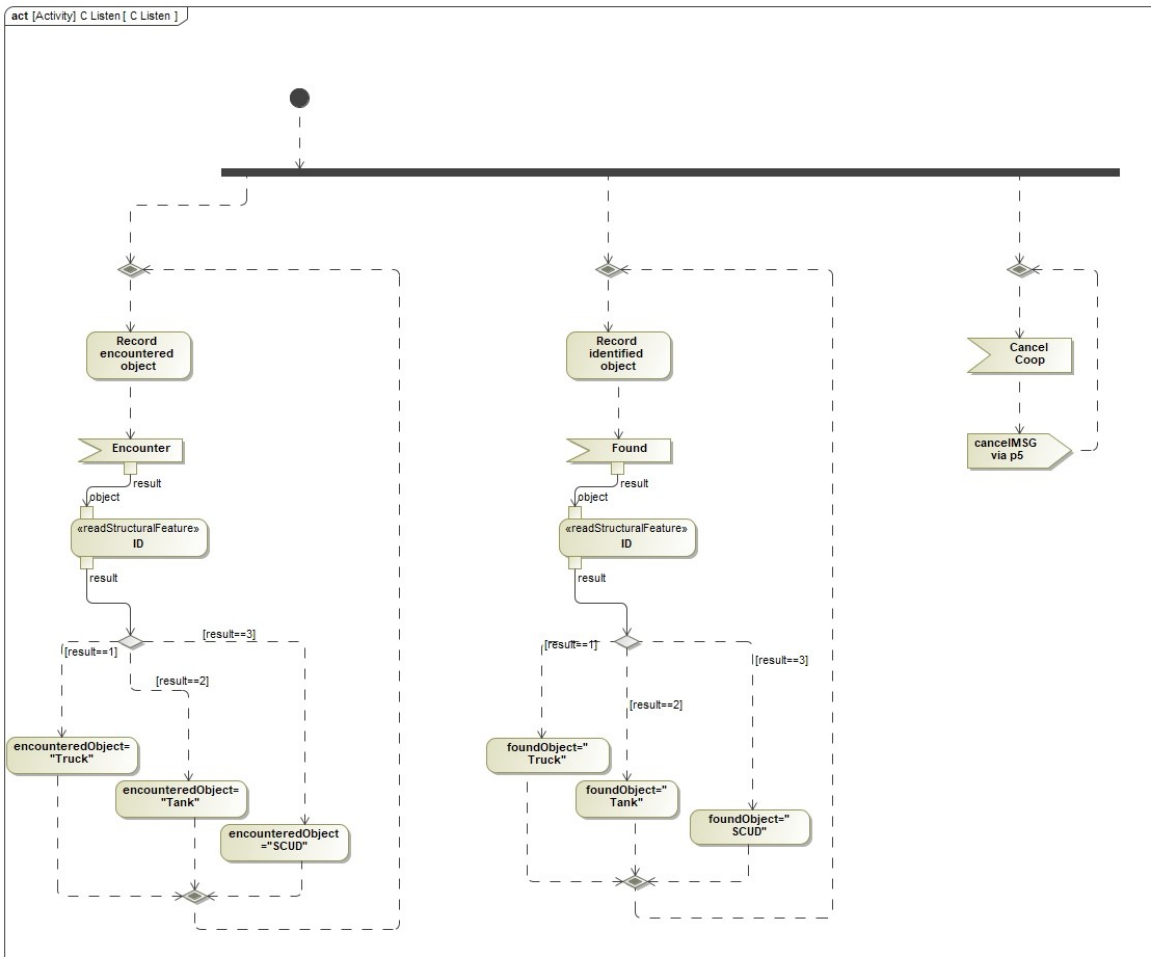

Figure 22. NCAM Coordinator Listen Activity Diagram

4.3 MBSE Simulation vs AFSIM Simulation

The MBSE tool used in this report has an integrated simulation toolkit that is useful to verify the logic of behavioral models in small scale, or run full User Interface (UI) simulations in a larger scale. The NCAM MBSE model has been constructed to a sufficient level to enable a systems engineering simulation of the NCAM mission scenario that mirrors the full physics based simulation created by Hatzinger in AFSIM [3]. The purpose of the MBSE simulation scenario is to inform the design and management team of the expected autonomous interactions of NCAM with respect to different munition configurations. As described in the previous section, behaviors are similar to the AFSIM model but abstracted for application in the MBSE model, so overall scenario behaviors should be consistent with the AFSIM simulated outcomes.

In the process of creating the behavior model for NCAM, numerous small simulations were utilized to confirm proper signal flow and valid activity logic. An important note for small simulations that rely on value properties is to simulate the block where the value property exists rather than the activity itself. Figure 23 depicts an MBSE simulation of the NCAM block where states are interacting to send a signal out from the coordinator to the external perimeter of the block. A simple block simulation will have to be controlled by manual signal inputs into the simulation pane. The green shaded elements indicate something that has been used while the red highlighted states are the currently running states. However, the purpose of the NCAM model is to represent collaborative munitions, so the next logical step is to get multiple NCAMs interacting in a small scale simulation. The following subsections describe the development process for creating a working NCAM scenario simulation in Cameo Systems Modeler.

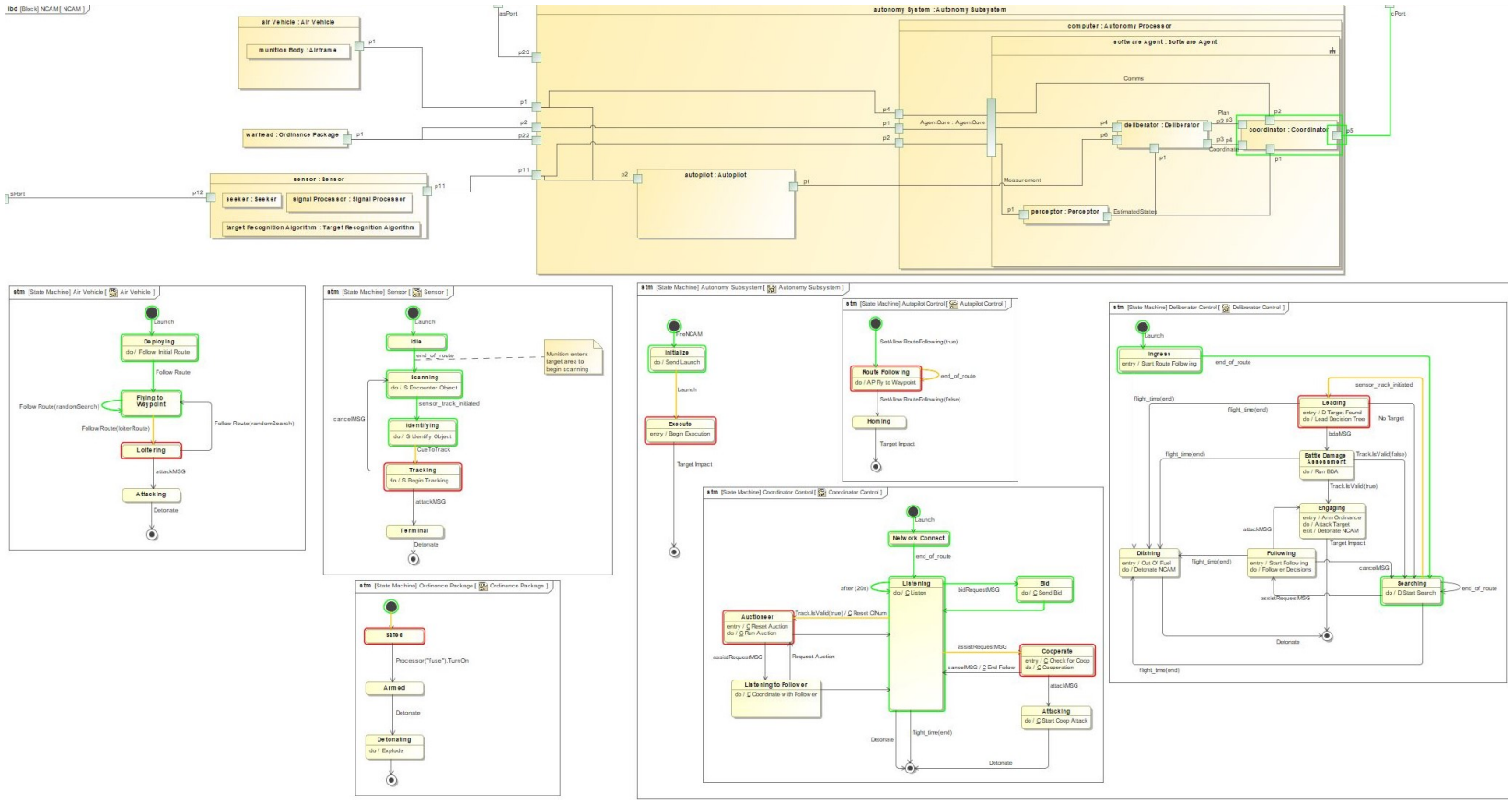

Figure 23. NCAM Block Simulation

4.3.1 Small Scale Development Scenario for Verification

Setting up a simulation scenario is similar to creating a logical structure like the NCAM domain. Figure 24 depicts a simple three-munition scenario that utilizes generalization to transfer all of the structure and behavior of the NCAM block built in Section 4.2 onto the three new munitions named NCAM 01, NCAM 02, and NCAM 03. In short, the scenario consists of three identical NCAMs. The next step for the scenario is to establish internal connections, namely, the NCAM coordinators need to have a way to communicate between the other NCAM coordinators. Figure 25 is an IBD that shows the three NCAMs connected by their external coordinator ports. The ports have a carrot ($\hat{()}$) meaning the port is a reference property from the original NCAM block and appear due to inheritance from the generalization relationship. At this point, the small scale scenario can be simulated by running the scenario block, however, manual signal inputs will not make sense with identical state machines running in parallel. So, a custom UI is critical to ensuring the correct input signals are sent to the right component.

The small scale UI presented in Figure 26 was developed over the course of dozens of simulations as behaviors were identified for verification. It is important to follow the process developed by this research to create the custom UI, otherwise the UI will not work properly. First, a background frame is placed and the scenario block used in Figure 24 is drag-and-dropped to type the frame to the scenario. Then, a grouped box is placed and typed with each NCAM part property from the scenario block representing the individual NCAMs. Each subsequent grouped box layer is then typed with the subsystem or component part property from the original NCAM block. The purpose of this process is to associate the signal buttons and image switchers with the correct NCAM in the scenario. A button, typed by a specific signal, placed in the grouped box will send the specified signal to the part property in an IBD to

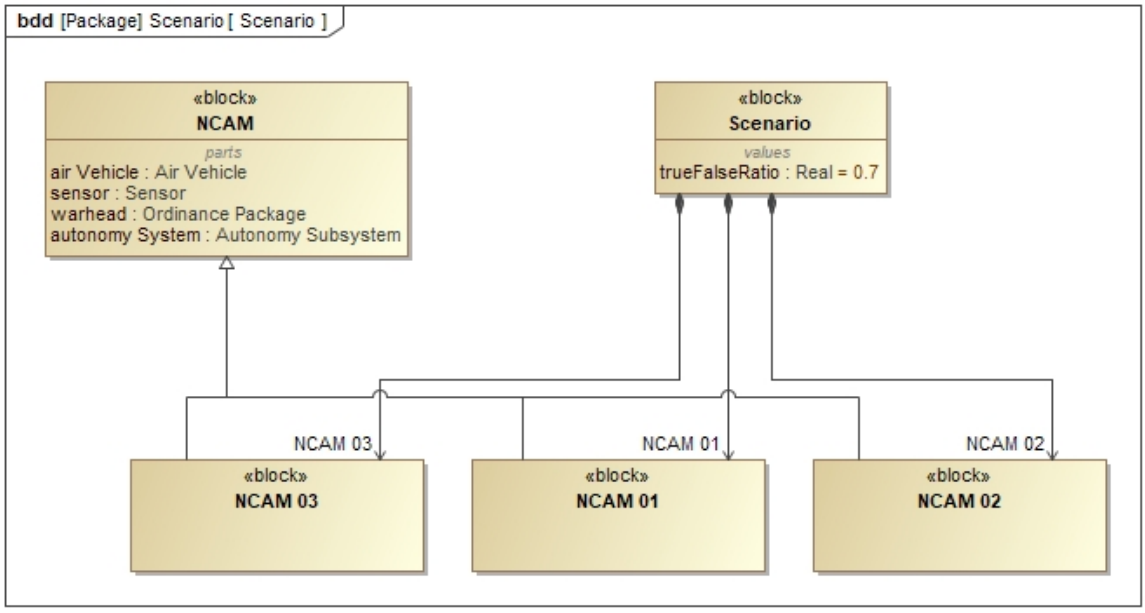

Figure 24. NCAM Small Scale Scenario Structure Diagram

interact with the behaviors of the block.

Image switchers and other simulation visual aids are created in a simulation configuration diagram. The simulation configuration diagram for the visual aids in this research is shown in Figure 27. The image switchers are created to represent a block and the current state of that block. For an NCAM, an important block to track is the deliberator because it acts as the brain of any individual NCAM. Each state from

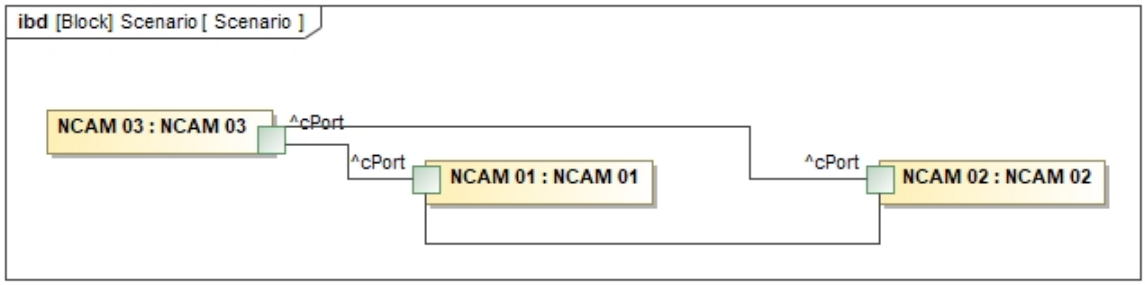

Figure 25. NCAM Small Scale Scenario Internal Structure Diagram

the deliberator state machine in Figure 9 has an image associated with it, so in simulation the current state of the deliberator will be visible to the user. Additionally, a similar method was used to display the encountered target and identified target, except the image switch is triggered by specific actions in the sensor. The actions are visible in Figure 28 as the "Encounter..." actions and the "Classify..." actions. The difference with this switcher is that the changes are global, so the images switch for all instances when the action occurs. The other elements in Figure 27 are the Simulation Config and Timeline Chart. The config block tells the simulation toolkit all of the parameters for the simulation each time it is run. Time units and step size are set to seconds and 1 for the NCAM simulations to be concurrent with the AFSIM engine timescale used with NCAM. The timescale charts are another way to visually show the states of blocks within the simulation. The overall NCAM timescale chart is visible on the right side of Figures 29 through 31 where the red lines track the states of the subsystems as the simulation progresses.

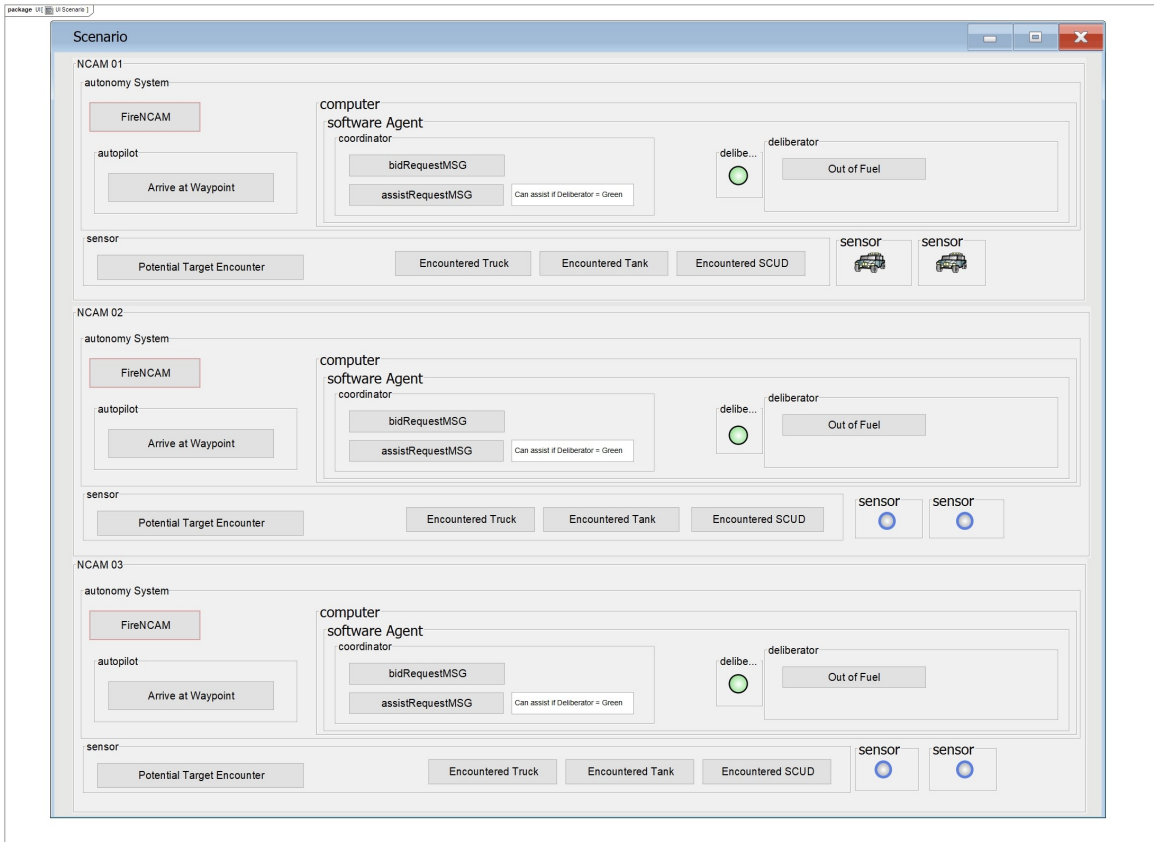

Figure 26. NCAM Small Scale Scenario User Interface Diagram

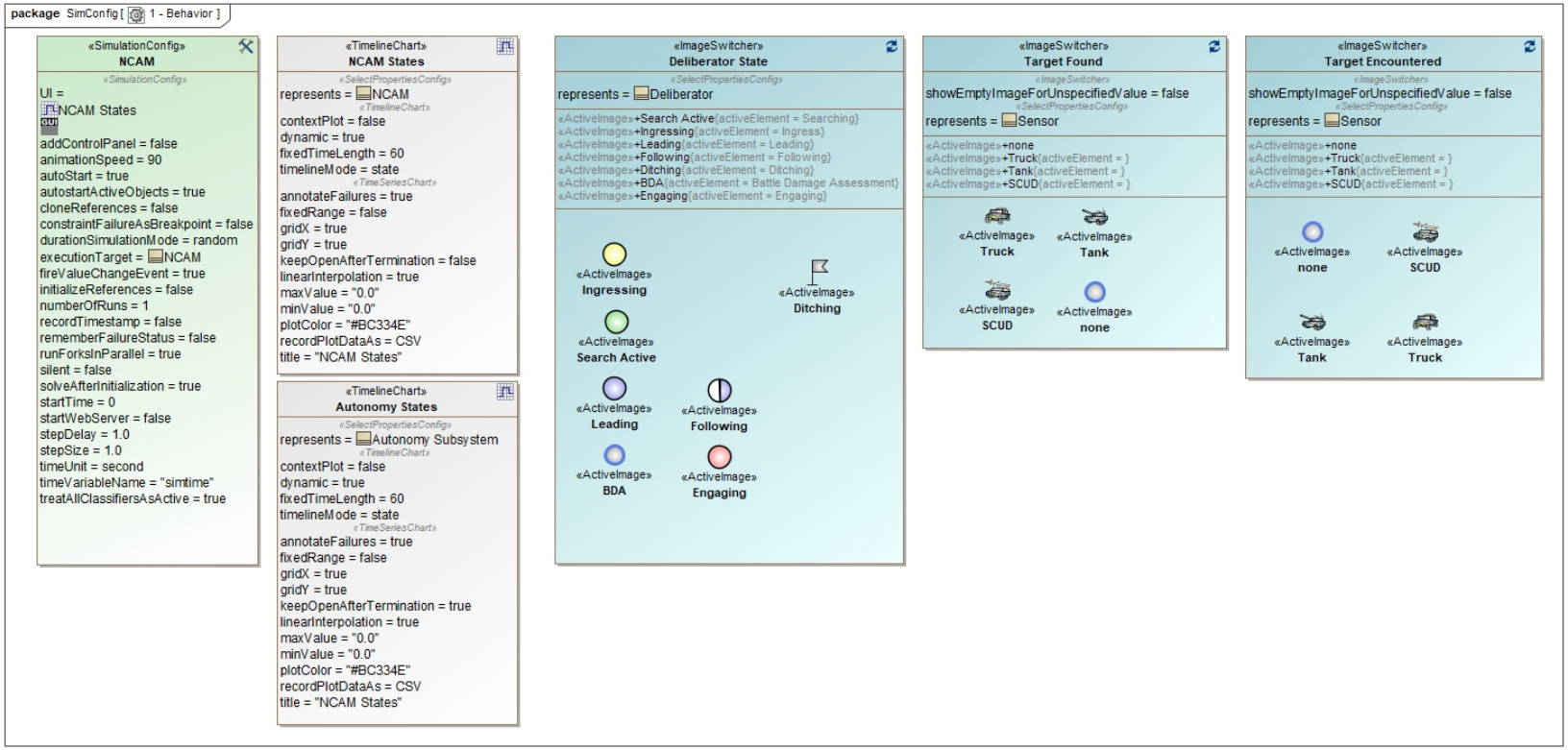

Figure 27. Simulation Configuration Items in Development Diagram

The progression of the small scale scenario logically follows the same mission profile as the AFSIM physics-based simulation. First, each NCAM requires a fire command (FireNCAM button) to launch from the mission aircraft and the deliberator enters the *Ingressing* state indicated by a yellow circle in the UI. Then, the munitions autonomously begin guiding to their first waypoint at the perimeter of the search area. Upon reaching the first waypoint (Arrive at Waypoint button), the NCAM begins autonomously searching for targets and the deliberator enters the *Searching* state indicated by a green circle in the UI. If an NCAM has a potential target encounter provided (Potential Target Encounter button), the target type needs to follow (Encounter Truck, Tank, or SCUD button). The NCAM simulation will run a confusion matrix, as shown in Figure 28, based on the sensor subsystem's probability of identification (probID) value property identical to the AFSIM confusion matrix. When the identified target signal is passed to the deliberator, the NCAM will become a lead munition indicated by a purple circle. After becoming the lead, the NCAM will send a bid request to all other NCAMs still active in the simulation. As described in the coordinator subsection, an NCAM will only respond to the assist request if it is ready to cooperate. After the lead munition completes the auction and sends the assist request with the winning bid value attached, the receiving NCAM with a matching bid sent value will become a follower as indicated by a purple half circle in the UI. Figure 30 depicts the UI at this point in the simulation. NCAM 01 is the lead munition that has encountered a Tank, identified the Tank, requested assistance, and NCAM 03 provided the highest bid to win the auction so it become the follower. Between Figure 30 and Figure 31, both the lead and follower NCAM attacked the identified Tank, then NCAM 02 encountered and identified a SCUD. Because no other munitions were available in the simulation, the bid request received no responses and no NCAMs won the auction. So, in accordance with the decision tree in Figure 3,

NCAM 2 moved into the ***Battle Damage Assessment*** state, indicated by the grey circle, in preparation to attack the target. After NCAM 02 attacks the SCUD, the small scale scenario simulation is effectively complete.

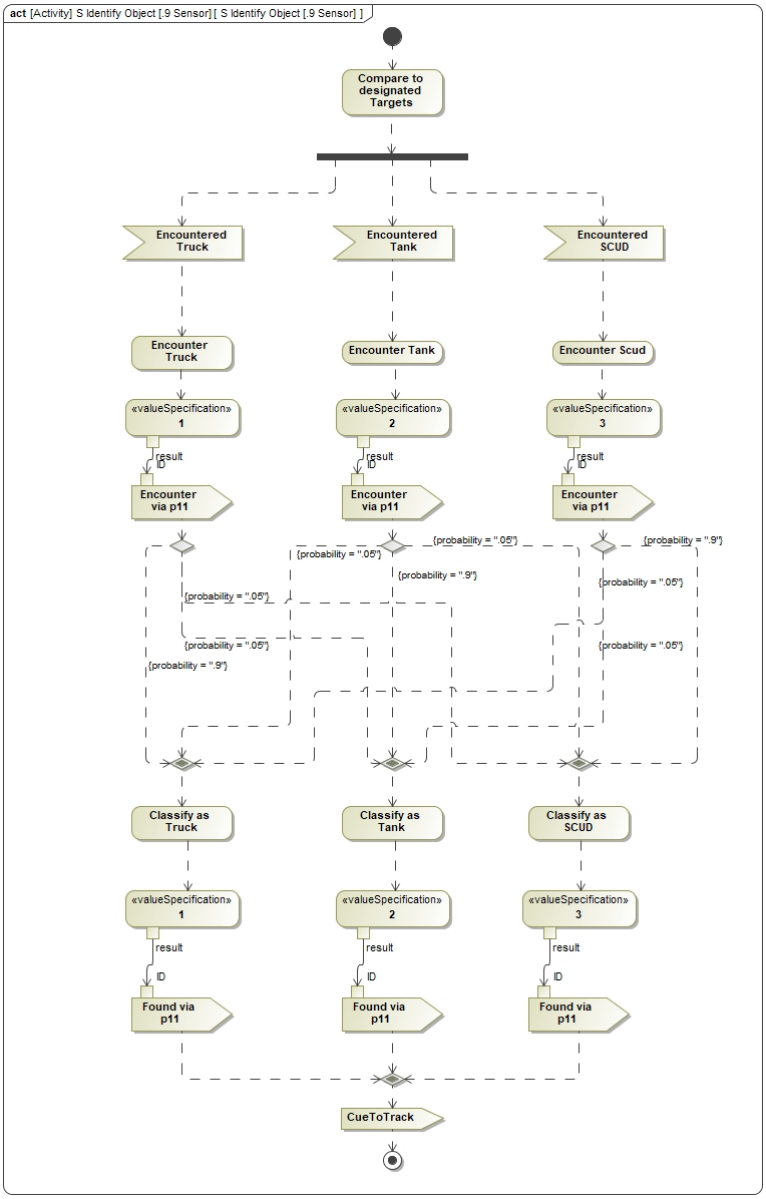

Figure 28. NCAM Sensor Target Identification Confusion Matrix for a 0.9 Probability of Identification Sensor

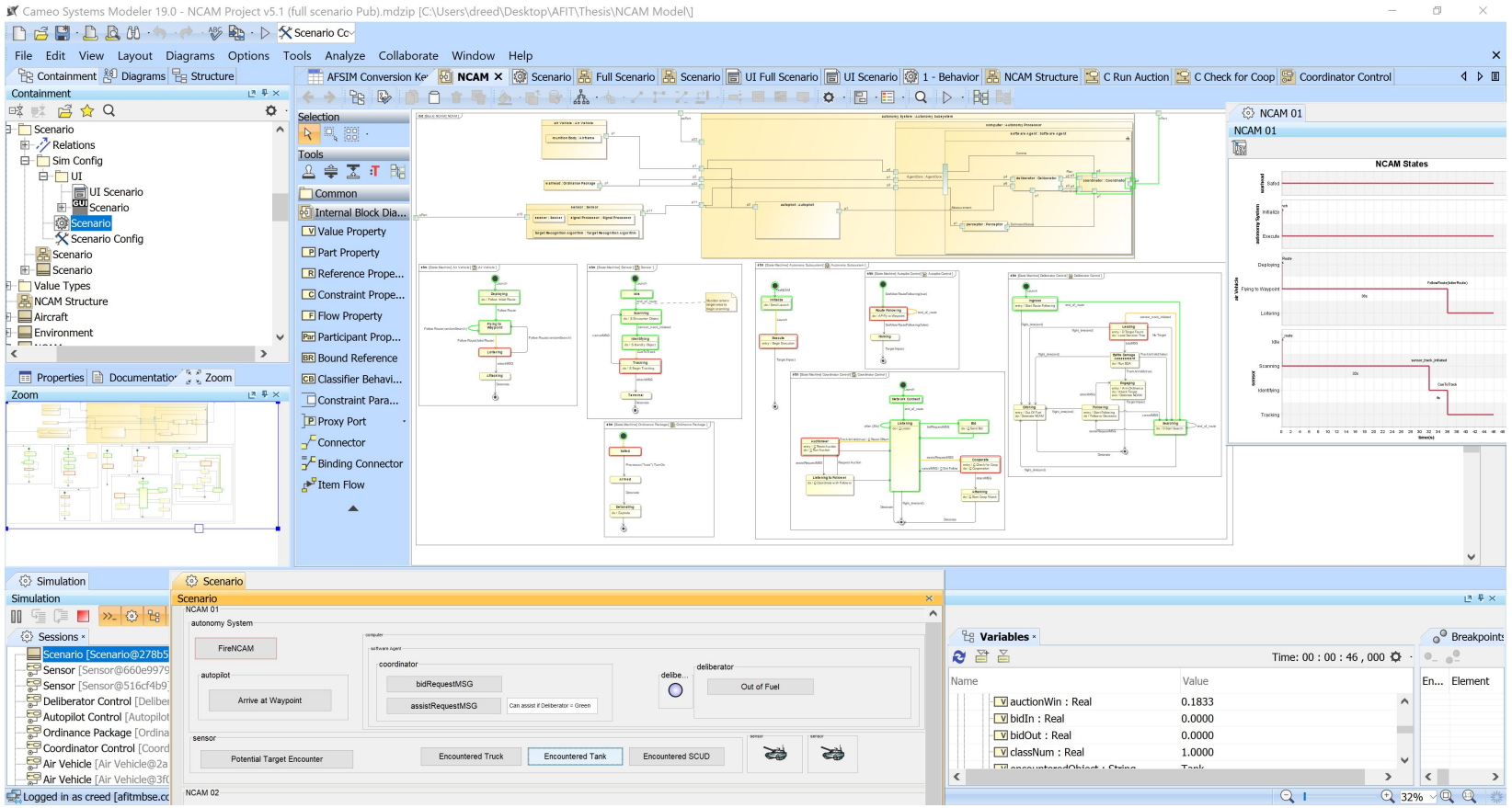

Figure 29. NCAM Small Scale Scenario Simulation: Leader Identified a Target

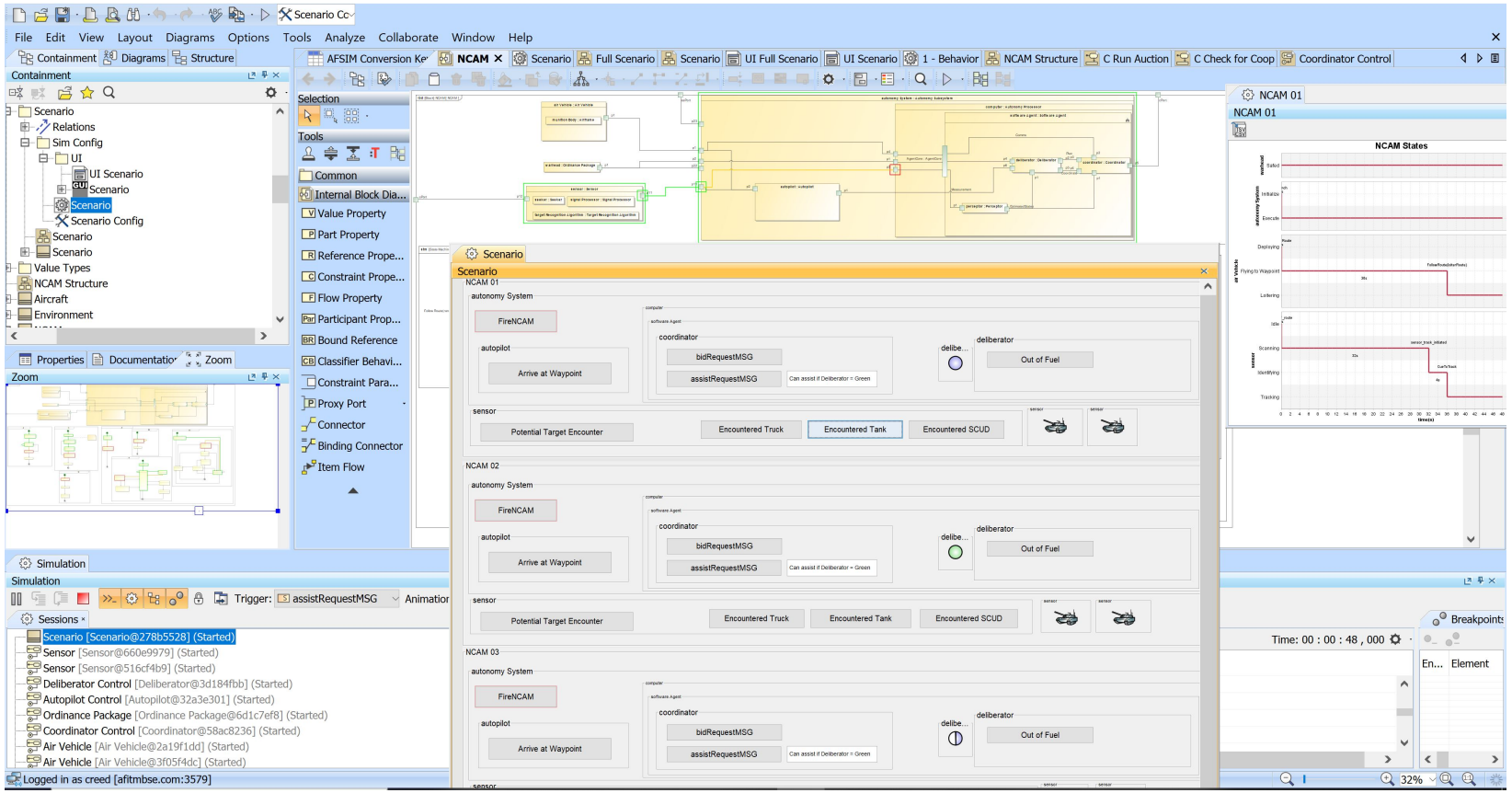

Figure 30. NCAM Small Scale Scenario Simulation: Leader and Follower Communicating

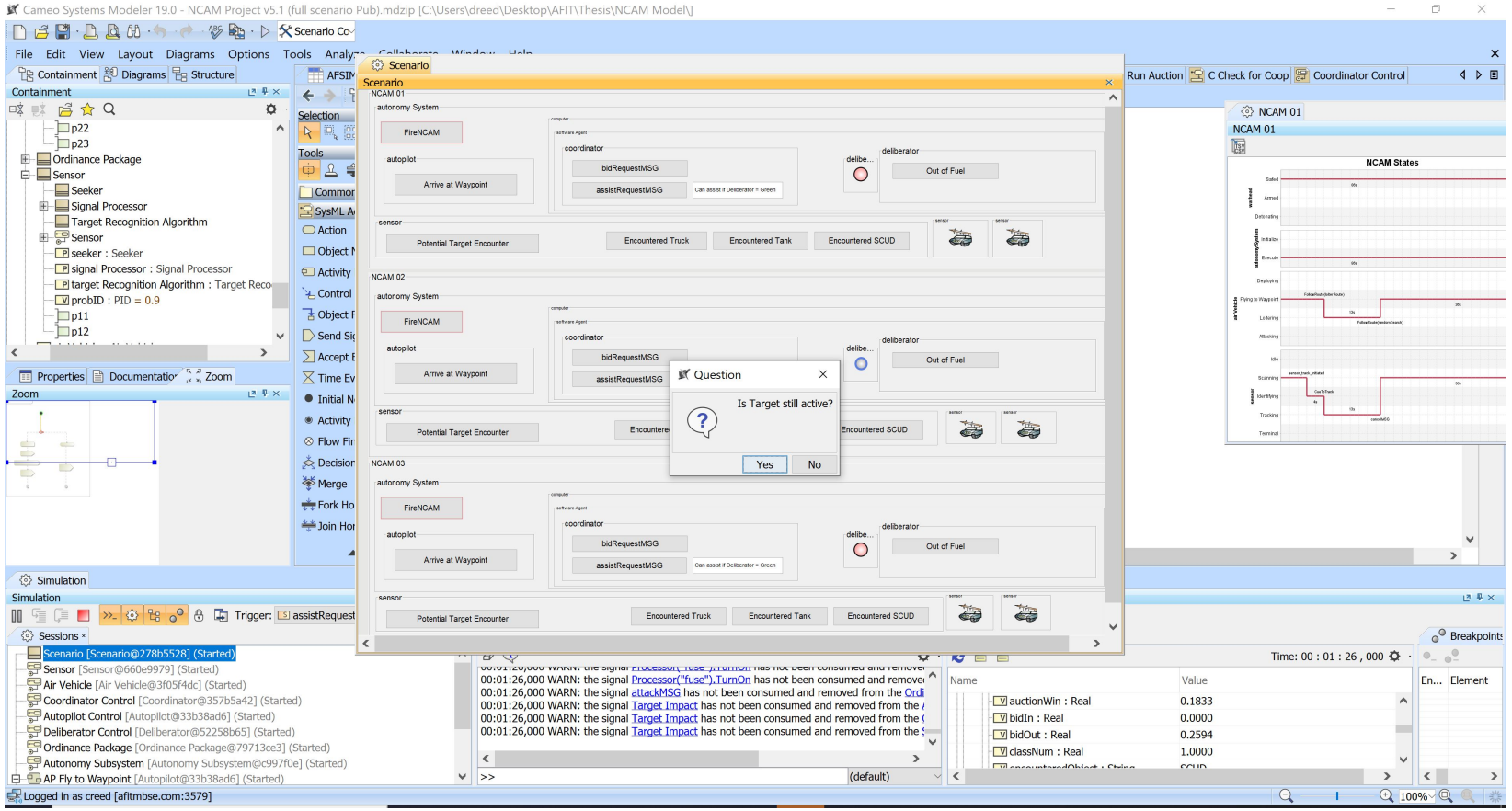

Figure 31. NCAM Small Scale Scenario Simulation: Separate Lead Munition Identified a New Target

4.3.2 Full NCAM Mission Profile Scenario Model

The small-scale scenario enabled the expansion into a full-scale scenario simulation to match the AFSIM NCAM mission profile as used by Hatzinger and Gertsman [3]. The scenario construction method is identical except with 14 individual NCAM, the addition of an aircraft to launch all 14 NCAM simultaneously, and a target to provide encounter types to the NCAM. The BDD and IBD structure of the full scenario is presented in Figure 32 and Figure 33 respectively. The BDD appears very similar to the small scale BDD from Figure 24 by including the 14 individual NCAMs (extending beyond the figure bounds) on the bottom half of the diagram and the original NCAM for generalization in the top left. The Target and Aircraft blocks from the NCAM domain are simply added as components of the full scenario to enable their effects in the simulation. The full scale IBD, however, is much more complex in appearance than the small scale IBD from Figure 25. The same principle of construction was followed though, where each NCAM coordinator port is connected to all other NCAM coordinator ports. Additionally, the target has logical connections to each NCAM sensor to provide the target type data, and the aircraft is logically connected to each NCAM autonomy subsystem to provide the fire command.

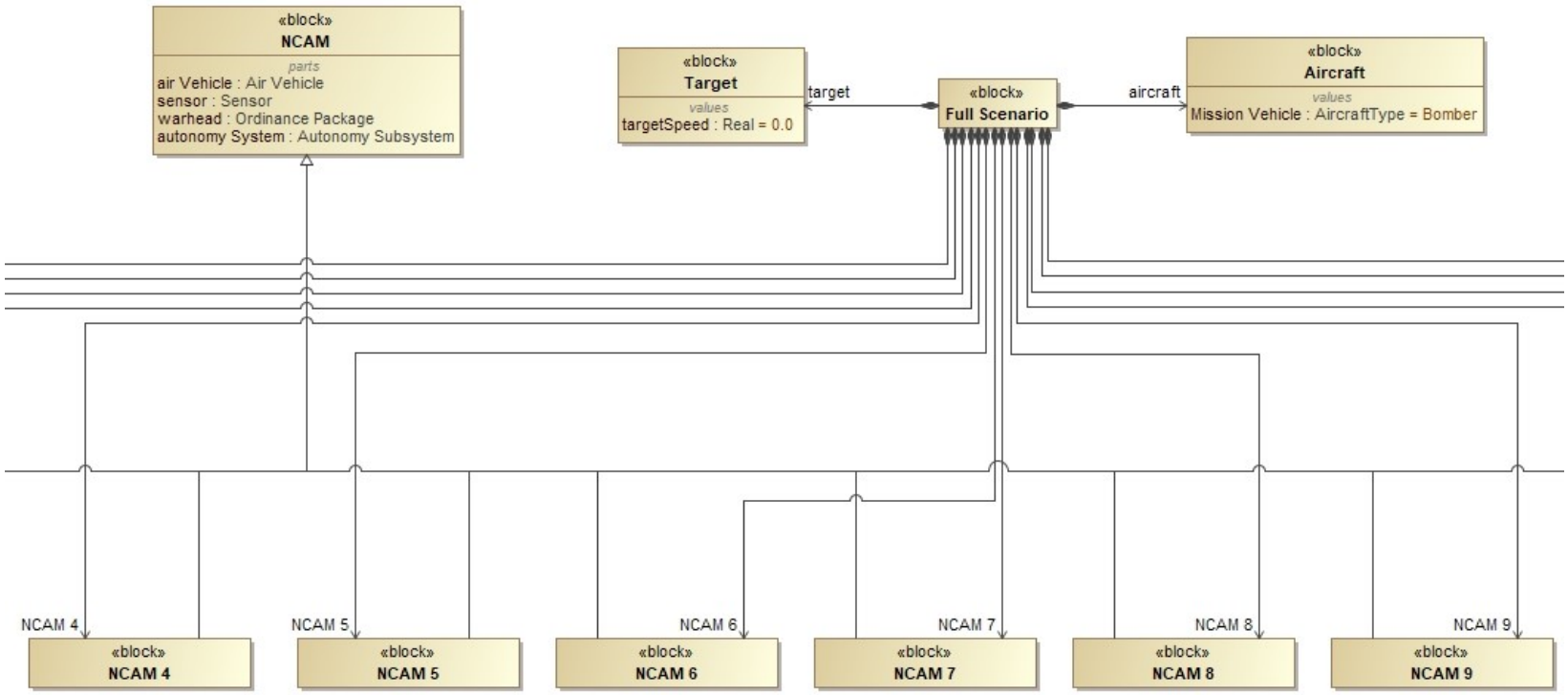

Figure 32. NCAM Full Scenario Structure Diagram

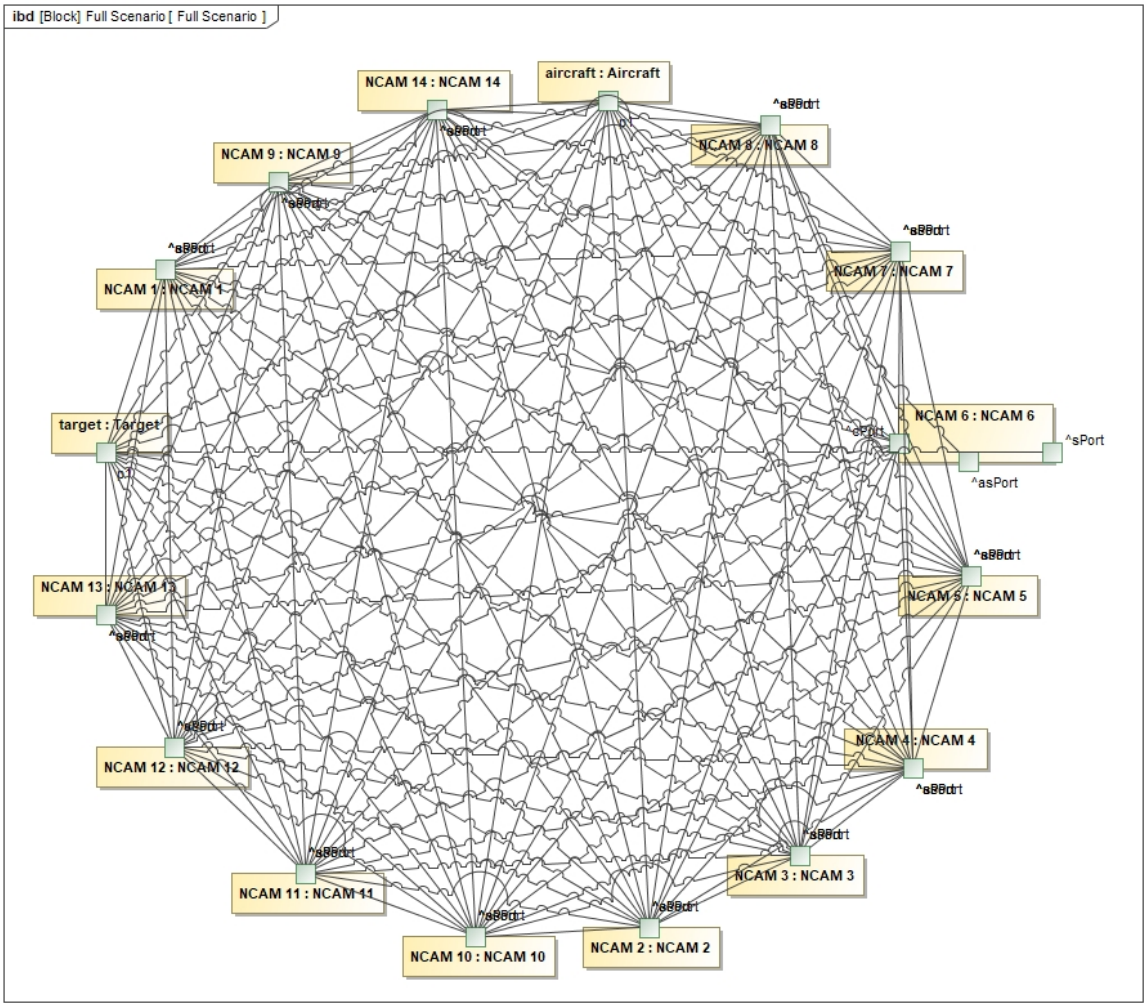

Figure 33. NCAM Full Scenario Internal Structure Diagram

The full scenario UI shown in Figure 34 is trimmed to only the controls necessary for normal scenario simulation. The addition of the aircraft and target to the scenario allows the control for firing each NCAM and control for encountered target type to be moved to a high level single point control on the left of the UI. Additionally, the unintended interaction between the image switcher and actions for the sensor enables a simplified encountered vs identified display for the overall scenario, also located on the left of the UI. Each NCAM then only requires the ability to independently arrive at

a waypoint or encounter a target as shown by the two buttons in each NCAM grouped box. The deliberator state indication circle is included in each NCAM grouped box to give the simulation operator a sense of what each munition is doing at any point in the simulation.

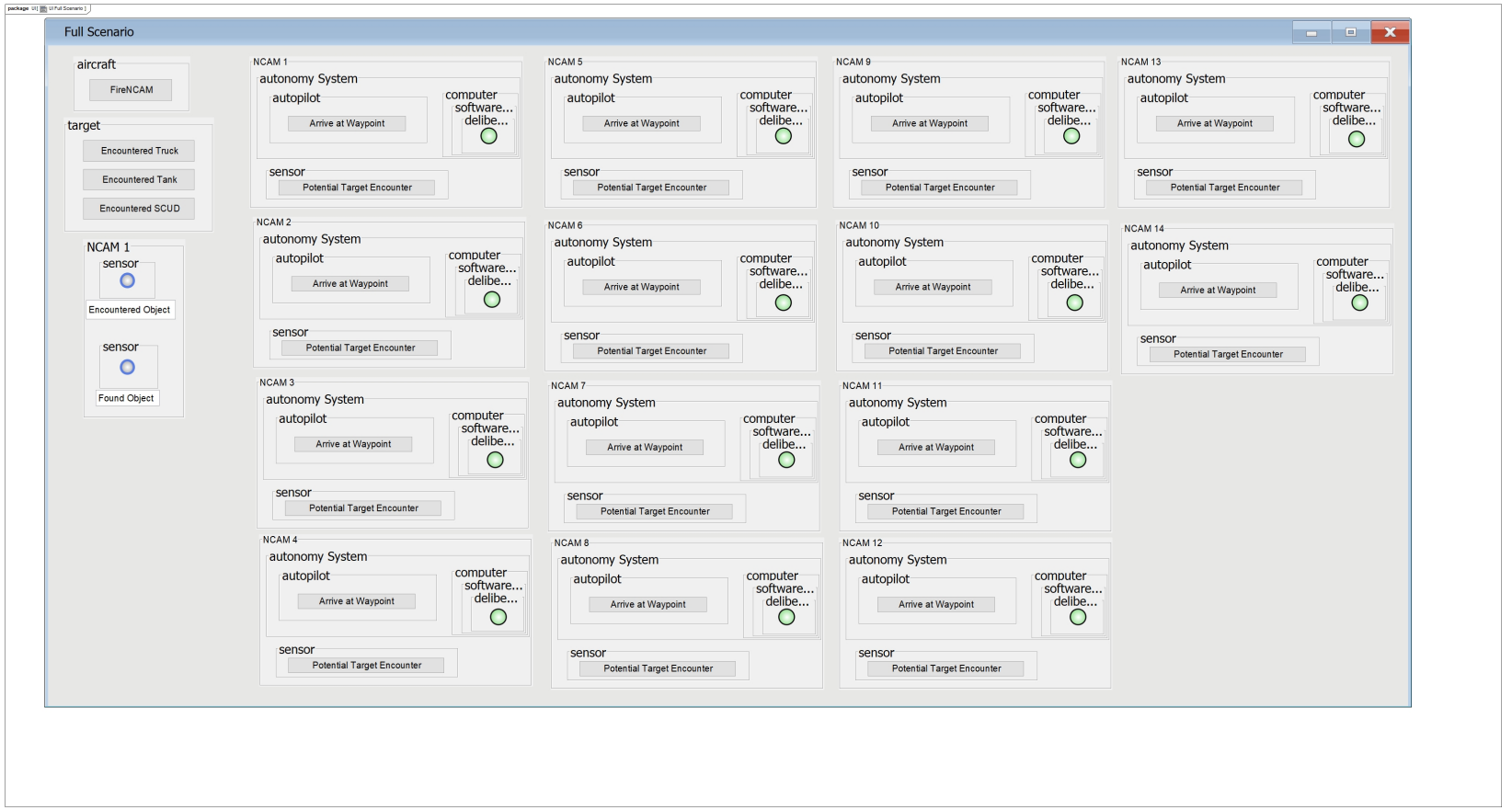

Figure 34. NCAM Full Scenario User Interface Diagram

Figures 35 and 36 show what can be expected in the full-scale simulation, starting with some munitions beginning a search, one munition finding a potential target, then autonomous decision making takes over with respect to the target cooperation thresholds. For example, NCAM 1 goes from being a lead munition requesting assistance in the first image, to next command NCAM 10 to complete the attack on the identified SCUD, to finally become a follower munition on a separate SCUD target that was identified by NCAM 5 in Figure 36 over the course of the simulation. The complex interactions occurring in this full scale simulation are still under human operator control via the UI providing encounters to the scenario or manual value property manipulation in the simulation variables.

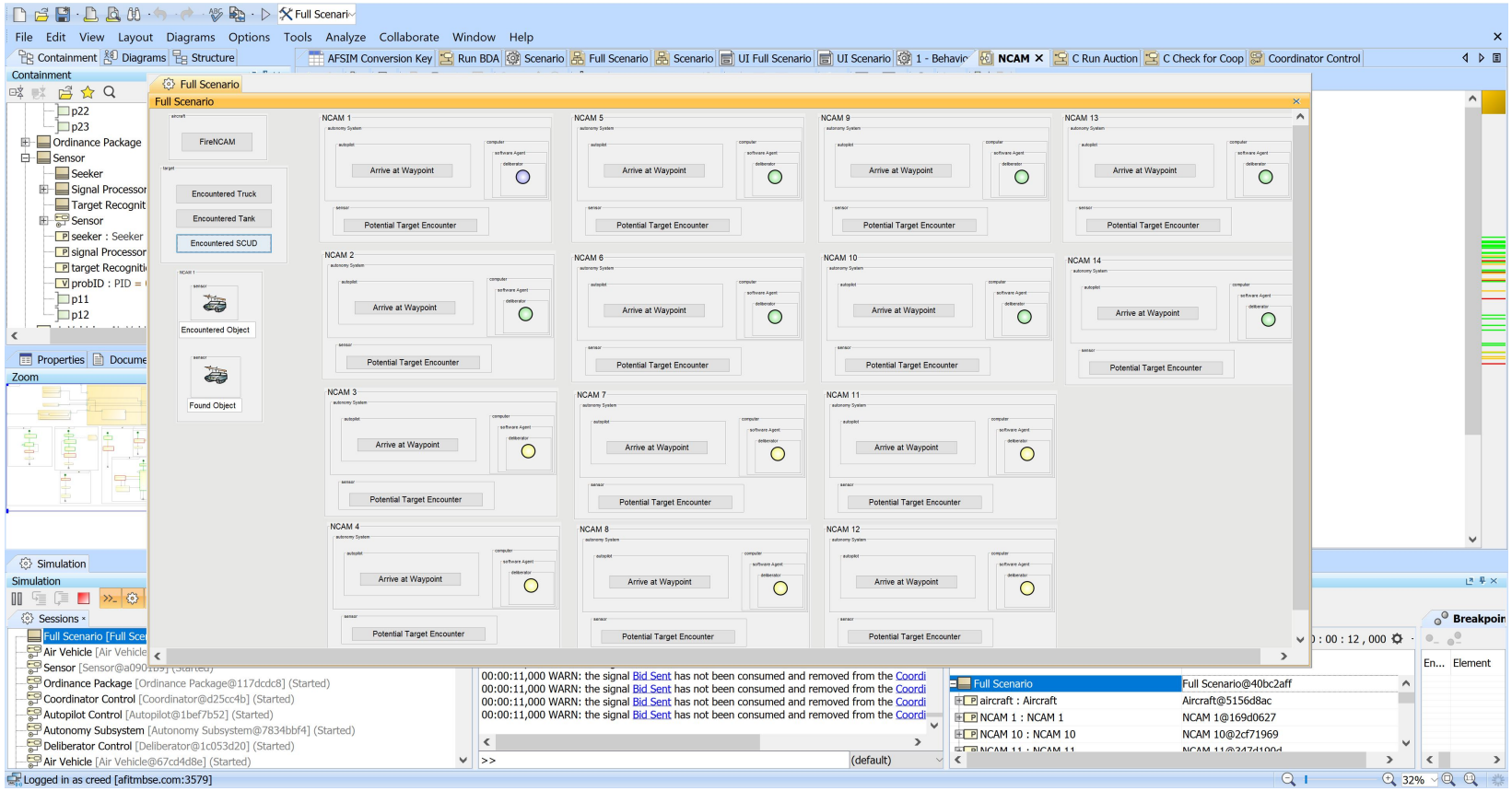

Figure 35. NCAM Full Scenario Simulation: Some Munitions Searching with One Target Found

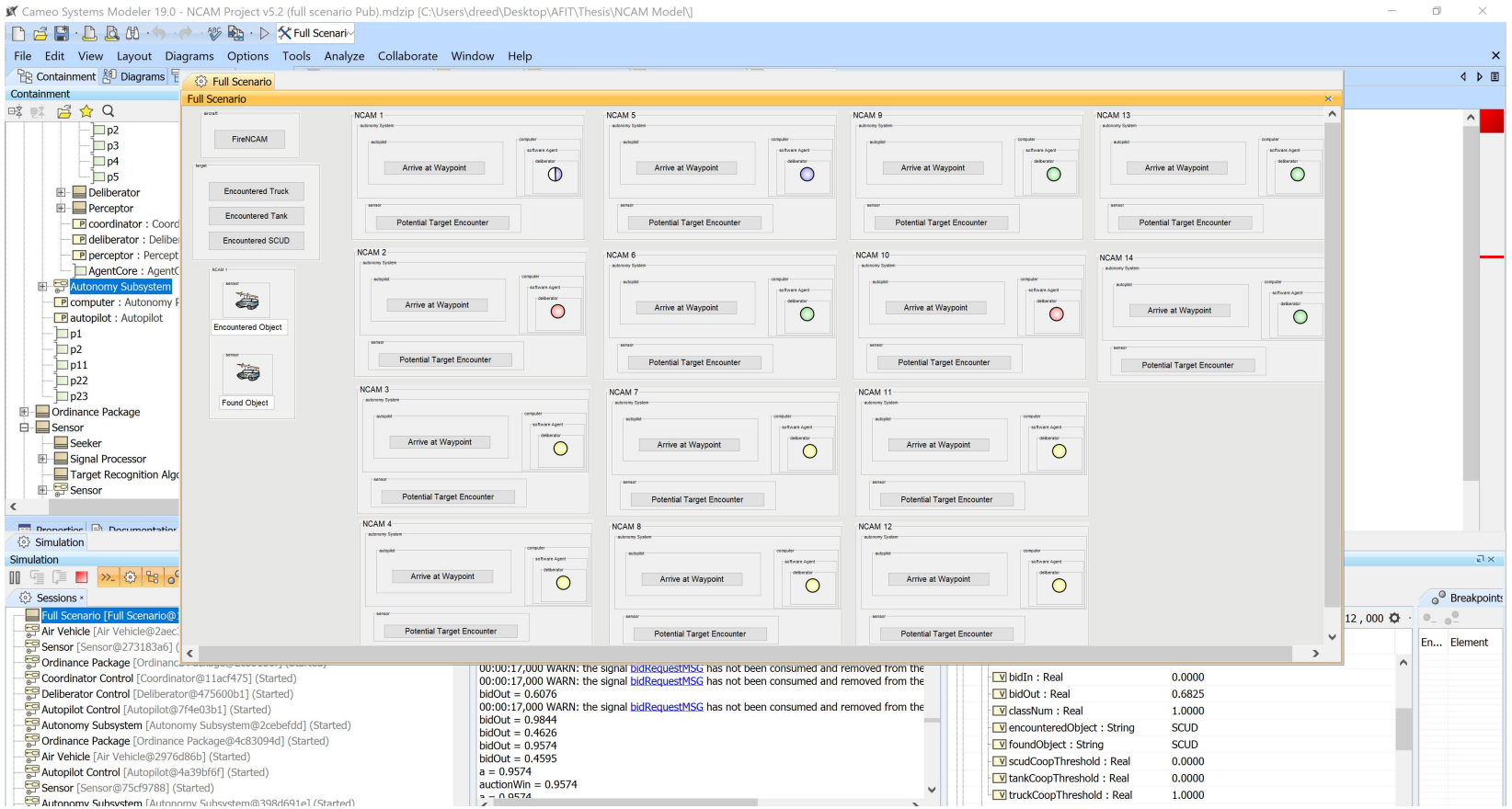

Figure 36. NCAM Full Scenario Simulation: Mid Simulation Status Variety

4.3.3 Simulation Comparison

One purpose of the MBSE model and MBSE scenario simulation is to give the model user the ability to trust and communicate the actions taken by the autonomous agents in response to stimuli. In the case of the physics-based AFSIM simulation, the same NCAM design displays a level of autonomy consistent with the (USD(R&E)) definition [4] , where munitions respond to an unknown target composition in a target area with self-governance. The munitions go through a decision-making process to determine which potential targets should be attacked. These decisions are based on characteristics of the munitions themselves such as sensor quality, warhead lethality, and fuel state. In essence, the pair of NCAM simulations provide significantly different value to the model users and each platform has advantages.

The MBSE NCAM model has the following advantages:

1. Provides visual traceability of behaviors to structure for more complete system definition and understanding;

2. Enables rapid prototyping of baseline changes and provides immediate, interactive feedback before starting detailed modeling in a physics-based environment;

3. Becomes a natural focal point for management and engineering efforts through interconnectivity between software programs;

4. Can include requirement traceability through the full design process;

5. Model fidelity can be expanded and customized to the level that provides the most value to the user.

Each MBSE model advantage traces back to the root definition of MBSE, "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and

continuing throughout development and later life cycle phases." [INCOSE SE Vision, 2020] Advantages 1 and 4 utilize the specific classes of SysML to enable clean system traceability with meaning to model users. Advantage 5 feeds into Advantage 2 because model users in charge of design can customize the model to any level of fidelity that would aid in verification and validation of design changes. A model user is then able to use Advantage 3 to translate the model into forms that are usable by non-model users, like documents or code.

The physics-based NCAM simulation in AFSIM has the following advantages:

1. Provides a purpose built environment for military scenario simulation;
2. Predefined, reusable components and behaviors enable rapid scenario generation;
3. Enables broad scope data collection through comprehensive simulation logs;
4. Provides a visual playback environment to observe any completed simulation;
5. Can efficiently run thousands of simulations per hour.

The advantages of working with AFSIM is that it is designed for simulation. All five advantages stem from the fact that AFSIM is a simulation framework that has tools purpose built to help build and run scenarios. These advantages are apparent when compared to the simulation environment in Cameo.

4.4 Analysis of Translation

The key to a simple method of translating variable data between Cameo and AFSIM is the *variables.txt* file presented in Hatzinger and Gertsman Appendix A.2 [3], which is identical to the text file generated by the MBSE model as shown in Figure 37. The *variables.txt* code creates pre-processor variables that are recalled every time

they are needed within the AFSIM scenario. Importantly, these variables are the elements that can be changed to alter autonomy behaviors within either simulation.

The *variables.txt* file can be generated using the Cameo Report Wizard Tool as described in Figure 6. This document generation method requires a custom template that is coded in the desired output format with a language called Velocity Template Language. For the AFSIM variable conversion, the template needs to have the ability to pull value property names and default values which populate the *variables.txt* file. Additionally, the phrase "$Define" needs to be included before each variable. The first challenge with the template is the difficulty with manipulating three different programming languages to create the desired output. Velocity uses the $ to define a referencing action, while AFSIM also requires the $ to begin the pre-processor definition action. So, a "do not parse" section has to be included in each output line within the template with the specific phrase, "$Define". Next, the Velocity referencing actions need to have the proper notation to find the exact element of the MBSE model. The reference uses dot notation to specify each level of the searched list of items, which is described by a search loop that lists every specified item from the model. For example, Line 2 of Figure 38 uses the "foreach" command to list every value property in the model and gives them the nickname "vp". The next line specifies that the output is only looking for value properties within the Scenario Variables block from the document generator diagram, Figure 6. Line 4 is the output, where each value property that meets the requirements is printed with: first the "$Define" phrase specified by the "#[[$Define]]#" section of passthrough code, followed by the name of the value property specified by "$vp.name" where the name is a first level element of the value property, and finalized by the default value of the value property specified by "$vp.defaultValue.value" because the number value itself is a second level element of the value property. The result of running this section of the template through the

Cameo Report Wizard is the first three lines of the *variables.txt* output in Figure 37. The same template process is repeated for each variable in the output file from the value properties shown in Figure 6. The only change in process was for the probID variable because its default value in the MBSE model is an enumerated option. Thus, the search parameters for the element are "$vp.defaultValue.instance.name" because the enumeration adds another layer to reach the number's "name" value.

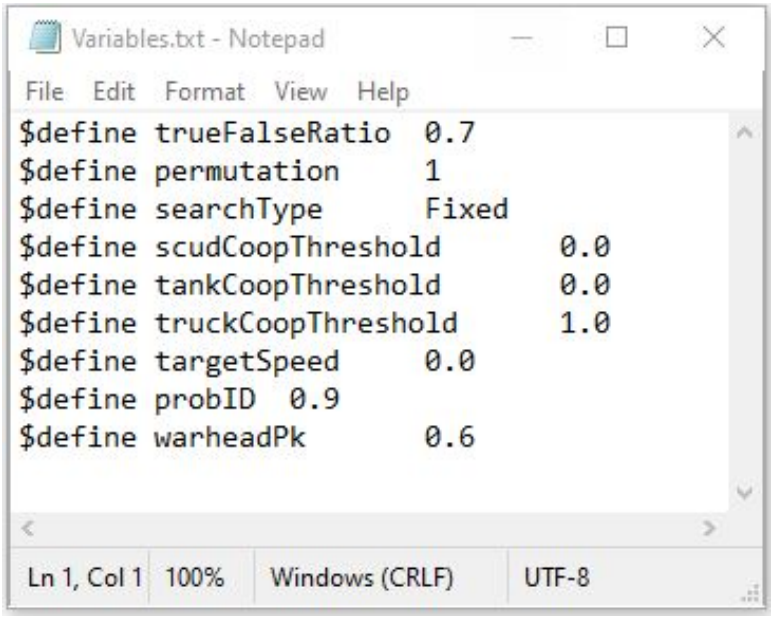

Figure 37. variables.txt output from NCAM MBSE model

A separate, automatic integration between Cameo and AFSIM is presented in Hatzinger and Gertsman Section 4.6 [3]. The integration process is similar to the Report Wizard method in terms of using specific value properties to populate a text file for AFSIM, but files are created through Jython scripts and the model automatically launches 30 runs of the AFSIM scenario in the background of the MBSE simulation execution. This process utilizes the full capabilities of both the AFSIM and MBSE model with an MBSE facing interface, including output results from the AFSIM simulations. The main drawback of this application is the need to manually

tune the Jython scripts to locate and utilize the correct directories for the AFSIM files. In summary, Hatzinger's method to integrate AFSIM with Cameo should be the first choice for the model user; however, the Report Wizard method provides a robust process to generate the desired text file for use in AFSIM simulation, perhaps on a separate computer.

4.5 Analysis of Parallel Model Representation

Overall, both the AFSIM coded scenario and the MBSE simulated NCAM models share nearly identical behavior due to a close pairing of structural composition and behavioral designs. An NCAM when presented with a potential target will use its sensors to identify the potential target within three identified classes, then become a leader. The now-lead NCAM will then begin the auction process with all available NCAM in the target area where the other NCAM send a bid measured on the suitability of assistance to the lead. The lead then checks for the highest bid above the cooperation threshold for the target type and, if there is a winner, requests assistance from the highest bidder which becomes a follower. The follower, upon arrival at the target, will run an independent identification of the target and send the information to the lead. The lead will then follow the decision tree from Figure 3 and complete the action prescribed by the tree. The minor differences identified throughout Chapter 4 of this report do not impact the mission level behavioral profile of NCAM in the simulation.

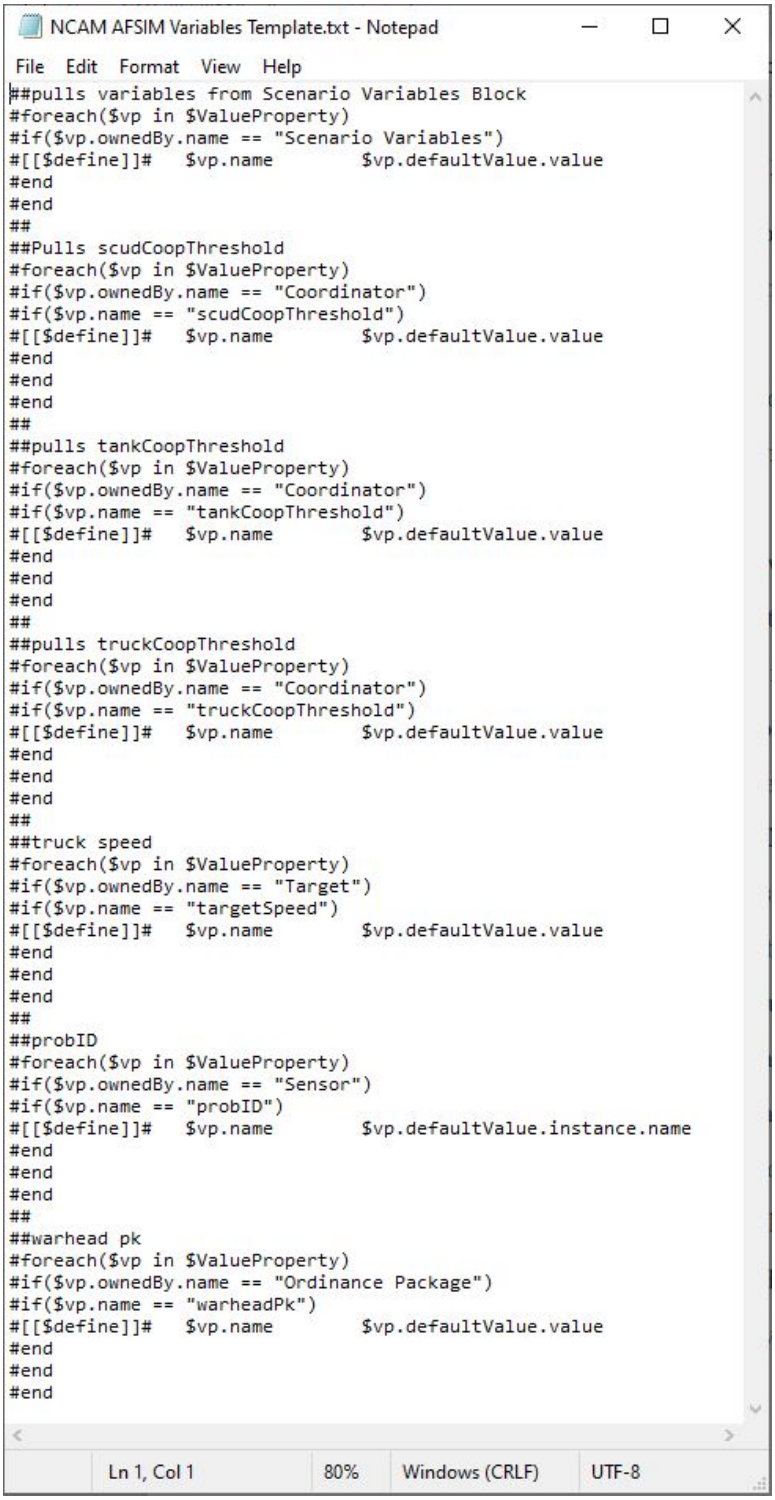

Figure 38. variables.txt Template in Velocity Template Language for use in the NCAM MBSE model

V. Conclusion

5.1 Research Findings

This research started with the goal of generating a practical methodology to model a behaviorally complex munition using SysML, then progressed into the creation of the NCAM MBSE model. Importantly, the MBSE NCAM model had to be consistent and parallel to the AFSIM NCAM model. Assessment of the complete model was done with behavioral analysis, automatic translation results, and structural comparisons between the models. The research has resulted in the following findings to the initial research questions.

- What are the strengths and weaknesses of SysML for behavior modeling?

The strengths of SysML are intrinsically tied to the benefits presented in the definition of MBSE, where systemic traceability of structure, behaviors, and requirements are the backbone of an MBSE model. With the NCAM MBSE model, complex munition behaviors are given a logical flow within a structure of NCAM components, including realistic signal production and reception. The flow is illustrated using Cameo simulations that help the model user control and understand the autonomous behaviors of each NCAM interacting with the simulated environment. Thus, the main strength of SysML for this effort is the ability to verify behavioral logic and communicate a large volume of information in an organized and traceable manner to system designers. Additional strengths extend from the OOSEM used in creation of the NCAM MBSE model, where objects like blocks and activities are reused, rather than recreated, as logically needed within the model. For example, the same NCAM block is reused in each simulation scenario, then SysML generalization relations further reuse the NCAM block design into the individual NCAM instances. SysML enabled the full

scenario simulation of 14 individual, interconnected NCAMs without creating new configurations of the NCAM design.

The weaknesses of a SysML behavioral model are apparent when considering its application in MBSE. MBSE is foremost a systems engineering modeling mindset, where abstractions of complex systems are visually presented in diagrams with simple shapes and lines. SysML is the primary language used to write these MBSE models and supports consistency within the diagrams. An NCAM in a SysML diagram is simply a block that has parts, properties, interfaces and behaviors. It is not a 1:1 digital representation of a physical NCAM. The activities associated with the NCAM block through the state machines of its parts describe the behavior of the block. Overall, these elements of the NCAM block can only interact in a closed system that contains elements that are specifically defined in the model. So, physics-based behaviors and equations need to be manually coded into the environment if the NCAM block wants to utilize the behavior. For example, spatial locations of the NCAMs in simulation is not included in the current MBSE model, as every aspect of the scenario would have to be manually defined. The additional work would include, at a minimum, creating a two-dimensional target area, randomly distributing targets within the area, defining the initial location and movement system of the NCAMs, defining the search parameters and sensor area of the NCAMs, and adding multiple extra value properties to each NCAM subsystem to track the information relevant to the simulation. However, the work required to create this basic functionality does not add value to the MBSE model when a much more realistic simulation has already been purpose-built in another simulation environment. AFSIM includes a broad scope of base features and equations to enable physics-based simulations, including three dimensional spatial locations and behaviors, with minimal manual specification. In summary, the weaknesses of SysML behavioral modeling derive from the strengths of

SysML and MBSE, where SysML's ability to simplify any complex systems for human comprehension and communication means the behavioral modeling environment is not designed to be specialized in one specific field.

- Which MBSE elements and/or properties are suitable for translation into AFSIM native language for scenarios?

Overall, the language used to create an MBSE model is inherently different to the language AFSIM uses to generate and run scenario simulations. So, a full model translation from MBSE to AFSIM is not easily generated. Instead, each environment will have a parallel model that utilizes some key variables to set parameters of the system's behaviors. For this research, the key variables were translated from value properties identified in Figure 6 from the MBSE model into the variables text file in Figure 37 using either one of two separate methods identified in Section 4.4 of this report. These variables were identified as design factors by Hatzinger and Gertsman for their Design of Experiments scope in Section 3.3 [3]. In summary, the *variables.txt* file passed from the NCAM MBSE model to AFSIM were the best suited properties to translate between models to maintain parallel behaviors from both models.

- To what extent can a SysML digital model represent the behaviors of the cooperative munition used in AFSIM simulation?

As discussed in the weaknesses portion of SysML behavior modeling, the SysML model can technically match every behavior from the AFSIM model; however, the additional work to define the specific, physics-based behaviors would not provide value to the design team. The NCAM MBSE model provides a reasonable point of behavioral representation for a SysML model compared to the NCAM AFSIM model because the model user has a systems engineering viewpoint of the autonomous behaviors for NCAM with the ability to control the simulation events and observe

outcomes. The NCAM AFSIM simulation built by Hatzinger can simply run the scenario without user input and results can then be analyzed visually or statistically. The different perspectives on NCAM behavior provided by each model provide value that is subjectively better for system comprehension than two identical capability behavioral models.

- What automatic and repeatable methods can be utilized for data exchange between a SysML model and AFSIM scenario?

The specific data exchange methods between SysML and AFSIM found by this research were generated, without middle-ware, through Cameo Systems Modeler and AFSIM. The first two methods involve the creation of the *variables.txt* file, one through the Cameo report wizard tool and the other through Jython scritpts. The report wizard utilizes Velocity Template language to convert value property name and default value elements from the model into plain text ready for AFSIM to compile. To create the *variables.txt* file using the report wizard, either follow the directions from Figure 6 or launch the "NCAM AFSIM Variables" option in the tools tab. The Jython script creates the text file within an opaque action that is fed value properties with default values as structural feature inputs. Running a simulation of the activity in context of the Scenario block is sufficient to generate the *variables.txt* file in the location specified within the Jython script.

An automated method to import data from AFSIM into Cameo is described by Hatzinger in Section 4.6 [3]. Essentially, an opaque action with Jython script compiles the results of the AFSIM runs into a comma separated value file, then averages the results for desired metrics, and finally imports the averaged results into value properties of the Scenario block. This action must be run after the AFSIM scenarios have completed, so Hatzinger developed a state machine that, when simulated, automatically creates the *variables.txt* file, executes a specified number of runs of the

AFSIM scenario, then finishes with the value property data import.

Although the data exchange methods described are not exhaustive of all capabilities with either tool, each method provides an automated and consistent capacity for a model user to transfer design information between the environments. A point of focus for these methods was the lack of middleware handling the translations. Each method utilizes base Cameo features to generate files and populate data.

5.2 Lessons Learned

Over the course of this research effort, many of the difficulties with programming the NCAM MBSE model resulted in hours of trial and error combined with MBSE forum searching. One major example was learning the velocity template language for the report wizard function in Cameo. There is very little online documentation on how to read specific values from the model. In the end, a short discussion with another researcher that had worked with the language in the past solved many of the mysteries around integrating velocity with a Cameo model. Essentially, the template language uses dot notation to search layer by layer through the model for the elements specified in the output line of the template. For example, if elements from a specific value property, like the name and default value of the probability of sensor identification (probID), are desired for the output file, the template language first needs to understand that a value property is desired by commanding a list of all value properties from the model using #foreach(). Then, the list of value properties can be reduced by including if statements that specify either the name of the value property and/or the block owning the value property. Finally, the desired elements of the value property can be output using dot notation for the exact field, like $vp.name to pull the name of the property.

Another example of learning by trial and error due to limited documentation is the

chain of actions often used in the NCAM MBSE model to pull value properties and their current value from a block into an activity diagram. A simple activity diagram to reference is Figure 14 where a bid value is updated in the first opaque action, then the value is read using the common chain of: a readSelf action that creates a reference to the block owning the activity; flowing into a readStructuralFeature action where the structural feature is linked to the value property of the block owning the activity; and the output of this action contains the current value of the value property. In the case of Figure 14, the "bidOut" value is flowed into the *Bid Sent* signal where it is sent to another activity diagram for use in decisions and actions. The chain of actions to read value properties into an activity diagram is not documented clearly in any help guide or forum, but was built into the program as a base feature. An alternate method to read value properties discovered by this research is the chain of: createObject action linked to the desired block; and flowing into a readStructuralFeature action where the structural feature is linked to the value property of the block. However, this method results in activity diagrams that do not reliably read the current value of the value property.

Overall, the trial and error used in generating the NCAM MBSE model can lead to a significant lesson learned for offices starting to explore the use of MBSE for digital engineering. That is, hire an expert in SysML modeling to train others in a cohesive and consistent manner, and have a base modeling style/methodology to follow. This is important because each novice modeler left to their own tools will come up with many different ways to model the same system. This lesson learned was previously encountered by Reed [6] for modeling structure and rediscovered in this research with behavior modeling. A byproduct of creating MBSE models in a vacuum is the end result being difficult to use and modify by future model users that were not included in the development of the model, similar to when software code is handed to another

programmer. The programmer handed this code may elect to start over from nothing instead of modifying the existing code because it is difficult for humans to read and interpret code.

5.3 Future Work

The NCAM MBSE model developed for this research acts as a reasonable entry point to digital design for a notional cooperative munition. Many simplifying abstractions were required due to time constraints of developing this model within program timelines as well as the notional nature of the design. An advantage of building the model in SysML with an OOSEM mindset is the ability to extend and modify elements within the model and have the modifications automatically propagate to all areas they were used. For example, if a future researcher wanted to add a higher fidelity bid generation system into the model, the only modification required would be in the Create Bid opaque action from Figure 14. A full list of areas that could use work to increase model fidelity are:

- Providing an automatic mix of targets and non-targets to a new MBSE scenario influenced by the trueFalseRatio value property, and track the removal of the entities within simulation as the munitions attack based on the warheadPk value property.

- Overhauling the coordinator with an individual identification system to enable individual messaging; a similar system to the "C Check for Coop" activity could be used with new coordinator value properties to track self identification and follower identifications.

- Creating a MBSE spatial framework for NCAMs to operate in, including coordinate grids and velocities.

- Updating the auction and bid system to better provide a realistic suitability to assist.

- Mapping components of the MBSE designed NCAM to physical components of an unmanned aerial system for hardware in the loop testing of the autonomous behaviors.

The first four identified areas for future work would bring the MBSE NCAM behavioral model even closer in performance to the physics based AFSIM model while still maintaining the systems engineering viewpoint desired for system cognition. Mapping components would identify areas within the MBSE model that are either missing or not realistic to a physical application. This mapping research would also act as the launching point for creating a digital twin of a future NCAM.

5.4 Final Thoughts

The combination of MBSE and AFSIM models for NCAM provides a powerful resource for understanding complex autonomous behaviors in a broad range of simulated situations. The visual feedback from each model helps to build trust in the autonomy through either a controlled systems engineering simulation or physics based scenario simulation. Additionally, each model provides value to the overall design team in different ways, where the MBSE model acts as a systems engineering hub for prototyping modifications and analyzing logical design decisions, and the AFSIM model provides an environment to collect large volumes of high-fidelity performance data for optimization analysis. The bridges that connect the two models for data transfer ensure that both models maintain parallel NCAM designs well into the future.

Appendix A. NCAM Internal Structure and State Machines

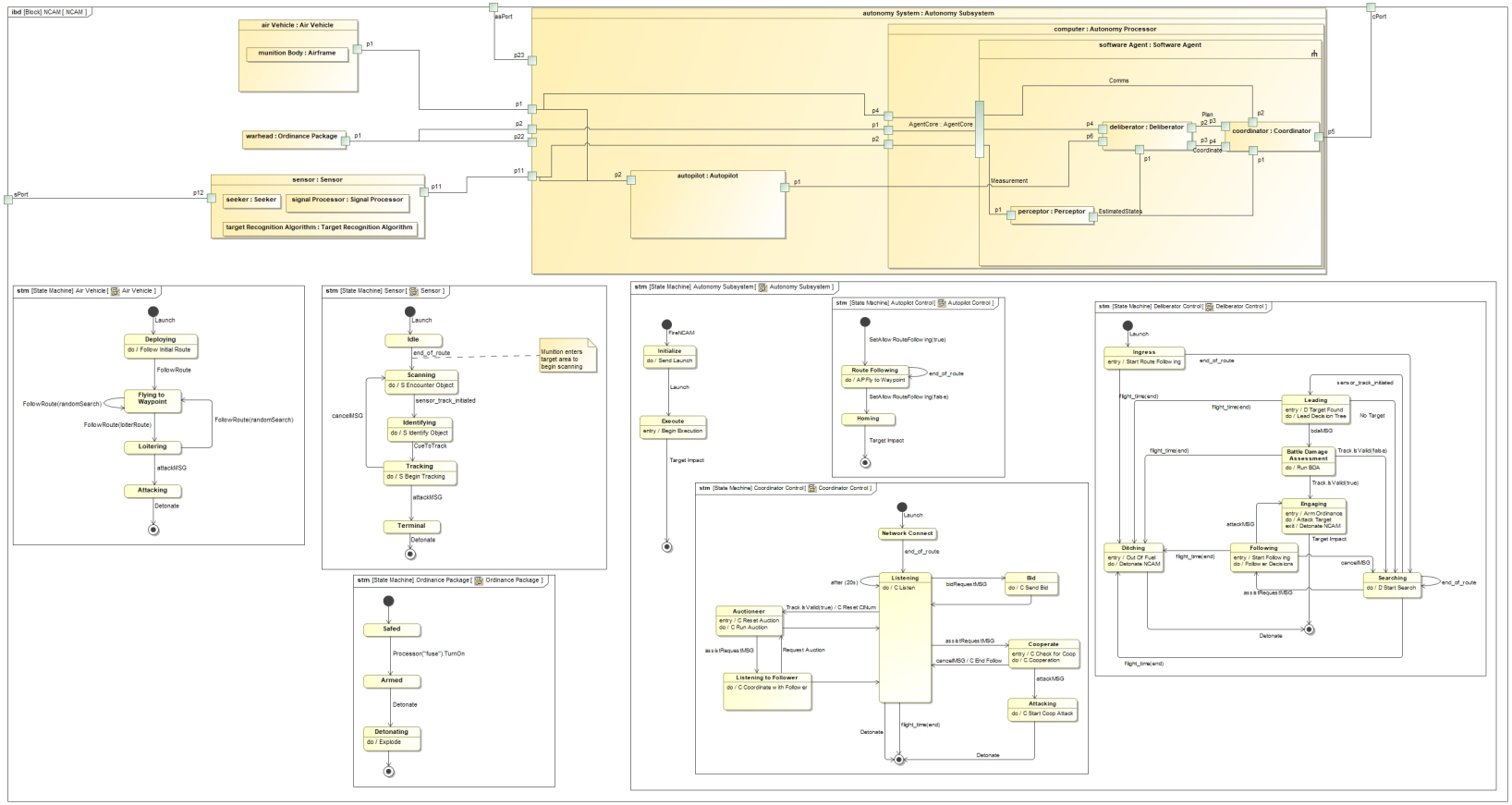

Figure 39. NCAM Internal Logical Structure

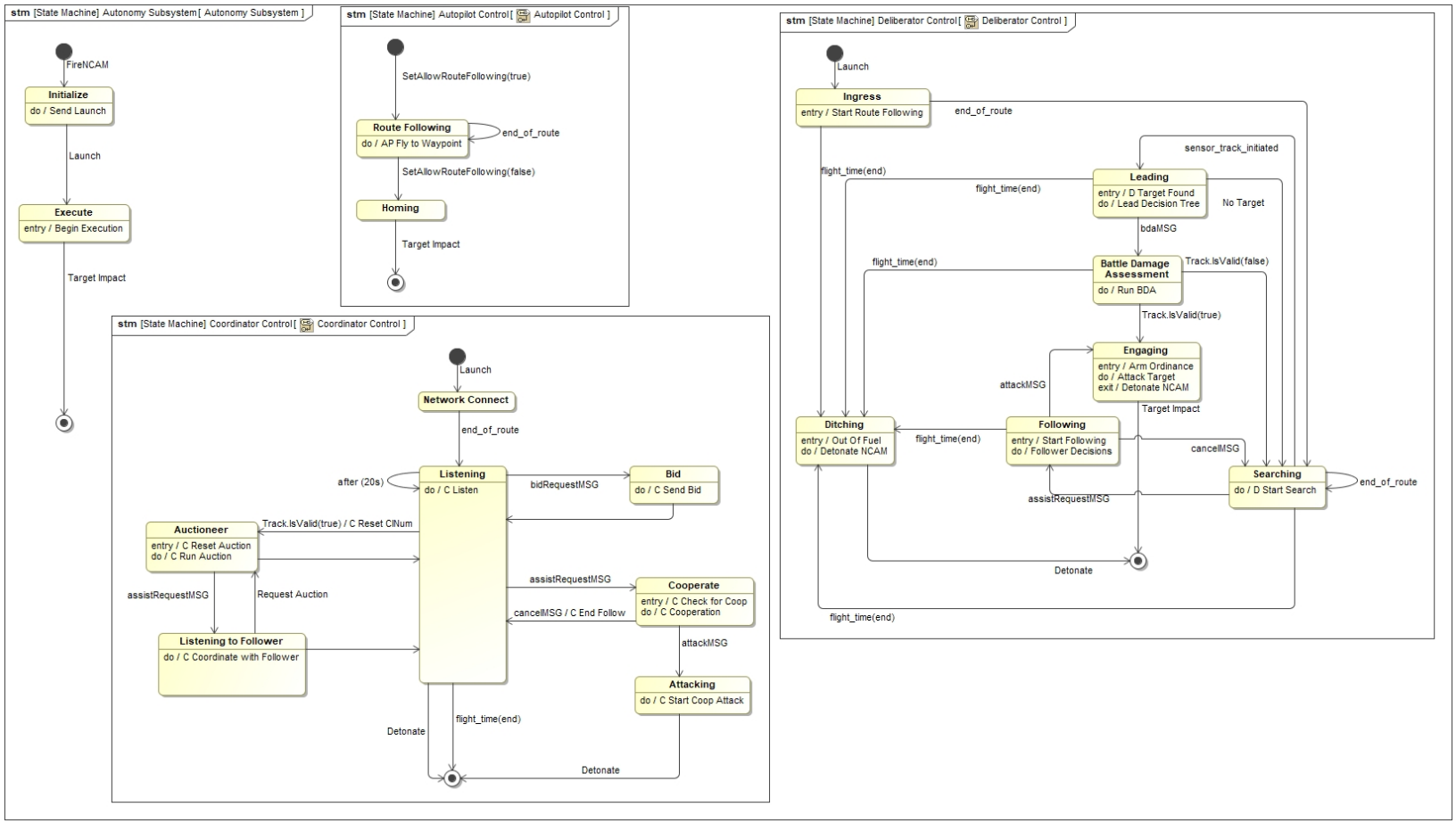

Figure 40. NCAM Autonomy Subsystem State Machine Diagram

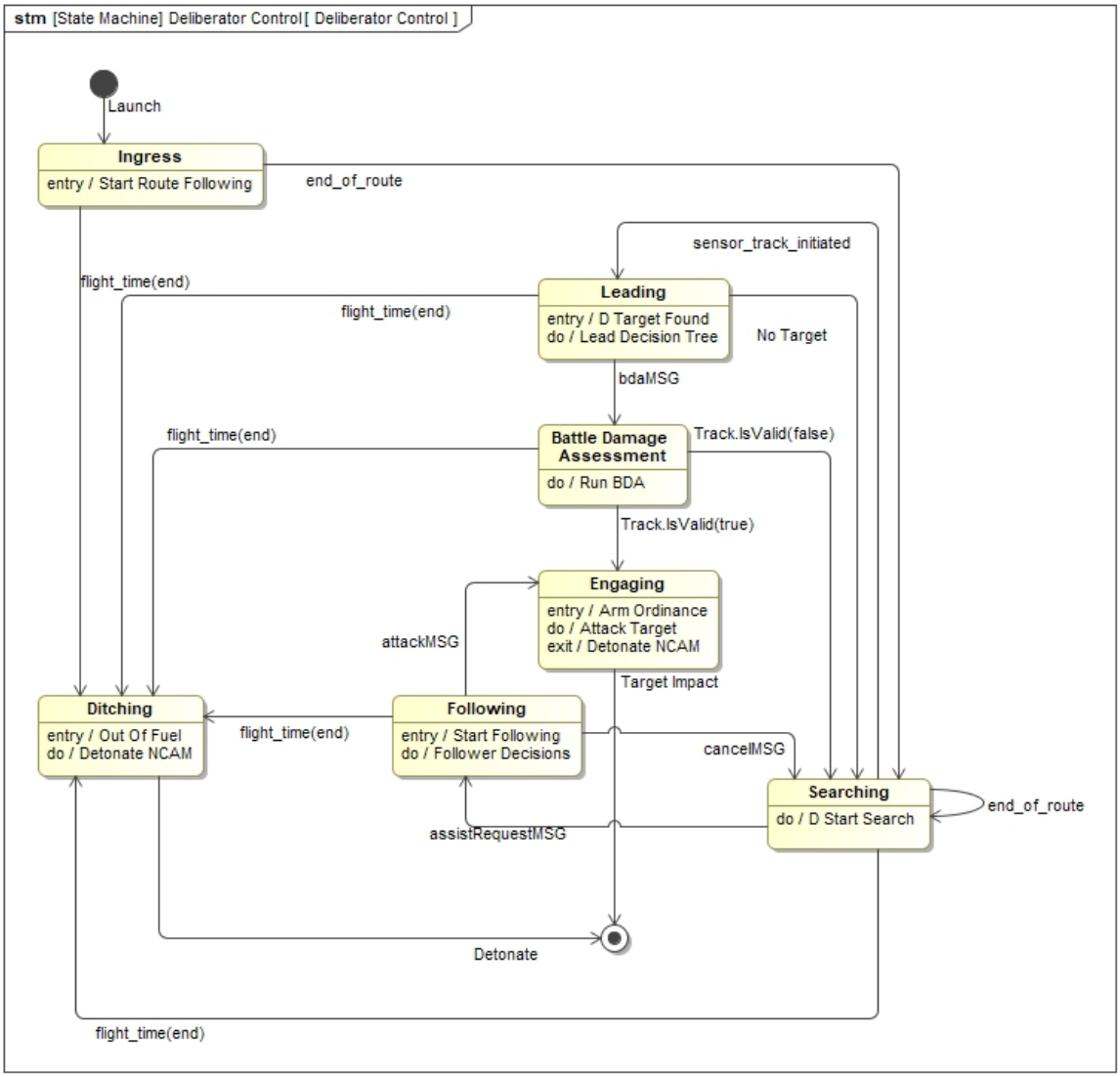

Figure 41. NCAM Deliberator State Machine Diagram

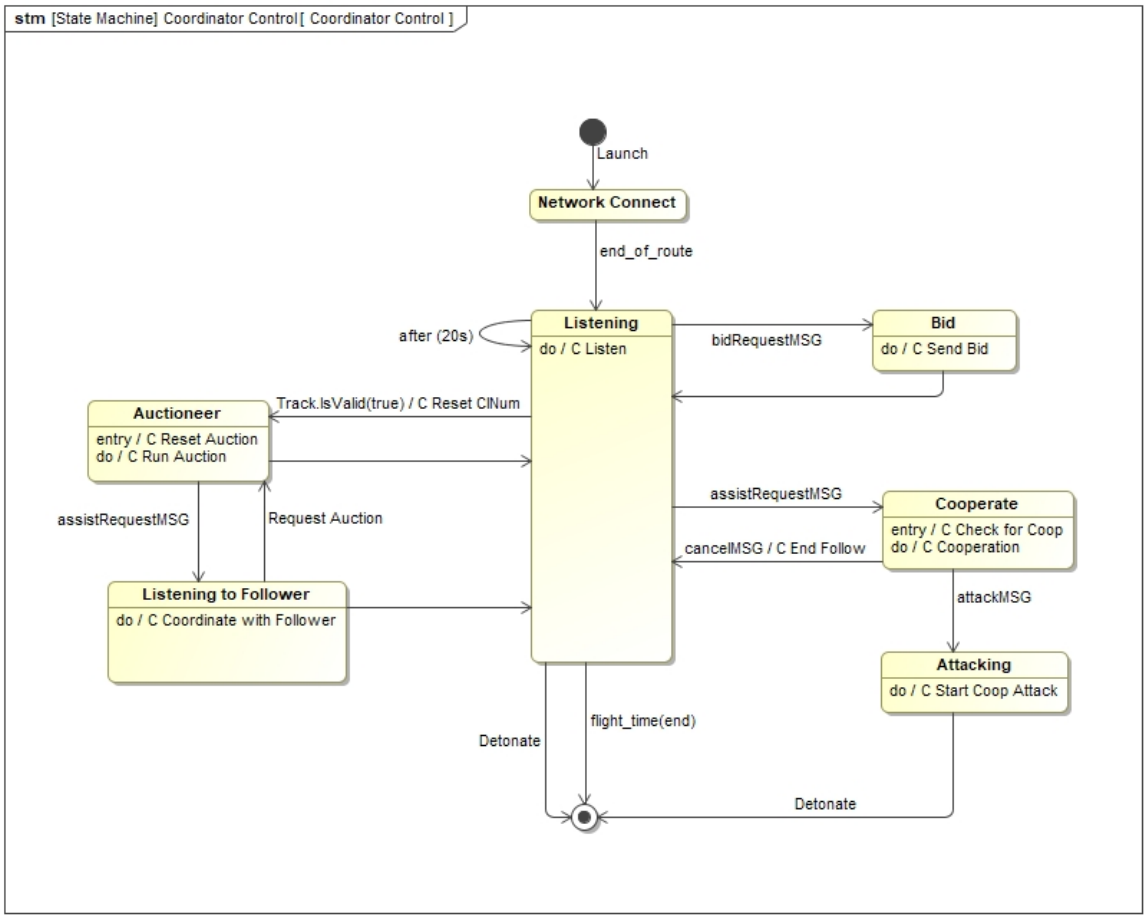

Figure 42. NCAM Coordinator State Machine Diagram

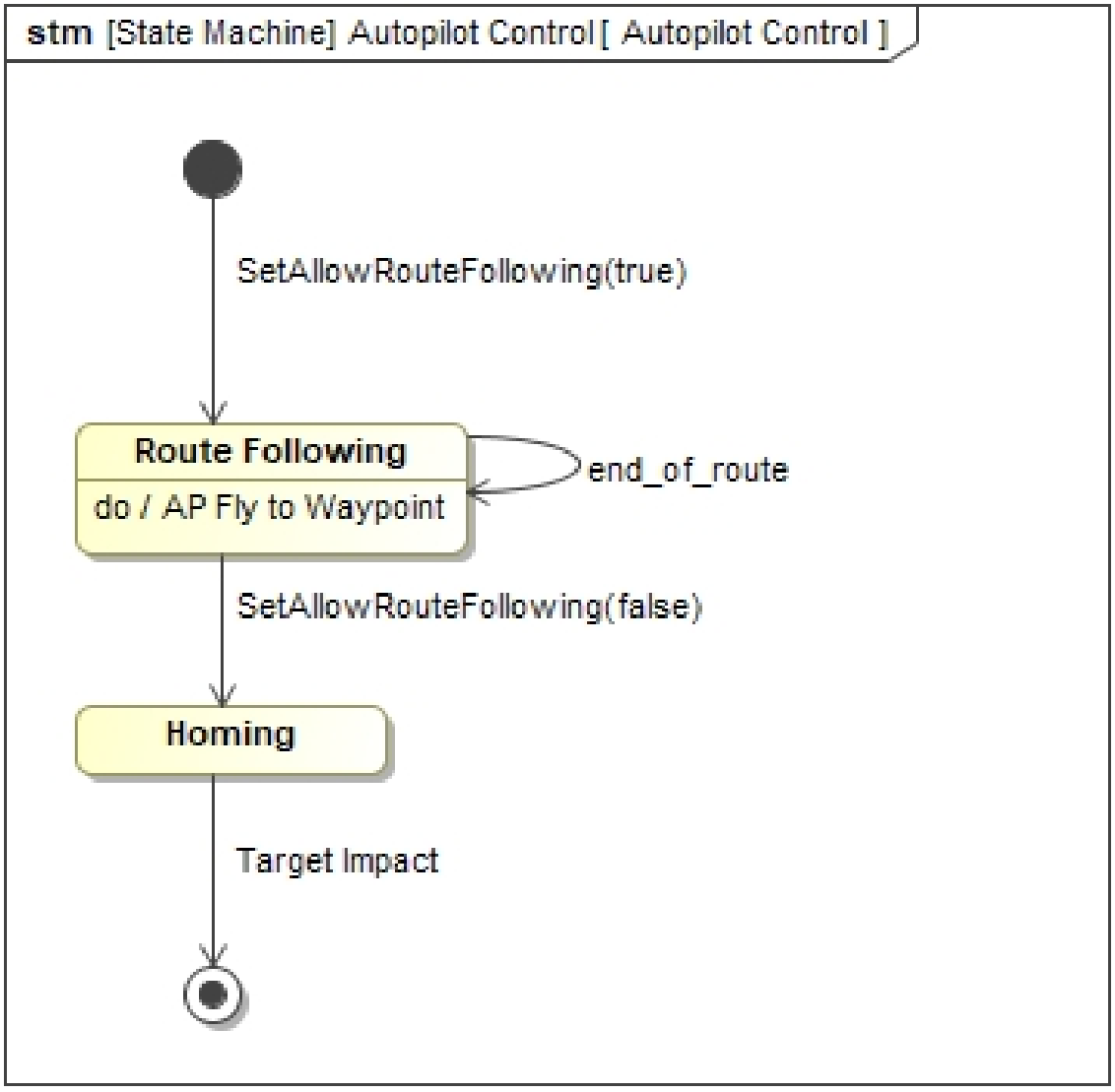

Figure 43. NCAM Autopilot State Machine Diagram

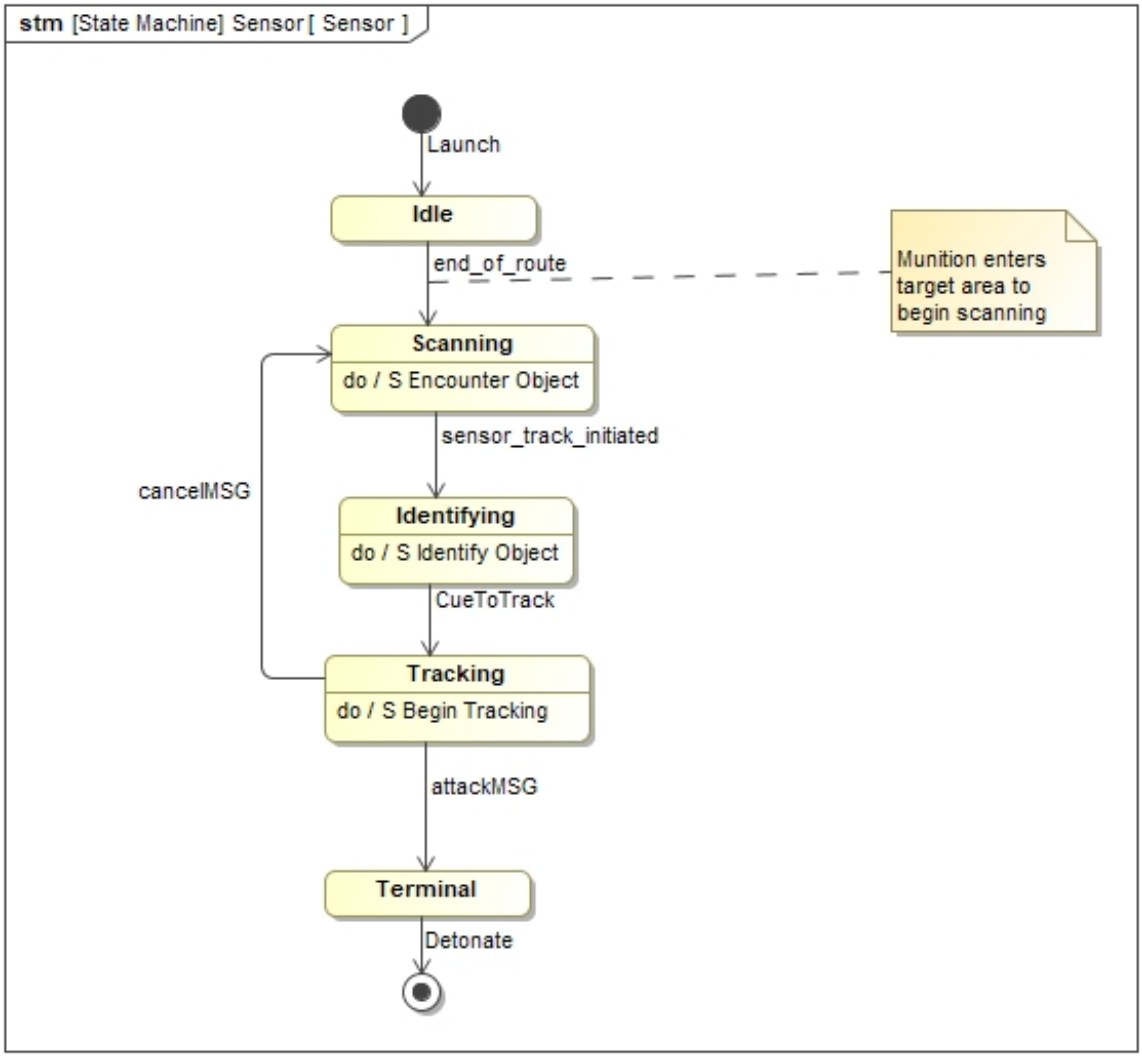

Figure 44. NCAM Sensor State Machine Diagram

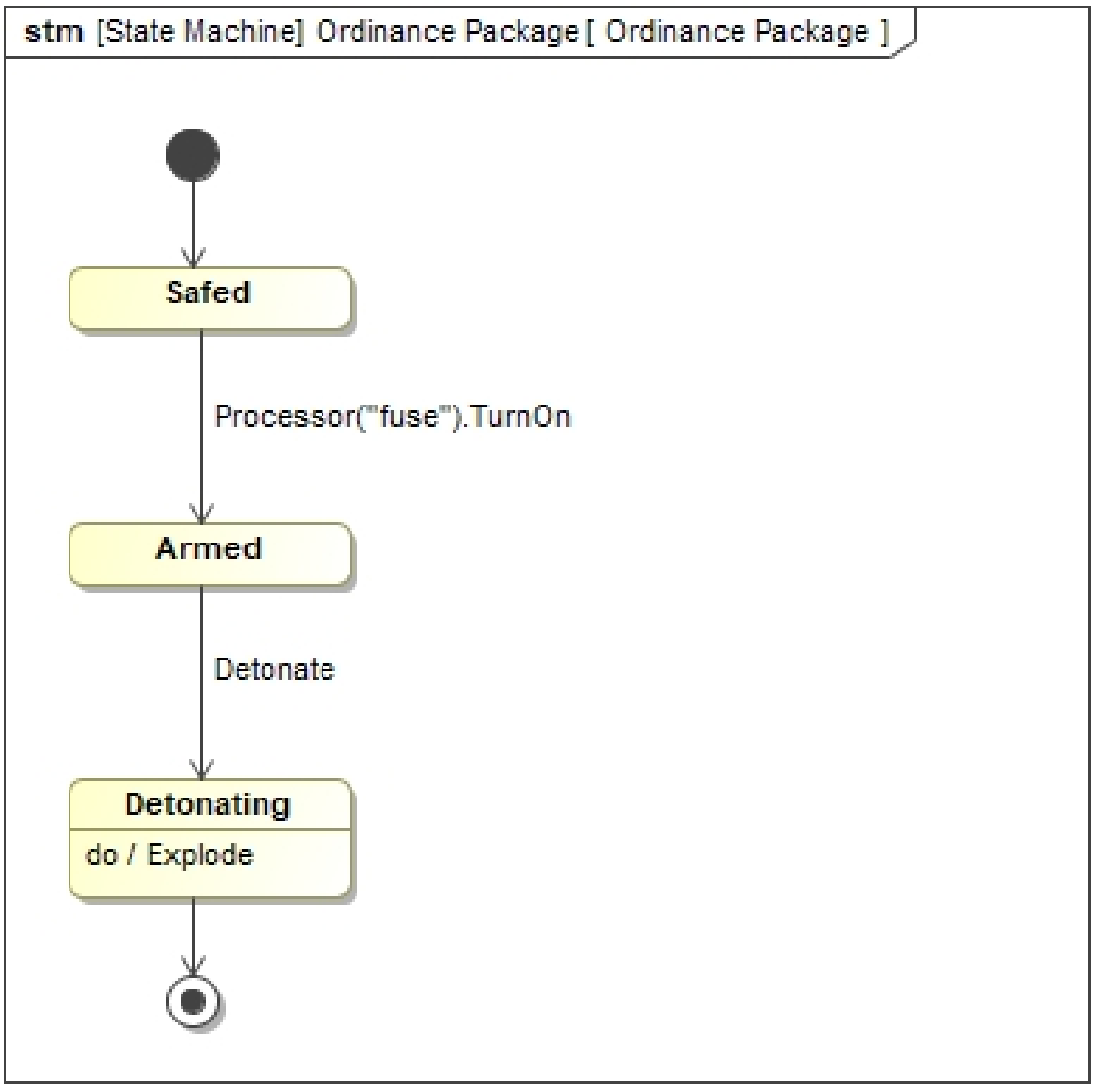

Figure 45. NCAM Ordinance Package State Machine Diagram

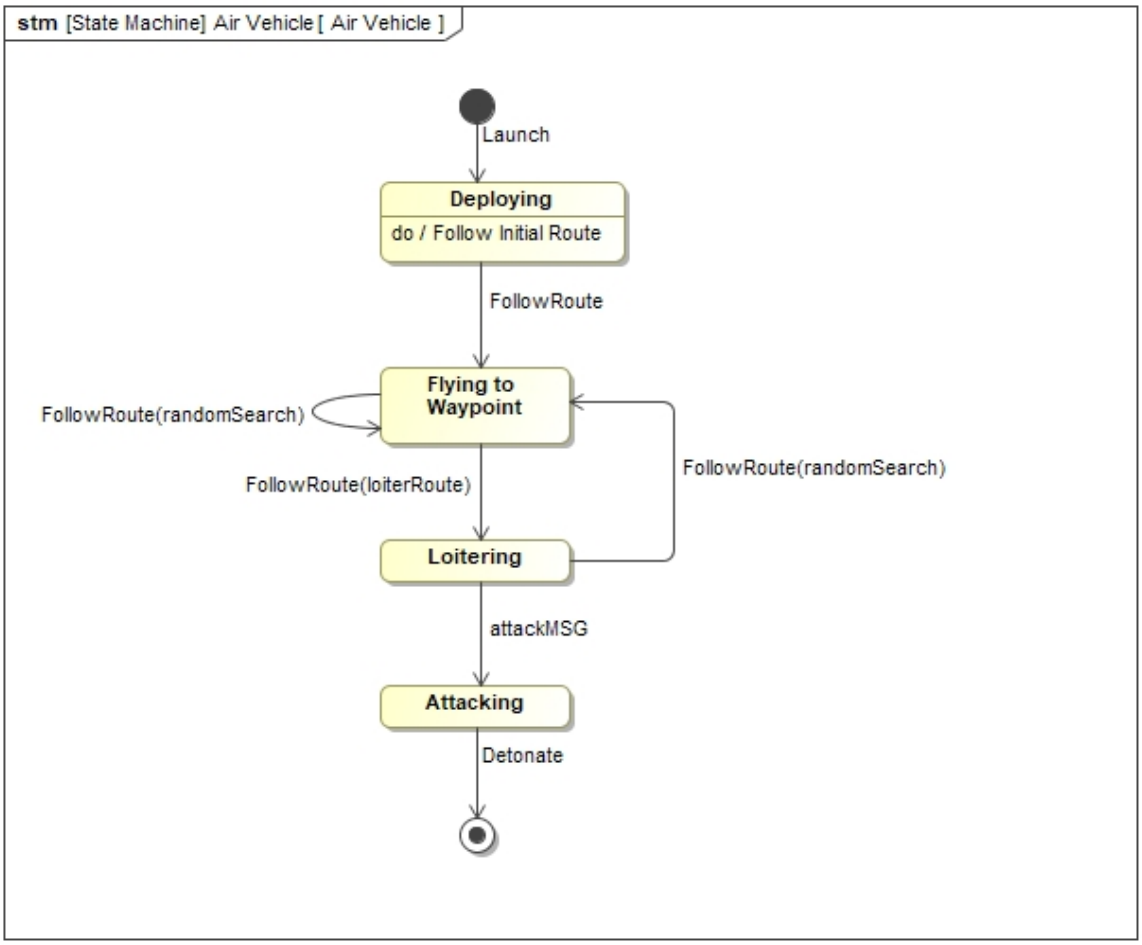

Figure 46. NCAM Air Vehicle State Machine Diagram

Bibliography

1. L. Hart, "Introduction To Model-Based System Engineering (MBSE) and SysML," in *Delaware Valley INCOSE Chapter Meeting*, 2015.

2. K. Cheney and D. King, "Development, Test, and Evaluation of Autonomous Unmanned Aerial Systems in a Simulated Wide Area Search Scenario: an Implementation of the Automnomous Systems Reference Archtitecture," Ph.D. dissertation, AFIT, 2020.

3. J. Hatzinger and I. Gertsman, "Mission Effectiveness Analysis of Networked Cooperative Munitions using Modeling and Simulation," Ph.D. dissertation, AFIT, 2022.

4. Office of the Assistant Secretary of Defense for Research & Engineering, "Technology Investment Strategy 2015-2018," *Autonomy Community of Interest (COI) Test and Evaluation, Verification and Validation (TEVV) Working Group*, no. May 2015, 2015. [Online]. Available: https://defenseinnovationmarketplace.dtic.mil/wp-content/uploads/2018/02/OSD_ATEVV_STRAT_DIST_A_SIGNED.pdf

5. B. Brackens, "Air Force to develop F-16 'digital twin'," AFLCMC, WPAFB, Tech. Rep., 2021. [Online]. Available: https://www.aflcmc.af.mil/News/Article-Display/Article/2677215/air-force-to-develop-f-16-digital-twin/

6. C. Reed, "The Foothold in the War of Cognition : The Operational Training Infrastructure Enterprise System Model," *I/ITSEC*, no. 19226, pp. 1–9, 2019.

7. T. J. Allen, "Design and Test of a UAV Swarm Architecture over a Mesh Ad-hoc Network," 2018.

8. C. A. Combs, "Verification of Autonomous Systems : Developmental Test and Evaluation of an Autonomous UAS Swarming Algorithm Combining Simulation , Formulation and Live Flight," 2019.

9. Curtis E. Lemay Center, "A IR FORCE DOCTRINE PUBLICATION (AFDP) 3-60 TARGETING," no. March, pp. 83–86, 2019.

10. "What is SysML?" [Online]. Available: https://www.omgsysml.org/what-is-sysml.htm

11. G. Peterson, "Dynamic Behavior Sequencing for Hybrid Robot Architectures," AFIT, WPAFB, Tech. Rep., 2011. [Online]. Available: https://doi.org/10.1007/s10846-010-9535-3

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24–02–2022	Master's Thesis	Sept 2020 — Feb 2022

4. TITLE AND SUBTITLE

Networked Cooperative Autonomous Munitions Parallel Modeling Utilizing Model-Based Systems Engineering

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Reed, Christopher R., Capt, USAF

5d. PROJECT NUMBER

AFIT-ENV-MS-22-M-250

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

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

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENV-MS-22-M-250

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

Air Force Research Laboratory Munitions Directorate
203 Eglin Blvd, Eglin AFB, FL 32542

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RW

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

An example of a high-complexity system is a swarm of Networked Cooperative Autonomous Munitions (NCAM) that prioritize wide area search and multiple view target confirmation. First, this research discusses methods toward building behavioral models within a Model-Based Systems Engineering (MBSE) tool. Then, this research presents the parallel modeling effort of NCAM in two environments: the MBSE model in Cameo Systems Modeler, and a physics-based model in the Advanced Framework for Simulation, Integration, and Modeling (AFSIM). Each digital model in its environment provides distinct benefits to the stakeholders of the design process, so the models must present consistent and parallel information. Thus, this research also presents automated methods to translate design information between models. Overall, the pair of models working in concert build trust with decision making authorities through understanding of the autonomous processes through systems cognition and digital scenario simulation.

15. SUBJECT TERMS

Digital Engineering, Model-Based Systems Engineering, Autonomy Behavior Modeling, Parallel Modeling

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	95	19b. TELEPHONE NUMBER *(include area code)* (937) 255-6565; david.jacques@afit.edu