2022-09-30

Modeling Expeditionary Advanced Base Operations in the Combined Arms Analysis Tool for the 21st Century (COMBATXXI)

Balogh, Imre; Blais, Curtis L.; Reeves, David; Stork, Kirk

Monterey, California. Naval Postgraduate School

NPS-MV-22-001

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

MODELING EXPEDITIONARY ADVANCED BASE OPERATIONS IN THE COMBINED ARMS ANALYSIS TOOL FOR THE 21ST CENTURY (COMBATXXI)

by

Dr. Imre Balogh, Dr. Curtis L. Blais, David Reeves, and Kirk Stork

30 September 2022

Prepared for: Capabilities Development and Integration
Operations Analysis Directorate
Marine Corps Combat Development Command
Quantico, VA

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
30 September 2022	Technical Report	29 July 2021	31 March 2022

4. TITLE AND SUBTITLE

Modeling Expeditionary Advanced Base Operations in the Combined Arms Analysis Tool for the 21st Century (COMBATXXI)

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
		M3070521POYQ7RE 1761784 RCK9K

6. AUTHOR(S)

Dr. Imre Balogh, Dr. Curtis L. Blais, David Reeves, and Kirk Stork

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

Modeling, Virtual Environments, and Simulation (MOVES) Institute
Naval Postgraduate School
1 University Circle
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

NPS-MV-22-001

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Capabilities Development and Integration Operations Analysis Directorate Marine Corps Combat Development Command Quantico, VA	OAD CD&I	

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution Statement A: Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The United States Marine Corps (USMC) is undergoing organizational and operational changes to adapt to new warfighting requirements in today's world. The USMC Force Design 2030 describes new concepts, such as Expeditionary Advanced Base Operations (EABO), with a focus on reconnaissance/counter-reconnaissance and maritime interdiction. To examine and evaluate new concepts of operation, force structures, weapon systems, tactics, techniques, and procedures, as well as other adaptations for such operations, the USMC requires models and simulations that can represent the full range of variations related to these expected changes. The Combined Arms Analysis Tool for the 21st Century (COMBATXXI) is a combat simulation jointly developed by the USMC and the US Army to support modeling and analysis. Developed over the past 20 years, COMBATXXI possesses many of the fundamental capabilities needed to study these new concepts but currently lacks realistic representation in some key areas, such as maritime surface combatants needed for examining critical aspects of the new role of maritime interdiction. Such representation requires platform identification, targeting, and assessment of damage that can lead to determination of their continued ability to perform operational missions. The purpose of this study is to examine new warfighting concepts related to EABO and to identify relevant modeling approaches using the COMBATXXI simulation. The study describes a modeling approach, initial implementation of that approach in COMBATXXI, and preliminary evaluation of the utility of the model for supporting scenarios and studies relevant to the new USMC concepts of operation. The study concludes with recommendations for follow-on work to further improve or employ the developed capability.

15. SUBJECT TERMS

Simulation, Combat Simulation, Expeditionary Advanced Base Operations, EABO, Anti-Access / Area Denial, A2/AD, Ship Damage Modeling, Loitering Munitions, Unmanned Systems, COMBATXXI

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	UU	93

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Dr. Imre Balogh	831-656-3805

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Ann E. Rondeau President	Scott Gartner Provost

The report entitled "Modeling Expeditionary Advanced Base Operations in the Combined Arms Analysis Tool for the 21st Century (COMBATXXI)" was prepared for and sponsored by the Marine Corps Combat Development Command Operations Analysis Directorate Capabilities Development and Integration.

Distribution Statement A: Approved for public release. Distribution is unlimited.

This report was prepared by:

Dr. Imre Balogh Director, MOVES Institute	Dr. Curtis Blais Faculty Associate - Research
David Reeves Faculty Associate - Research	Kirk Stork Faculty Associate - Research

Reviewed by:	**Released by:**
Dr. Gurminder Singh, Chairman Computer Science Department	Kevin B. Smith Vice Provost for Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The United States Marine Corps (USMC) is undergoing organizational and operational changes to adapt to new warfighting requirements in today's world. The USMC Force Design 2030 describes new concepts, such as Expeditionary Advanced Base Operations (EABO), with a focus on reconnaissance/counter-reconnaissance and maritime interdiction. To examine and evaluate new concepts of operation, force structures, weapon systems, tactics, techniques, and procedures, as well as other adaptations for such operations, the USMC requires models and simulations that can represent the full range of variations relating to these expected changes. The Combined Arms Analysis Tool for the 21st Century (COMBATXXI) is a combat simulation jointly developed by the USMC and the US Army to support modeling and analysis. Developed over the past 20 years, COMBATXXI possesses many of the fundamental capabilities needed to study these new concepts but currently lacks realistic representation in some key areas, such as maritime surface combatants needed for examining critical aspects of the new role of maritime interdiction. Such representation requires platform identification, targeting, and assessment of damage that can lead to determination of their continued ability to perform operational missions. The purpose of this study is to examine new warfighting concepts related to EABO and to identify relevant modeling approaches using the COMBATXXI simulation. The study describes a modeling approach, initial implementation of that approach in COMBATXXI, and preliminary evaluation of the utility of the model for supporting scenarios and studies relevant to the new USMC concepts of operation. The study concludes with recommendations for follow-on work to further improve or employ the developed capability.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

The Marine Corps Combat Development Command (MCCDC) Operations Analysis Directorate (OAD) runs the Marine Corps Study System (MCSS), which solicits study nominations from across the Marine Corps on a quarterly basis. Each year, there are several studies that require modeling with high-resolution combat simulations. The Combined Arms Analysis Tool for the 21st Century (COMBATXXI) is a high-resolution analytic combat simulation that has been co-developed by OAD and the US Army The Research and Analysis Center, White Sands Missile Range (TRAC-WSMR) since 1998. The combined arms simulation represents individual entities (i.e., vehicles, aircraft, riflemen, ships, landing craft, etc.) at the tactical level of operations up to reinforced battalion-sized units.

COMBATXXI offers the capability to model across multiple domains, to include amphibious operations, combined arms operations, and integrated air defense. The simulation can be used to conduct detailed sensor-to-shooter analyses to include direct and indirect fires as well as critical command, control, communications, computers, intelligence, surveillance, and reconnaissance (C4ISR) interactions. Detailed analysis of combat platforms in multiple domains and the combined arms fight are primary functions of COMBATXXI. This capability has been previously demonstrated by OAD in the Amphibious Combat Vehicle (ACV) Analysis of Alternatives (AoA), the Anti-Personnel Landmine/Cluster Munition (APL/CM) Study, Amphibious Assaults in a Contested Environment Study, the Future Vertical Lift Capability Set 3 AoA, the ACV Medium Cannon Capability Study, and the Advanced Reconnaissance Vehicle (ARV) AoA.

Currently, OAD is supporting several aspects of Force Design 2030 (Commandant of the Marine Corps 2020). New scenarios with conceptual tactics, techniques, and procedures (TTPs) are being used to conduct various analyses. Complex behaviors that encompass various threats and tactical situations are required.

The Naval Postgraduate School (NPS) Modeling, Virtual Environments, and Simulation (MOVES) Institute possesses unique technical expertise to support and extend OAD's analytic use of COMBATXXI. MOVES has developed innovative tools over the

years that have vastly improved the efficiency and effectiveness of analyst use of the COMBATXXI simulation. The NPS MOVES Institute is assigned tasks to enhance OAD's ability to more fully employ the analytic capabilities of COMBATXXI by developing and maintaining required functionality, providing technical support for conducting OAD studies, and training on analytic techniques. MOVES supports OA with development, maintenance, and enhancement of tools and capabilities such as Behavior Studio, Workbench, Observer/Sensor tool, entity and unit behaviors, and the Monterey Extensions software package. OAD provides the COMBATXXI simulation, existing behaviors, data, test scenarios, and documentation as government furnished information (GFI) for NPS use. Scenarios and associated databases for assigned tasks can be up to the Battalion Landing Team (BLT) or Marine Expeditionary Unit (MEF) level, and may include capabilities of all Marine Air Ground Task Force (MAGTF) elements (e.g., Command Element (CE), Ground Combat Element (GCE), Air Combat Element (ACE), and Logistics Combat Element (LCE)).

B. SCOPE AND OBJECTIVE

Force Design 2030 encompasses new organizations such as the Maritime Littoral Regiment (MLR) and new operational concepts such as Expeditionary Advanced Base Operations (EABO) with a focus on reconnaissance/counter-reconnaissance and maritime interdiction. The objective of the present effort is to examine new concepts related to EABO and identify relevant modeling approaches using the COMBATXXI simulation. The study describes a modeling approach, initial implementation of that approach in COMBATXXI, and preliminary evaluation of the utility of the approach for supporting scenarios and studies relating to the new USMC concepts of operation.

C. PROBLEM STATEMENT

The United States Marine Corps (USMC) is undergoing organizational and operational changes to adapt to new warfighting requirements in today's world. The USMC Force Design 2030 describes new concepts, such as Expeditionary Advanced Base Operations (EABO), requiring examination of force structures, missions, and warfighting capabilities. To examine and evaluate new concepts of operation, force structures, weapon systems, tactics, techniques, and procedures, as well as other adaptations for such operations, the USMC requires models and simulations that can

represent the full range of variations relating to these expected changes. Developed over the past 20 years, COMBATXXI possesses many of the fundamental representations needed to study these new concepts but is lacking realistic representation in some key areas, such as the representation of maritime surface combatants needed for examining critical aspects of the new role of maritime interdiction. Such representation requires identification, targeting, and assessment of damage against those platforms that can lead to determination of their continued ability to perform operational missions. Work is needed to examine new concepts related to EABO and to identify relevant modeling approaches using the COMBATXXI simulation.

D. TECHNICAL APPROACH

To address this need, the present study examines the EABO concepts and describes a notional scenario that captures key aspects of the concepts. The study examines current COMBATXXI capabilities to determine what additional or modified capabilities are needed to address the new concepts. The study describes a modeling approach (capabilities needed in COMBATXXI), initial implementation in COMBATXXI, and preliminary evaluation of the utility of the model for supporting scenarios and studies relating to the new USMC concepts of operation. The study concludes with recommendations for follow-on work to further improve or employ the developed capability.

At the direction of the sponsor, new functionality developed should be implemented without modifying existing Java code, if technically feasible. NPS MOVES must notify the OAD study sponsor and gain approval in advance of any development work that requires new code or a modification to existing code in the COMBATXXI core model. Such notification gives OAD opportunity to coordinate potential code changes with TRAC-WSMR and the COMBATXXI configuration advisory board.

E. ORGANIZATION OF THIS DOCUMENT

Chapter I provides an introduction to this study, giving background information on the basis for the work, study scope and objective, statement of the problem, and the general technical approach. Chapter II provides an overview of EABO as a conceptual foundation for the study and describes a notional scenario for purposes of identifying functional capabilities that must be attained to represent operational conditions of

interest, such as maritime interdiction (e.g., ship representation, targeting ship capabilities, assessing damage to ship capabilities, and determining continuing mission effectiveness based on the damage sustained). Chapter III shows how the capabilities needed for representation of the notional scenario can be implemented in COMBATXXI. Chapter IV walks through an example execution of the notional scenario in COMBATXXI and identifies sample study variants to examine application of the added capabilities. Chapter V presents study conclusions and recommendations for follow-on work. Appendix A is a glossary of terms and acronyms used in the report. Appendix B provides a listing of the python script used to initiate the hierarchical task network (HTN) process in COMBATXXI for executing entity behaviors in the notional scenario.

II. KEY CONCEPTS OF EXPEDITIONARY ADVANCED BASE OPERATIONS

A. INTRODUCTION

The Tentative Manual for Expeditionary Advanced Base Operations provides "a baseline of information, focused on Force Design 2030, to inform the live, virtual, and constructive experimentation that will test and refine force structure and capabilities" while also providing "an educational primer on the ideas, logic, context, and terminology associated with EABO [and] a foundation for expansion into formal naval doctrine" (HQ USMC 2021, iii).

B. FOUNDATIONS OF EABO

EABO are defined as "a form of expeditionary warfare that involves the employment of mobile, low-signature, persistent, and relatively easy to maintain and sustain naval expeditionary forces from a series of austere, temporary locations ashore or inshore within a contested or potentially contested maritime area in order to conduct sea denial, support sea control, or enable fleet sustainment" (HQ USMC 2021, 1-3, 1-4). The EABO concept "seeks to address challenges created by potential adversary advantages in geographic location, weapons system range, precision and capacity while creating opportunities by improving our own ability to maneuver and exploit control over key maritime terrain…by fully integrating Fleet Marine Force (FMF) and Navy capabilities to enable sea denial and sea control, as well as support sustainment of the fleet" (Office of the Chief of Naval Operations and Headquarters, US Marine Corps, 3).

The Tentative Manual identifies the following missions that may be assigned during EABO (HQ USMC 2021, 1-4):

- Support sea control operations;
- Conduct sea denial operations within the littorals;
- Contribute to maritime domain awareness;
- Provide forward command, control, communications, computers, combat systems, intelligence, surveillance, reconnaissance, targeting (C5ISRT), and counter-C5ISRT capability;
- Provide forward sustainment.

To perform these missions, the Tentative Manual identifies the following tasks to be performed in EABO (HQ USMC 2021, 1-4):

- Conduct surveillance and reconnaissance;
- Conduct operations in the information environment;
- Conduct screen/guard/cover;
- Deny or control key maritime terrain;
- Conduct surface warfare operations;
- Conduct air and missile defense;
- Conduct strike operations;
- Conduct antisubmarine warfare;
- Conduct sustainment operations;
- Conduct forward arming and refueling point (FARP) operations.

Many of these missions and tasks have not been the focal point of past studies using COMBATXXI and, therefore, are not well represented in COMBATXXI functional capabilities. Principal limitations fall into the maritime warfighting areas such as sea denial, maritime domain awareness, and antisubmarine warfare.

Key characteristics of EABO include the following:

- Stand-in Forces: These are forces operating in the littorals *within* a potential adversary's weapons engagement zone (WEZ).
- Mobility: "Forces conducting EABO have the organic resources and platforms sufficient to transit within a theater and conduct tactical maneuver across the seaward and landward portion of the littoral to accomplish assigned missions" (HQ USMC 2021, 1-5).
- Persistence: Forces operate forward with high flexibility and minimal support, "protecting themselves from detection and targeting" (HQ USMC 2021, 1-5).
- Low Signature: Forces operate at all times with highly managed signatures, presenting the lowest possible susceptibility to detection by adversary sensors.
- Integrated Naval Forces: Resources for EABO are allocated from joint and coalition forces as part of an integrated naval force – "integrated naval units executing assigned tasks within and from expeditionary advanced bases (EABs) are referred to as *littoral forces*" (HQ USMC 2021, 1-5).

- Cost-effective: “Forces executing EABO are small, numerous, dispersed, and relatively inexpensive and difficult to target, thus inverting an adversary’s cost-benefit calculation when deciding whether to engage and upsetting the cost-imposition strategy” (HQ USMC 2021, 1-5).

In order to determine specific functionality to add to COMBATXXI, the team laid out a notional scenario providing a framework for representation and study of the key warfighting tasks and other considerations listed above.

C. NOTIONAL EABO SCENARIO FOR EXAMINING AND ENHANCING COMBATXXI CAPABILITIES

Because EABO concepts are evolving, simulation and analysis personnel need a concrete example that can be used to tease out the full essence of the warfighting environment. One such scenario is introduced in this section to indicate essential functional capabilities that need to be provided to support EABO analyses. The scenario description includes assumptions made to simplify this initial investigation. Follow-on studies can relax any of the assumptions as appropriate to advance understanding of important aspects of the problem. Also, for purposes of this initial investigation, nations, forces, weapon systems, platforms, and other components of the scenario are described generically to facilitate open discussion of essential characteristics of the operations. Geographically, the scenario is placed in the Hawaiian Islands as a mix of land, sea, and littoral areas that are representative of places where such operations could occur, rather than into a geographic region that may imply real-world operational planning against known or anticipated threats.

1. Operational Situation

Blue assets are stationed on a chain of islands with the mission to disrupt Red maritime force "control of the sea" operations. The Red force is conducting armed patrol of the area with the intent to control any ship traffic, whether military or non-military. The Red force consists of three vessels: one destroyer and two frigates. It is assumed that these ships have anti-ship weapons with the destroyer having the primary anti-ship capability. Anti-ship weapon capabilities do not play into this scenario other than making the destroyer the priority target for the Blue force operations. The only Red surface systems that matter to this study are the sensing and defensive systems.

2. Red Force Considerations

Each Red ship has a generic long-range sensor that can detect both ground and air entities. We assume that the ships can share situation awareness (SA) information and there is a common operational picture (COP) shared by all the ships. How this is maintained is not explicitly modeled. If partial kills are modeled, such as communication kills, we can make it so that a ship cannot obtain updates to the COP from other platforms. The ships have surface-to-air missile (SAM) launchers that can be used against "traditional" air threats. In addition, they have Close-in Weapon Systems (CIWS) that are used against threats that are fairly close to the ship. When the CIWS is used, the locations of the other ships need to be taken into consideration because the system could damage a neighboring ship. Also, the placement of the CIWS on the ship matters in how it can be employed because the ship's superstructure can get in the way. The modeling needs to be done in a way where the location of the CIWS can be easily changed. For the first iteration of this scenario, Blue will not employ assets that will trigger the use of the SAMs, so those systems will not be used; only the CIWS will be employed against detected threats. Blue uses an unmanned aerial system (UAS) for reconnaissance and surveillance that could be targeted with the SAMs but for simplification purposes it is assumed that Red will not engage the UAS. Table 1 provides parametric information defining route, speed, and formation data for Red surface platforms. Table 2 provides parametric data for describing sensors.

Table 1. Red Surface Ship Route, Speed, and Formation Parameters

Parameter Name	**Description**	**Type**	**Notes**
Red route	The route the red ships will follow. The lead ship follows this and the other ships move in formation	CXXI route or list of waypoints	There is no need for red to use dynamic route finding at this point
Red speed	Speed at which ships are moving	Knots (Kts)	Between 10 and 30 Kts
Red formation	Formation for red ships	Formation type	Could use the table def from CMIS work?

Table 2. Sensor Data

Column Name	Description	Type	Notes
System name	The system type this data is for		
Max Range	The max range this system's sensor can see	Meters	
FOV	The Field of View of the sensor	degree	This can be 360 deg for ship sensors
FOV time	Time for one scan of the whole FOV	Sec	For simplicity, we assume that the FOV and field of regard (FOR) are the same
P-Detect Param	The parameters that describe the probability function. Details TBD		Future consideration: should this be a band-based description or some simple function

For the CIWS modeling we assume that once the ship's sensor has identified threats that are to be engaged by the CIWS, this info is passed to the CIWS and it uses its own integrated sensing system (e.g., fire control radar). The sensing of the CIWS is not explicitly modeled. The effectiveness of the sensor is rolled into the performance data of the system. We assume the data that is provided for the CIWS will be a range-dependent damage function against the incoming munitions based on the type of munition. The simplest damage function is a probability of kill (P-kill) value which is used in this study, but the approach will allow for partial kills to be assessed in the future. The damage is computed per burst and is based on the distance to the target at the time the burst is fired. The ship is modeled as a system-of-systems where the key components are modeled as separate entities. The CIWS is assumed to be either fully operational or not; that is, while in principle, the CIWS model could have both a sensor and a weapon, we assume that if the sensor is not functioning the CIWS cannot be employed. In addition to the P-Kill data, there is also range data based on the type of incoming munition specifying the max range at which the CIWS will engage. Other performance parameters for the CIWS include burst rate, burst duration, number of rounds, slew rate, and total number of rounds available. In addition to this, there needs to be some way to model how the CIWS knows if a target is disabled/killed. This will be based on ground truth with some probability that the assessment is wrong allowing for both false positives and false

negatives. If the target is not killed in an engagement, it can be reengaged, and a new damage assessment is calculated based on the distance at the time of the subsequent engagement. It is assumed that there is no "partial" damage of the munition by the CIWS, so no notion of cumulative damage is included in the modeling. The orientation of the CIWS barrel is explicitly represented and points towards the target being engaged (with an initial setting being level at the middle of the field of fire for the system). Since the Blue attack mode uses swarms of attacking systems, the CIWS needs to be able to engage multiple targets in sequence. As an initial methodology for how the system will determine which target to engage out of the swarm, the best approach is to attack the closest first. This is probably acceptable if the swarming entities are all of the same type but for heterogeneous swarms, other strategies might be better so the behavior needs to be designed so the selection algorithm can easily be changed. The CIWS also needs to keep track of the other ships in its unit and not fire in directions that could hit the other ships. The assumption is that the visibility will be good enough that ground truth locations of the other ships can be used to determine the no-fire area. Where the CIWS is placed on the ship impacts its operational effectiveness, so the behavior that controls the system needs to take into consideration the location of the weapon system on the ship; i.e., if the CIWS is on the port side of the ship, it cannot engage targets that are coming towards the starboard side. Table 3 lists information describing the configuration of a CIWS on a platform. Table 4 describes a particular type of CIWS for use in the simulation. Table 5 lists data describing the performance of CIWS.

Table 3. CIWS Configuration Data

Column Name	**Description**	**Type**	**Notes**
Ship name	Name of ship	String	
Subsystem #	Which subsystem this is on the ship	int	
CIWS Type	What kind of CIWS is this	String	
Min Field of Fire Angle	The leftmost angle of the allowed Field of Fire. Angle is relative to the heading of the ship.	Degrees	e.g., -180 to 0 covers the entire port side
Max Field of Fire Angle	The rightmost angle of the allowed Field of Fire. Angle is relative to the heading of the ship.	Degrees	

Table 4. CIWS Type Data

Column Name	Description	Type	Notes
CIWS Type	The name of the type of CIWS	String	
CIWS burst duration	Time to fire a burst	Seconds	Could be a distribution around a mean
CIWS burst pause	Time between bursts being fired	Seconds	Could be a distribution around a mean
CIWS rounds per burst	The number of rounds fired in a single burst	integer	Assume all bursts are same. Only needed to track usage
CIWS Slew rate	The speed at which CIWS can slew to a new target	Degree/sec	Slew calculated as angle between current orientation and direction to new target
CIWS rounds stored	This is the total number of rounds a CWIS holds	integer	The assumption is that once the rounds are used up there is no reload
CIWS target switching time	This is the time to pick a new target	seconds	Could be ignored

Table 5. CIWS Performance Data

Column Name	Description	Type	Notes
CIWS Type	The system type this row of data is for	String	
Target Type	The type of the target this data is for	String	For this scenario, this is the name of the munition that attacks the ship
Max sensing range	The max range at which the CIWS sensor can detect this type of target	Meters	This should be longer than the engagement range. It is assumed that the sensor always senses the target within this range.

Column Name	Description	Type	Notes
P-False Positive	Probability that the sensor reports a kill when the target was not damaged. 1 minus this is the probability that the damage is assessed correctly.	Real [0.0-1.0]	For simplicity, the assumption is that this is not range-dependent.
P-False Negative	Probability that the sensor reports a no-kill when the target was killed. 1 minus this is the probability that the damage is assessed correctly.	Real [0.0-1.0]	For simplicity, the assumption is that this is not range-dependent.
Num Engagement bands	The number of engagement bands for which data is defined.	Int	This table supports up to 10 bands
Max Engagement range	Max range at which the CIWS can engage this type of target with a chance of damaging it.	Meters	
Damage band 1 min range	Minimum distance for damage band 1.	Meters	
Damage band 1 P-Kill	The probability of one burst killing the target in the range band between the min damage band 1 and the max engagement range	Real [0.0-1.0]	
Damage band 2 min range	Minimum distance for damage band 2.	Meters	
Damage band 2 P-Kill	The probability of one burst killing the target in the range band between min Damage band 2 and min Damage band 1	Real [0.0-1.0]	
…	…		…
Damage band 10 min range	Minimum distance for damage band 10.	Meters	
Damage band 10 P-Kill	The probability of one burst killing the target in the range band between min Damage band 10 and min Damage band 9	Real [0.0-1.0]	

The Red ships use an integrated air defense system, so there is coordination between the ships regarding how the CIWS is employed. There needs to be a behavior that coordinates the different systems by assigning Areas of Responsibility (AORs), but each CIWS should control the details of its specific behavior. Each ship should have the

ability to operate its own weapons without external input if there is a communications failure between the ships.

As introduced above, ships are modeled as systems of systems where specific components of the ship are modeled as distinct entities that can be seen and targeted individually. The spatial separation of the components will be explicitly modeled by having each component have a different location offset from the center of the ship. For the initial formulation, movement of the subcomponents will be accomplished with "magic moves" based on the center of the ship (perhaps formation move would also work). This distributed representation will require a special behavior for sensing to account for subsystems being obscured by the ship's superstructure. The configuration info will be in tables to allow for easy changes to the ship configurations. This methodology will be side-independent. Table 6 lists the data needed to describe the configuration of subsystems on a ship.

Table 6. Ship Configuration Data

Column Name	**Description**	**Type**	**Notes**
Ship Name	The name of the ship this info is for	String	
Subsystem Type	What kind of subsystem this is	String	
Entity Name	The name in the scenario of the entity that corresponds to this subsystem	String	
Subsystem #	Numeric id for this subsystem	int	
Offset angle	Offset angle of the location of the subsystem relative to center of the ship. The angle is from the heading of the ship.	Degrees	
Offset distance	Offset distance of the location of the subsystem relative to center of the ship.	Meters	
Min observation angle	The leftmost angle from which the subsystem can be seen. Angle is relative to the heading of the ship.	Degrees	
Max observation angle	The rightmost angle from which the subsystem can be seen. Angle is relative to the heading of the ship.	Degree	

The Red ships move in formation through the area of interest. When the attack occurs, they do not take evasive actions—they continue moving on their assigned routes.

In the future, evasive actions could be added, so the behaviors that are developed for the present effort should avoid depending on this Red behavior and it should be noted whenever something is built with this assumption.

3. Blue Force Considerations

The Blue mission is to disrupt the Red operation by attacking the Red forces with swarming munitions. The Blue force is widely distributed on the islands, so communication links are critical to Blue operations. While this scenario is not focused on the details of the communications network, future developments based on this scenario could look at communications implications, so the behaviors should be developed whenever possible with this future development in mind. For this iteration, the only aspect of communications that needs to be considered is the range limitation on the comms systems. We will assume perfect communications when the systems are within the communications ranges and no communications if the communicating systems are too far apart. The communications behaviors will need to represent the full sequence of messages that flow through the Blue systems. Because of the distributed nature of the Blue force, different functionalities are spread across the force, therefore how the information gets to the attack force needs to be explicitly modeled. The Blue force consists of a reconnaissance UAS, a Blue ship serving as the Maritime Operations Center (MOC), a MAGTF Combat Operations Center (COC) / Fire Support Coordination Center (FSCC) in a ground vehicle, a long-range unmanned surface vessel (LRUSV) control station (LRUSVCS) in a ground vehicle, and 5 LRUSVs armed with swarming loitering attack munitions. The MOC, FSCC, and LRUSVCS are mobile, but do not move in this scenario. They are deployed within communications range of each other. The LRUSVCS is located where it can communicate with the LRUSVs in their initial locations. The LRUSVs initial locations are such that the Red forces are beyond the range of the munitions on the LRUSVs. To be able to engage Red, the LRUSVs need to move to a location where the munitions can be launched. Table 7 identifies parametric data defining communications capabilities. Table 8 identifies statistical parameters for communications processing delays for the scenario.

Table 7. Communications Parameters

Column Name	Description	Type	Notes
System name	The system type this data is for		
Max Range	The max range this system's comms can reach	Meters	
Needs LOS	Flag to indicate if this system's communications require line of sight or not	True/False	

Table 8. Communications Processing Times

Column Name	Description	Type	Notes
Unit Name	The name of the unit processing the message	String	
Mean Delay time	The mean time to pass a message on to the next unit	Seconds	A lognormal distribution should be used
Standard Deviation	The standard deviation used to calculate the distribution		

At the start of the scenario, the Blue force UAS is observing the area in which the Red ships are expected to move. It is flying in a search pattern traversing this area. The specifics of the path need to be an input to the behavior controlling the UAS. It has line-of-sight (LOS) sensors with a range-based probability of detection (P-detect). The P-detect should also be a function of the aspect angle of the observation. Once the UAS detects a Red platform, it passes the type, location, speed, and heading of the detected systems to the MOC. The UAS only detects the "whole" ship and passes this info to the MOC—it does not sense individual components of the ships. It is assumed that the UAS's path keeps it within communications range of the MOC. The MOC passes the information to the FSCC. In this scenario, this is the only information that is passed to the FSCC, but the assumption is that other information could be coming into the FSCC as well. As the information is passed between entities, a time delay should be added to account for the processing that needs to be done before the information can be sent on. The FSCC takes the information passed in, prioritizes the targets, and passes this prioritized targeting information to the LRUSVCS. In this initial scenario, this will

always be the observed Red ships, but this step should be included since other variations and broader capabilities likely will be needed in future scenarios. The LRUSVCS then computes an attack location from which the LRUSVs can launch their munitions. This calculation needs to take into consideration the last known location, heading, and speed of the Red ships and the desired direction from which to attack the ships. This information is passed to the deployed LRUSVs which all move in formation to the attack location. Table 9 identifies data defining the Blue UAS route and speed, as well as the formation for the munition swarm.

Table 9. Blue UAS and Munition Swarm Formation

Parameter Name	Description	Type	Notes
Blue UAS route	The route the Blue UAS will follow	CXXI route or list of waypoints	There is no need for the UAS to use dynamic route finding at this point. The route points can be traversed in a loop.
Blue UAS speed	Speed at which the UAS flies	Knots	
Swarm formation	This is the formation the munitions fly towards the target ship(s)		This parameter may need to have multiple components that cover both the geometry and spacing.

Once the LRUSVs are at the attack location, the loitering munitions are launched to attack the Red force. The munitions fly in a "swarm" formation as a large group but do not necessarily exhibit full swarming behavior (coordination amongst themselves). Since the munitions have the ability to loiter, once they are close to the target vessels (this range is an input parameter), they go into a loiter mode to do a final assessment of the ships and to wait for the best time and geometry to attack the target. This allows for refined targeting using the sensors on the munitions. Once at the loiter distance, the munitions send the information from their sensors back to the LRUSVCS for final targeting. The LRUSVCS sends specific targeting information back to the munitions designating what targets to attack. At this point it is not clear how the targets are allocated to the munitions, but it is assumed that multiple targets will be engaged by multiple munitions. On the final attack, the munitions move in a sea-skimming mode.

Once all the munitions have been destroyed either by hitting their target or by being shot down by the CIWSs, the scenario ends. Table 10 identifies data defining damage to a subsystem of a shop from a particular munition.

Table 10. Ship Subsystem Damage

Column Name	Description	Type	Notes
Subsystem Type Name	This is the name of the type of system being damaged	String	
Munition Name	Munition doing the damage	String	
P-Disable	The probability of disabling the system	[0-1.0]	This means that the system no longer can be used and cannot be repaired within the time of the scenario

If the attack location for the munitions is beyond communications range to the LRUSVCS, then one or more of the munitions will play the role of being an airborne communications relay. At the time the LRUSVs are sent out to the attack location, it can be determined if this relay behavior will be needed. The relay munitions are launched and loiter in an area where they can perform the airborne relay function. Given that they will not participate in the attack, it is assumed that the communications relay munitions can loiter for much longer than the munitions that attack the targets. The planning of the relay munitions should happen when the mission is planned. It is acceptable for the relay munitions to have a zero speed and just "hang" in the air. For this initial cut, the use of the relay munitions could be “faked,” by just reducing the number of available munitions by the number of relay munitions and allow the munitions to talk to the LRUSVCS even if it is beyond the communications range. Table 11 lists the data describing munitions performance for this study.

Table 11. Munitions Performance Data

Column Name	Description	Type	Notes
Munition type	Type of munition this data is for	String	
Launch range	The distance from the target at which the munition is launched	Km	

Column Name	Description	Type	Notes
Loiter Range	The distance from the target when the munitions go into loiter mode	Km	It is not clear if this should be a range or time after launch
Max loiter time	The max time the munition can be in loiter mode	Sec	After this time there must be a switch to attack mode
Max comms range	The max range at which the munition can communicate	Km	This can be to the base station or another munition
Cruise speed	Speed at which munition flies towards target	Kts	
Loiter speed	The speed the munition changes to when in loiter mode	Kts	
Attack speed	The speed at which the munition attacks the target	Kts	
Flight profile	The profile the munition flies		Future capability will be to dynamically generate the flight profile

The next chapter shows how COMBATXXI is used to represent these elements of the notional scenario.

III. COMBATXXI MODELING APPROACH

A. INTRODUCTION

Chapter III presents design and implementation of capabilities in COMBATXXI needed to represent functional elements of the notional scenario described in Chapter II. Many of the capabilities are designed and implemented as behaviors using the hierarchical task network (HTN) notation using software available in COMBATXXI through the Behavior Studio or Behavior Development Studio graphical user interface tools developed by NPS MOVES for OAD. The following sections describe the various components developed (or, in some cases, reused) for this study.

B. GEOGRAPHIC REGION

For purposes of this study, a terrain file and geographic region around the Hawaiian Islands was requested and obtained from TRAC-WSMR. Figure 1 shows a view of the map in the COMBATXXI Scenario Integration Tool Suite (SITS). Note the Blue force structure and control points are shown in the far-left panel of the image.

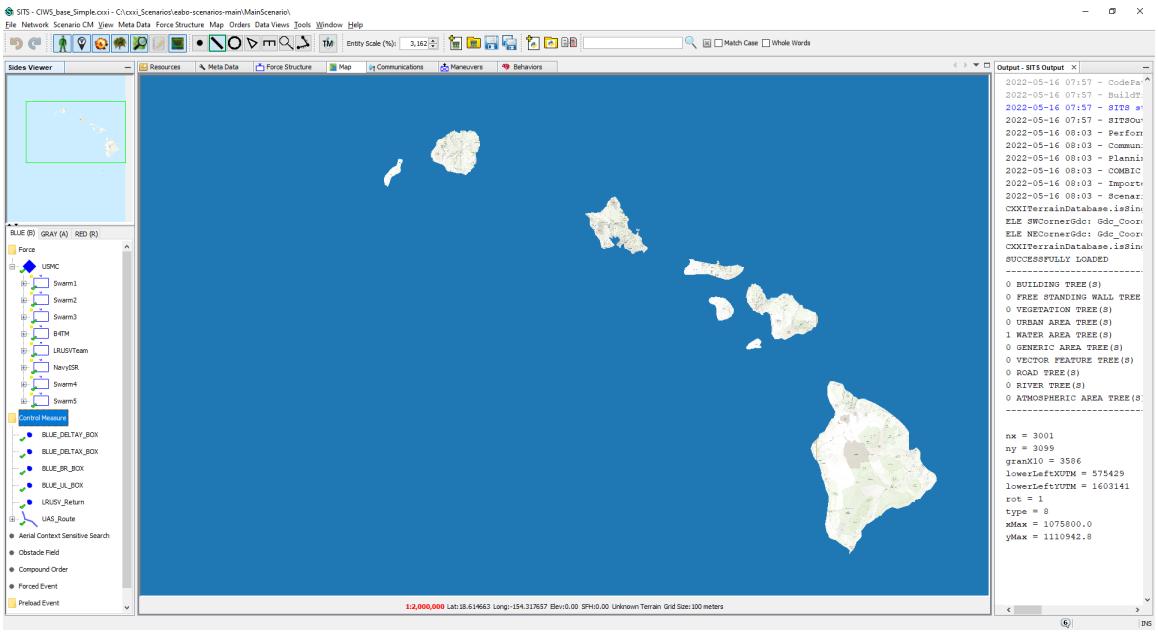


Figure 1. View of the geographic region for the notional scenario as displayed in SITS (1:2,000,000 scale).

Terrain file information is listed below:

- Valid TRNDInfo: TRNDFile:C:\cxxi_Scenarios\eabo-scenarios-main\MainScenario\terrains\BigWater_elevSRTM_12as.trnd [note: local file folder location]
- TerrainFile=BigWater_elevSRTM_12as.ele
- SWLatitude=14.500000000000009
- SWLongitude=-164.3
- NELatitude=24.5
- NELongitude=-154.3
- DPP=0.0
- DPPH=0.0
- AOSFileName=null
- BSPTreeFolder=BigWater_elevSRTM_12as_BSPTrees
- RoadNetwork=null
- EnvironmentRepresentation=Cxxi
- ClimateZone=4
- Terrain SWCorner: (GDC) Lat: 14.500000 Long: -164.300000 Elev: 00.00
- SWCorner as UTM: (UTM) Zone: 3 isNorthernHemisphere: true Easting: 575428.8963025599 Northing: 1603140.3467505418 Elev: 5.9750862419605255E-5
- Terrain NECorner: (GDC) Lat: 24.500000 Long: -154.300000 Elev: 00.00
- NECorner as UTM: (UTM) Zone: 5 isNorthernHemisphere: true Easting: 368284.0192399012 Northing: 2710205.0458879783 Elev: 1.2930948287248611E-4
- isSingleUTMGridZone: false
- Terrain Origin: (GDC) Lat: 14.500006 Long: -164.299999 Elev: 00.00

Figure 2 provides a screenshot of this region from the COMBATXXI Viewer user interface.

Figure 2. COMBATXXI map for the geographic region of this study, as displayed in the COMBATXXI Viewer user interface (also shows the Red ship movement path).

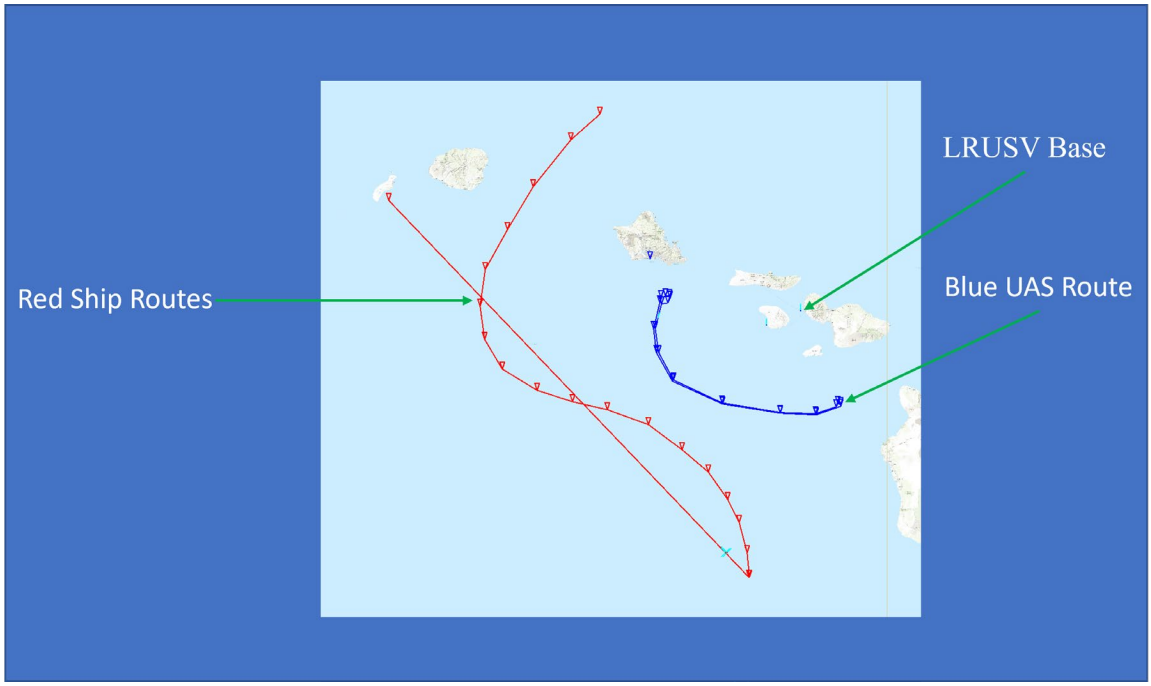


Figure 3. Scenario region showing Blue and Red movement paths and location of the LRUSV base on Maui.

C. HTN INITIATION AND SPECIALIZED PYTHON SCRIPTS/UTILITIES

The HTN behaviors, described in the next section, are initiated through a conventional jump_start.py script. The full content of this script is provided in Appendix B. In SITS, the jump_start.py script is assigned as a behavior to a notional entity named 'BehaviorBaseEntity,' whose commander is a unit in the GRAY force structure named 'mediator.' As usual, the jump_start behavior is set to start when all entities have been initialized in COMBATXXI.

The jump_start script starts the following HTNs:

(1) the commMediator HTN (no data passed) on the BehaviorBaseEntity (using the CommsMediator named behavior stack);

(2) the BlueCoordinator HTN (no data passed) on the ControlShip entity in the Blue force (using the Coordination named behavior stack);

(3) if the BLUE_SWARM_TESTING flag is set to True, the TestingDriver HTN (no data passed) on the BehaviorBaseEntity (using the TestDriver named behavior stack), followed by scheduling of a SendDelayedLaunchEvent to each of the LRUSVs in the scenario;

(4) the RedCoordinator HTN (passing the redRoute, formationLeader, formation, shipSpeed, and startDelay values initialized in the jump_start script for a specific execution of the scenario) on the DDGCore entity in the Red force (using the Coordination named behavior stack).

The next section of this document provides a description of each HTN in the distribution of the study software, as well as identifying key Python code developed to perform specific logic in execution of the notional scenario.

D. HTN BEHAVIOR TREES

1. BasicMove

a. Description

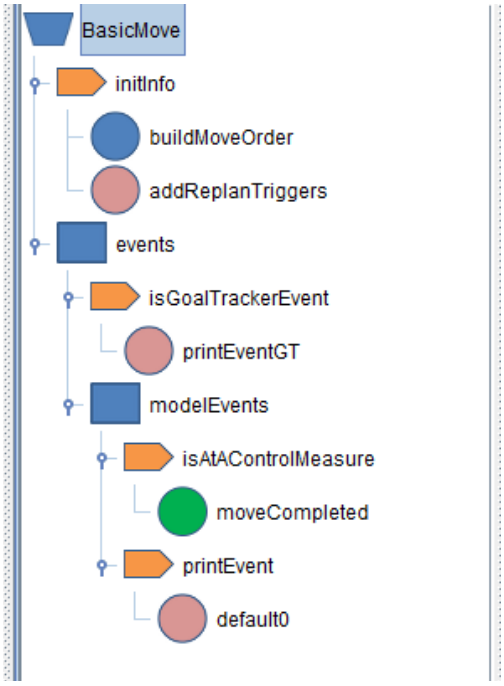
The BasicMove HTN is an existing behavior that was considered for reuse for this study. The behavior is assigned to an entity to initiate movement along a path defined by a list of locations. Instead, we took an approach that provides more explicit control over the movement on a point-by-point basis and made use of the movement network capabilities, so this HTN was ultimately not used in the final implementation

(but remains in the HTN folder path in the code in case it is needed in variations on the current scenario logic). As existing code, no further detail is provided here on the design of the BasicMove HTN.

b. Data Map

Param	Type	Comment
locs	java.util.ArrayList	list of locations to move to (Location)
speed	java.lang.Double	speed to move at m/s
toNotify	java.lang.String	name of entity to notify on completion
msgToSend	java.lang.String	name of message to send
formationName	java.lang.String	name of formation
orientation	java.lang.Double	final orientation
useOrientation	java.lang.Boolean	use orientation or not

c. HTN Structure



2. BlueC2

a. Description

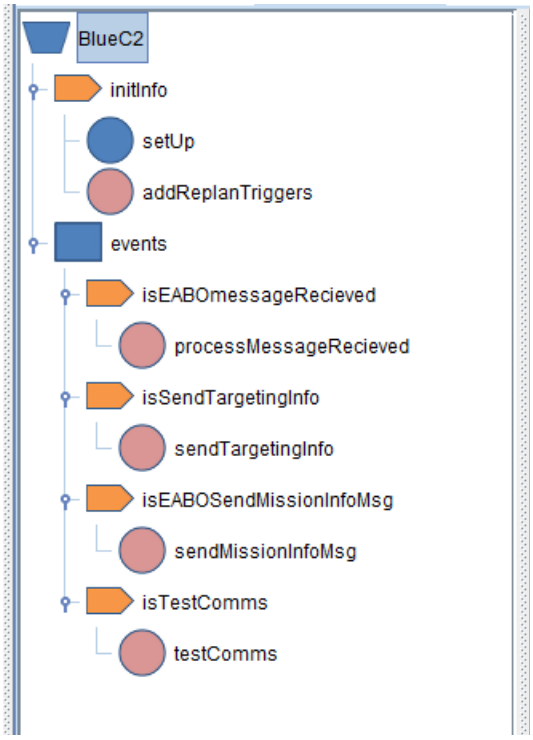
The BlueC2 HTN controls all Blue Command and Control (C2) functions. It contains functionality that may be found in a Maritime Operations Center (designated MOC0), the MAGTF COC/FSCC, and the LRUSV Control Station (LRUSVCS). No

communications between these nodes are modeled at this stage of the development but are expected in follow-on development to support EABO studies.

b. Data Map

Param	Type	Comment
swarmC2NetworkName	java.lang.String	The network on which to talk to the swarms
ISRNetworkName	java.lang.String	The network the drone uses
LRUSVC2NetworkName	java.lang.String	The network on which to talk to the LRUSVs
listOFLRUSVs	java.util.ArrayList	List of LRUSVs available to send out on a mission

c. HTN Structure



3. BlueCoordinator

a. Description

The BlueCoordinator HTN sets up the Blue forces. The original intent was that this tree would represent the higher command and coordinate the processes between

the UAS and the LRUSV Control Center, but this level of coordination was not implemented for purposes of this study.

At initialization, the BlueCoordinator HTN sets up initial definition of several communications networks using the createNetwork Python method of the commMediator class. The commMediator class handles communications events to enable developers to modify communications performance based on the situation; for example, to represent details of degraded or denied communications. The createNetwork method instantiates a new _EABOnetwork object from an internal class that provides an initial representation of considerations important to support EABO studies, such as the maximum range and the COMBATXXI propagation model to apply (i.e., perfect comms, probabilistic comms, line-of-sight restricted comms, range-based comms). Representation of these specialized networks in code separate from the COMBATXXI network representations (in contrast to the use of SITS) provides greater flexibility in designing considerations important for exploring these new operational concepts. The commMediator class also contains an inner class called _EABOMessage for representing elements of individual messages passed over an EABO network.

The BlueCoordinator HTN initialization logic instantiates three networks, one for the swarm C2 network, one for the ISR network, and one for the LRUSV C2 network (the ControlShip entity is made a member of each of these networks). Note that the parameters defining each network can be different in accordance with the configuration of systems and platforms being represented for a particular study. The BlueCoordinator initialization logic then starts the following HTNs:

(1) the loitMun HTN (passing the ControlShip entity and the swarm C2 network name) on each loitering munition entity;

(2) the LRUSV_C2 HTN (passing the ControlShip entity and the LRUSV C2 network name) on each LRUSV entity;

(3) the ISR_UAV HTN (passing the ControlShip entity, the ISR network name, the assigned IRS route, and the assigned movement speed) on the UAS1 entity;

(4) the BlueC2 HTN (passing the swarm C2 network name, the IRR network name, the LRUSV C2 network name, and a list of the

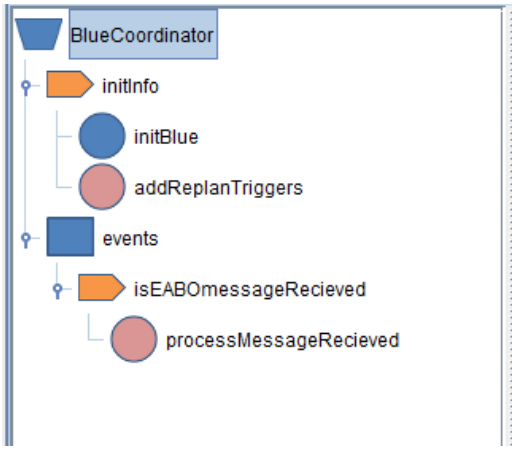
LRUSVs) on the ControlShip entity (using the "blueC2" named behavior stack).

There are no replan triggers defined for the BlueCoordinator HTN, so its processing effectively ends following its initialization (i.e., the processMessageReceived node is not implemented).

b. Data Map

The BlueCoordinator HTN has no data map.

c. HTN Structure



4. CIWS

a. Description

The CIWS HTN controls the behavior of CIWS onboard surface ships. It models both the sensing and engagement capabilities of the system. For this notional scenario, much of the processing dynamics of the simulation occurs in this HTN.

During initialization of the HTN, for the ShipObject and numbered ship subsystem passed into the HTN, the logic reads performance data on sensors and weapon systems from a number of SQL relational database tables:

(1) From a CIWSConfig database table, the minimum field of fire angle and the maximum field of fire angle. From these the logic computes a central Field of View (FOV) value between the two values and a width of FOV as the difference of the two values. From the initial orientation passed to the HTN, the logic orients the ship object and reads its location.

(2) From a SensorSpecs database table, the logic reads sensor specifications for the given ship object and numbered ship subsystem, obtaining the maximum sensor range, ability to see over-the-horizon, and a FOV width (which is constrained to the width of the FOV computed from the min and max fire angles if the sensor FOV width is larger), and a scan rate.

(3) From a SensorTGTPerformance database table, the logic reads target-specific sensor performance data, obtaining for each sensor a target profile name, the ability to see over-the-horizon, and a maximum identification range.

(4) From a CIWSPerfParams database table, the logic reads weapon system performance data, obtaining the CIWS burst duration, the CIWS burst pause, the CIWS rounds per burst, the CIWS slew rate, the CIWS number of stored rounds, and the CIWS target switching time.

(5) From a CIWS_TGTPerformance database table, the logic reads target-specific performance data for the ship, obtaining, for each target type, values of maximum number of engageable targets in range, probability of false positive kill assessment, probability of false negative kill assessment, the number of engagement bands, and the maximum engagement range. For each of the engagement bands, the logic reads in values for range and probability of kill at that range.

(6) From an EntityDamage database table, the logic reads (for the given ship object and subsystem) the value of the number of damage points to kill that subsystem.

A second part of the initialization code initializes and stores flags that will be used to control sensing and engaging, an initial (empty) target list, and the core ship entity object.

The initialization concludes with declaration of replan triggers (events being listened for):

(1) EABO_EntitySeen – scheduled when the simulation determines that an entity has been seen and needs to be processed accordingly. When this event occurs, the CIWS HTN adds the entity to the target list if it is not already on the list and if the range to the entity is less than the maximum engagement range of the weapon system.

(2) CIWS_SensorScanComplete – scheduled in the CIWS HTN startSensor interrupt node each time a sensor is started and in the ProcessFOV interrupt node to schedule the next scan interval.

(3) CIWS_StartSensor – scheduled in the ShipC2 HTN (processUpdatedObsList interrupt node) for each CIWS on any ship assigned that HTN.

(4) CIWS_StopSensor – scheduled in the CIWS HTN ProcessFOV interrupt node when the sensor process is discontinued (e.g., upon loss of a sensor) and in the ShipC2 HTN when there are no targets within twice the maximum range of the CIWS.

(5) CIWS_processTargetList – scheduled in the CIWS HTN ProcessFOV interrupt node when the mediator has determined that a target on the target list can be engaged.

(6) CIWS_EngageTarget – scheduled in the CIWS HTN selectTarget interrupt node when a target has been selected from the target list.

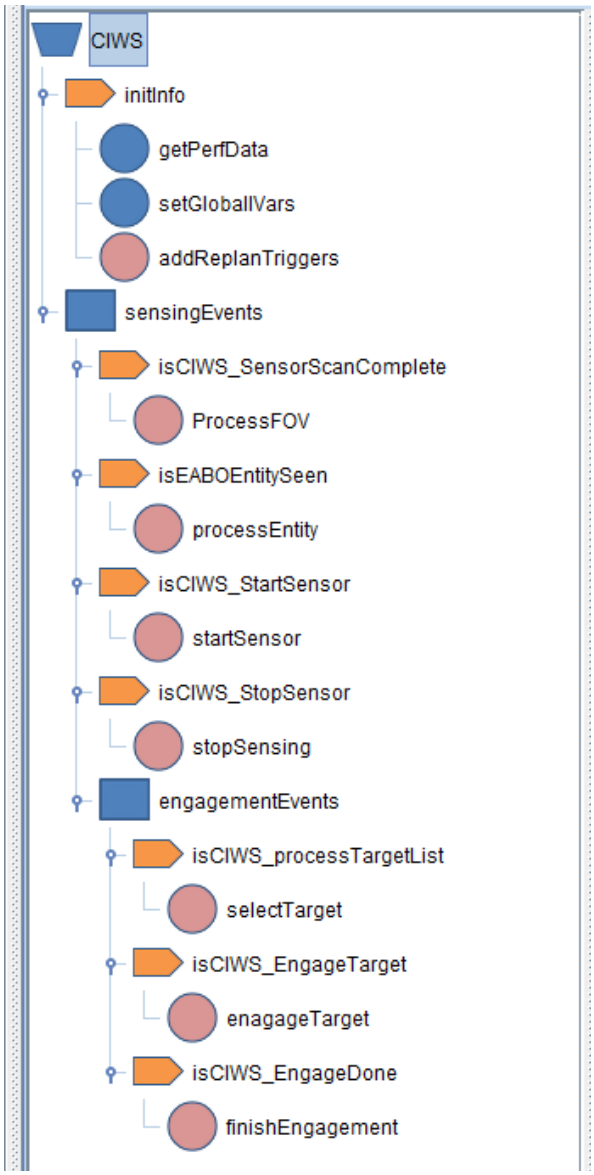
(7) CIWS_EngageDone – scheduled in the CIWS HTN engageTarget interrupt node when an engagement has started to signal completion of the engagement. The entity is reengaged (scheduling

CIWS_EngageTarget) if the target remains alive and there are CIWS rounds remaining.

b. Data Map

Param	Type	Comment
ShipObject	java.lang.Object	The python object that holds the ship of which this is a part
SubsystemNum	java.lang.Integer	The index of which subsystem this is on the ship
MinObservationAngle	java.lang.Double	The leftmost angle from which the subsystem can be seen
MaxObservationAngle	java.lang.Double	The rightmost angle from which the subsystem can be seen
InitialOrientation	java.lang.Double	The direction the subsystem faces relative to the ship heading when the ship is set up

c. HTN Structure



5. commMediator

a. Description

The jump_start.py script adds the commMediator HTN to the GRAY force BehaviorBaseEntity entity on the "CommsMediator" HTN stack. No data are passed. On startup, the commMediator behavior initializes *networkIsBusy* and *networkQueue* lists to empty lists and then defines a *sendTheMessage* method. The HTN sets replan triggers to listen for events "EABO_COM_sendMessage" and "EABO_COM_messageSendComplete" before suspending execution (interrupt node).

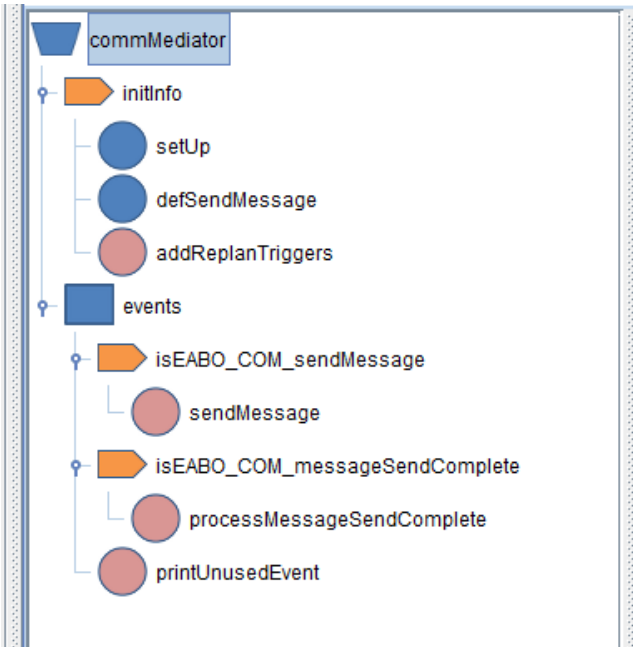
When receiving the “EABO_COM_sendMessage” trigger, the behavior is passed a *networkName*, list of *recipients*, *timeToSend*, and *message* content. If the network is not busy, the logic uses the *sendTheMessage* method to communicate with the list of recipients. Otherwise, the data are added to the *networkQueue* for later sending.

When receiving the “EABO_COM_messageSendComplete” trigger, the behavior is passed the *networkName* and reads the *networkIsBusy* and *networkQueue* data from local storage. If there are no messages to send, the networkIsBusy variable is cleared (set False). Otherwise, the next item on the networkQueue is read (and deleted from the queue) and the logic uses the *sendTheMessage* method to communicate the saved message with the list of recipients for that queued message.

b. Data Map

The commMediator HTN has no data map.

c. HTN Structure



6. EABO_MunitionInteractions

a. Description

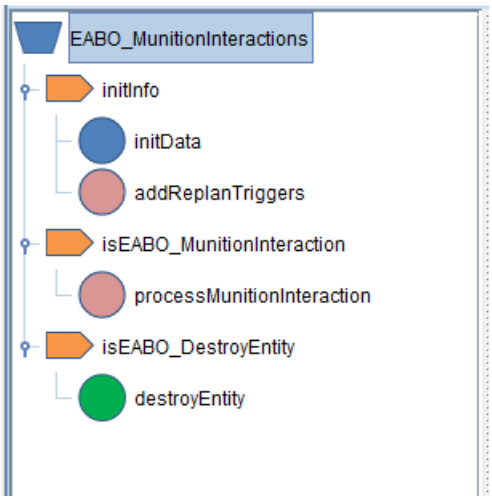
The EABO_MunitionInteractions HTN processes general EABO interactions that are not too frequent. For performance reasons, events that occur very frequently (on the order of every second or less), should have event_handlers that are directly mapped to scripts.

The EABO_MunitionInteraction HTN is assigned to each entity participating in the EABO scenario, from the logic of the EABOMediator python class created in the jump_start python script. On initialization, the initData node reads in health status information from the EntityDamage database table for the subject component of the given platform, storing that information in a local variable for use later in the logic. The initialization logic concludes with declaration of two replan triggers: (1) EABO_MunitionInteraction and (2) EABO_DestroyEntity. The EABO_MunitionInteraction event is scheduled in the EABOMediator python script methods damageShipComponent and damageLoiterMunition to determine if the component has been hit and to accumulate damage to the component. The EABO_DestroyEntity is scheduled in the processMunitionInteraction node of the EABO_MunitionInteractions HTN when cumulative hits on a component drive the health to zero.

b. Data Map

The EABO_MunitionInteractions HTN has no data map.

c. HTN Structure



7. ISR_UAV

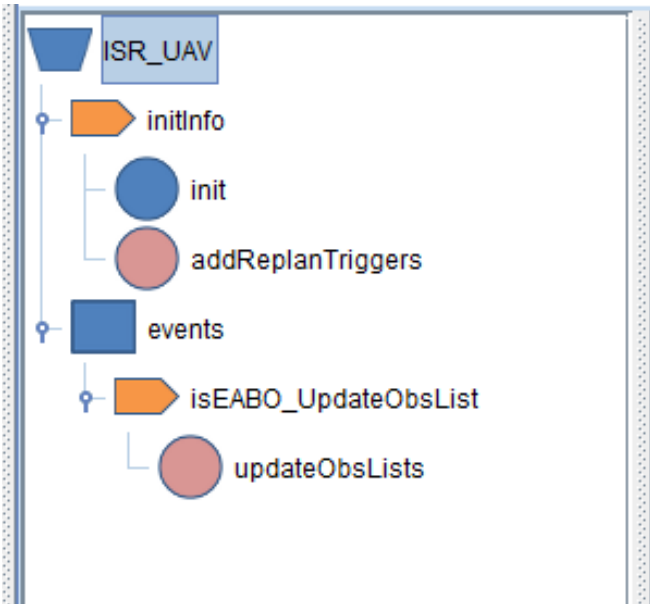
a. Description

The ISR_UAV HTN defines the behavior of the ISR unmanned aerial vehicle. This behavior is assigned to each UAV entity (there can be more than one) during initialization of the BlueCoordinator HTN. On initialization of the ISR_UAV HTN, the entity is assigned its flight route and speed, and is assigned the LongRangeSensor HTN. The logic creates an initial observedList to store observations and saves this in a local variable for later processing. Initialization concludes with declaration of the EABO_UpdateObsList replan trigger. The EABO_UpdateObsList event is scheduled in the processFOV node of the LongRangeSensing HTN upon completion of each field of regard scan by the sensor. In the subsequent updateObsLists node of the ISR_UAV HTN, the logic calls the createMessage method of the commMediator python script to send the observations to the UAV's Blue C2 entity over the assigned ISR network.

b. Data Map

Param	Type	Comment
blueC2Entity	java.lang.Object	Command and control entity for the UAV
ISRNetwork	java.lang.String	Network assigned to the UAV for ISR communications
routeName	java.lang.String	Name of the route assigned to the UAV
speed	java.lang.Double	Movement speed assigned to the UAV

c. HTN Structure



8. LoitMun

a. Description

The LoitMun HTN specifies the launch, fly-out, routing, communications, sensing, attack, and impact actions of the loitering munitions used in the scenario. This behavior is assigned to each loitering munition entity during initialization of the BlueCoordinator HTN.

On initialization, the HTN initializes state information on the munition entity, including its sensor scan rate, its target set (empty), and the "swarm" common operational picture (initially empty, but will contain what may be known by other members of a swarm of loitering munitions). The logic reads performance information from the LoitPerfParams relational database table, to include maximum endurance, launch range, maximum communications range, cruise speed, cruise altitude, final attack range, attack speed, and attack altitude. From the SwarmParams relational database table, the logic reads the horizontal and vertical spacing between members of the swarm. From the SensorSpecs relational database table, the logic reads the maximum range, field of view (FOV) width, and sensor scan rate. From the SensorTGTPerformance relational database table, the logic reads the target profile name, if the munition can sense over the

horizon and the maximum range to obtain identification on the target. From the Loit_TGTPerfromance [*sic*] relational database table, the logic reads the target type, attack profile, the apogee offset, climb angle, probability of hit, and the minimum and maximum damage per hit.

In the setupComms node, the initialization logic proceeds to add the loitering munition entity to its assigned C2 network.

In the defineFunctions node, the initialization logic defines two python script functions: (1) getMoveToDestination, returning a movement order, and (2) getSwarmDestination, returning a waypoint.

Initialization concludes with declaration of the following replan triggers. Actions taken in the LoitMun HTN when those events occur are also described.

(1) EABO_EntitySeen – this event is scheduled when the simulation determines that an entity has been seen and needs to be processed accordingly. When this event occurs, the processEntity node of the LoitMun HTN adds the entity and the range to the entity to the munintion's target set.

(2) EABOmessageRecieved [*sic*] – this event is scheduled in the commMediator HTN when an entity sends a message to another entity over a network. The LoitMun HTN processMessage node checks for use of the munition's C2 network, and broadcasts "Targeting_assignments" and "Final_Approach_Info" messages to other members of the swarm over the swarm network. If the received message is on the swarm network, three types of messages are handled: [a] for a "COP_update" message, then the message content is used to update the swarm COP; [b] for a "Final_Approach_Info" message, the processMessage node schedules a LoitMun_AdjustRoute event, passing the assigned priority ship; and [c] for a "Targeting_assignment" message, the processMessage node updates the targeting list and schedules the LoitMun_StartAttackRun event.

(3) LoitMun_Launch – this event is scheduled in the LaunchMunitions node of the LRUSV_C2 HTN after the LRUSV has reached its

assigned launch point. When the event occurs, the Launch node of the LoitMun HTN separates the loitering munitions from the transporting LRUSV and sets up the initial movement orders toward an identified goal location, applying any horizontal and vertical offsets for a swarm of munitions. Using the maximum endurance time, the logic schedules the LoitMun_Obtime_Expired event and the initial LoitMun_SensorScanComplete event (now that the munitions are on their own missions).

(4) LoitMun_AdjustRoute – this event is scheduled in the processMessage node of the LoitMun HTN on receipt of a "Final_Approach_Info" message. The event parameter passes the priority ship for this munition, from which the logic can calculate the current distance to the ship. Given the munition's attack speed over that distance, the logic computes a future estimated location to use in modifying the munition's destination

(5) LoitMun_SensorScanComplete – this event is scheduled in the ProcessFOV and Launch nodes of the LoitMun HTN to for completion of the next sensor scan. When the event occurs, the ProcessFOV node executes, passing information to the EABOMediator to determine if detections occur. When not in attack mode, the logic creates and sends a message through the commMediator python script and updates and sends the swarm COP when applicable, also through the commMediator. If not in attack mode, the logic schedules the next LoitMun_SensorScanComplete event for this munition.

(6) LoitMun_StartAttackRun – this event is scheduled by the processMessage node of the LoitMun HTN upon receipt of a "Targeting_assignment" message. When the event occurs, the startAttackRun node calls the isTargetable method in the EABOMediator python script. If that returns true, then the startAttackRun node schedules a LoitMun_ExecuteFinalAttack event

on the target entity (with a delay set to either 0.0 or the time to move at attack speed over the remaining distance to the target).

(7) LoitMun_ExecuteFinalAttack – this event is scheduled by the startAttackRun node of the LoitMun HTN. When the event occurs, the executeFinalAttack node of the LoitMun HTN sets up final attack approach information to generate a new movement representing the final attack approach angle for the munition against the target. The logic creates a COMBATXXI generic event for the impact using the LoitMun_MunitionImpact label.

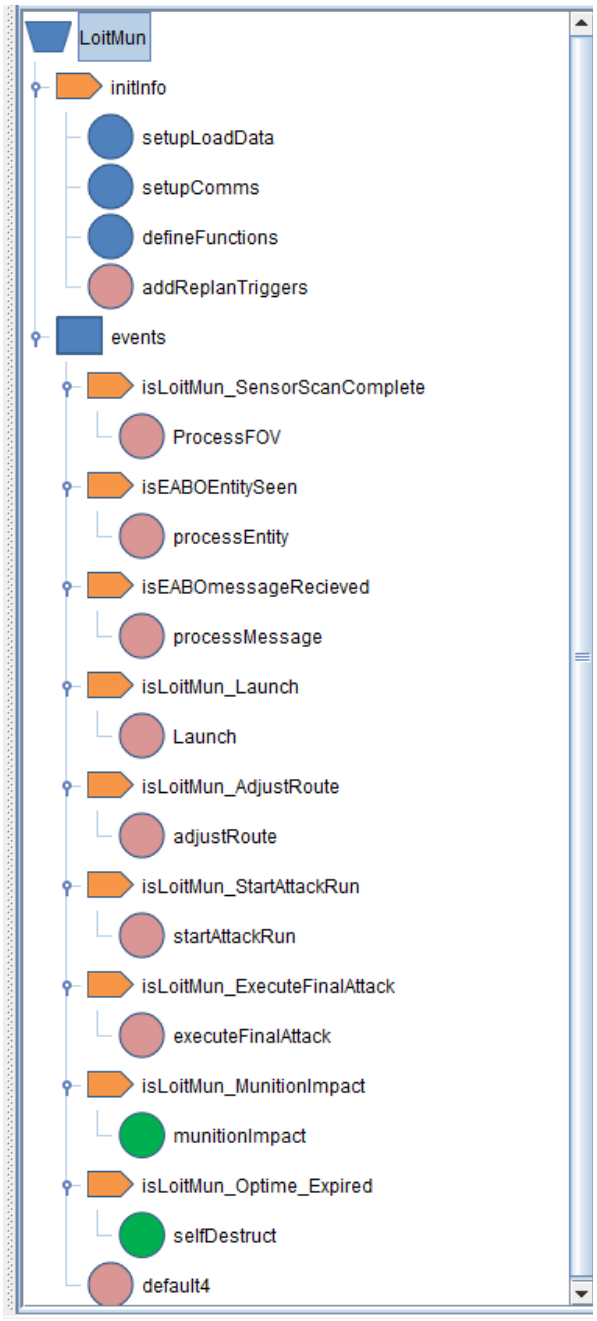
(8) GenericEvent – this event with label LoitMun_MunitionImpact is set up in the executeFinalAttack node of the LoitMun HTN to represent impact of the munition on the target. The munitionImpact node of the LoitMun HTN calls the processMunitionImpactOnTarget method of the EABOMediator python script to assess the damage from the munition against this target. This terminates the HTN execution (goal node) for this munition.

(9) LoitMun_Optime_Expired – this event is scheduled by the Launch node of the LoitMun HTN. The event occurs if the maximum endurance of the munition is reached after launch. If that occurs, the selfDestruct node of the LoitMun HTN calls the HTNUtilities method killEntityAt to mark the munition as destroyed after a random delay of up to 60 seconds.

b. Data Map

Param	**Type**	**Comment**
C2Entity	java.lang.Object	The entity that is doing the C2 for this entity
C2NetworkName	java.lang.String	Name of the network to use for C2

c. HTN Structure



9. LongRangeSensor

a. Description

The LongRangeSensor HTN defines the behavior for a long-range sensor that can be used by entities in the scenario. The LongRangeSensor HTN is assigned to the ISR UAV in the init node of the ISR_UAV HTN and to a ship in the setUpShip node of the ShipC2 HTN. Upon initialization, the LongRangeSensor HTN initializes a list of

observed entities (empty initially) and reads sensor specification information from the SensorSpecs relational database table to obtain the maximum sensor range, the sensor field of view width, and the sensor scan rate. This information, including the initial orientation of the sensor relative to the platform heading, is stored in local variables for later processing. The logic schedules the initial LongRangeSensor_FORScanComplete event to begin looking for other entities. Initialization logic also reads sensor target performance data from the SensorTGTPerformance relational database table to obtain the target profile name, the ability to detect over-the-horizon, and the maximum range at which a target can be identified.

Initialization concludes with declaration of the following replan triggers:

(1) LongRangeSensor_FORScanComplete – this event is scheduled by the init node and the processFOV node of the LongRangeSensor HTN. When the event occurs, the processFOV node initializes a list of observed entities and schedules the EABO_UpdateObsList event (processed in the ISR_UAV or ShipC2 HTNs) for possible detections. The logic sets the sensor orientation with respect to the ship or UAV operating this sensor. The logic then calls the processFOV method of the EABOMediator python script. Finally, the logic reschedules the LongRangeSensor_FORScanComplete for the next scan.

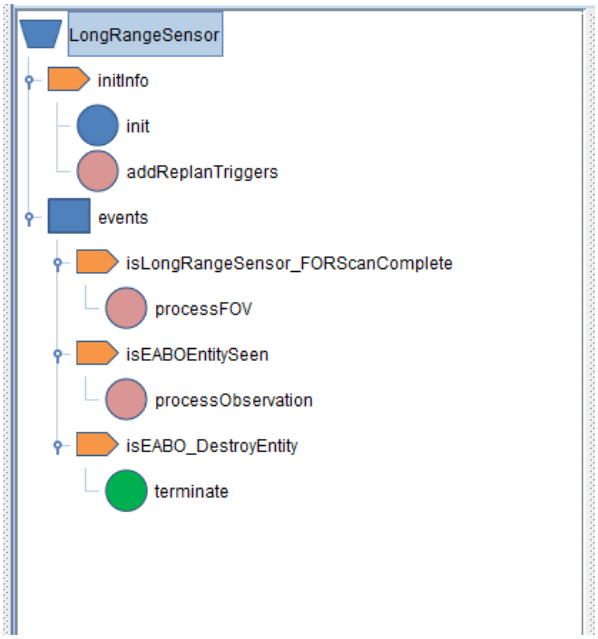
(2) EABO_EntitySeen – this event is scheduled when the simulation determines that an entity has been seen and needs to be processed accordingly. When this event occurs, the processFOV node of the LongRangeSensor HTN adds the entity to its observed list, including the entity's location and velocity.

(3) EABO_DestroyEntity – this event is scheduled in the processMunitionInteraction node of the EABO_MunitionInteractions HTN when cumulative hits on a component drive the health to zero. The terminate node of the LongRangeSensor HTN ends the behavior node (goal node).

b. Data Map

Param	Type	Comment
isPartOfShip	java.lang.Boolean	
ShipObject	java.lang.Object	The name of the entity this sensor is part of
InitialOrientation	java.lang.Double	The direction this component faces relative to the ship heading when the ship is set up

c. HTN Structure



10. LRUSV_C2

a. Description

The LRUSV_C2 HTN defines the behavior of the LRUSV control center. The HTN is assigned to the LRUSV entity in the initBlue node of the BlueCoordinator HTN. On initialization, the setUp node of the LRUSV_C2 HTN reads loitering munitions parameters from the LoitPerfParams relational database table, obtaining the munition cruise speed and the maximum endurance of the munition. These values are stored in local variables for later processing. The initialization logic then declares the following replan triggers:

(1) LRUSV_At_Launch_Point – this event is scheduled by the SendDelayedLaunchEvent node of the TestingDriver HTN to initiate testing of swarming munitions. When the event occurs, the processAtLaunchPoint node of the LRUSV_C2 HTN creates and sends a broadcast message (using the commMediator python script) over the C2 network announcing arrival at the launch location and then schedules the LRUSV_LaunchMunitions event.

(2) LRUSV_LaunchMunitions – this event is scheduled in the processAtLaunchPoint node and in the processMessage node of the LRUSV_C2 HTN. When the event occurs, the LaunchMunitions node of the LRUSV_C2 HTN obtains information on the observed target, aborting the mission if the target is beyond the maximum range for the munitions to reach at cruise speed based on the maximum endurance. Otherwise, the logic schedules a LoitMun_Launch for each munition assigned to the LRUSV. Finally, the processAtLaunchPoint node assigns the NetworkMove HTN to the LRUSV to return the platform to its base.

(3) LRUSV_Returned_To_Base – this event is scheduled (indirectly, using the SendEntityEventDelay method of the HTNBehaviors python script) by the moveComplete node of the NetworkMove HTN upon movement to the LRUSV platform's base location. When the event occurs, the missionDone goal node executes, completing the LRUSV_C2 behavior.

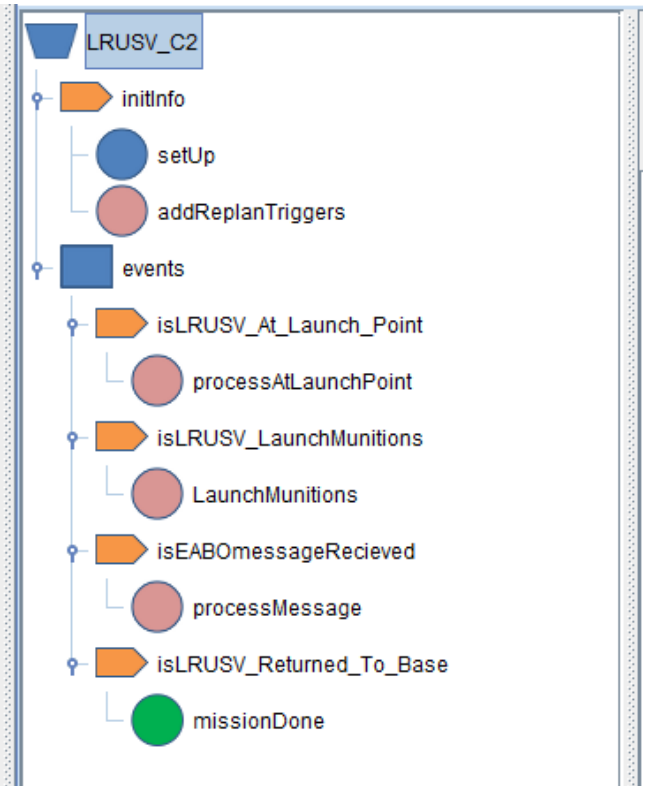
(4) EABOmessageRecieved [*sic*] – this event is scheduled by the defSendMessage node of the commMediator HTN (using the sendTheMessage python script defined therein). When this event occurs, the processMessage node of the LRUSV_C2 HTN obtains information from the message. If the received message is "MissionData," the processMessage node reads information on the observed target, sets up parameters for heading and range to the target, computes an intercept location, and assigns the NetworkMove HTN to

move to the calculated launch point. If the message is "ArrivedAtLaunchPoint," the processMessage node schedules the LRUSV_LaunchMunitions event.

b. Data Map

Param	Type	Comment
C2Entity	java.lang.Object	The entity that is doing the C2 for this entity
C2NetworkName	java.lang.String	Name of the network to use for C2

c. HTN Structure



11. NetworkMove

a. Description

The NetworkMove HTN set ups a movement order for the assigned platform. The HTN is assigned to the LRUSV in the processMessage node (to move the LRUSV to its launch point) and in the LaunchMunitions node (to return the LRUSV to

its base) of the LRUSV_C2 HTN. The HTN uses a movement network computed by the terrain reasoning utilities in the Monterey extensions to COMBATXXI (specifically, the HierarchicalTaskNetwork.PythonUtilities.AStarCM.TerrainNetwork.TerrainSearch package). On initialization, the buildMoveOrder node of the NetworkMove HTN uses the _py_search method from that package to obtain a path to the goal waypoint passed to the NetworkMove HTN (final point in the movement points). The logic sets up a move order for the assigned platform using the speed and formation passed to the HTN.

The initialization logic declares the following replan triggers:

(1) AtAControlMeasure – this event is scheduled by the COMBATXXI core model when an entity performing a movement reaches a control measure (e.g., waypoint) on the movement path. When the event occurs, the HTN checks to see if the movement has reached the final waypoint and, if so, schedules the GoalTracker_NWMoveComplete event using the SendEntityEvent method of the HTNBehaviors python script.

(2) GoalTracker_CancelMove – this event is scheduled in the processMessage of the LRUSV_C2 HTN when it is determined that the current movement will not reach an effective launch point, requiring the current movement to be cancelled so that a new move order can be set up. When this event occurs, the CancelMove node of the NetworkMove HTN stops the current movement and sends "GoalTracker_MoveCanceled" to the name of the entity passed to the NetworkMove HTN. As a goal node in the NetworkMove HTN, when the CancelMove node ends, the HTN behavior terminates.

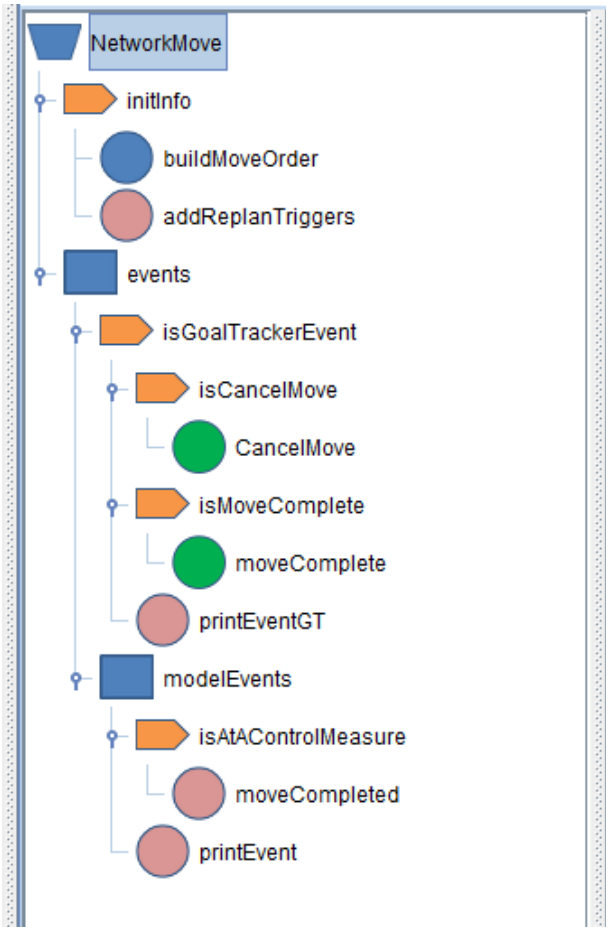
(3) GoalTracker_NWMoveComplete – this event is scheduled in the moveCompleted node of the NetworkMove HTN. When the event occurs and the HTN has been passed the name of an entity to notify, the moveComplete node of the NetworkMove HTN notifies that entity using the SendEntityEventDelay method of the HTNBehaviors python script, passing the name of the message to send (e.g., "LRUSV_Returned_To_Base"). As a goal node in the NetworkMove

HTN, when the moveCompleted node ends, the HTN behavior terminates.

b. Data Map

Param	Type	Comment
locs	java.util.ArrayList	list of locations to move to (start and end)
speed	java.lang.Double	speed to move at m/s
toNotify	java.lang.String	name of entity to notify on completion
msgToSend	java.lang.String	name of message to send
formationName	java.lang.String	name of formation

c. HTN Structure



12. RedC2

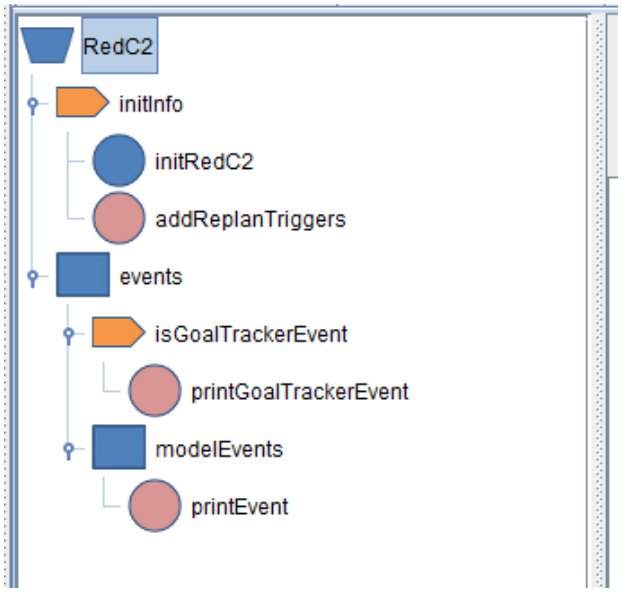
a. Description

The RedC2 HTN defines the C2 behavior for the Red force to provide high-level coordination for the whole force. The original idea was that this would allow for the Red to share information when communications systems were functional. For the current scenario, this is not important, but in a more complex situation where the Red had more complex reactions this coordination would be critical.

b. Data Map

The RedC2 HTN has no data map.

c. HTN Structure



13. RedCoordinator

a. Description

The RedCoordinator HTN initiates behaviors for the Red platforms in the scenario. The HTN is assigned to the primary Red ship entity ("DDGCore" in the study scenario) in the jump_start python script.

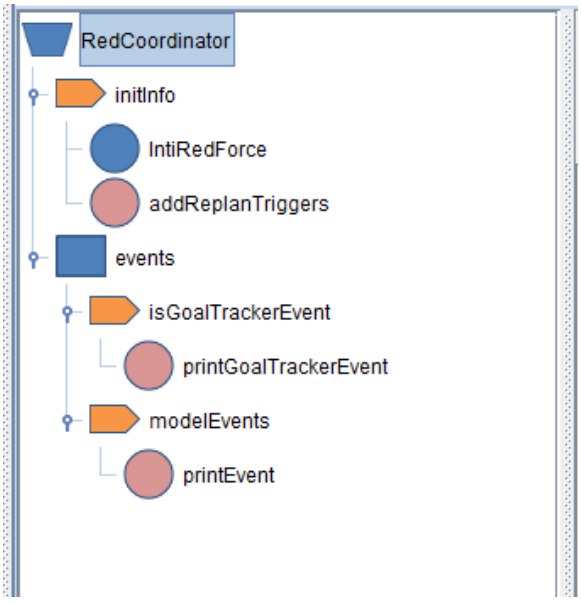
On initialization, the IntiRedForce [*sic*] node of the RedCoordinator HTN assigns the RedC2 and ShipC2 HTNs to the primary Red ship, and then assigns the

ShipC2 HTN to other Red ships participating in the scenario ("FFG1Core" and "FFG2Core" in the study scenario). The logic schedules the EABO_ExecuteMoveOrder (processed in the ShipC2 HTN) to initiate movement on the assigned route for the ship formation. There are no replan events declared in the RedCoordinator HTN.

b. Data Map

Param	Type	Comment
routeName	java.lang.String	Name of the route the ships are supposed to go on
formationLeaderName	java.lang.String	Name of the entity that leads the formation
formation	java.lang.Object	The formation object for the ships
shipSpeed	java.lang.Double	The speed at which the ships will be moving (kts)
startDelay	java.lang.Double	Allows for changing when the ships start moving to affect when the attack occurs

c. HTN Structure



14. ShipC2

a. Description

The ShipC2 HTN defines the C2 behavior for an individual ship. It controls the ship subsystems and interacts with the RedC2 HTN. This behavior is

assigned to the "hull" (or *core*) entity of the ship. On initialization, the setUpShip node of the ShipC2 HTN uses the shipCoreMapping method from the EABOMediator python script to initialize the data structure for associating components with the core ship entity. The logic reads ship configuration data from the ShipConfiguration relational database table to obtain each subsystem type, entity name for that subsystem, the subsystem number, minimum observation angle, maximum observation angle, initial orientation (relative to the ship), and the horizontal offset angle, vertical offset angle, and offset distance used to position the component subsystem with respect to the center of the platform. Each component subsystem is added to the mapping to the core entity. For each "LR_RADAR" subsystem type, the logic assigns the LongRangeSensor HTN to that subsystem entity. For each "CIWS" subsystem type, the logic assigns the CIWS HTN to that entity and reads data from the SensorSpecs relational database table to obtain the sensor maximum range and maximum sensing range for the CIWS subsystem. If the subsystem type is "COMMS," the logic assigns the ShipComms HTN to the subsystem entity. If the subsystem type is "SHIPCORE," the registerShipCore method of the EABOMediator python script is called to associate the core entity with the overall ship object. The logic finds and saves the maximum range of the CIWS sensor subsystems.

Initialization ends with declaration of the EABO_UpdateObsList and EABO_ExecuteMoveOrder replan triggers.

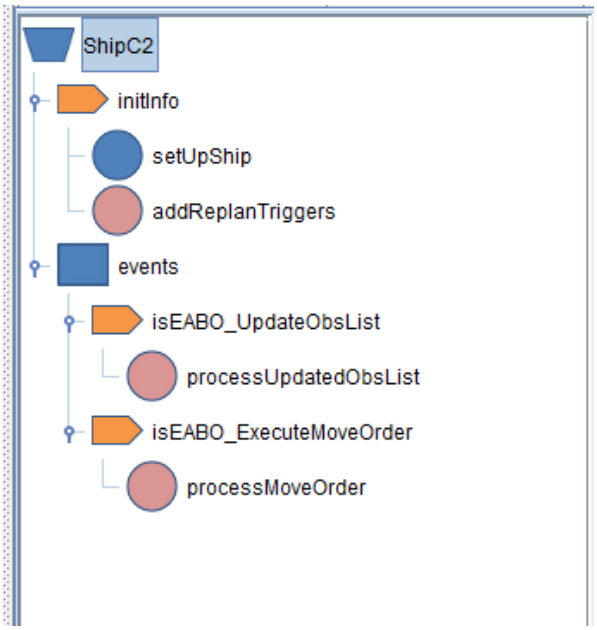
The EABO_UpdateObsList event is scheduled by the processFOV node of the LongRangeSensor HTN at the completion of each scan. When the event occurs, the processUpdatedObsList node of the ShipC2 HTN obtains the observed entity list from the event parameters. If none of the targets have been seen more recently than the all-clear time, then the logic calls the SendEntityEventDelay method from the HTNBehaviors python script to stop the CIWS sensors. Otherwise, if the CIWS is not active, then it becomes active if the distance to the observed entity is less than the maximum range of the CIWS and the SendEntityEventDelay method in the HTNBehaviors python script is called to start the CIWS sensors. If the CIWS is active and the distance to the observed entity is great than twice the maximum range of the CIWS, then the SendEntityEventDelay method from the HTNBehaviors python script to stop the CIWS sensors.

The EABO_ExecuteMoveOrder event is scheduled in the IntiRedForce [*sic*] node of the RedCoordinator HTN. When the event occurs, the processMoveOrder node of the ShipC2 node calls the moveShipsOnRoute method of the EABOMediator python script to set up movement of the Red ships based on the ordered route, formation, and speed.

b. Data Map

The ShipC2 HTN has no data map.

c. HTN Structure



15. ShipComms

a. Description

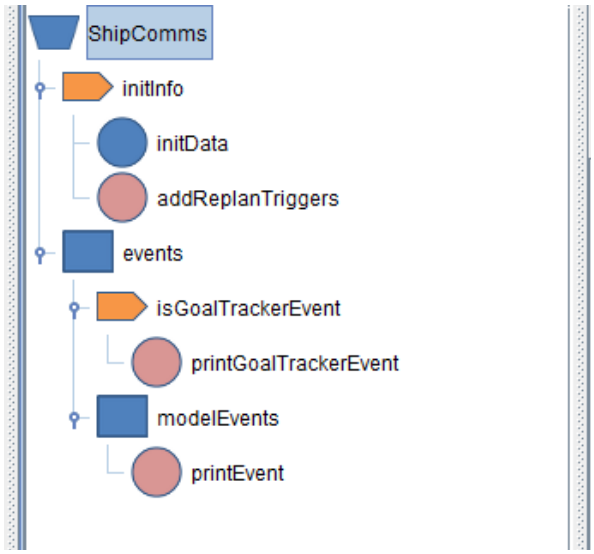
The ShipComms HTN is a shell at this time and is not assigned to any entity in the scenario. A ShipComms behavior may be useful in the future in further developing logic for EABO and other emerging operations, such as for specifying particular doctrinal logic or processes.

b. Data Map (NOTE: not applicable to this HTN)

Param	Type	Comment
ShipName	java.lang.String	The name of the ship this CIWS is part of

ShipCoreEntity	cxxi.model.objects.entity.Entity	The entity that is the core of the ship
SubsystemNum	java.lang.Integer	The index of which subsystem this is on the ship
MinObservationAngle	java.lang.Double	The leftmost angle from which the subsystem can be seen
MaxObservationAngle	java.lang.Double	The rightmost angle from which the subsystem can be seen
InitialOrientation	java.lang.Double	The direction the subsystem faces relative to the ship heading when the ship is set up
shipObject	java.lang.Object	The python object that holds the ship of which this is part

c. HTN Structure



16. ShipFormationMove

a. Description

The ShipFormationMove HTN controls the movement of a group of ships that are moving in formation. On initialization, the initMoveTree node of the ShipFormationMove HTN obtains the movement parameters from the data map and schedules the SFM_StartMovingOnRoute event. The initialization logic defines a

PassMoveInfoToFormation python script that, when called, is used to schedule the SFM_ProcessNextMoveStep event for each member of the formation. The initialization concludes with declaration of the following replan triggers:

(1) CompoundOrderCompleted – this event is scheduled by the COMBATXXI core model when an entity completes an assigned compound order. When the event occurs for the leader of the formation, the processNextMoveInRoute node of the ShipFormationMove HTN obtains the next point in the route, computes the time to reach the next point and the assigned movement speed, and calls the PassMoveInfoToFormation python script to continue the movement. Note that if the distance to the first point is more than 5000 meters, a new first point is inserted in the route that is 5000 meters from the current location along the direction of the movement to the original first point. If the movement has reached the end of the movement route, the logic schedules the SFM_EndMove event for each entity in the formation.

(2) SFM_StartMovingOnRoute – this event is scheduled in the initMoveTree node of the ShipFormationMove HTN. When the event occurs for the leader of the formation, the startMovingOnRoute node of the ShipFormationMove HTN sets up a movement order to initiate movement to the first point in the assigned movement route. Note that if the distance to the first point is more than 5000 meters, a new first point is inserted in the route that is 5000 meters from the current location along the direction of the movement to the original first point. The logic creates a move primitive order to move to the first point at the assigned speed.

(3) SFM_StopMoving – this event is not scheduled in the study scenario. The stopMoving node of the ShipFormationMove HTN is written to create a stop move primitive on the entity processing the event.

(4) SFM_ResumeMoving – this event is not scheduled in the study scenario and its logic is not implemented at this time.

(5) SFM_ChangeSpeed – this event is not scheduled in the study scenario and its logic is not implemented at this time.

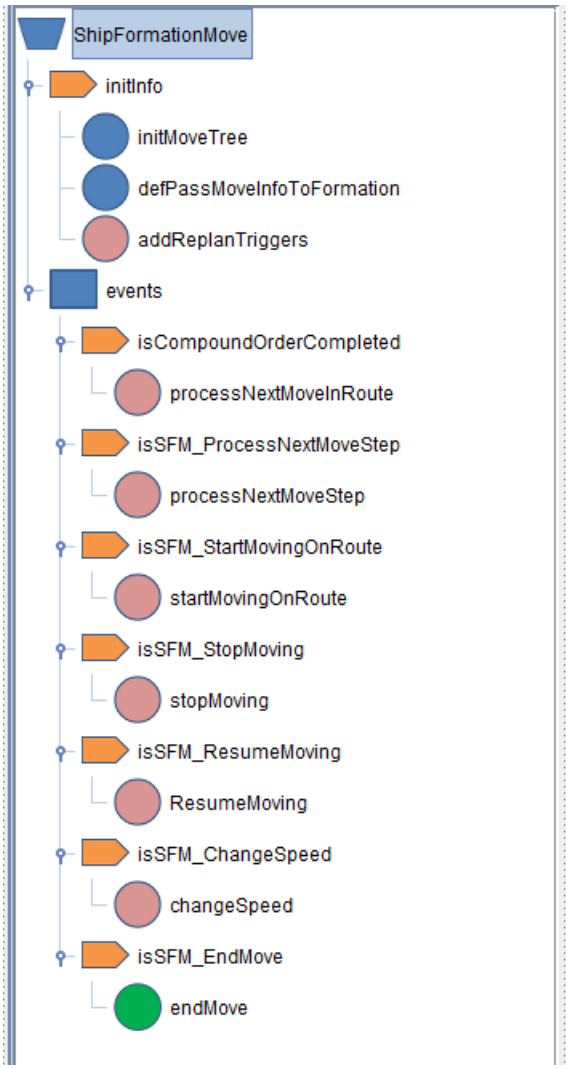
(6) SFM_ProcessNextMoveStep – this event is scheduled in the PassMoveInfoToFormation method defined in the initialization logic of the ShipFormationMove HTN. When the event occurs for ships *other than the leader of the formation*, the processNextMoveStep calls the getNextMoveLoc method on the ship formation object (defined in the shipFormation python class) to calculate the next point in the route and creates a movement order to move to that location.

Note that there is a conditional node in the ShipFormationMove HTN to respond to an SFM_EndMove replan trigger, but this trigger is not declared in the HTN. Thus, for the study scenario, the logic coded in the endMove node of the ShipFormationMove is never called and, since this is the goal node of the HTN, the behavior does not terminate. For more robust scenarios in the future, the logic can be further developed to provide additional dynamics in Red ship behaviors.

b. Data Map

Param	**Type**	**Comment**
formationLeaderName	java.lang.String	
shipFormation	java.lang.Object	the python object that had the info about the formation
route	cxxi.model.objects.features.CMPolyline	
speed	java.lang.Double	

c. HTN Structure



17. ShipGlue

a. Description

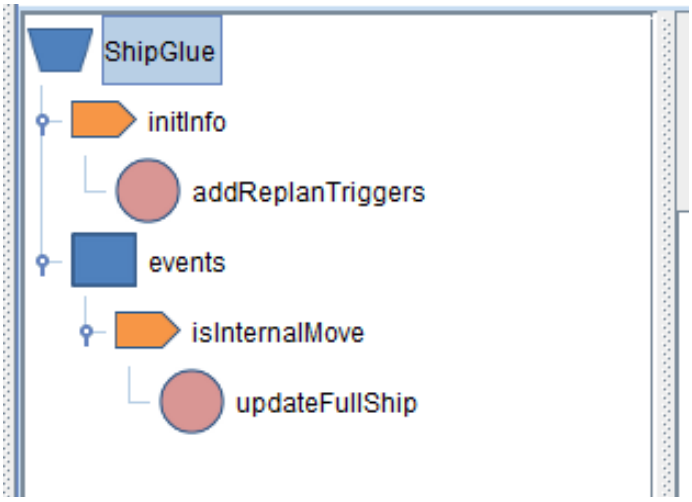
The ShipGlue HTN holds the ship entities together, as in updating the positions of component entities attached to ship platforms. (NOTE: This will probably not be needed once a more complete solution is found for modeling ships as collections of components.) The ShipGlue HTN is not assigned to any entity. Its intended functionality is implemented in the shipGlue python script as supported by the shipModel python script (where the updateFullShip method is found). The ShipGlue HTN declares a replan trigger for an InternalMove event (scheduled by the MoveFM [COMBATXXI functional module]), but the logic in the setUpShip node of the ShipC2 HTN maps this

event to the shipGlue python script (i.e., resulting in a call to the updateFullShip method of the shipModel python script).

b. Data Map

The ShipGlue HTN has no data map.

c. HTN Structure



18. TestingDriver

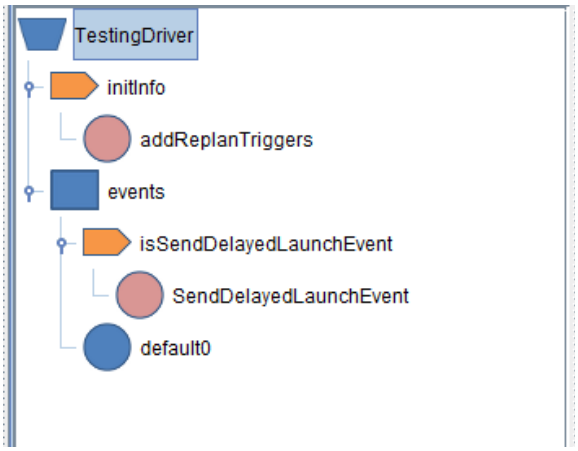
a. Description

The TestingDriver HTN is assigned to xxx in the jump_start python script for the study scenario. The purpose of the behavior is to setup information specific to a test scenario, such as the example scenario used in this study. Upon initialization, the TestingDriver HTN declares a replan trigger for SendDelayedLaunchEvent. This event is scheduled in the jump_start python script when the user seeks to conduct a test of swarming munitions launched from the LRUSV. When this event occurs, the SendDelayedLaunchEvent node of the TestingDriver HTN obtains information on the primary Red ship and schedules the LRUSV_At_Launch_Point event for the LRUSV to occur after the assigned launch delay time.

b. Data Map

The TestingDriver HTN has no data map.

c. HTN Structure



E. SUMMARY OF DEVELOPED CAPABILITIES

The software described in the previous sections implement the following key capabilities to support the notional EABO scenario:

- Targeting of individual component systems onboard Red ships by attacking munitions
- Behaviors for Red ship self-defense, loitering munitions movement and attack, and management of component systems on Red ships
- Red Close-In Weapon System (CIWS)
- Restrictions on component system operation based on ship superstructure and friendly ship positions
- Positioning of component systems on the representation of the Red surface ships
- Formation movements of launch platforms and loitering munitions
- Representation of each component of interest as a first-class simulation entity; for example see Figure 4
    - Each element is individually visible and targetable (note: the communications entity is a placeholder at this time)

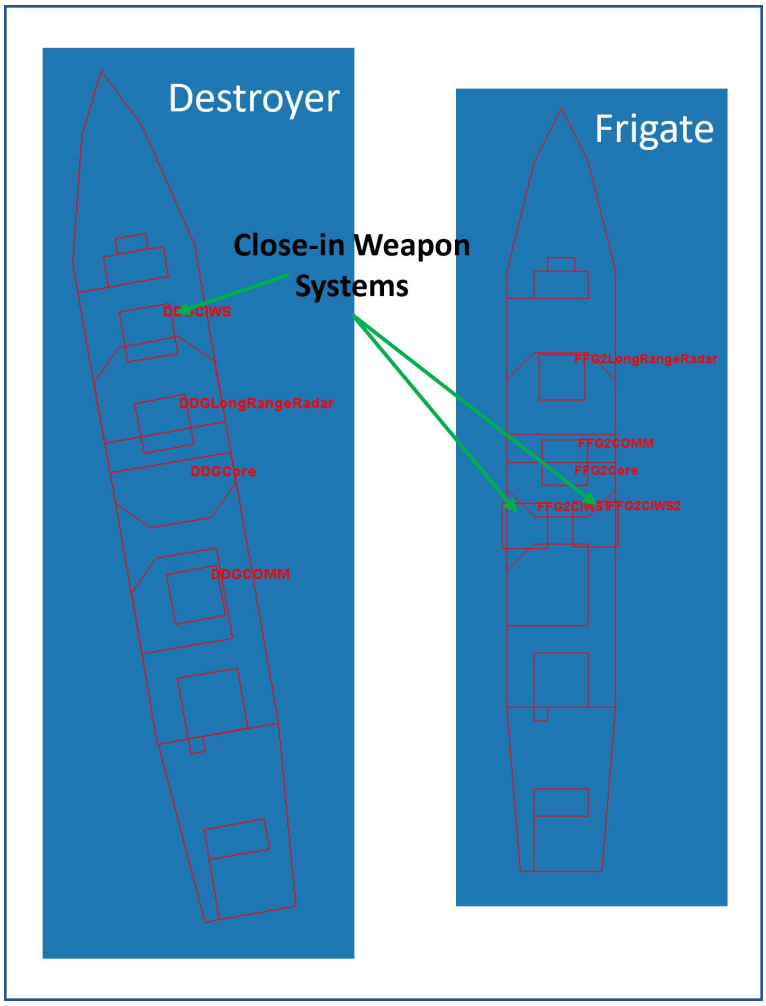


Figure 4. Example Red ship component system configuration

- Movement synchronized via internal move events
- Ship formation; for example, Figure 4 illustrates a parameterized Red ship formation

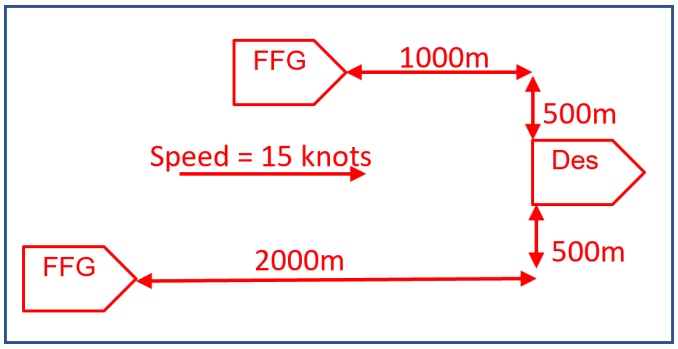


Figure 5. Example Red Ship Movement Formation

THIS PAGE INTENTIONALLY LEFT BLANK

IV. INITIAL EVALUATION OF DEVELOPED CAPABILITIES

A. INTRODUCTION

This chapter describes application of the developed capabilities to exploration of the notional scenario under various options. The examples shown here certainly are not exhaustive, merely illustrative, of the many explorations that could be performed using the new capabilities. ***Parameters used in the model runs are notional and used for proof of concept only.*** Refer to Chapter II of this report for a description of the data.

B. IMPLEMENTATION FILES

The main folder for the scenario developed for this study is titled *MainScenario*. The folder contains the following folders and files:

- *databases*: folder containing the *StandardMobility* folder, *cxxi_demo.odb*, *jcombic.odb*, and *templateCOMMS.odb* as commonly delivered with COMBATXXI distributions
- *maps* folder containing map and terrain data supporting this scenario:
  - *BigWater_bath_6as.jgw*
  - *BigWater_bath_6as.jpg*
  - *BigWater_bath_6as.prj*
  - *BigWater_bath_6as.trni*
  - *BigWater_topo_3as.jgw*
  - *BigWater_topo_3as.jpg*
  - *BigWater_topo_3as.prj*
  - *BigWater_topo_3as.trni*
  - *EARTHMAP.jpg*
  - *EARTHMAP.trni*
- *scripts* folder containing:
  - *data* folder containing
    - *grid_net.srz*
    - *grid_net.txt* file associating named edges with pairs of named grid points

        - *grid_net_points.txt* file providing latitude-longitude-elevation data for each named grid point
    - *HTN* folder containing:
        - *Scripts* folder containing:
            - *__init__$py.class*
            - *__init__.py* python script
            - *commMediator$py.class*
            - *commMediator.py* python script
            - *EABOLogging$py.class*
            - *EABOLogging.py* python script
            - *EABOMediator$py.class*
            - *EABOMediator.py* python script
            - *processBeingObserved.py* python script
            - *shipFormation$py.class*
            - *shipFormation* python script
            - *shipGlue* python script
            - *shipModel$py.class*
            - *shipModel.py* python script
        - *Trees* folder containing the 18 HTNs described in Chapter III of this report
        - *__init__$py.class*
        - *__init__.py* python script
    - *__init__.py* python script
    - *HTNBehaviors$py.class*
    - *HTNBehaviors.py* python script
    - *jump_start.py* python script
- *terrains* folder containing:
    - *BigWater* folder containing
        - *BigWater_ocean.dbf*
        - *BigWater_ocean.prj*
        - *BigWater_ocean.shp*

        - *BigWater_ocean.shx*
    - *BigWater_elevSRTM_12as_BAPTrees* folder
        - *BigWater_elevSRTM_12as_Compilation_Manifest.txt*
        - *BigWater_ocean.bspw*
        - *BigWater_ocean.linesw*
        - *BigWater_ocean.objw*
    - *BigWater_elevSRTM_12as.ele*
    - *BigWater_elevSRTM_12as.trnd*
- *CIWS_base_Simple.bbp2*
- *CIWS_base_Simple.cxxi* base scenario manifest file
- *CIWS_base_Simple.xml* base scenario definition file
- *CIWS_base_Simple_ORIG.txt*
- *CIWS_base_Simple_testbase.bbp2*
- *CIWS_base_Simple_testbase.cxxi* test scenario manifest file
- *CIWS_base_Simple_testbase.xml* test scenario definition file
- *EABO_data* Microsoft Access database file
- *EABO_data.odb* Open Database file

We emphasize that this work was exploratory in nature; as discussed in the previous chapter, the files are not "production-quality" but show what was done and generally provide starting points for further development.

C. SCENARIO EXECUTION

This section describes examines execution of the scenario specified in the *CIWS_base_Simple* COMBATXXI scenario file. The goal of these explorations into the developed capabilities is to examine sensitivity of the model to changes in certain aspects of the representation. We examine the model outputs under two variations:

- **Angle of Attack** – the direction from which the munitions approach the target ships
    - 0 deg to 180 deg tested in 45 deg steps as shown in Figure 6

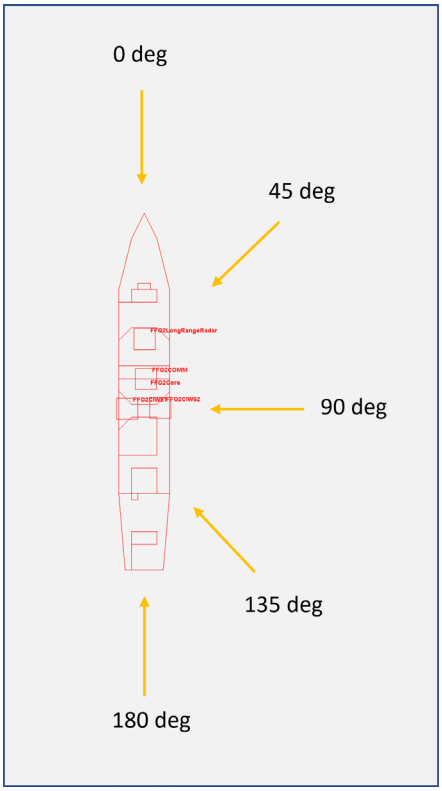


Figure 6. Angle of attack.

- **<u>Loitering Munition Formation Spacing</u>**
  - Two-layer linear formation
  - Four spacing settings:
    - Tight – 25 m horizontal, 10 m vertical – full formation width ~125m
    - Normal – 100 m horizontal, 20 m vertical – full formation width ~600m
    - Wide – 200 m horizontal, 40 m vertical – full formation width ~1200m
    - Extra Wide – 400 m horizontal, 40 vertical – full formation width ~ 2400m

Each variant was executed 100 times to generate the outputs. Figure 7 shows a screenshot of an early stage in the scenario where the scenario has been initialized and the Red ships and the ISR UAS have started their respective movements.

Figure 7. Scenario execution: Red ships and ISR UAS have started respective movements.

Figure 8 shows the scenario situation after detection of the Red ships by the Blue ISR UAV, prompting start of LRUSV movements.

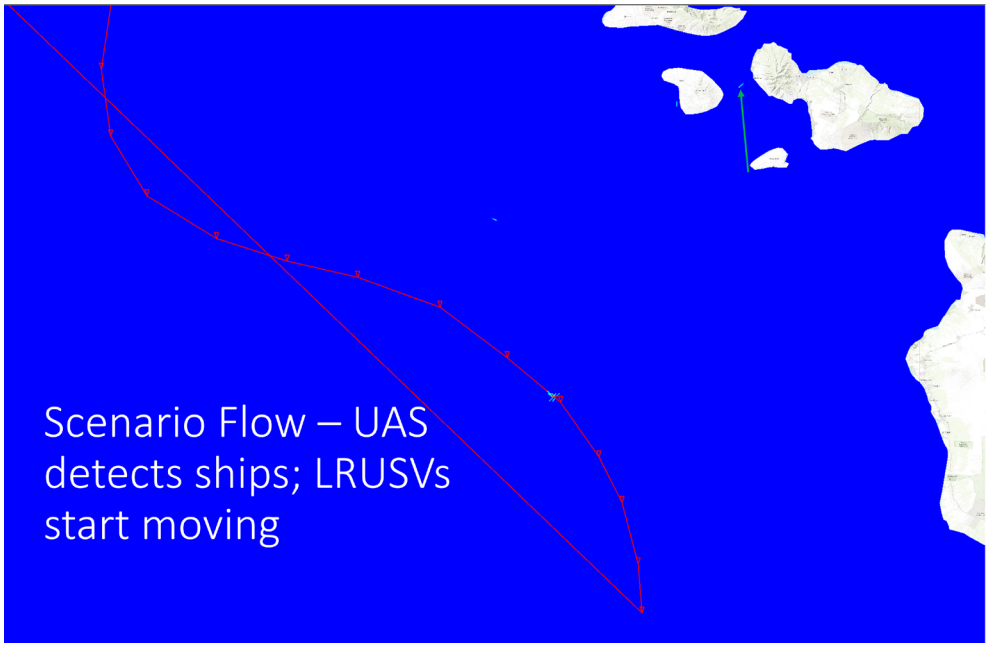


Figure 8. Scenario Execution: Detection of Red ships prompts movement of LRUSVs.

Figure 9 shows the scenario situation shortly after the LRUSVs have reached launch points and have launched the munitions.

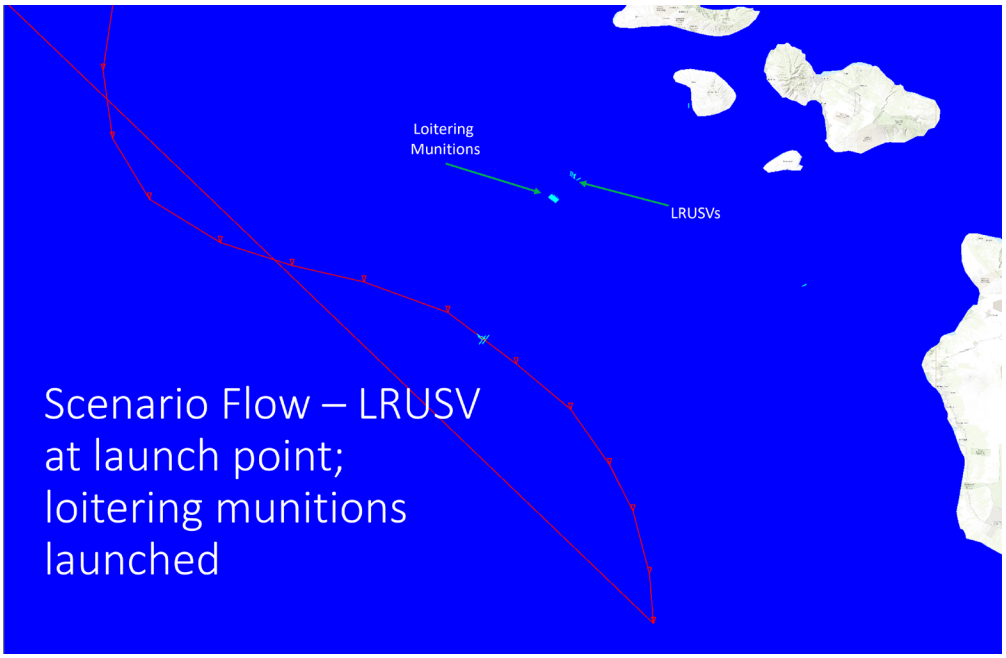


Figure 9. Scenario execution: LRUSVs have reached launch positions and have launched munitions.

Figure 10 provides a close-up of the movement of the munitions toward the target ships.

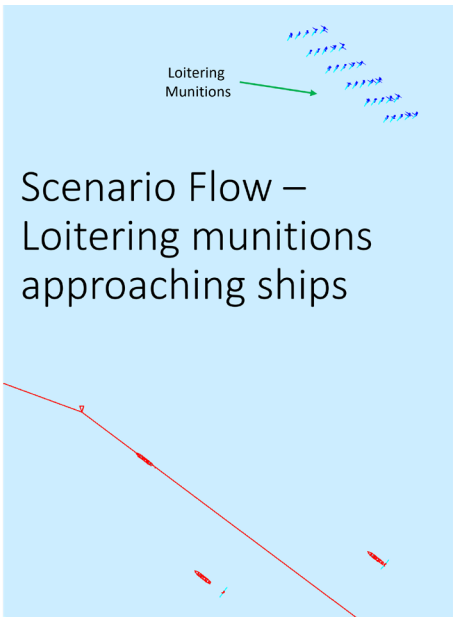


Figure 10. Scenario execution: Munitions approaching Red ships.

In Figure 11, the approaching munitions have separated to begin attacking the target ships from certain angles of attack. Some of the munitions have been engaged and destroyed by Red ship self-defense systems.

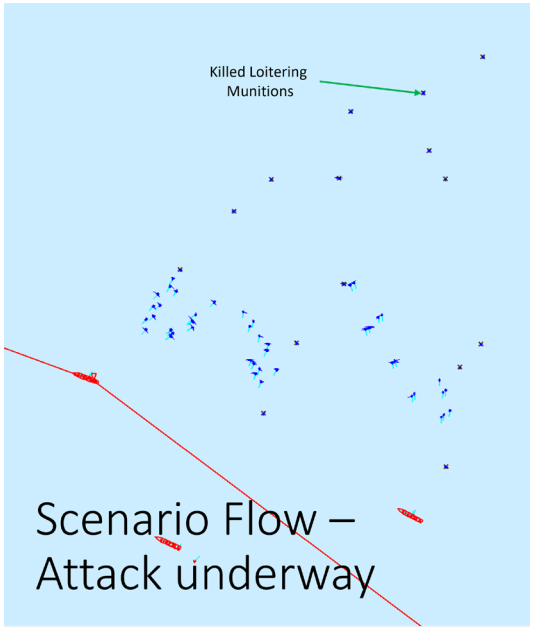


Figure 11. Scenario execution: Munitions maneuvering to attack Red ships from assigned angles of attack; Red ships have detected incoming munitions and have started firing self-defense weapons systems (i.e., CIWS).

In Figure 12, attacking munitions that have made it through the Red ship self-defense fires are striking and damaging or destroying specific component systems on board the Red ships.

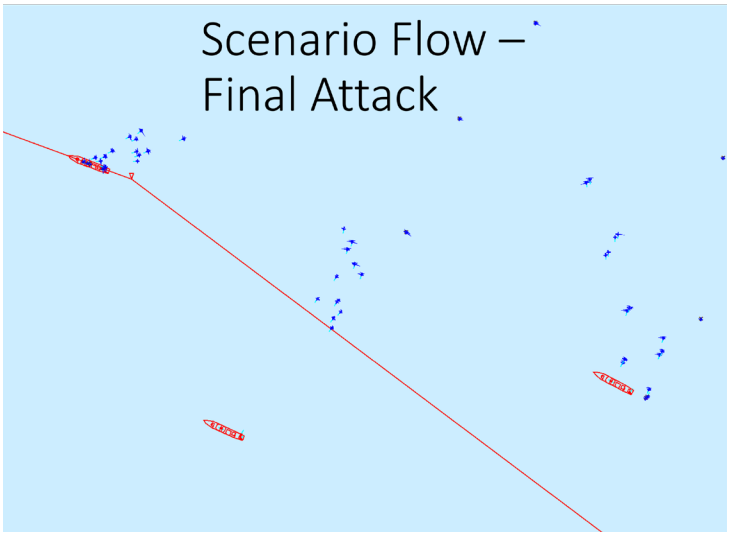


Figure 12. Scenario execution: Munitions that survive the Red ship self-defense fires strike component systems aboard the Red ships.

D. EVALUATION OF SCENARIO OUTCOMES

With respect to the angle of attack and configuration of component systems on the ship as shown in the representation in Figure 6, it is reasonable to expect that an attack from 0 degrees would have the best outcomes since only one CIWS can be employed for most of the attack. It is expected that an attack from 90 degrees would have the worst outcomes since the CIWS from all three ships can engage the incoming munitions. Attacks from aft of the ships also would have poor outcomes since multiple CIWS on the frigates can engage the attacking munitions.

Plots in Figures 13 and 14 show simulation results in terms of number of incoming munitions engaged by the Red ships (Red Engage), number of incoming munitions killed by the Red ships (Red kill), number of hits on the Red ships (Ship hit), and number of component system kills on the Red ships (Ship Comp Kill). Figure 13 shows the simulation results varying angle of attack for each of the four formations. Figure 14 shows simulation results for each formation at angles of attack at 0 degrees (front), 90 degrees (side), and 180 degrees (rear).

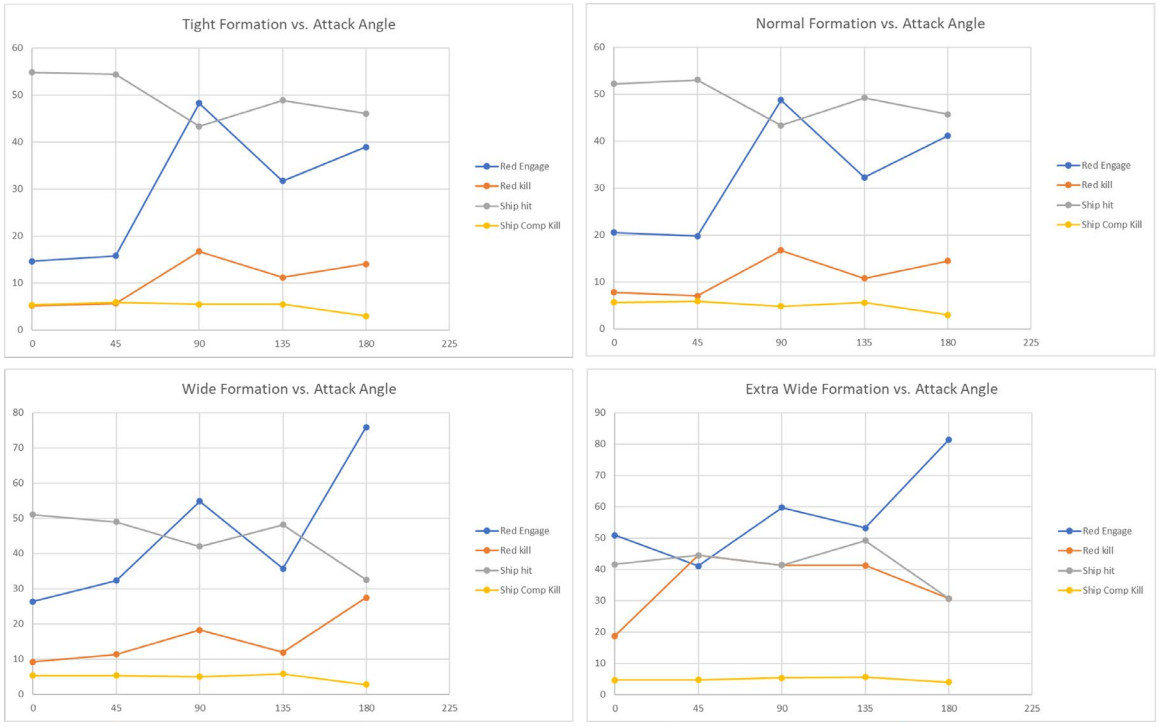


Figure 13. Simulation results: effect of angle of attack for each formation.

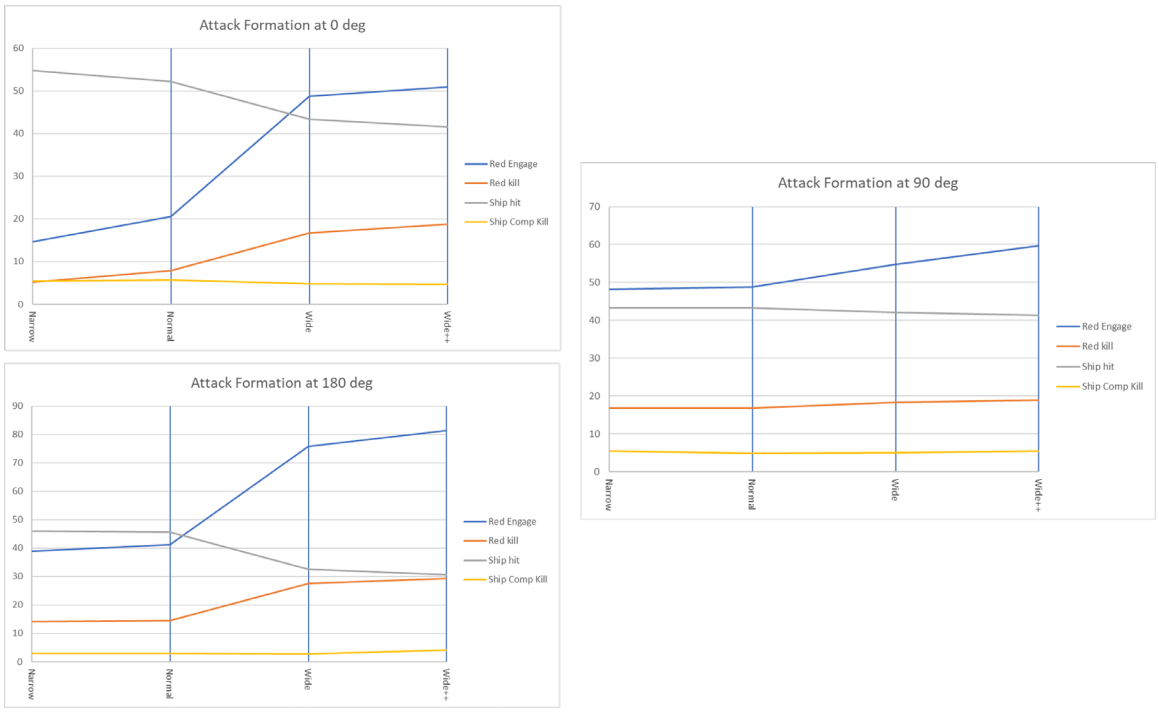


Figure 14. Effect of attack formation

The simulation results show sensitivity of outcomes based on the variations of attack angles. For example, attacks at 90 degrees generally were less effective. Moreover, attacks using the Wide and Extra Wide formations were less effective since the CIWS on multiple ships could engage more readily at all attack angles. Attacks from 180 degrees generally were less effective since all 4 CIWS on the frigates (2 CIWS on each of 2 frigates) could engage the incoming munitions. More detailed analysis of the combinations can be performed in follow-on studies.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

While the scope and level of effort of this study restricted the level of detail that could be developed, it is clear that COMBATXXI can be used to represent important aspects of operations needed to study EABO concepts. The initial set of capabilities developed demonstrate the kinds of sensitivities to tactics, systems, and geometries that are expected in these situations. Further system behavior modeling can be used to explore complex physical model development and newly emerging concepts of operation.

B. RECOMMENDATIONS

The work performed in this study only begins to scratch the surface in representing the wide array of considerations present in new warfighting concepts pertinent to Marine Corps Force Design 2030. Indeed, several student thesis research efforts in progress at the Naval Postgraduate School are considering issues related to logistics support to EABO.

Areas for further study and development include the following:

- Improve modeling of swarming behaviors
- Enhance representation of the command and control of loitering munitions based on guidance from OAD
- Explore explicit communications capabilities/limitations to deal with degraded and denied environments
- Request, obtain, and apply authoritative data for system performance parameters, munition employment and maneuvering, C2 procedures, and other real-world considerations

Furthermore, we recommend the representation of component systems on entities be integrated into the core COMBATXXI simulation code to enable these representations to be used in a wider array of scenarios and studies.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. GLOSSARY OF ACRONYMS AND ABBREVIATIONS

ACE	Air Combat Element
ACV	Amphibious Combat Vehicle
AOA	analysis of alternatives
AOR	area of responsibility
APL/CM	Anti-Personnel Landmine/Cluster Munition
ARV	Advanced Reconnaissance Vehicle
BLT	Battalion Landing Team
BSP	binary space partitioning
C2	command and control
C4ISR	command, control, communications, computers, intelligence, surveillance, and reconnaissance
C5ISRT	command, control, communications, computers, combat systems, intelligence, surveillance, reconnaissance, targeting
CD&I	Capability Development and Integration Directorate
CIWS	Close-in Weapon System
CM	measure
COC	Combat Operations Center
com, comm, comms	communications
COMBATXXI	Combined Arms Analysis Tool for the 21st Century
COP	common operational picture
DDG	guided missile destroyer
EAB	Expeditionary Advanced Base
EABO	Expeditionary Advanced Base Operations
elev	elevation
FFG	guided missile frigate
Deg	degree
FARP	forward arming and refueling point
FM	functional module
FMF	Fleet Marine Force
FOR	field of regard
FOV	field of view
FSCC	Fire Support Coordination Center
GCE	Ground Combat Element
GDC	geodetic coordinate
GFI	Government-Furnished Information
int,Int	integer
HTN	Hierarchical Task Network
HQ	Headquarters
ISR	Intelligence, Surveillance, Reconnaissance
km, Km	kilometer
Kts, kts	knots
Lat	latitude

LCE	Logistics Combat Element
LCT	Littoral Combat Team
Long	longitude
LOS	line-of-sight
LRUSV	long-range unmanned surface vehicle
LRUSVCS	LRUSV control station
Loit	loitering
LoitMun	loitering munition
m	meter
MAGTF	Marine Air Ground Task Force
max	maximum
MCCDC	Marine Corps Combat Development Command
MCSS	Marine Corps Study System
MEF	Marine Expeditionary Force
min	minimum
MLR	Marine Littoral Regiment
MOC	Maritime Operations Center
MOVES	Modeling, Virtual Environments, and Simulation
NE	northeast
NPS	Naval Postgraduate School
OAD	Operations Analysis Division
P-	probability (of)
P-kill	probability of kill
Py	python
SA	situational awareness
SAM	surface-to-air missile
sec	seconds
SQL	Structured Query Language
SITS	Scenario Integration Tool Suite
SW	southwest
TRAC-WSMR	The Research and Analysis Center – White Sands Missile Range
TTP	tactics, techniques, and procedures
UAS	unmanned aerial system
USMC	United States Marine Corps
UTM	universal transverse Mercator
WEZ	weapons engagement zone

APPENDIX B. HTN BEHAVIOR INITIATION

jump_start.py script:

```
import cxxi.model.behavior.OrderUtilities as OrderUtilities
import java.util.Vector as Vector
import mtry.cxxi.model.HierarchicalTaskNetwork.HTNUtilities as HTNUtilities
import mtry.cxxi.model.HierarchicalTaskNetwork.GoalContainer as GoalContainer
import mtry.cxxi.model.behaviorextensions.ExternalActionEventMapper as ExternalActionEventMapper
import cxxi.model.knowledge.group.holders.NewUnitHolder as NewUnitHolder
import cxxi.model.objects.holders.CMHolder as CMHolder
import cxxi.model.objects.playboard.PlayBoard as PlayBoard
import mtry.cxxi.model.HierarchicalTaskNetwork.PythonUtilities.AStarCM.TerrainNetwork.TerrainNetworkSerializer as
TerrainNetworkSerializer
from java.util import ArrayList
import UtilityFuncs
from HTN.Scripts import EABOMediator
from HTN.Scripts import commMediator
from HTN.Scripts import shipFormation

from HTNBehaviors import CreateBasicMove

print info.getMyAssignedName(),": JUMP START!"
print state.getCurrentCommander(), ": CMDR"

# initialize all the entities HTN systems
GoalContainer.initAllEntities()

# set up mobility network for pathfinding
borg.roadNetwork = TerrainNetworkSerializer.loadPointNetworkFromBox("grid_net", "BLUE", "BLUE_UL_BOX",
"BLUE_BR_BOX", "BLUE_DELTAX_BOX", "BLUE_DELTAY_BOX", 10000.0)

borg.roadNetwork.printStats()

if not borg.roadNetwork.wasDeserialized:
   TerrainNetworkSerializer.saveNetwork(borg.roadNetwork, "grid_net.srz")

printMessage("Created movement network", True)

# list of all the profiles that the EABO modeling will use.  This is needed for the sensing and engagement models
# used by the behaviors
EABOProfiles = ["CommAntennas", "LoitMun", "LongRangeRadar", "Type730C", "LRUSV", "Type054A", "Type052D"]
# these are the profiles of the "core" of the ship modeling - this is needed to be able to detect and locate
# a ship as a whole.
EABOShipProfiles = ["Type054A", "Type052D"]

# Set up the "mediator" object for the EABO modeling
borg.mediator = EABOMediator.EABOMediator(EABOProfiles, EABOShipProfiles)

########################################
########################################
#
# Testing flags - used to test behavior functionality
#               these should all be "False" for normal runs
#
########################################
########################################

borg.testComms = False

DO_SWARM_TESTING = False

borg.mediator.holdFireCIWS = False

# if True, use simple target allocation to test the actual targeting process
borg.testMunitionTargeting = False
```

```
########################################
########################################
#
#  Parameters to control scenario
#
########################################
########################################

# the formation to use for the swarms - needs to match the names in the EAB database table SwarmParams
# uncomment the formation to be used
formation = "SWARMCRUISE"
#formation = "SWARMCRUISE_TIGHT"
#formation = "SWARMCRUISE_WIDE"
borg.swarmFormationToUse = formation

# tell the behaviors to try to synch the launch of the munitions, so they arrive at the targets at about
# the same time
borg.synchLaunch = True

# The route and speed in kts the Navy Drone uses
borg.droneRouteName = "UAS_Route"
borg.droneSpeed = 45.0  # speed in Kts

# this is the top priority target.  This could be derived from the DB, but is set here for simplicity
# this is the name of the ship as set in the database - this name corresponds to the unit name that
# contains all the components of the ship
borg.topPriorityShip = "DDGCore"

# speed the LRUSVs travel in KTS
borg.lrusvSpeed = 40.0

# The angle in degrees relative to the heading of the priority ship from which the attack should come.  0 deg implies
# an attack from head on and 180 from the rear.  1-179 is attack from the port side and 181-359 from the starboard side.
borg.attackAngle = 0

# This is the range in meters from the priority ship at which the launch of the munitions should take place.
borg.launchRange = 80000.0

# The max range at which a LRUSV mission is launched - this accounts for the max range the LRUSVs can travel plus the
# attack range for the loitering munitions.
borg.maxMissionRange = 300000.0

# Set up the route red will use - these need to be control measures in the scenario file
# This is the location from where the ships start to move.  At the start of the run, the ships are all "magic moved"
# to this location.
redStartLoc = "RedStart"
# the route on which the red ships will move
redRoute = "redRouteLinear"
#redRoute = "RedRouteCurved"
shipSpeed = 15.0 # Kts
startDelay = 5.0 # allows for changing when the ships start moving to affect when the attack occurs

# set up the formation object for the ship movement
formation = shipFormation.shipFormation()
formationLeader = "DDGCore"

# Offset in meters from Formation leader    Lateral      Fore/Aft  negative values mean to Port side or to aft
formation.addOffsetMapping("DDGCore",          0.0,          0.0)  # formation leader always has 0, 0 offset
formation.addOffsetMapping("FFG1Core",      -500.0,       -1000.0)
formation.addOffsetMapping("FFG2Core",       500.0,       -2000.0)


########################################
########################################
#
# End Parameters to control scenario
#
########################################
########################################
```

```
# Set up the communication mediator
borg.commMediator = commMediator.commMediator(info.getMySelf(), borg)

UtilityFuncsExp.addSpecificGoal(
  info.getMyAssignedName(),
  0.0,
  "HTN/Trees/commMediator.xml",
  "CommsMediator",
  [],
  None)


# set up the EABO base interaction behavior for all the EABO entities
units = NewUnitHolder.getAllUnits()
for i in range(units.size()):
        unit = units.get(i)

        for ii in range(unit.getMembers().size()):
                ent = unit.getMembers().get(ii)
                profileName = ent.getBaseProfile().getName()
                if profileName in EABOProfiles:
                        borg.mediator.connectEABOEntities(ent)
                        # The units for these EABO entities are not used in the same way as the normal use,
                        # so we need to turn off the default command succession process.  The assumption
                        # is that if an entity in a unit is an EABO entity, then the unit is an EABO unit.
                        unit.setAutomaticCommandSuccession(False)

# start the blue force activities
UtilityFuncsExp.addSpecificGoal(
        "ControlShip",  # entity name string
        0.0,    # delay double
        "HTN\Trees\BlueCoordinator.xml",    # goal path string
        "Coordination",
        [],     # arguments to tree hashable list
        None)       # variables to pass to the tree usually None


##############################################################################
##############################################################################
##############################################################################
#
# Behavior testing  --- should be turned off when running in non testing mode
#
##############################################################################
##############################################################################
##############################################################################


if DO_SWARM_TESTING:

        blueSector = "SW"

        UtilityFuncsExp.addSpecificGoal(
          info.getMyAssignedName(),
          0.0,
          "HTN/Trees/TestingDriver.xml",
          "TestDriver",
          [],
          None)

        # get the red force moving
        #redStartLoc = "RedWest"
        #redRoute = "RedWestEast"
        #ObsTime = 25000.0  # for testing

        #redStartLoc = "RedSouth"
        #redRoute = "RedSouthNorth"
        #ObsTime = 12100.0  # for testing
```

```
		#redStartLoc = "RedEast"
		#redRoute = "RedEastWest"
		#ObsTime = 21400.0  # for testing

		redStartLoc = "RedNorth"
		redRoute = "RedNorthSouth"
		ObsTime = 15000.0  # for testing

		#redStartLoc = "RedStart"
		#redRoute = "redRoute3"
		#ObsTime = 21400.0  # for testing



		# start the test - set params to modify testing context
		for i in range(5):
			#i = 4
			LRUSVName = "LRUSV" + str(i+1)
			delayToLaunch = 600.0
			aPoint = CMHolder.retrieveControlMeasureByName("Blue" + blueSector + str(i+1)).getLocation()
			ent = PlayBoard.getSingleton().getEntityByName(LRUSVName)
			ent.getPhysicalState().setGroundTruthLocation(aPoint)

			args = ArrayList()
			args.add(LRUSVName)
			args.add(delayToLaunch)
			args.add(formation)
			UtilityFuncsExp.scheduleEvent(dm, info.getMyAssignedName(), "SendDelayedLaunchEvent", ObsTime, args)


############################
############################3
#
# End testing code
#
############################
############################3


# Start the red ships' behavior

aPoint = CMHolder.retrieveControlMeasureByName(redStartLoc).getLocation()
ent = PlayBoard.getSingleton().getEntityByName("DDGCore")
ent.getPhysicalState().setGroundTruthLocation(aPoint)
ent = PlayBoard.getSingleton().getEntityByName("FFG1Core")
ent.getPhysicalState().setGroundTruthLocation(aPoint)
ent = PlayBoard.getSingleton().getEntityByName("FFG2Core")
ent.getPhysicalState().setGroundTruthLocation(aPoint)

UtilityFuncsExp.addSpecificGoal(
		"DDGCore", # entity name string
		0.0,	# delay double
		"HTN\Trees\RedCoordinator.xml",	# goal path string
		"Coordination",
		[redRoute, formationLeader, formation, shipSpeed, startDelay],
		None)	# variables to pass to the tree usually None
```

LIST OF REFERENCES

Commandant of the Marine Corps (2020). Force Design 2030. Washington, D.C. March.

Headquarters United States Marine Corps (2021). Tentative Manual for Expeditionary Advanced Base Operations. Washington, D.C.:HQ USMC. February.

Office of the Chief of Naval Operations and Headquarters, US Marine Corps (2019). (U) Concept for Expeditionary Advanced Base Operations, classified. Washington, DC: US Department of the Navy.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Research Sponsored Programs Office, Code 41
   Naval Postgraduate School
   Monterey, CA 93943

4. Name of Addressee
   Organization of Addressee
   City, State

5. Name of Addressee
   Organization of Addressee
   City, State

Provide a separate copy of the INITIAL DISTRIBUTION LIST w/ email addresses only if you want the Research and Sponsored Programs Office (RSPO) to send the report electronically.