



**NAVAL  
POSTGRADUATE  
SCHOOL**

**MONTEREY, CALIFORNIA**

**THESIS**

**ENHANCING ENTITY LEVEL KNOWLEDGE  
REPRESENTATION AND ENVIRONMENTAL SENSING IN  
COMBATXXI USING UNMANNED AIRCRAFT SYSTEMS**

by

James C. Teters II

September 2013

Thesis Advisor:

Imre Balogh

Second Reader:

Peter Nesbitt

**This thesis was performed at the MOVES Institute  
Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
<b>1. AGENCY USE ONLY</b> (Leave blank)		<b>2. REPORT DATE</b> September 2013	<b>3. REPORT TYPE AND DATES COVERED</b> Master's Thesis	
<b>4. TITLE AND SUBTITLE</b> Enhancing Entity Level Knowledge Representation And Environmental Sensing In COMBATXXI Using Unmanned Aircraft Systems			<b>5. FUNDING NUMBERS</b>	
<b>6. AUTHOR(S)</b> Teters, James C. II				
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Naval Postgraduate School Monterey, CA 93943-5000			<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>	
<b>9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> N/A			<b>10. SPONSORING/MONITORING AGENCY REPORT NUMBER</b> N/A	
<b>11. SUPPLEMENTARY NOTES</b> The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number N/A.				
<b>12a. DISTRIBUTION / AVAILABILITY STATEMENT</b> Approved for public release; distribution is unlimited			<b>12b. DISTRIBUTION CODE</b> A	
<b>13. ABSTRACT (maximum 200 words)</b> Current modeling and simulation techniques may not adequately represent military operations using unmanned aircraft systems (UAS). A method to represent these conditions in a combat model can offer insight to the use and application of UAS operations, as well as understanding the sensitivity of simulation outcomes to the variability of UAS performance. Additionally, using combat model simulations that do not represent UAS behavior and conditions that cause this variability may return misleading or incomplete results. Current approaches include explicit scripting of behaviors and events. We develop a proof of principle search, targeting, and acquisition (STA) model for use with UAS within COMBATXXI, leveraging existing STA research conducted at the MOVES Institute at the Naval Postgraduate School. These dynamic behaviors are driven by events as they unfold during the simulation run rather than relying on preplanned events as in the scripted approach. This allows these behaviors to be highly reusable since they do not contain scenario or incident specific information. We demonstrate the application of the new STA model in a tactical convoy scenario in COMBATXXI. A design of experiments and post analysis quantifies the sensitivity of the measures of effectiveness of success to conditions contributing to variability in UAS performance.				
<b>14. SUBJECT TERMS</b> UAS, UAVs, Unmanned Aircraft Systems, Unmanned Aerial Vehicle, Sensors, Electronics & Electronic Warfare, Information Systems Technology, COMBATXXI, HTNs, Hierarchical Task Networks, Ontology, Reasoning, Knowledge Representation, Training			<b>15. NUMBER OF PAGES</b> 145	
			<b>16. PRICE CODE</b>	
<b>17. SECURITY CLASSIFICATION OF REPORT</b> Unclassified	<b>18. SECURITY CLASSIFICATION OF THIS PAGE</b> Unclassified	<b>19. SECURITY CLASSIFICATION OF ABSTRACT</b> Unclassified	<b>20. LIMITATION OF ABSTRACT</b> UU	

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited**

**ENHANCING ENTITY LEVEL KNOWLEDGE REPRESENTATION AND  
ENVIRONMENTAL SENSING IN COMBATXXI USING UNMANNED AIRCRAFT  
SYSTEMS**

James C. Teters II  
Major, United States Army  
B.A., Marshall University, 2002

Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF SCIENCE IN  
MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION (MOVES)**

from the  
**NAVAL POSTGRADUATE SCHOOL  
September 2013**

Author: James C. Teters II

Approved by: Imre Balogh  
Thesis Advisor

Peter Nesbitt  
Second Reader

Christian Darken  
Chair, MOVES Academic Committee

Peter J. Denning  
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

## **ABSTRACT**

Current modeling and simulation techniques may not adequately represent military operations using unmanned aircraft systems (UAS). A method to represent these conditions in a combat model can offer insight to the use and application of UAS operations, as well as understanding the sensitivity of simulation outcomes to the variability of UAS performance. Additionally, using combat model simulations that do not represent UAS behavior and conditions that cause this variability may return misleading or incomplete results. Current approaches include explicit scripting of behaviors and events. We develop a proof of principle search, targeting, and acquisition (STA) model for use with UAS within COMBATXXI, leveraging existing STA research conducted at the MOVES Institute at the Naval Postgraduate School. These dynamic behaviors are driven by events as they unfold during the simulation run rather than relying on preplanned events as in the scripted approach. This allows these behaviors to be highly reusable since they do not contain scenario or incident specific information. We demonstrate the application of the new STA model in a tactical convoy scenario in COMBATXXI. A design of experiments and post analysis quantifies the sensitivity of the measures of effectiveness of success to conditions contributing to variability in UAS performance.

THIS PAGE INTENTIONALLY LEFT BLANK



---

---

# Contents

---

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	BACKGROUND . . . . .	1
1.2	RESEARCH PROBLEM . . . . .	1
1.3	MOTIVATION . . . . .	1
1.4	OBJECTIVES . . . . .	2
1.5	THESIS ORGANIZATION . . . . .	2
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>5</b>
2.1	U.S. MILITARY UNMANNED AIRCRAFT SYSTEMS IN OPERATION AND SIMULATION . . . . .	5
2.2	UNMANNED AIRCRAFT SYSTEM . . . . .	5
2.3	IMPROVISED EXPLOSIVE DEVICES . . . . .	7
2.4	TARGET ACQUISITION MODELING . . . . .	12
2.5	COMBATXXI . . . . .	15
2.6	HIERARCHICAL TASK NETWORKS . . . . .	17
2.7	KNOWLEDGE REPRESENTATION . . . . .	20
2.8	SUMMARY . . . . .	24
<b>3</b>	<b>METHODOLOGY</b>	<b>25</b>
3.1	INTRODUCTION . . . . .	25
3.2	SCENARIO . . . . .	25
3.3	FORMULATION OF BEHAVIORS . . . . .	27
3.4	UAV NOTIONAL SENSOR MODEL . . . . .	32
3.5	ONTOLOGY OF IED . . . . .	32
3.6	DESIGN OF EXPERIMENTS . . . . .	51

<b>4 ANALYSIS</b>	<b>53</b>
4.1 INTRODUCTION . . . . .	53
4.2 DESIGN OF EXPERIMENTS ANALYSIS IN COMBATXXI . . . . .	53
4.3 SUMMARY . . . . .	57
<b>5 CONCLUSION</b>	<b>59</b>
5.1 OUTCOMES . . . . .	59
<b>Appendix A JOHNSON CRITERIA</b>	<b>61</b>
<b>Appendix B CONCEPT OF OPERATIONS</b>	<b>63</b>
<b>Appendix C CONVOY MOVE IN FORMATION</b>	<b>65</b>
<b>Appendix D OPERATE THE IED HTN</b>	<b>77</b>
<b>Appendix E OPERATE UAS HTN</b>	<b>83</b>
<b>Appendix F UAV SENSING HTN</b>	<b>89</b>
<b>Appendix G IED ONTOLOGY</b>	<b>95</b>
<b>List of References</b>	<b>117</b>
<b>Initial Distribution List</b>	<b>121</b>

---

---

## List of Figures

---

Figure 2.1	Categories of Unmanned Aircraft Systems [5, p. 12]. The U.S. Army UAS categorization is from the “DoD 2009-2034 Unmanned Systems Integrated Roadmap” [5, p. 3]. As of 2010 the Army does not have a Group 2 UAS. . . . .	6
Figure 2.2	RQ-11B Specifications from Aerovironment [7]. The RQ-11 can fly “low and slow” traveling only 17 to 44 knots which equates to 30 to 60 mph. It is very lightweight 4.2 pounds, making it easy to employ by hand-launching. . . . .	7
Figure 2.3	Marine preparing to hand launch the 4.2lb Raven UAV. From [8] The tripod in the back is part of Aerovironment’s Ground Control System (GCS), which “improves situational awareness of the ground, provides operators cue to potential threats, and enhances airborne intelligence, surveillance and reconnaissance (ISR)” [9]. . . . .	8
Figure 2.4	Scan Eagle and launch system. From [10] . . . . .	9
Figure 2.5	RQ-7B Shadow taking off. From [12] . . . . .	9
Figure 2.6	U.S. Army Gray Eagle in flight. From [14] . . . . .	10
Figure 2.7	MQ-9 Reaper in flight with landing gear down. From [16] . . . . .	10
Figure 2.8	Soldier looking through an image device in order to identify an object. From [21, p. 1] This illustrates the point that the M&S community has to take real instruments and tools—like a camera—and create an abstraction (model) the light waves hitting the tank bouncing into the lens of the sensor (here a camera) and then travel back to observer’s eyes. . . . .	12
Figure 2.9	Illustration of number of pixels required for detection, recognition, and identification of a human being. From [24, p. 3] . . . . .	14

Figure 2.10	Relationship of behavior (static, dynamic) vs usability (easy, difficult) when considering different programming methods in COMBATXXI. From [1, p. 317] . . . . .	16
Figure 2.11	Planning to go to store without accounting for changes in the environment. From [30, slide 8] . . . . .	19
Figure 2.12	Going to store with an interrupt node. From [30, slide 19] . . . . .	20
Figure 2.13	Semantic Network from Collins and Quillian modeling how information would be stored in a computer. This is a modified remake of their illustration. From [36, p. 241] . . . . .	24
Figure 3.1	Mission Overlay in COMBATXXI. . . . .	26
Figure 3.2	Screen shot of NAI 1. . . . .	27
Figure 3.3	<b>ConvoyMoveInFormationTree</b> . This tree will form a plan to get the convoy from the COP Ramrod to OP X-Ray. . . . .	28
Figure 3.4	<b>OperateIEDTree</b> . Picture of <b>OperateIED</b> HTN. . . . .	30
Figure 3.5	<b>OperateUASTree</b> . Picture of <b>OperateUAS</b> HTN. . . . .	30
Figure 3.6	<b>UAVSensor</b> Tree. . . . .	31
Figure 3.7	A depiction of the UAV Sensor Model used in this study. Courtesy from Imre Balogh [40]. . . . .	33
Figure 3.8	Screen picture of Protégé 4.3 program with the main tabs used circled. The Classes tab is shown. . . . .	34
Figure 3.9	<b>Thing</b> class shown with its subclasses. <b>IED</b> class is a subclass of class <b>Thing</b> . . . . .	34
Figure 3.10	<b>DeliveryMethod</b> class shown with its disjoint sibling classes. . . . .	35
Figure 3.11	<b>Human</b> class shown with its disjoint sibling classes. . . . .	36
Figure 3.12	<b>IEDIndicators</b> class shown with with its subclasses contained in the yellow box. . . . .	36
Figure 3.13	<b>Trigger</b> class with subclass <b>CellPhone</b> highlighted. . . . .	37
Figure 3.14	The <b>hasLocation</b> object property is selected with Domain <b>IED</b> and Range <b>Location</b> . The <b>Functional</b> box is checked as well. . . . .	38

Figure 3.15	Necessary and Sufficient description. Taken from Pizza Tutorial [43, p. 56]. . . . .	39
Figure 3.16	The “Equivalent To” box is highlighted showing what must be true in order for an individual to be a member of the <b>IED</b> class. . . . .	40
Figure 3.17	<b>PackageIED</b> shown. . . . .	41
Figure 3.18	<b>SBIED</b> shown. . . . .	41
Figure 3.19	<b>VBIED</b> shown. A vehicle borne IED delivery happens by a vehicle via <b>hasVehicleDeliveryMethod</b> . . . . .	42
Figure 3.20	<b>ConfirmedIED</b> shown. . . . .	42
Figure 3.21	<b>PossibleIED</b> shown. . . . .	43
Figure 3.22	<b>WeakBeliefIED</b> shown. . . . .	43
Figure 3.23	<b>StrongBeliefIED</b> shown. . . . .	44
Figure 3.24	<b>ied_0</b> individual shown. . . . .	45
Figure 3.25	<b>disturbedsoil_1</b> individual being assigned type <b>DisturbedSoil</b> . . . . .	45
Figure 3.26	<b>disturbedsoil_1</b> indicator being assigned string “dirt” shown. . . . .	46
Figure 3.27	<b>wires_1</b> individual shown. . . . .	47
Figure 3.28	<b>ied_1</b> individual shown. . . . .	47
Figure 3.29	Assigning IED Indicators to <b>ied_1</b> . . . . .	47
Figure 3.30	<b>isolatedbox_1</b> individual shown. . . . .	48
Figure 3.31	<b>ied_2</b> individual shown. . . . .	48
Figure 3.32	<b>ied_3</b> individual shown. . . . .	49
Figure 3.33	Reasoner classifies object “Possible IED.” based on its object and data properties. . . . .	49
Figure 3.34	Reasoner classifies object as “Strong Belief IED.” based on properties associated with <b>ied_2</b> . . . . .	50
Figure 3.35	The “possible, strong belief IED” has been confirmed an IED by explosive ordnance disposal (EOD). . . . .	51

Figure 4.1	Heat Map of Mission Success Rate versus parameters False Positive ( $\alpha$ ) and False Negative ( $\beta$ ). Notice that the Mission Success Rate does not change as alpha increases from 0.0 to 1.0, which demonstrates that the False Positive parameter $\alpha$ does not have an effect on mission success. Mission failures occurred even with $\beta$ at zero because on some trials the UAV never sees the IED, and as a result the IED was not identified. . . . .	53
Figure 4.2	Boxplot of Mission Success Rate versus False Positive Rates. Notice that the Mission Success Rate does not change as alpha increases from 0.0 to 1.0. This shows that $\alpha$ does not have an effect on mission success since False Positive represents the identification of non-IED objects. . . . .	54
Figure 4.3	Boxplot of Mission Success Rate versus False Negative Rate ( $\beta$ ). Notice that the Mission Success Rate decreases as $\beta$ increases from 0.0 to 1.0. There is no variance in the success rate either, which is why the boxes appear as lines instead of boxes per earlier boxplots when looking at Mission Success versus False Positive Rate. This shows that $\beta$ has a great effect on mission success rate versus $\alpha$ . . . . .	54
Figure 4.4	Heat Map of Mean Time to Complete Mission   Mission Success for different values of False Positive ( $\alpha$ ) and False Negative ( $\beta$ ). Mean time to complete the mission takes the longest when $\alpha$ is 1.0 and $\beta$ equals 0.0. Mean time for a successful mission increases as $\alpha$ increases and decreases as $\beta$ increases. . . . .	55
Figure 4.5	Boxplot of Mission Completion Time versus False Positive Rates. Notice that median time to complete mission increases as the False Postive Rate ( $\alpha$ ) increases. There appears to be a linear relationship; the variance for each value of $\alpha$ appears to be equal to one another as well. This means that $\alpha$ will always have an effect on amount of time it completes the mission. . . . .	56
Figure 4.6	Boxplot of Mission Complete Time versus False Negative Rate ( $\beta$ ). The False Negative parameter $\beta$ does not have as much of an effect on average time it takes the convoy to complete the mission. Notice that the Mission Completion Time is not heavily influenced on the change in $\beta$ from 0.0 to 1.0. A time of 150 minutes could be found for each $\beta$ value. Only when the value of 0.8 is reached is there a drop in median time of completing the mission. This is because the observer has a higher probability of incorrectly identifying the IED as a non-IED, resulting in fewer times to successfully complete the mission and have an observed time. . . . .	56

Figure 4.7	Heat Map of Risk for different values of False Positive ( $\alpha$ ) and False Negative ( $\beta$ ). Lowest risk takes place at $\alpha$ and $\beta$ being 0.0, 0.0. Given the IED is detected, the user has capability to correctly identify the IED and does not waste time clearing non-IEDs. When $\alpha$ , $\beta$ is 1.0, 1.0, risk is highest (less likely to correctly identify IED and more likely to incorrectly identify non IEDs as being IEDs). . . . .	57
Figure A.1	Normalizing resolved line pairs for critical target dimension. After [44, p. 264] . . . . .	61
Figure A.2	Number of minimum line pairs required for the four detection, orientation, recognition, identification. From [44, p. 264] . . . . .	61

THIS PAGE INTENTIONALLY LEFT BLANK



---

---

## List of Tables

---

Table 2.1	HTN Task and Nodes. From [30] . . . . .	17
Table 2.2	Propositional Calculus Symbols. . . . .	22
Table 2.3	English sentences being used in predicate calculus. After [35, p. 60] . .	23

THIS PAGE INTENTIONALLY LEFT BLANK

---

## List of Acronyms and Abbreviations

---

AGL	Above Ground Level
AI	Artificial Intelligence
AK-47	Avtomat Kalashnikova
AT	Anti-Tank
AO	Area of Operations
AOI	Area of Interest
AMSAA	Army Materiel Systems Analysis Activity
BDE	Brigade
BSL	Behavior Specification Language
CO	Company
COE	Center of Excellence
COMBATXXI	Combined Arms Analysis Tool for the 21st Century
COP	Command Outpost, Common Operating Picture
DFP	Deployable Force Protection
DOE	Design of Experiments
DoD	Department of Defense
EOD	Explosive Ordnance Disposal
EO	Electro-Optics
FT	Feet
FOB	Forward Operating Base
FOL	First-order logic
FOR	Field of Regard
FOV	Field of View
GCS	Ground Control Station
HMG	Heavy Machine Gun
HMMWV	High-Mobility Multipurpose Wheeled Vehicle
HQ	Headquarters
HTN	Hierarchical Task Network
KB	Knowledge Base
KPH	Kilometers Per Hour
KR	Knowledge Representation
IED	Improvised Explosive Device
IN	Infantry
IR	Infrared
IOT	In Order To
ISO	In Support Of
ISR	Intelligence, Surveillance, Reconnaissance
LNA	Local National Army
MCCDC	Marine Corps Combat Development Command

M	Meters
MM	Millimeters
MOVES	Modeling, Virtual Environments and Simulation
MSR	Main Supply Route
M&S	Modeling and Simulation
NAI	Named Area of Interest
NPS	Naval Postgraduate School
NVESD	Night Vision and Electronic Sensors Directorate
PBIED	Person-Borne IED
PLT	Platoon
OEF	Operation Enduring Freedom
OP	Observation Post
OWL	Web Ontology Language
RIP	Relief-In-Place
RPG	Rocket-Propelled Grenade
RPK	Ruchnoy Pulemyot Kalashnikova
SA	Small Arms
SPG	SPG-9 Kopye
SOCOM	United States Special Operations Command
STA	Search and Target Acquisition
SWA	Southwest Asia
TTPM	Targeting Task Performance Metric
TRAC	United States Army TRADOC Analysis Center
TRADOC	United States Army Training and Doctrine Command
UA	Unmanned Aircraft
UAS	Unmanned Aircraft Systems
UAV	Unmanned Aerial Vehicle
USAF	United States Air Force
USG	United States Government
UXO	Unexploded Ordnance
WFF	Well Formed Formula
WSMR	White Sands Missile Range
WWII	World War II
W3WC	World Wide Web Consortium

---

---

## Executive Summary

---

Current modeling and simulation techniques may not adequately represent military operations in respect to unmanned aircraft systems (UAS), to include the unmanned aerial vehicle (UAV) and its human operator. The performance of a UAS in the field can vary greatly, and this performance can have an enormous affect on the outcome of a military operation. This complex and variable behavior may be sensitive to considerations not represented in contemporary combat models. These conditions can include, but are not limited to, pilot training and experience. Combat models may require a means to model the effects of this behavior, particularly when analyzing the skill or efficiency of a UAS in providing information to other entities in the scenario.

A method to represent these conditions in a combat model can offer insight to the use and application of UAS in operations, as well as understanding the sensitivity of simulation outcomes to the variability of UAS performance. Additionally, using combat model simulations that do not represent UAS behavior and conditions that cause this variability may return misleading or incomplete results. Understanding when and how to model these conditions contributes to quality simulation and analysis.

Current approaches include explicit scripting of behaviors and events. Combat models, including Combined Arms Analysis Tool for the 21st Century (COMBATXXI) [1], provide opportunities for analysis. The ACQUIRE model is the most prevalent in U.S. Army simulations, allowing for the modeling of sensors used for search and targeting and acquisition (STA) [2]. Hierarchical Task Networks (HTNs) allow for automated planning based on changing environments in the simulated world [3]. Ontologies can show the effects that detailed information may play on decision making when utilizing a UAS. This thesis proposes using dynamic behaviors to include hierarchal task networks, ontologies, and sensor models to represent the inherent complexity of a UAS in a simulation. We developed a proof of principle search, targeting and acquisition (STA) model for use with UAS within COMBATXXI leveraging existing STA research conducted at the Modeling, Virtual Environments and Simulation Institute (MOVES) at the Naval Postgraduate School. We showed that this methodology can result in analysis that gives insight into the effects the capabilities of a UAS offer regarding deployable force protection (DFP). An ontology provides a means of representing pieces of information that, when combined with reason, was able to output relevant knowledge of the world and affect the out-

come. This study used a scenario where understanding the ability to identify a target has a relationship to successful execution.

We demonstrated application of the new STA model in a tactical convoy scenario in COMBATXXI. The ability to model a UAS in a working environment is fundamental to a study concerning tactical convoy operations. A design of experiments and post-execution analysis quantified the sensitivity of the measures of effectiveness of success to conditions contributing to variability in UAS performance. Future work is given to identify possible studies that can extend this research.

---

---

## Acknowledgements

---

I want to thank Dr. Imre Balogh for his tireless efforts in assisting me with the programming and instruction on HTNs, COMBATXXI, and for making this daunting task not as stressful as it could have been. Also I would like to thank Major Peter Nesbitt, whose help, insight and encouragement have been a tremendous blessing. To Curtis Blais, “Thank you, sir” for your assistance and instruction on building ontologies with Protégé and providing additional eyes on the thesis. Many thanks to the wonderful people at TRAC-Monterey: Lieutenant Colonel John Alt, Major Ed Masotti, and Dr. Tom Anderson for your assistance and a place to work. I also want to thank the awesome faculty at The MOVES Institute for providing a great and wonderful learning experience. It has been a blast.

Thank you Mom, Dad, Nanny and Papaw for believing in me, I hope to never let you down.

Most of all thank you, Bobbi, for your patience, understanding and most importantly, the love you have poured out for me during the last two years; you are the best woman God could have ever given me to call my wife. Thanks Tyler and Emma for being great kids and making me smile at the most perfect times. I love you both.

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

THIS PAGE INTENTIONALLY LEFT BLANK



---

---

# CHAPTER 1: INTRODUCTION

---

## **1.1 BACKGROUND**

The United States Military is considered the most powerful and effective fighting force when fighting a traditional red on blue, symmetrical war. Since the September 11, 2001, attacks, our enemies have been successful in mitigating our fighting strength through the use of improvised explosive devices (IEDs). The ability to counter such threats demand an adaptable and resilient force. Modeling and simulation (M&S) plays a vital role in developing scenarios that help leaders make decisions that can have lasting impacts on the battlefield. It is key that M&S provides decision makers with more fidelity and plausible results when studying the potential effects of systems used in theater.

## **1.2 RESEARCH PROBLEM**

Current modeling techniques do not adequately provide the means to modify and measure the effects of changing the properties inherent to human/machine components and processes in an unmanned aircraft system (UAS). This research proposes an innovative way to represent the value of information and training in a constructive simulation, COMBATXXI (Combined Arms Analysis Tool for the 21st Century). Sensors in COMBATXXI do not allow for accurate modeling of IED detection when deploying UAS in a doctrinally correct manner. This may result in improper system deployment, overconfidence of capabilities, or unrealistic expectations causing more harm than good to our Soldiers. This study will attempt to model decision making capabilities of a UAS in COMBATXXI, utilizing newly developed sensor modeling, dynamic planning methods, and ontology based reasoning.

## **1.3 MOTIVATION**

The ability of a unit to effectively perform its mission is dependent on many factors including presence, posture, and safety. The U.S. Army relies on force protection at all times. Unmanned aerial vehicles (UAVs) are gaining in prevalence use since the War on Terrorism began after the tragic events of 9-11. As of 2010, Unmanned Aircraft Systems (UAS) are in use at echelons below battalion and below [5, p. 21]. Analyzing the effective use of a UAS can be performed using the U.S. Army's high-resolution simulation COMBATXXI. This simulation models scenarios ranging from squad level building clearing operations to brigade-size amphibious assaults [1].

COMBATXXI is extremely important for analysis of systems and can be improved to represent the dynamics of a UAS's information-gathering and its effects. Understanding how information and training affect decision making may provide better insight on how we train, equip; and employ systems such as a UAS based on this new prototype. The study utilizes a new means of creating behaviors in COMBATXXI by way of Hierarchical Task Networks (HTNs).

## **1.4 OBJECTIVES**

In order to develop a means to represent information and study the effects of UAS operator skills when using this operation, we identify the following objectives:

- Utilize HTNs to assist in the formulation of behaviors more representative of complex systems in operation.
- Describe a new method to represent knowledge in COMBATXXI through use of an ontology and a new nontraditional sensor model.
- Gather more knowledge on the effects of information fidelity and varying capabilities of sensors utilization of Unmanned Aircraft Systems in a deployable force protection scenario.

## **1.5 THESIS ORGANIZATION**

Chapter 1 lays out the plan to completion. The motivation for the thesis is given and provides the reader with guidance in the direction the study will take in order to answer the research questions.

Chapter 2 provides a review of the UASs that the U.S. Army has in operation and other systems used elsewhere in the Department of Defense (DoD). The development of target acquisition models used in simulations is discussed. Hierarchical task networks are described to provide the reader with an understanding of the value they provide to the developer when compared to previous methods. A brief history of knowledge representation (KR) and its applications are discussed as well.

Chapter 3 details the methodology used in the thesis from its inception to completion of the study. Novel ideas of using HTNs and an ontology approach are discussed, providing the reader with a new method in KR implemented simulations.

Chapter 4 describes the modeling of the Raven (unmanned aircraft of the Raven UAS) using a new STA model. This chapter discusses how an ontology of an IED could allow an entity

to make a “realistic” determination of a threat based on the knowledge it is given through the sensor provided by the new STA model.

Chapter 5 provides overall conclusion of the research and a discussion of possible further work.

THIS PAGE INTENTIONALLY LEFT BLANK

---

---

# CHAPTER 2: LITERATURE REVIEW

---

## **2.1 U.S. MILITARY UNMANNED AIRCRAFT SYSTEMS IN OPERATION AND SIMULATION**

This chapter provides a review of the UAS in use by the U.S. Military. This includes systems that the U.S. Army has in operation and other systems only being used by other members of the U.S. Armed Forces. A short history of IEDs, their makeup, and means of detection will be examined along with the development of target acquisition models used in simulations. Hierarchical task networks are described in detail to provide the reader with an understanding of the value they provide to the developer when compared to previous methods. A brief history of knowledge representation and its applications are discussed as well.

## **2.2 UNMANNED AIRCRAFT SYSTEM**

The term UAV (unmanned aerial vehicle) is the term most popularly used when discussing an unmanned aircraft that flies with a remotely located pilot. The UAV is part of a system-of-systems called an unmanned aircraft system. According to the U.S. Army [5, p8] “a UAS is comprised of an unmanned aircraft (UA), payload, human operator, control element, display, communication architecture, life cycle logistics, and the supported Soldier.” A UAS is capable of bringing the Warfighter value added service with systems such as electro-optical (EO) and infrared (IR) sensors to enhance intelligence, surveillance, and reconnaissance (ISR) capability. According to the U.S. Army 2010 UAS Roadmap, the five different groups of UASs being used are based on the following UA attributes: weight, altitude, and airspeed, see Figure 2.1 [5, p. 12]. These attributes were approved by the Joint Chiefs of Staff on November 25, 2008 [5, p. 12].

### **2.2.1 Raven RQ-11B**

Ravens offer ISR capabilities for units at brigade (BDE) and lower echelons and currently sees high usage in oversea deployments. A report provided to Congress in 2012 reports that the U.S. Army, Navy and United States Special Operations Command (SOCOM) has 5,346 RQ-11 vehicles in the inventory [6, p. 8]. The characteristics of this UAS are given in Figure 2.2. This hand-launched UAV flies low and slow with cruising speeds between 17–44 knots (Figure 2.3). It scans areas of interest (AOI) at an operating altitude of approximately 100–500 feet (30–152

UAS Category	Max Gross Takeoff Weight	Normal Operating Altitude (Ft)	Airspeed	Current Army UAS in Operation
Group 1	< 20 pounds	< 1200 above ground level (AGL)	<100 Knots	RQ-11B Raven
Group 2	21-55 pounds	< 3500 AGL	<250 Knots	No current system
Group 3	< 1320 pounds	<18,000 mean sea level (MSL)		RQ-7B Shadow
Group 4	> 1320 pounds	> 18,000 MSL	Any Airspeed	MQ-5B, MQ-1C
Group 5				No current system

Figure 2.1: Categories of Unmanned Aircraft Systems [5, p. 12]. The U.S. Army UAS categorization is from the “DoD 2009-2034 Unmanned Systems Integrated Roadmap” [5, p. 3]. As of 2010 the Army does not have a Group 2 UAS.

meters) above ground level (AGL). The sensor package allows the operator to gather data via EO or IR cameras using a continuous pan with a +10- to -90-degree tilt. “The [Raven] can be operated manually or programmed for autonomous operation” [7]. The Raven would be the ideal asset for ISR capabilities for any squad-size mission (based on its specs) such as searching for IEDs along a route at known locations.

### 2.2.2 Scan Eagle

The Scan Eagle UAS is a Group 2 category UAS that performs missions for the U.S. Air Force and the U.S. Marine Corps (see Figure 2.4). This UAV operates at altitudes up to 16,000 feet AGL (above ground level) and can stay aloft for more than twenty hours at speeds between 48–70 knots. The Scan Eagle payload consists of a high-resolution day/night camera and thermal imager. It catapults into operation with a launch and recovery system called Skyhook. This system requires four personnel (two operators, two maintainers) to run [10].

### 2.2.3 RQ-7B Shadow

The RQ-7B Shadow UAS belongs to the UAS Group 3 category. This UAV can fly to altitudes of 15,000 feet AGL for nine hours of flight. Cruising speed is 90 knots with loitering capabilities at 65 knots. The unmanned aircraft can carry up to 80 pounds of sensors [11]. Figure 2.5 shows a RQ-7 Shadow catapult from its hydraulic rail launcher.

### 2.2.4 MQ-1C Gray Eagle

The MQ-1C Gray Eagle (also known as “Warrior”) belongs in the Group 4 category. It is the upgrade of the MQ-1 Predator. The “M” and “Q” designation mean “multi-role” and “unmanned,” respectively. It has a wingspan of 56 feet and a length of 28 feet. This system can perform ISR

<b>Standard Payloads</b>	Dual Forward and Side-Look EO Camera Nose, Electronic Pan-tilt-zoom with Stabilization, Forward and Side-Look IR Camera Nose (6.5 oz payloads)
<b>Range</b>	10 km
<b>Endurance</b>	60–90 minutes (Rechargeable Battery)
<b>Speed</b>	32–81 km/h, 17–44 knots
<b>Operating Altitude (Typ.)</b>	100–500 ft (30-152 m) AGL, 14,000 ft MSL max launch altitude
<b>Wing Span</b>	4.5 ft (1.4 m)
<b>Length</b>	3.0 ft (0.9 m)
<b>Weight</b>	4.2 lbs (1.9 kg)
<b>GCS</b>	Lightweight, Modular Components, Waterproof Softcase, Optional FalconView Moving Map and Mission Planning Laptop Interface, Digital Video Recorder and Frame Capture
<b>Launch &amp; Recovery Method</b>	Hand-Launched, Deep Stall Landing

Figure 2.2: RQ-11B Specifications from Aerovironment [7]. The RQ-11 can fly “low and slow” traveling only 17 to 44 knots which equates to 30 to 60 mph. It is very lightweight 4.2 pounds, making it easy to employ by hand-launching.

and communications relay missions, or be equipped with four Hellfire missiles. It has the ability to fly to 29,000 feet and stay aloft for 25 hours with a maximum speed of 167 knots, see Figure 2.6 [13].

### 2.2.5 MQ-9 Reaper

Introduced in 2001, the Reaper primarily performs the role of intelligence collection in support of strike, coordination, and reconnaissance missions [15]. It does not belong in the U.S. Army’s inventory but can support army missions. The Reaper is a Group 5 UAS with a 66-foot wingspan and is 36 feet from nose to tail. Its maximum takeoff weight is 10,500 pounds. The hand-thrown Raven weighs a minuscule 4 pounds comparatively. The Reaper (Figure 2.7) flies at a maximum altitude of 50,000 feet and cruises at approximately 230 miles per hour (200 knots) at ranges up to 1,250 miles. It can fire Hellfire missiles or guided bombs, depending on mission requirements.

## 2.3 IMPROVISED EXPLOSIVE DEVICES

In this section, we provide a definition of an IED, its components, and means of detection in order to assist in making the IED ontology for the scenario to be used in the study.



Figure 2.3: Marine preparing to hand launch the 4.2lb Raven UAV. From [8] The tripod in the back is part of Aerovironment’s Ground Control System (GCS), which “improves situational awareness of the ground, provides operators cue to potential threats, and enhances airborne intelligence, surveillance and reconnaissance (ISR)” [9].

### 2.3.1 History of IEDs

The history of IEDs goes farther back than the “Global War on Terror.” They are a direct ambush weapon extremely effective in killing soft targets such as troops and/or lightly armored vehicles such as the High-Mobility Multipurpose Wheeled Vehicle (HMMWV). The DoD states that, “An IED is a device placed or fabricated in an improvised manner incorporating destructive, lethal, noxious, pyrotechnic, or incendiary chemicals and designed to destroy, incapacitate, harass, or distract. ...[the IED] may incorporate military stores, but is normally devised from nonmilitary components” [17, pII-14]. According to the *Washington Post* [18] as of August 29, 2013, 2,503 out of 6,668 [fatalities] 37.5% were caused by IEDs in support of Operation Iraqi Freedom and Operation Enduring Freedom. If an IED explosion does not kill the entire crew in the vehicle, there is a very strong chance that survivors will likely suffer from traumatic brain injury (TBI), loss of limb, burn trauma, etc. The ability to detect and defeat (making IEDs inoperable once found) is paramount to current and future mission successes.





Figure 2.4: Scan Eagle and launch system. From [10]



Figure 2.5: RQ-7B Shadow taking off. From [12]



Figure 2.6: U.S. Army Gray Eagle in flight. From [14]



Figure 2.7: MQ-9 Reaper in flight with landing gear down. From [16]

### 2.3.2 Makeup of IEDs

In general, an IED [19, para 6-71] is made up of the following components:

1. Main charge (explosive)
2. Casing (materials around explosive)

3. Initiators (what triggers the explosion)
  - (a) Command wired
  - (b) Radio controlled
  - (c) Victim operated
  - (d) Timed

### **2.3.3 Detection of IEDs**

It is crucial that modeling detection of IEDs be performed to make it easier for U.S. Army leadership to understand the most beneficial ways to mitigate their effects when using UASs such as the Raven RQ-11B. The most dangerous way of detecting IEDs visually is from close proximity fewer than 5 meters. At distances of 100 meters or more detection is much safer, but more difficult. The enemy is continuously adapting their means of implementing IEDs, which make detection more difficult. Shepherd states in [20] that IEDs were first buried underground, then placed in and behind objects above ground, and then below ground again. Later, they were found hidden in dead animals by the road, vehicles (vehicle-borne IEDs or VBIEDs), and humans (person-borne or PBIEDs). The enemy then began countering our mitigation by placing the IEDs underneath culverts or burying them underneath the roadway. One can see that winning the IED battle is a zero-sum game that costs millions of dollars, and more importantly, the lives and limbs of our soldiers.

According to Shepherd [20] there are four methods to detecting IEDs:

1. “Observing the [observation post] or triggerman.”
2. “Identifying the explosives [IED indicators such as loose soil, wires, containers out of place, etc.] or where they are hidden.”
3. “Gathering intelligence from the local population.”
4. “Being attacked.”

Shepherd [20] says that while leaders can use dismounts in staggered column formation or dismounted with vehicles in traveling overwatch, it is not always practical. Another means of detecting IEDs from a safe distance is through the use of unmanned aerial vehicles (UAVs). He sums up the use of aerial reconnaissance below when detecting IEDs:

[UAVs] assigned to cavalry troops, as well as scouts in helicopters, can provide successful aerial reconnaissance. Aerial surveillance can move quicker and provide

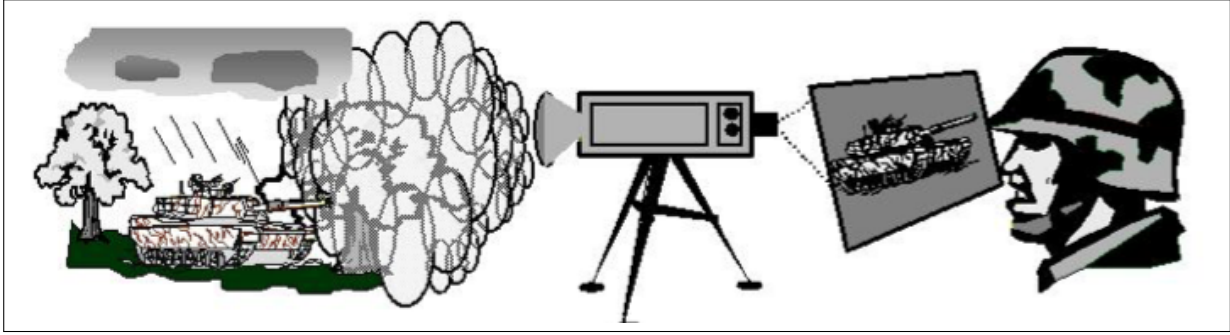


Figure 2.8: Soldier looking through an image device in order to identify an object. From [21, p. 1] This illustrates the point that the M&S community has to take real instruments and tools—like a camera—and create an abstraction (model) the light waves hitting the tank bouncing into the lens of the sensor (here a camera) and then travel back to observer’s eyes.

advanced warning to reconnaissance elements on the ground. Scout elements on the ground can then move forward to confirm or deny information provided by air elements. [20]

This concludes the discussion regarding the makeup and some detection methods for IEDs. Employing UASs to assist in detection of IEDs appears to be useful and promising. Since aerial reconnaissance is important during military operations, it may be necessary to develop methods of modeling this in a simulation. Next, we will look at how human sight has been modeled in the context of target acquisition modeling. This provides a better understanding of the dynamics and limitations of modeling human eyesight in current simulations.

## 2.4 TARGET ACQUISITION MODELING

In order to engage the enemy, one has to know the enemy’s whereabouts. Throughout history, commanders on land or at sea continue to deploy sentries/patrols to gain knowledge of the enemy’s location. Since WWII the use of electro-optic (EO) devices have become widespread, and are being utilized ubiquitously throughout the military. Vollermerhausen and Jacobs state: “The history of modeling EO imagers traces back almost 60 years to the pioneering work of Otto Schade... and [he] specifies that an EO image must be produced based on the capabilities and optical characteristics of the eye” [21, p. 9]. Schade had shown that it is possible to model the eye and created an analog model of the eye, which is “patterned after a television system” [22, p. 721]. Objective methods are used to test these characteristics and the “numerical values obtained by calculation...require correlation with the subjective impressions: graininess, tone

scale, and sharpness” [21, p. 9]. While his work is pioneering and a great foundation for modeling EO systems, it has been proven to be “complex and difficult to adapt to changing conditions” [21, p. 10]. Figure 2.8 illustrates a soldier looking through an image device to assist in his/her ability to assess whether an object is a threat or not. Not only is the human eye itself complicated, but adding to this complexity is trying to model photons hitting an object like a tank, bouncing into a device (UAV sensor), which in turn takes that image and makes a duplicate image. However, the M&S community is asked to come up with ways to best represent this real phenomena in simulations.

### 2.4.1 Johnson Criteria

Johnson Criteria assists analysts in understanding the critical dimensions necessary for an observer to “see” a target. The United States Army Night Vision and Electronic Sensors Directorate (NVESD) state: “John Johnson, a Night Vision scientist, worked to develop methods of predicting target detection, orientation, recognition, and identification. Johnson worked with volunteer observers to test each individual’s ability to identify targets through image intensifier equipment under various conditions” [23]. His models allow the ability to create predictive models of imagery systems. Johnson believed that alternating lines of a fixed width and spacing enable one to characterize the level of detail that can be discerned from an object. The thinner the line and closer the spacing, the more detail a person can perceive. He developed thresholds of line pairs to predict a person’s ability to perform detection, orientation, recognition, and identification [21, p. 7]. The Johnson Chart can be found in Appendix A, which shows the minimum number of line pairs for the probability of 50% of observers to obtain detection, orientation, recognition and resolution when looking at the target. Figure 2.9 illustrates the number of pixels needed for detection, recognition and identification of a human being. It makes sense that the level of detection, recognition, and identification is proportional to the number of pixels that cover the respective area of acquisition. Vollmerhausen and Jacobs [21, p. 7] also state:

The Johnson metric uses limiting bar-chart resolution as an indicator of sensor goodness for target acquisition purposes. Predictive accuracy of this metric is best when comparing “like” sensors and conditions. The metric is not compatible with many features found in modern sensors. For example, it is not compatible with sampled imagers. Further, the Johnson metric fails to predict the impact of frequency boost on range performance.

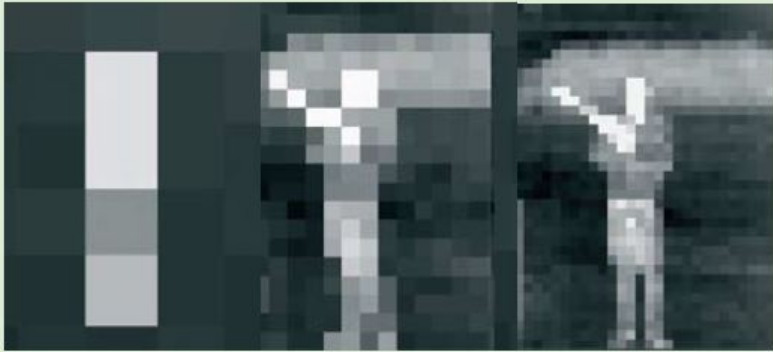
The Johnson Criteria allows for the modeling of target acquisition predictability for different sensors currently in use by the U.S. Army. The metric does have its limitations that may inhibit accurate prediction of what humans will see when using imagery systems. The ACQUIRE model performs target acquisitions in COMBATXXI and requires the Johnson Criteria critical dimension for target detection. Since the Johnson Criteria are in need of an update to reflect the effects of STA when operating UASs, it may be beneficial to understand more about ACQUIRE and possibly construct a model without using the Johnson Criteria.

**Example :**

The Johnson Criteria assume that the critical dimension for a human being is 0.75 meters. To get DRI, you need 1.5 pixels, 6 pixels and 12 pixels across 0.75 meters in the object plane. That means:

1.5 pixels/0.75m = 2 pixels per meter  
 6 pixels/0.75m = 8 pixels/meter  
 12 pixels/0.75m = 16 pixels/meter

Let us assume a man is 1.8m by 0.5m. So the man should be covered by:



<p><b>Detection =</b>  <b>3.6 pixels by 1 pixel</b>          You can see something is there.</p>	<p><b>Recognition =</b>  <b>14.4 pixels by 4 pixels</b>          You can see that a person is there.</p>	<p><b>Identification =</b>  <b>28.8 pixels by 8 pixels</b>          You can see that the person is holding a rifle.</p>
--	--	---

*Images only intended to represent the concept.*

Figure 2.9: Illustration of number of pixels required for detection, recognition, and identification of a human being. From [24, p. 3]

### 2.4.2 ACQUIRE

The most prevalent target acquisition model the U.S. Army operates in combat simulations is the ACQUIRE model. The ACQUIRE model was formulated in 1990 and is a continuation of the Johnson Criteria [25, pp. 1-2] According to Darken [26, p. 263] “ACQUIRE computes the

detection probability as a function of the brightness...of the target, the brightness of the background of the target, and the subjective size of the target, in terms of its ‘number of resolvable cycles.’” The ACQUIRE model uses physical sensor characteristic data and human perceptual data along with environmental factors to compute a probability of detection for a specific observer-target configuration. In combat models such as COMBATXXI, this data coupled with a stochastic process allows for determining an actual outcome of a simulated observation. For example, ACQUIRE computes the probability that one will see a target given a target’s range and the entity’s sensor as .78. COMBATXXI then randomly draws a (uniform) number from 0–1 (where all values between 0 and 1 have an equal chance of being selected). If the number drawn is equal or less than .78 the simulation says that target is “seen” otherwise, the target is not seen even though there is clear line of sight between the sensor and the target.

LTC Baez [27] identifies some major drawbacks to the ACQUIRE model:

- “Current [model has] not been calibrated or developed for close-in targets within 200 meters” [27, p. 5].
- “Simulated soldiers can scan large fields of regard (FOR) quicker and make significantly quicker detections and identifications than a real soldier” [27, p. 5].
- Unvalidated workarounds have been made for limitations such as moving targets, multiple targets, clutter effects, color effects.
- Lack of updated visual perception experiments for targets not in database (i.e., components of IEDs).

These four items listed above hinder the ability to accurately predict the detection of targets by a UAS modeled in this study. The UAS that will be modeled in COMBATXXI for this study is the Raven RQ-11B. The Raven generally operates between altitudes of 30–152 meters [7]. When operating the Raven soldiers scan the hand-held screen and probably do not always detect (within a reasonable amount of time) objects such as a person holding a cell phone, misplaced box along the road, wires coming out of the ground. To compound the problem, there are no known human experiments that have been run to accurately model the detection of items using a sensor such as this UAS.

## **2.5 COMBATXXI**

The Combined Arms Analysis Tool for the 21st Century (COMBATXXI) is a detailed event-driven simulation developed by U.S. Army Training and Doctrine Command White Sands Mis-



sile Range (TRAC-WSMR) and Marine Corps Combat Development Command (MCCDC). The COMBATXXI User’s Guide describes this simulation as “A Joint, high-resolution, closed-form, stochastic, discrete event, entity level structure analytical combat simulation” [1, p. 14]. Some of the major model functions include “Ground Combat (Light and Heavy Forces), Future forces, Fixed-wing and rotary wing” [1, p. 14]. One of the goals of COMBATXXI is to provide “a simulation that can represent information flow in a way that allows the analysis of its impact on operational effectiveness” [1, p. 14].

Behaviors of entities in COMBATXXI can be specified by model users using [1, p. 317]:

- Orders
- BSL (Behavior Specification Language, the native scripting language in COMBATXXI).
- Python scripts

Orders are the easiest for the programmer to use and are grouped into compound orders to perform more complex actions. BSL provides more complex behavior structures and a way to connect to complex internally defined behaviors. However, BSL restricts access to certain objects and data outside the model, which limits its use of making complex, dynamic behaviors. Python allows for the most flexible method of developing dynamic, “realistic” behaviors in COMBATXXI. To the non-programmer this may be seem a little intimidating. Figure 2.10 relates the types of behavior being programmed to the difficulty of implementation. The next section provides information on how this study will utilize dynamic planning methods and tools to assist in creating these dynamic behaviors.

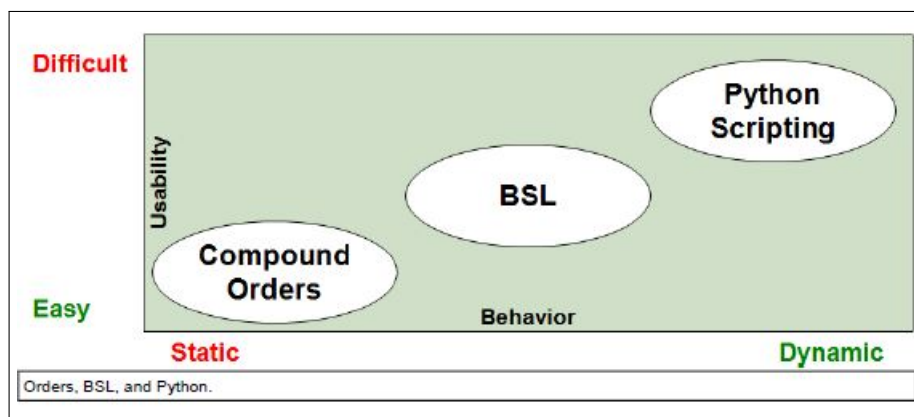


Figure 2.10: Relationship of behavior (static, dynamic) vs usability (easy, difficult) when considering different programming methods in COMBATXXI. From [1, p. 317]







HTN Tasks and Shapes	
Tasks	Shape
Primitive Task	
Compound Tasks	
Goal Tasks	
Constraint Node	

Table 2.1: HTN Task and Nodes. From [30]

## 2.6 HIERARCHICAL TASK NETWORKS

The father of modern-day hierarchical task networks is Earl Sacerdoti, who describes HTNs as “procedural nets” [28]. HTNs are a means of planning actions in a program via the traversal of a series of networks in order to reach a goal. Sohrabi, Baier, McIlraith [29] state: “the planner is provided with a set of tasks to be performed, possibly together with constraints on those tasks. A plan is then formulated by repeatedly decomposing tasks into smaller and smaller subtasks until primitive, executable tasks are reached.” Traditional HTNs consider what is to be the desired state of world, finds the plan (based solely on the current state of the world) which then is executed to reach the goal state (the desired state of the world).

### 2.6.1 Basic Terminology of HTNs

In order to fully appreciate the potential of HTNs, some basic terminology must be defined. *Primitive tasks* are tasks that cannot be broken down any further. A *compound task* consists of a collection of tasks that may be primitive or complex. *Goals* or *goal tasks* represent the final world state the HTN is trying to achieve. *Constraints* are the conditions that must be true in order to execute a specific branch of the HTN. See Table 2.1 for the graphics that correspond to an HTN task and/or nodes.

HTN trees are read from left to right and then top to bottom. Traversal of the HTN is complete once the goal node is reached. If the goal node is not reached, then the HTN might not finish and thus the goal state is never reached. Proper HTN fabrication allows for goal nodes to always be reached [3]. In the next section, an example of a static HTN in operation provides an opportunity to understand traditional automated planning.

### **2.6.2 Example of Static HTN Planning**

A HTN creates a plan to implement the behavior of someone going to the store (see Figure 2.11). The plan is based on the current state of the world and is valid as long as the state of the world does not change; remains static. The traversal of this tree illustrates the methodology of HTNs. First, the “Go to Store” compound task begins execution. Next it moves to the first and only constraint “Find Keys.” If “Find Keys” has value of “True” then the tree traverses to the first primitive (non-compound) task “Get in Car.” After this task the program automatically executes “Drive to Store.” Once “Arrive at Store” executes, the goal has been reached and the HTN terminates. Going back to the “Find Keys” constraint node, if one does not have keys (value of “Find Keys” constraint has value of “False”) then the “Walk to Store” primitive task is reached and then “Arrive at Store” performs and the HTN terminates. Using HTNs this way is great if and only if the conditions (known state of the world) have not changed since the plan was formulated and execution was started. But cars do break down and/or run out of gas. Road construction causes detours resulting in alternating course (changing plan) in order to ensure that the goal “Go to Store” is met. To address these types of problems, modifications need to be made to the HTN methodology. In the next subsection the use of dynamic planning is discussed.

### **2.6.3 Example of Dynamic HTN Planning**

If the state of the world changes after a developed plan starts execution the entity may never reach its goal. To account for a possible change, the programmer may add an interrupt goal to force the HTN planner to reassess and replan actions to ensure a feasible plan is made. Interrupt nodes for this demonstration are labeled red. Balogh et al. state that interrupt nodes serve two purposes: “represent tasks that takes some time to complete and are likely places where the plan become invalid and allow for the ‘lazy generation’ of the plan only part of the plan needs to be generated” [3, p. 4]. In addition to the interrupt node, the idea of a replan event needs to be introduced. A “replan event” will cause the tree to “suspend execution of the current plan and reevaluate the HTN being used to determine how the plan should be changed based on the current state of the world” [30, slide 13].

Looking at Figure 2.12, it displays the “Go to Store” task with the addition of interrupt goals. For this example the idea is that one is at home (at the start of execution) and along the way to the store, a problem will arise causing the planner to account for this unexpected event, which would normally cause the plan to be void and the goal never met. The replan event indicates

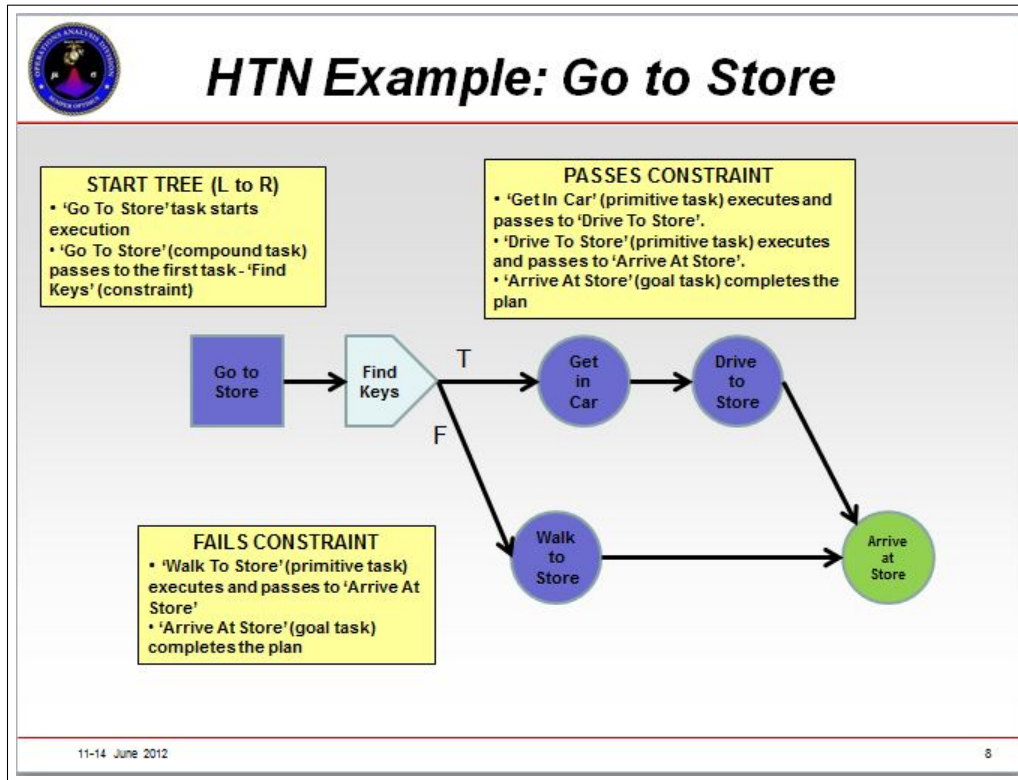


Figure 2.11: Planning to go to store without accounting for changes in the environment. From [30, slide 8]

that a state is reached where a change in the state of the world could occur and the whole plan will re-evaluate. When an interrupt node is hit, the execution of the tree halts, pending a replan event causing the tree being reevaluated. The replan event for this tree is “Stopped Moving.” First the “Go to Store” compound task executes and reaches the “At Store?,” which has a value of “False.” Next the “At Home?” (interrupt node) has value “True” and tree planner moves to constraint node “Find Keys.” Since that is true “Get in Car” task is planned, which then triggers the interrupt node “Drive to Store,” causing the planner to stop executing. Since the replan event “Stop Moving” was triggered after “Drive to Store” event executed, the tree re-executes at the root node “Go to Store.” The tree looks at constraint node “At Store?,” whose value is “False,” then moves to “At Home?” constraint node whose value is also “False” then plans an “Walk to Store” node. The “Walk to Store” node triggers the “Stop Moving” event causing the tree to be executed again. The task “Go to Store” executes and moves to constraint node “At Store?” being “True” causing the “Arrive at Store” goal node to execute allowing tree to be completed and finished. The interrupt node allows the tree to stop executing to allow for replan triggers to

cause the tree to form a new plan in order to reach a goal.

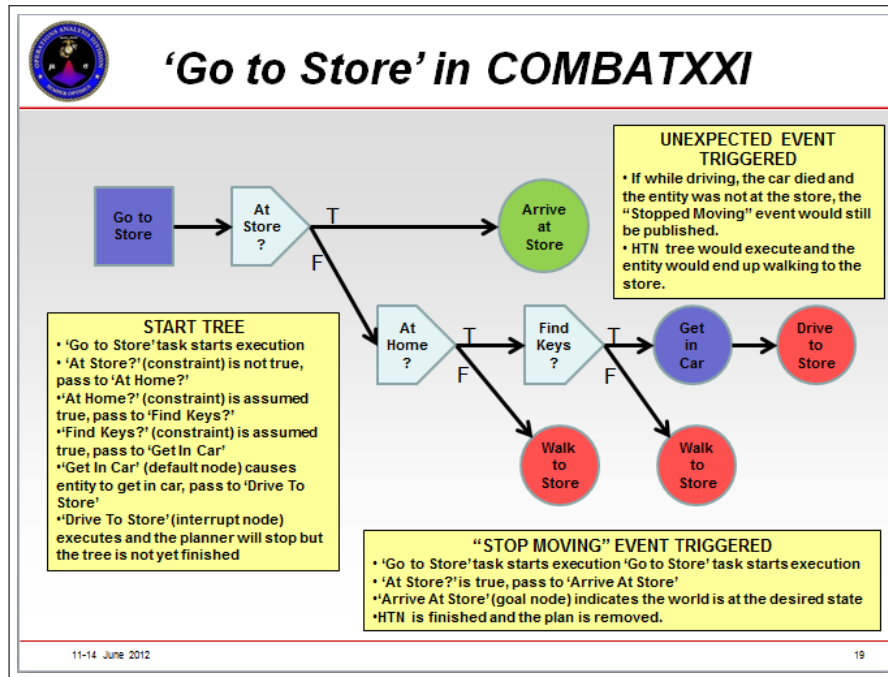


Figure 2.12: Going to store with an interrupt node. From [30, slide 19]

## 2.7 KNOWLEDGE REPRESENTATION

Information can be pieces of fact and fiction that describe the world around us. Knowledge is the collection of information that allows for intelligible action to take place. It is imperative that one knows how to apply knowledge and relate it to ideas that can accurately reflect the message one is trying to convey. Brachman and Levesque say knowledge representation is “the field of study concerned with using formal symbols to represent a collection of propositions believed by some putative agent” [31, p. 4]. Davis, Shrobe, and Szolovits see knowledge representation (KR) as “a surrogate, a substitute for the thing itself, that is used to enable an entity to determine consequences by thinking rather than acting that is, by reasoning about the world than [*sic*] taking action on it” [32, p. 17]. They explain one of the most interesting aspects of reasoning is that one must decide what the surrogate represents and what level of fidelity is the surrogate given when describing the entity [32, p. 18]. Their example consists of an entity planning on assembling a bicycle. There are different parts of the bicycle the person would have to think about (internally) that exist (externally) [32, p. 18]. Ways of representing knowledge via computers have been around for over fifty years. Some means of KR have included “neural networks, theorem proving and expert systems” [33]. In order to communicate

an idea effectively, one needs to be able to speak in a manner that both parties can understand (a common language). There are languages (old and new) that have been adopted and created to help present the understanding of the world. A unique problem with Artificial Intelligence (AI) is that a program needs to have precise understanding of what the digit “2” is versus the string “two.” We, as humans, must know how and when to use “2” and “two” when talking to one another. For example, concepts such as aircraft, flying, and travel can be closely related or very far apart. It all depends on the idea that you are trying to represent. If one is trying to plan a trip to Disneyland the first thought is the modes of “travel.” Travel could happen by driving, sailing, swimming (if you are a good swimmer) or flying. By flying, concepts such as “soaring,” “being lighter than air,” “flapping of wings” may come to mind. However, since the context is in the realm of transporting a person from their home town to Disneyland, it is most likely that “flapping of wings” is not the true method of flying in this scenario. A depiction of being enclosed in a fuselage resembling something like a Boeing 737 or Airbus 350 and soaring through the clouds may be more accurate. The aircraft engineer might visualize movement of air molecules over the wings creating lift opposing the downward force of gravity. There are infinite ways to represent knowledge, unlike many problems that have a constrained solution space. Below are some areas that have been used to formally represent knowledge.

### **2.7.1 Propositional Calculus**

Propositional calculus (aka propositional logic) is said to have been first created by the philosopher Aristotle [34]. In general, it is made up of symbols, truth symbols, and connectives. Propositional logic “is the branch of logic that studies ways of joining and/or modifying entire propositions, statements or sentences to form more complicated propositions, statements or sentences” [34]. These propositional sentences declare if the proposition is true or false [34]. Well-formed formulas (WFFs) are legal sentences in PL. Table 2.2 shows the symbols and well-formed formulas. Propositional logic is concerned with the assigning of truth-values to a proposition (true or false). A PL sentence cannot be both true and false. The “Internet Encyclopedia of Philosophy” provides a very thorough review of Propositional logic [34].

### **2.7.2 Predicate Calculus**

Predicate calculus allows one to tie relationships together and make clearer connections to objects, persons, ideas,...etc. Sentences are formed in predicate calculus just like they are in propositional calculus using the same connectives as found in Table 2.2. In propositional logic the atoms “P” and “Q” represent a proposition that only has meaning to itself and itself alone.

Symbols of Propositional Calculus		
Propositional Symbols	P Q	"The sun is shining." "Today is Friday."
Truth Symbols	true false	"True" "False"
Connectives	$\neg$ $\wedge$ $\vee$ $\rightarrow$ $\equiv$	negation, "not" conjunction, "and" disjunction, "or" implication, "if...then..." equivalence, "equivalent to"
Well-Formed Formulas (WFFs)	P, Q, R $\neg P$ $P \rightarrow Q$ $P \wedge Q \equiv R$	Propositions "not P" "P implies Q" "P and Q is equivalent to R"

Table 2.2: Propositional Calculus Symbols.

However, with predicate calculus, values can be assigned to the propositions individually, which allows the ability to make new inferences by explicitly referring to the values assigned to the symbols. If "X" is a variable representing hours of the day and sentence "weather(X, sunny)" is true then literally one is expressing "the weather is sunny all the hours of the day." Two new symbols are introduced in predicate calculus: the universal quantifier,  $\forall$ , and existential quantifier,  $\exists$ . The universal quantifier states that a sentence is true for all values of the variable. The existential quantifier states that there exists at least one value of the variable that make a sentence true. Luger states, "A major challenge for AI programmers is to find a scheme for using these predicates that optimizes the expressiveness and efficiency of the resulting representation" [35, p. 60]. Table 2.3 shows the power of representing "knowledge" using predicate calculus. Predicate calculus allows for a formalized fabrication of information enabling the structuring of knowledge bases, which will be shown later in the thesis.

### 2.7.3 Semantic Networks

A semantic network can be used to represent knowledge by using meaningful relationships with nodes and arcs. According to associationists, "when humans perceive an object, that perception is first mapped into a concept. This concept is part of our entire knowledge of the world and is connected through appropriate relationships to other concepts" [35, p. 229]. AI pioneers Collins and Quillian fabricate a semantic network based on human information and storage times. It is said that by using inheritance, humans are able to decrease the size of needed knowledge

Predicate Calculus and the English Language	
If it does not rain on Monday, Tom will go to the mountains.	$\neg \text{weather}(\text{rain}, \text{monday}) \rightarrow \text{go}(\text{tom}, \text{mountains})$
Emma is a Doberman pinscher and a good dog.	$\text{gooddog}(\text{emma}) \wedge \text{isa}(\text{emma}, \text{doberman})$
All basketball players are tall.	$\forall X (\text{basketball\_player}(X) \rightarrow \text{tall}(X))$
Some people like anchovies.	$\exists X (\text{person}(X, \text{anchovies}))$
If wishes were horses, beggars would ride.	$\text{equal}(\text{wishes}, \text{horses}) \rightarrow \text{ride}(\text{beggars})$
Nobody likes taxes.	$\neg \exists X \text{likes}(X, \text{taxes})$

Table 2.3: English sentences being used in predicate calculus. After [35, p. 60]

base by this form of abstraction. Figure 2.13 shows a Collins and Quillan hierarchy of two distinct type of birds, characteristics of birds, and how they both relate to animals representing the idea of inheritance. A node could represent concepts such as objects, ideas, or situations. The arc represents the relationship between two objects, ideas or situations. Figure 2.13 shows a representation of how human information could be stored in memory of a computer via a semantic network. This is why it is important to understand how a semantic network can be built to better understand beings, ideas, and the relationships they share.

#### 2.7.4 Ontologies

An ontology is a way of creating our knowledge base (KB) of the world and having the agent of interest act upon the knowledge it knows of the world and make new inferences. We can use predicate calculus to help create the “facts” of our known world. *Merriam-Webster* defines ontology as “a branch of metaphysics concerned with the nature and relations of being” [37]. Brachman and Levesque state “[ontology represents] the kinds of *objects* that will be important to the agent and the *properties* those objects will be thought to have, and the *relationships* among them” [31, p. 32]. There are software programs that can assist in developing simple to rather complex ontologies that would be otherwise impossible for a human to store in his or her brain. The Web Ontology Language (OWL) is a high-level knowledge-based language used to construct knowledge base for many domains. It allows for machines (computers) to automatically process information based on the predicate’s expressiveness. Childers discusses the benefits of using ontologies “[because ontologies]...allows programs to understand, process, and infer new information from coherent data” [38, p. V]. An ontology should assist in the creation of a knowledge base enabling a better understanding of the importance of acquiring accurate and useful information such from sensors in a UAS.

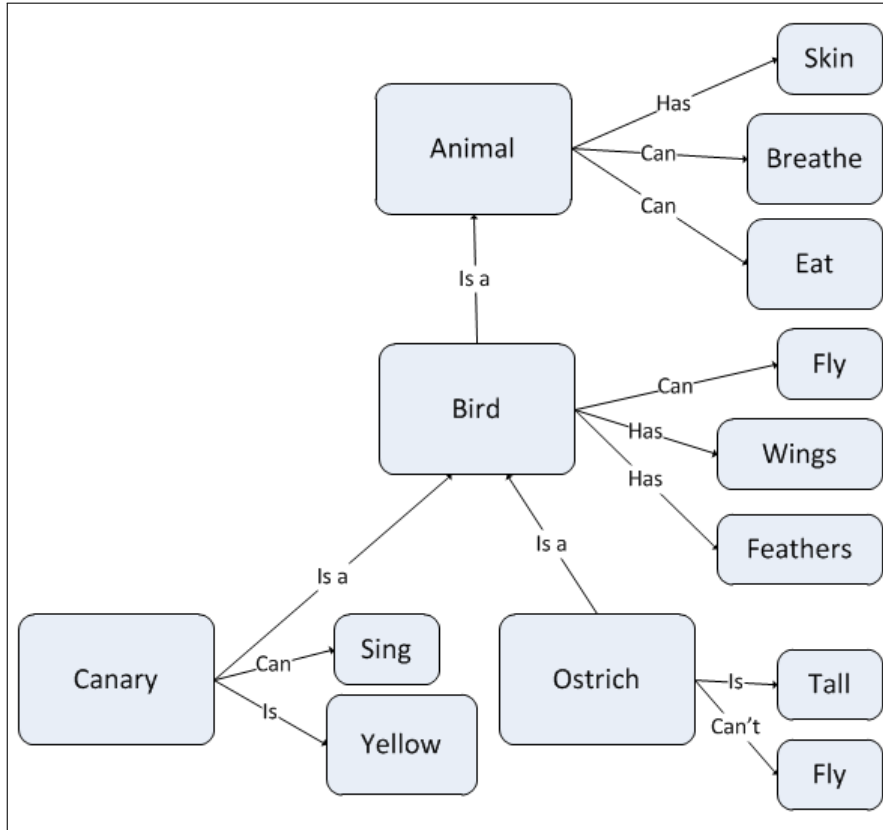


Figure 2.13: Semantic Network from Collins and Quillian modeling how information would be stored in a computer. This is a modified remake of their illustration. From [36, p. 241]

## 2.8 SUMMARY

The DoD has use for many types of UASs, of which one is reconnaissance. There are methods to model the sensors and sensing systems on these aircraft, however, they have drawbacks. In modeling, there are many ways to represent knowledge and decision making.



---

---

# CHAPTER 3: METHODOLOGY

---

## 3.1 INTRODUCTION

The methodology begins with developing a scenario that implements the Raven UAS in an typical Southwest Asian (SWA) environment. HTNs form the basis for planning and creating behaviors of the entities in the scenario. The creation of design of experiments and an ontology assist in measuring the effects of UAV operator skill and decision making.

## 3.2 SCENARIO

In order to understand the capabilities, limitations, pros and cons of an UAS, it is imperative that we consider the context of its usage. For this thesis, the scenario takes place in Southwest Asia. The 5-Ws (who, what, when, where and why) give the reader the understanding of what kind of mission will take place. Below is a brief description of the scenario [39] to be modeled in COMBATXXI. It represents a scenario that a small unit living on a combat outpost (COP) may have to perform while conducting operations with a host nation's army in SWA (see Appendix B).

- Who: Squad of U.S. Forces from COP Ramrod consisting of 7 vehicles and 1 RQ-11B RAVEN UAS.
- What: Conduct Resupply Run to Outpost X-Ray
- When: 0800 Hours Local Time
- Where: On an unimproved route consisting mainly of dirt and some gravel.
- Why: To resupply a joint American/Host Nation Outpost consisting of approximately 30 personnel.

Along the route there are areas known to have a higher than normal likelihood of containing an IED. There are four named areas of interest (NAIs) that have an equal chance of having an IED. For this study the real IED can be found in NAI # 2. The Raven UAS provides the added ability for the convoy commander to search the area at a safer distance than without a Raven. In the study it is assumed that a convoy will not see signs of a possible IED until they are within close proximity of one (approximately 50 meters). The mission can be seen in Figure 3.1 with NAIs and the convoy's start and end points labeled.

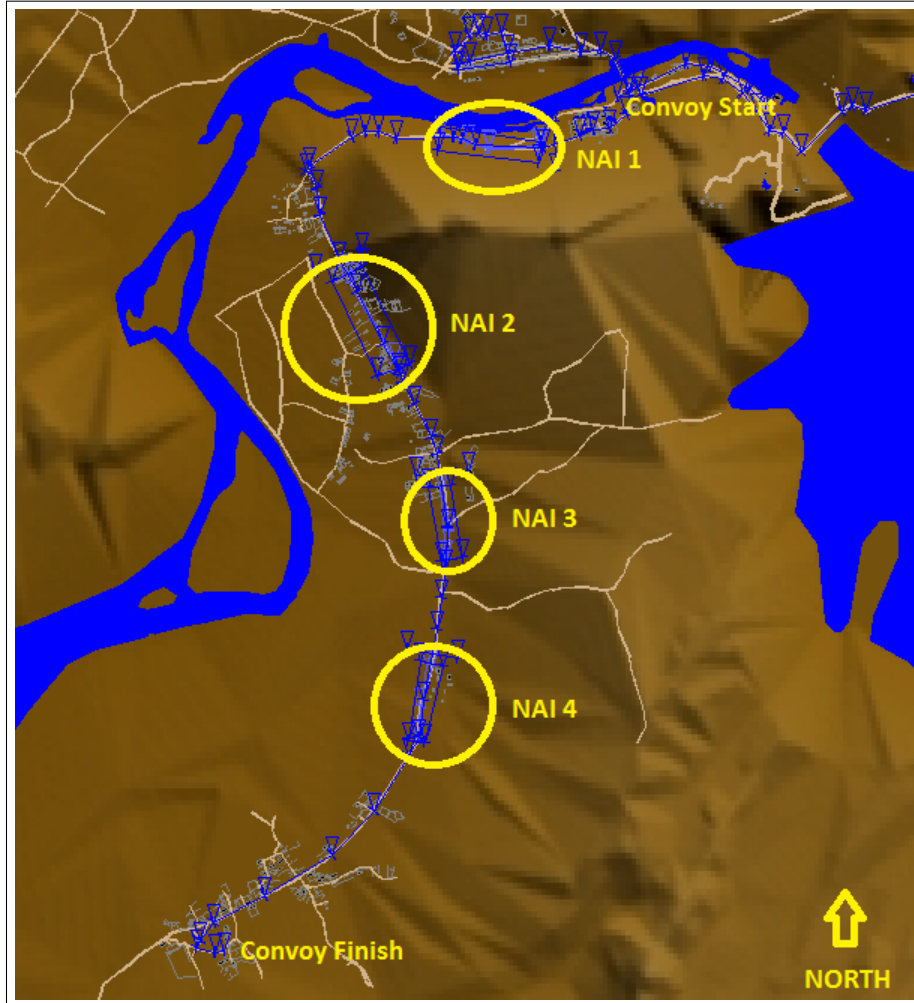


Figure 3.1: Mission Overlay in COMBATXXI.

When the convoy passes each phase line (PL) at the start of each NAI, they will stop and deploy the Raven. It will search each NAI by flying along a pre-determined path in search of an IED, see Figure 3.2. An object representing the IED will only detonate with an assigned standard uniform probability of .8 given that a convoy entity enters IED's sensor range of 10 meters. If an object is outside of the 10 meter range, the IED will never activate. The Raven UAS will defeat the IED if 1) the UAV detects the object that "could" be an IED (based on new UAV sensor model) and 2) the operator believes that the object detected *is* an IED and takes appropriate action by calling the unexploded ordnance clearing team (EOD). Calling EOD will always result in a 60-minute time penalty and IED being destroyed. If the convoy is hit by an IED, then mission is over. Success is the ability for convoy to safely arrive without taking any casualties.

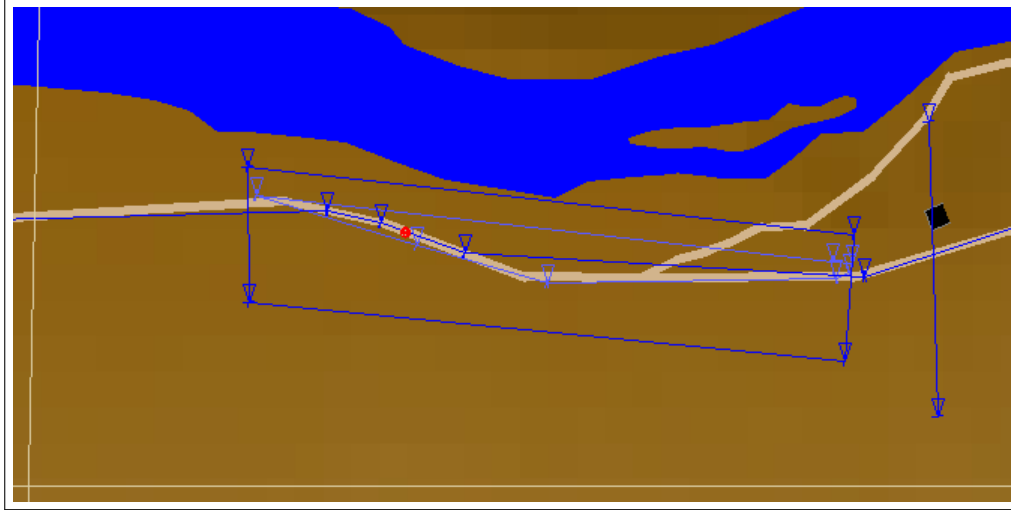


Figure 3.2: Screen shot of NAI 1.

Next, the research will look in to developing behaviors of our entities in the COMBATXXI scenario.

### 3.3 FORMULATION OF BEHAVIORS

The study uses HTNs to define behaviors for the entities modeled in the research. A tool called "Behavior Studio," developed by the MOVES Institute, provided for formation of behaviors. HTNs promise to make coding behaviors less time intensive, more efficient, etc., and in doing so allow for quicker formation of behaviors to be analyzed. An HTN tree is one separate entity that has a root node and at least one goal node. Each tree is organized with an initialization branch and execution branches. The execution branches consist of **events** and **modelEvents**. The events generally consist of events that are organic to the planner, whereas modelEvents plans for the execution of events directed by COMBATXXI exclusively. The list of trees for this study are as follows, with a broad description of the behavior to be modeled.

- **ConvoyMoveInFormation**: Behavior of moving convoy along the route.
- **OperateIED**: Behavior enabling detonation of IED when appropriate.
- **OperateUAS**: Behavior for operation of the UAV and convoy.
- **UAVSensor**: Behavior of the Raven UAV sensors.

#### 3.3.1 Moving the Convoy HTN

The behavior tree labeled **ConvoyMoveInFormation** is responsible for ensuring the convoy reaches its final destination (Outpost X-Ray), see Figure 3.3. When the tree executes it

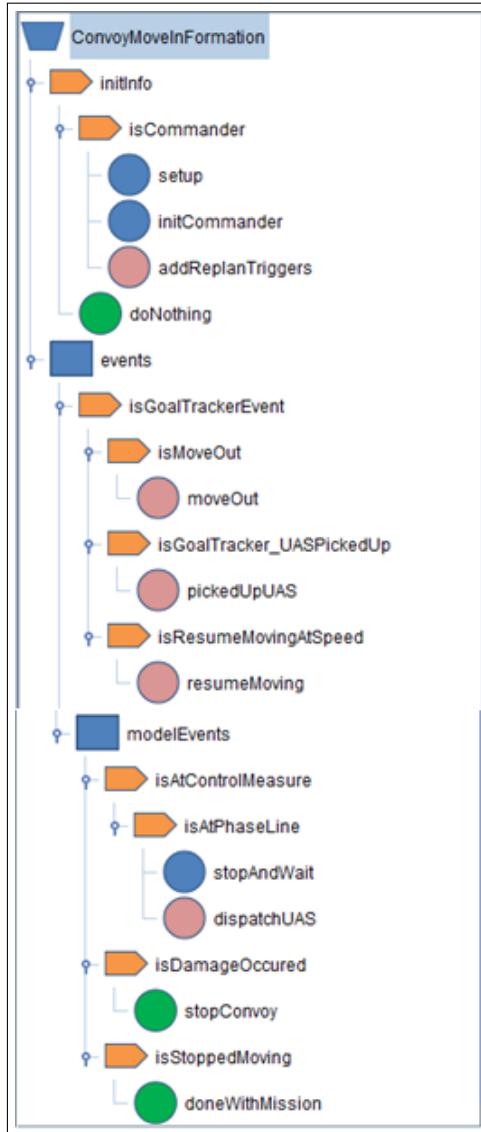


Figure 3.3: **ConvoyMoveInFormation**Tree. This tree will form a plan to get the convoy from the COP Ramrod to OP X-Ray.

first “initializes” using the **initInfo** and runs to the first condition **isCommander**, which tells that there are the NAI phaselines to pay attention to and schedules a “Move Out” event. Then several replan triggers are added for the tree listen to via **addReplanTriggers** node. The **doNothing** node tells the tree to end planning if someone other than the commander is assigned this tree. Once the **initInfo** branch completes, the execution is interrupted and is waiting for a replan event.

When a replan event occurs different nodes may execute. The **moveOut** node orders the convoy

to move out. The **pickedUpUAS** interrupt node informs the convoy commander if the UAS (Raven) is physically back with convoy. If the UAS has not seen any artifacts that could be dangerous (such as potential IEDs) it schedules an “ResumeMovingAtSpeed” event. If the condition **isResumeMovingAtSpeed** is true the plan passes to the **resumeMoving** interrupt node that has the convoy resume normal convoy speed.

Next, moving on to **modelEvents** node, if conditions are true in nodes **isAtControlMeasure** and **isAtPhaseLine** then the convoy will come to a “halt” (**stopAndWait**). Raven dispatches in search of IEDs, respectively. The **dispatchUAS** interrupt node will assign the **OperateUAS** tree to the the UAV entity to execute and search the current NAI.

If the **isDamageOccured** node is “true,” then the convoy has been hit by an IED, meaning the mission is over and the plan terminates appropriately. And finally, if the convoy has reached its final destination, the **doneWithMission** node is executed and the tree terminates. The complete **ConvoyMoveInformation** tree can be seen in Appendix C.

### 3.3.2 Operating the IED HTN

The HTN tree labeled **OperateIED** forms the behavior of the improvised explosive device. Figure 3.4 shows the tree in its entirety. First the tree jumps into the **initInfo** node and checks to see if the object is a real IED (**isRealIED**) by checking to see if probability of detonation is greater than 0. If object is not an IED the tree moves to goal node **setIsNotIEDInfo** and finishes. However, if **isRealIED** condition is “true” the tree moves to the **startObservation** node, activating the IED if convoy entity is within the activation (sensor) range of IED. Next the **addReplanTriggers** executes and ensures **OperateIED** tree is replanned if entities enter IED’s area of interest (AOI), the simulation publishes a “GoalTracker\_Detonate Now” event. Now moving along into the **events** branch, the IED detonates as long as “doGoalTracker\_DetonateNow” is true. The IED will detonate with a probability (0.8 for the study) passed in as a parameter and is ALWAYS catastrophic (no survivors). The **modelEvents** node schedules a replan trigger called “GoalTracker\_DetonateNow” when the second vehicle enters the IED’s killzone. The complete **OperateIED** node can be seen in Appendix D.

### 3.3.3 Operating the UAS HTN

The HTN tree **OperateUAS** forms the behavior of operating the UAS, see Figure 3.5. Like earlier HTNs, the tree first initializes and adds replan triggers that will cause the tree to replan when the UAS is finished searching or scanning the NAI. Next, the tree plans routes for

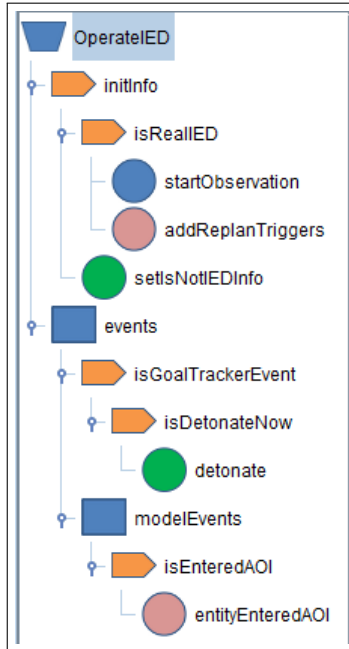


Figure 3.4: **OperateIED**Tree. Picture of **OperateIED** HTN.

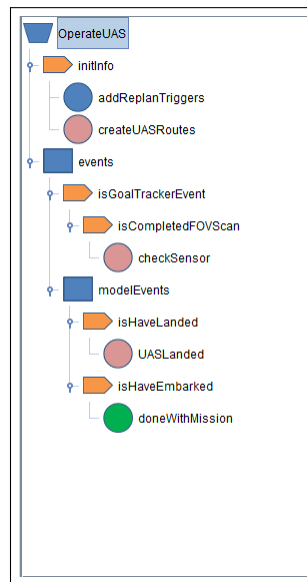


Figure 3.5: **OperateUAS**Tree. Picture of **OperateUAS** HTN.

the UAV to operate, see interrupt node **createUASRoutes**. Moving to **events** node, the branch is responsible for checking the sensor for objects found once it finishes scanning. The **isCompletedFOVScan** checks if UAV scanning its field of regard (FOR). If **isCompletedFOVScan** has value of “true” then the interrupt node **checkSensor** executes. In this node, if the UAV

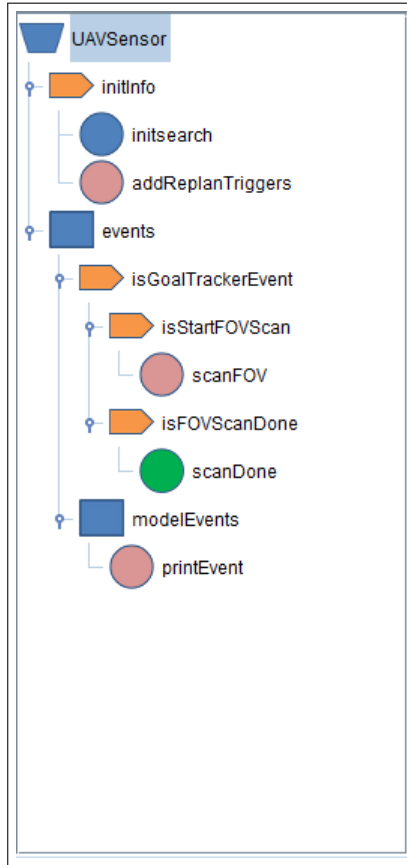


Figure 3.6: **UAVSensor** Tree.

entity has not returned to the convoy or is greater than 100 meters out from vehicle it disembarked, the tree will push the UAV sensor tree to **UAVSensor** tree where the UAV entity continues its search of FOR with necessary parameters.

When the tree traverses down to the **modelEvents** once the constraint node **isHaveLanded** is true, the **UASLanded** node executes—meaning that the Raven UAV has landed. Lastly, once the UAV embarks the convoy (it just completed its mission), then the goal node **doneWithMission** executes and the human operator determines if this is either an IED or not an IED. A more detailed depiction of the tree can be given in Appendix E.

### 3.3.4 UAV Sensing HTN

The **UAVSensor** HTN is responsible for modeling the sensor of our Raven UAV (see Figure 3.6). The goal here is to model a user observing through a single field of view. First the tree initializes and the **initsearch** node computes a scan time for the FOR (field of regard),

which is simply what the person is focusing on the screen. The field of view (FOV) is the entire screen that the observer cannot see due to the eyes abilities to only focus on a few degrees at a time. Moving down into the **events** branch if the **isGoalTrackerEvent** node and **isStartFOVScan** node are true the interrupt node **scanFOV** executes. In this node, the UAV searches for objects on the ground using an EO sensor. It has the probability of detecting objects in its path based on the distance. Looking at the **isFOVScanDone** node is true, it passes to the goal node **scanDone**. This node finishes the tree and the “observations” will be looked at in the **OperateUAS** node (mentioned earlier) for determination if it is or is not an IED. A more detailed depiction of the tree in its entirety can be found in Appendix E.

### 3.4 UAV NOTIONAL SENSOR MODEL

To achieve the representation of a UAS modeling the effects of operator skill, we need a sensor. The sensor must be able to report to the **OperateUAS** HTN (representing human behavior) all possible IEDs. Currently the sensor in COMBATXXI is overly sensitive and cannot determine if an object found is an IED unless it is hard-scripted into that object as a real IED. Therefore the study uses a simple notational sensor developed by Imre Balogh (see Appendix F). The UAV sensor model can detect objects in COMBATXXI using some simple geometry and probability. Looking at Figure 3.7 one can see how the UAV detects an object. Probability of detection is based on the value of  $r$ . For this model the limit to detect an object is 220 meters. If the range of an object is 110 meters then the observer has a .50 chance of detecting object given that it is in the field of view of the sensor. Next, the study will look at the idea of creating a knowledge base (an ontology) that would assist the UAS operator in making decisions based on clues such as “disturbed earth,” and “loose wires,” instead of finding an object and flipping a coin to say that it is or is not an IED (as explained for COMBATXXI).

### 3.5 ONTOLOGY OF IED

Ontologies are a useful way of expressing knowledge of the world. They help to formalize an understanding of what objects will be represented and the relationships the objects share together [31, p. 32]. The agent in this study is the UAS and its ability to discern threat-type based on information received.

#### 3.5.1 Developing an Ontology

This study seeks to develop a constraint that will allow our HTN to categorize the objects it observes. This categorization defines a state based on indicators in the ontology. This perceived state determines the selection of behaviors to follow in the HTN.



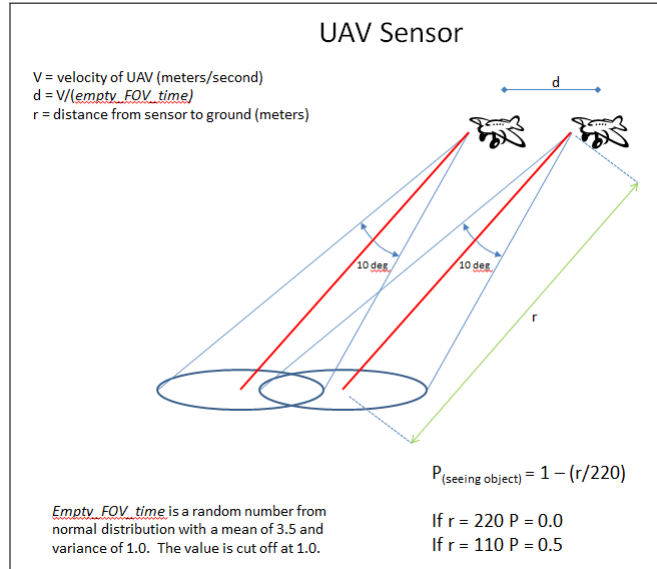


Figure 3.7: A depiction of the UAV Sensor Model used in this study. Courtesy from Imre Balogh [40].

This research uses the free, open-source program Protégé 4.3 to create a knowledge base (KB) of the world of IEDs [4]. According to the Protégé website, “Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats”[4]. There are two methods that ontologies can be formed: (1.) Protégé-Frames editor “*frame-based*, in accordance with the Open Knowledge Base Connectivity protocol (OKBC)” and (2.) Protégé-Owl editor, which “enables users to build ontologies for the *Semantic Web*, in particular the W3C’s [(World Wide Web Consortium)] Web Ontology Language OWL” [4]. The W3C website states “The OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms” [41]. The study uses Protégé-OWL editor (version 4.3, build 304) to create the IED ontology. According to the Protégé-OWL website, “OWL ontology may include descriptions of classes, properties and their instances. Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e., facts not literally present in the ontology, but entailed by the semantics” [4]. These logical consequences are derived by using the reasoner. A reasoner is “a piece of software able to infer logical consequences from a set of asserted facts or axioms” [42]. This study uses the Fact++ (fact plus plus) description logic (DL) reasoner.

### 3.5.2 Representing Information in an OWL Ontology

There are three main concepts when building an OWL ontology: classes, properties, and instances. The screen you see in Figure 3.8 shows the Protégé 4.3 user interface; the tabs used to create the ontology along with the reasoner are circled.

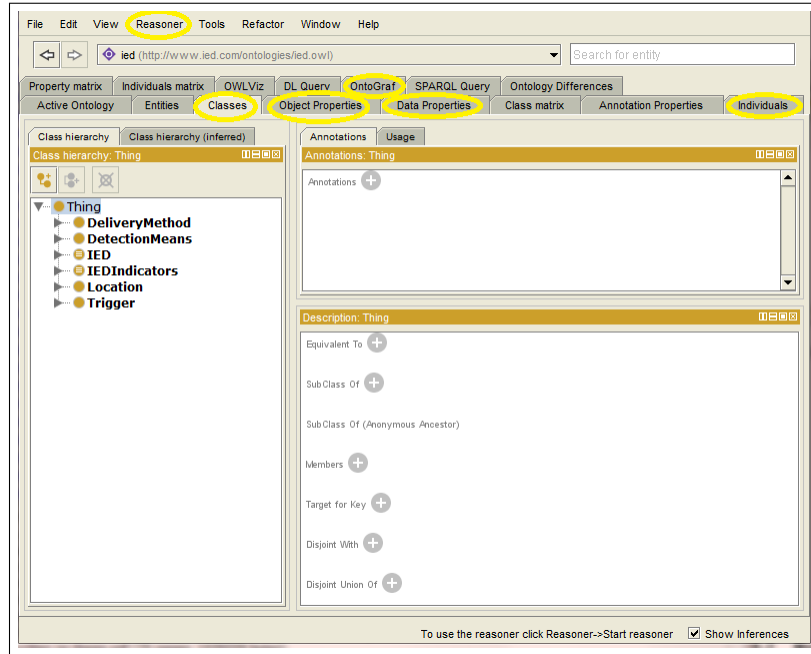


Figure 3.8: Screen picture of Protégé 4.3 program with the main tabs used circled. The Classes tab is shown.

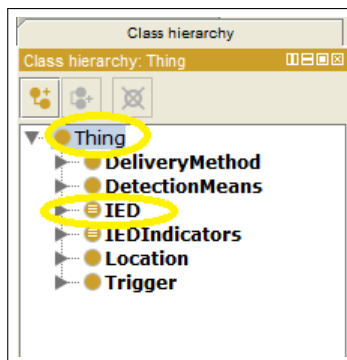


Figure 3.9: **Thing** class shown with its subclasses. **IED** class is a subclass of class **Thing**.

### 3.5.3 Defining Classes in IEDOntology

For this research, a preliminary ontology of IEDs was developed. The ontology is not fully defined, but provides a limited set of class and property definitions to demonstrate use of the

ontology and automated reasoning to make inferences about detected IEDs based on information reported from human or automated observers in the battlespace.

In the Ontology (see Appendix G), the main classes can be seen in Figure 3.9. The **IED** class is shown (circled) with the other classes in the IED ontology. The main class is always the **Thing** class [43, p. 15]. The ultimate goal is to classify something as being some type of “IED,” but in order to have a better understanding of the makeup of an IED, it is imperative that its sibling classes be defined first. The first sibling of class **IED** is the **DeliveryMethod**, which defines

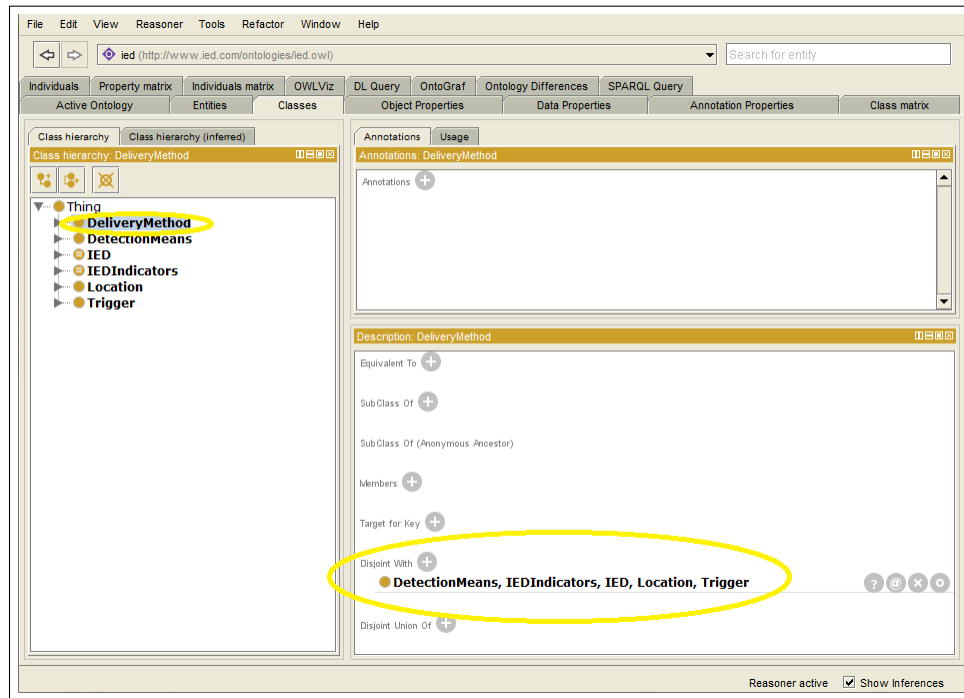


Figure 3.10: **DeliveryMethod** class shown with its disjoint sibling classes.

delivery method of an IED to its intended target (via human, package, or vehicle). The classes **Human**, **Package**, and **Vehicle** are subclasses of the parent class **DeliveryMethod** and are siblings to each other. This class is marked as disjoint from its sibling classes in the class hierarchy, as shown in Figure 3.10. We do not make them disjoint because the reasoner will fail. We make them disjoint because they truly are disjoint from each other. That information is used by the reasoner for its processing. The **Human** class is seen as a subclass of **DeliveryMethod**, disjoint with siblings **Package**, **Vehicle**, see Figure 3.11. It is not too difficult to understand how the tree can assist in visualizing the hierarchy of a class such as **DeliveryMethod**. The class **DetectionMeans** is a means of finding an IED, but is not explicitly used in defining an IED. The **IEDIndicators** class is a depiction of objects that a

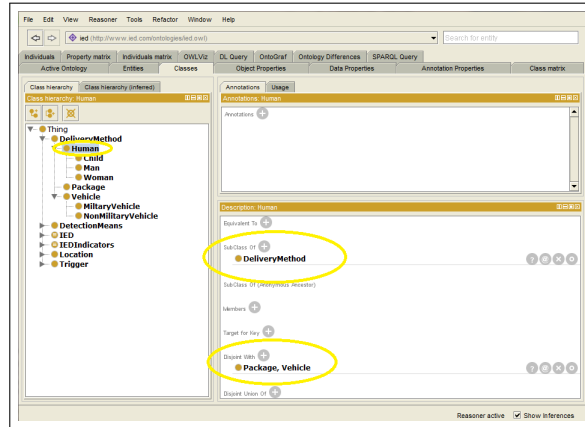


Figure 3.11: **Human** class shown with its disjoint sibling classes.

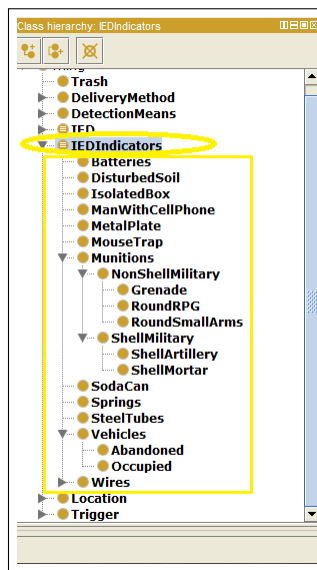


Figure 3.12: **IEDIndicators** class shown with its subclasses contained in the yellow box.

user may be able to see when utilizing a UAS such as a Raven (see Figure 3.12). The **Trigger** class represents means to initiate an IED, see Figure 3.13. It would be plausible for a user of a UAS to see these types of objects while scanning an NAI for an IED. In **Trigger** class the subclass **CellPhone** is highlighted, notice that it is disjoint with its siblings.

### 3.5.4 Defining Properties in IED Ontology

Properties are what relate an individual to another individual in the ontology. An individual is an instance of a class. For example if there was a class “Dog” containing an individual named “Rascal” and he belongs to individual “John” from class “Owner,” “John” would share the

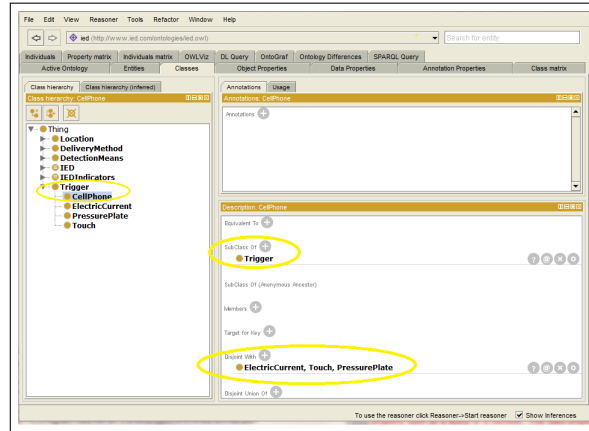


Figure 3.13: **Trigger** class with subclass **CellPhone** highlighted.

object property “hasPet” with individual “Rascal.” In predicate calculus this takes on the form **hasPet(John,Rascal)**. The same logic applies to defining an IED. There are four object properties that make up an IED ontology. The tutorial states “properties link individuals from the *domain* to individuals from the *range*” [43, 33]. Below are the four object properties defined in this ontology:

- **hasDeliveryMethod** Domain: **IED** Range: **DeliveryMethod**
- **hasIEDIndicators** Domain: **IED** Range: **IEDIndicators**
- **hasLocation** Domain: **IED** Range: **Location**
- **hasTrigger** Domain: **IED** Range: **Trigger**

The object property **hasDeliveryMethod** has three subproperties:

- **hasHumanDeliveryMethod** Domain: **IED** Range: **Human**
- **hasPackageDeliveryMethod** Domain: **IED** Range: **Package**
- **hasVehicleDeliveryMethod** Domain: **IED** Range: **Vehicle**

The **hasLocation** object property is selected in the Object Properties tab (see Figure 3.14) The **Functional** box checked implies that each individual in the **IED** class can be related to only one individual from **Location** class. The second class of properties in an OWL-Ontology are datatype properties. Datatype properties “...can be used to relate an individual to a concrete data value that may be typed or untyped” [43, 76]. The two data properties are **hasIEDIndicatorName** and **isConfirmed**, which are further summarized below:

- **hasIEDIndicatorName** Domain: **IEDIndicators** Range: **string**

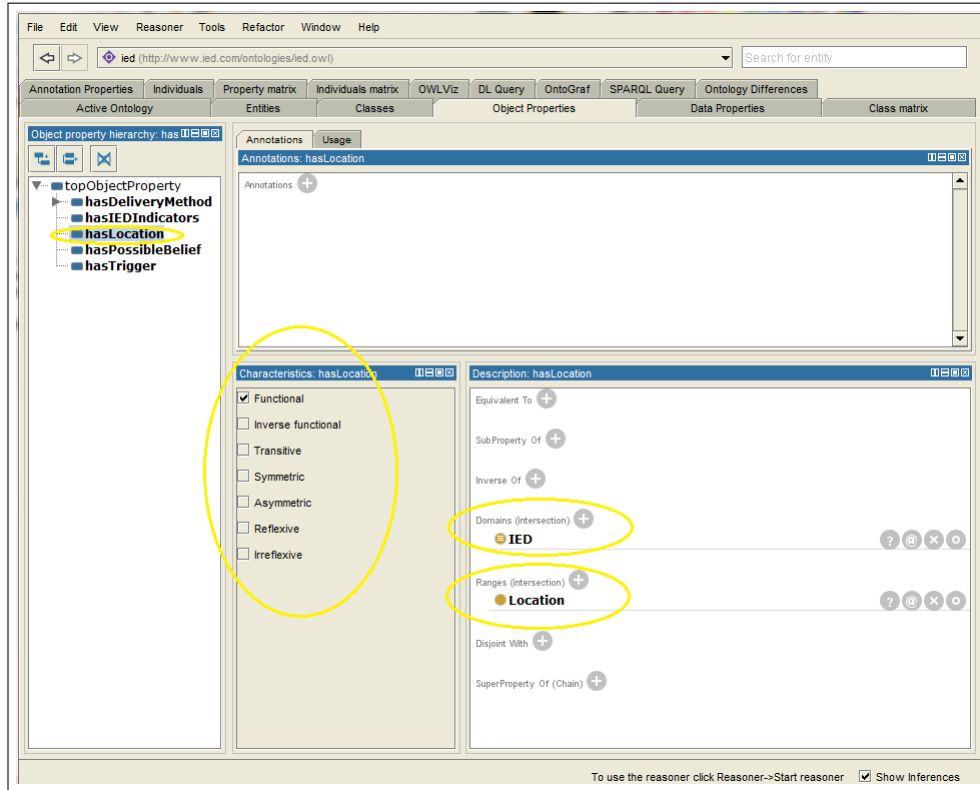


Figure 3.14: The **hasLocation** object property is selected with Domain **IED** and Range **Location**. The **Functional** box is checked as well.

- **isConfirmed** Domain: **IED** Range: **boolean**

The **hasIEDIndicatorName** range “string” will allow user to manually input some type of name for an **IEDIndicator** such as “dirt, iPhone, iron pipe.” The **isConfirmed** allows the user to set if an **IED** individual is a confirmed IED (**isConfirmed** value equals “true”) or not a confirmed IED (**isConfirmed** value equals “false”).

### 3.5.5 Deeper Look in to IED Class

Now that the definition of classes and properties has been given, it is time to look further into **IED** class. The idea is to have the UAV sensor “see” objects (“clues”) in its environment and the operator would take the “clues” and classify something as an IED. In order for something to be classified as being a member of the IED when using Protégé-Owl, the class has to be *necessary and sufficient* aka **Defined Class** [43, p. 55]. A necessary condition is one that needs to be present to be able to make the classification, but may need other conditions [43, 53]. A sufficient condition is one that by itself is enough to make the classification without needing other

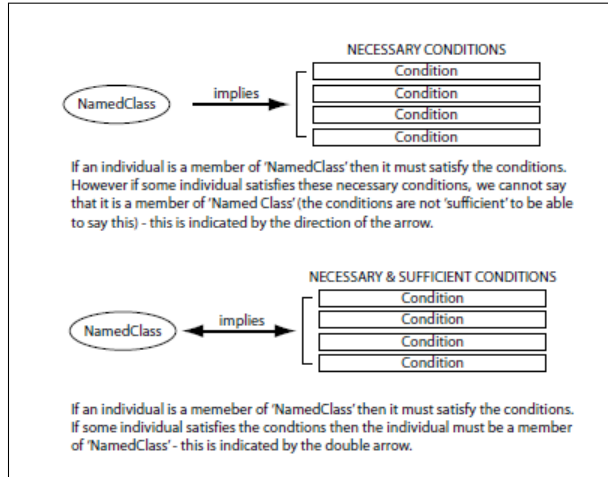


Figure 3.15: Necessary and Sufficient description. Taken from Pizza Tutorial [43, p. 56].

conditions to be true [43, 56]. See Figure 3.15 for an illustration of necessary and sufficient conditions. A class that is defined is indicated by white lines in the orange node circle [43, 57]. The necessary and sufficient criteria for **IED** class is seen in Figure 3.16. The word *some* represents an existential restriction. This says that there is a relationship between individuals of IEDs that has “at least one” relationship along the property (for example) “**hasDeliveryMethod**” to an individual that is a member of the class “**DeliveryMethod**” [43, 59]. The “and” represents an intersection where all of the object properties overlap. Next, we will look at the three different types of IEDs based on how they are delivered: Package (**PackageIED** class), Suicide/Human (**SBIED** class), and Vehicular (**VBIED** class).

The **PackageIED** class is a subclass of **hasPackageDeliveryMethod** and **IED** and it inherits the same four characteristics of an IED based on being a subclass of **IED**. The **PackageIED** class represents delivery by some package, one that is not a suicide bomber (**SBIED**) or vehicular (**VBIED**). One can see that it is also disjoint with its siblings **SBIED** and **VBIED** (see Figure 3.17). The **SBIED** class is a subclass of **IED** and **hasHumanDeliveryMethod some Human**, and like its sibling **PackageIED** it inherits the four properties that make an IED an IED. It is also disjoint with **VBIED** and **PackageIED** (see Figure 3.18). The purpose of **SBIED** is to represent characteristics of a suicide-borne IED. A suicide borne IED delivery could be performed by a child (**ChildSBIED**), Woman (**FemaleSBIED**) or a man (**MaleSBIED**). Since **ChildSBIED** is a subclass of **SBIED** it inherits the properties of **SBIED**, which is defined in the middle of the “Description: ChildSBIED” box labeled

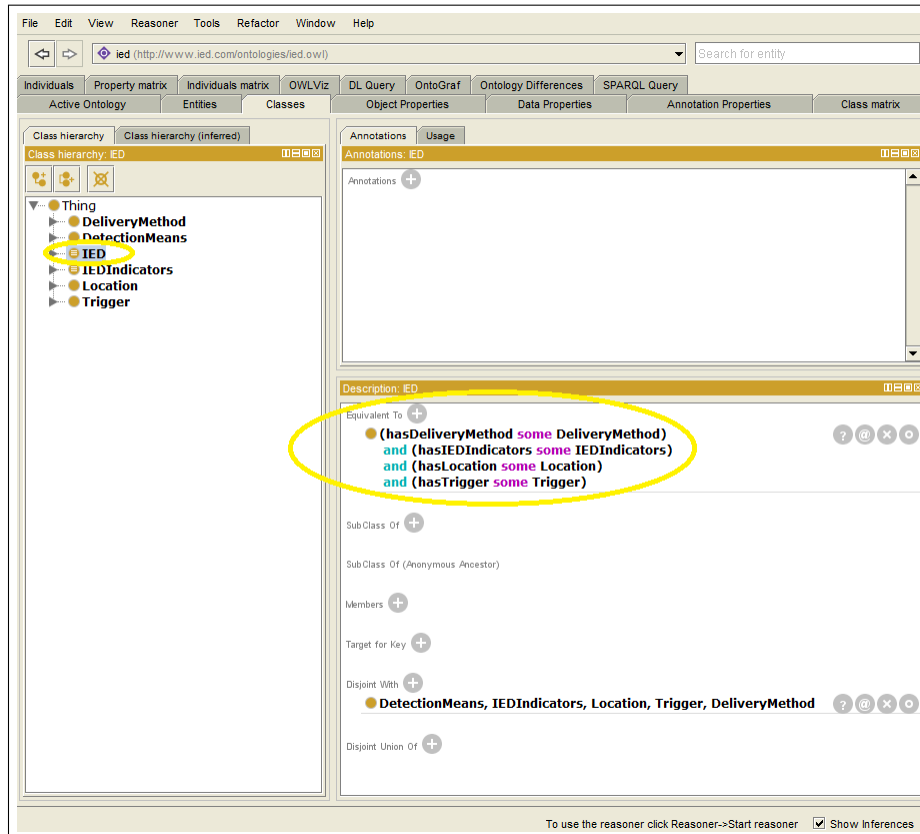


Figure 3.16: The “Equivalent To” box is highlighted showing what must be true in order for an individual to be a member of the **IED** class.

“Sub Class of (Anonymous Ancestor).” The property that explicitly defines **ChildSBIED** is **hasHumanDeliveryMethod** and qualifies the delivery method as a child with the “some child” clause. It is also disjoint with its siblings **MalesBIED** and **FemalesBIED**. The classes **MalesBIED** and **FemalesBIED** are similar to **ChildSBIED** with **hasHumanDeliveryMethod** qualified as Female and Male, respectively. The **VBIED** class is a subclass of **IED** and **hasVehicleDeliveryMethod** and like its siblings **PackageIED** and **SBIED** it inherits the four properties that make an IED an IED. It is also disjoint with **SBIED** and **PackageIED** (see Figure 3.19). The purpose of **VBIED** is to represent characteristics of a vehicle-borne IED. The class **ConfirmedIED** is to be used when an operator sees characteristics of an IED (via UAV) and calls EOD, who then “confirms” that it is a real IED. It inherits the properties of **IED** and has the value **isConfirmed** with a “true” value. The **ConfirmedIED** is disjoint with its siblings **SBIED**, **VBIED**, and **PossibleIED** (see Figure 3.20). If the IED is not confirmed, it will then be classified as a “PossibleIED,” which is discussed next.



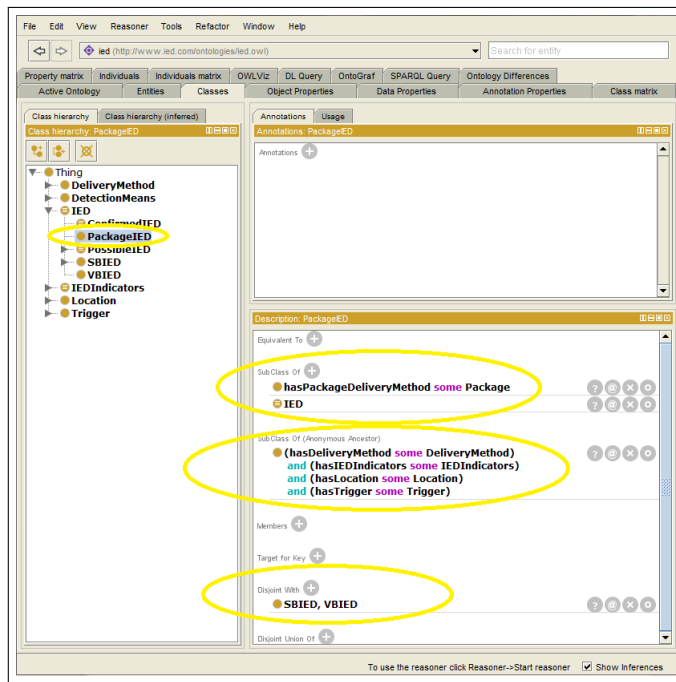


Figure 3.17: **PackageIED** shown.

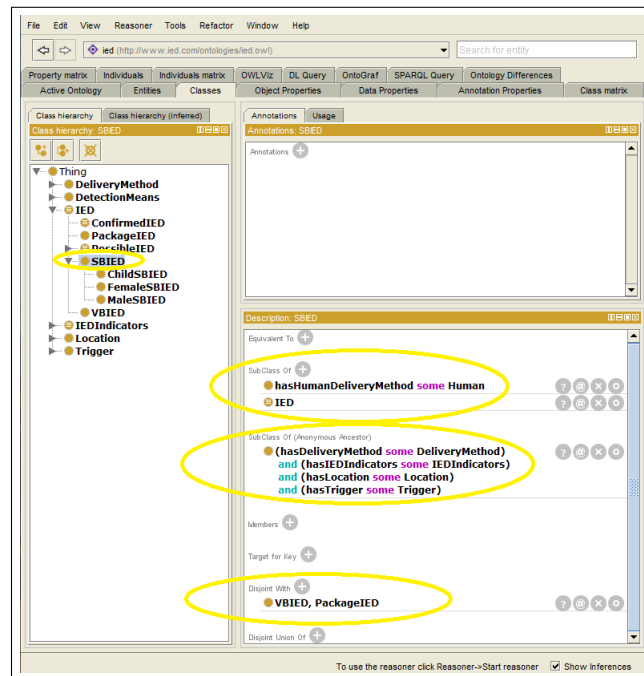


Figure 3.18: **SBIED** shown.

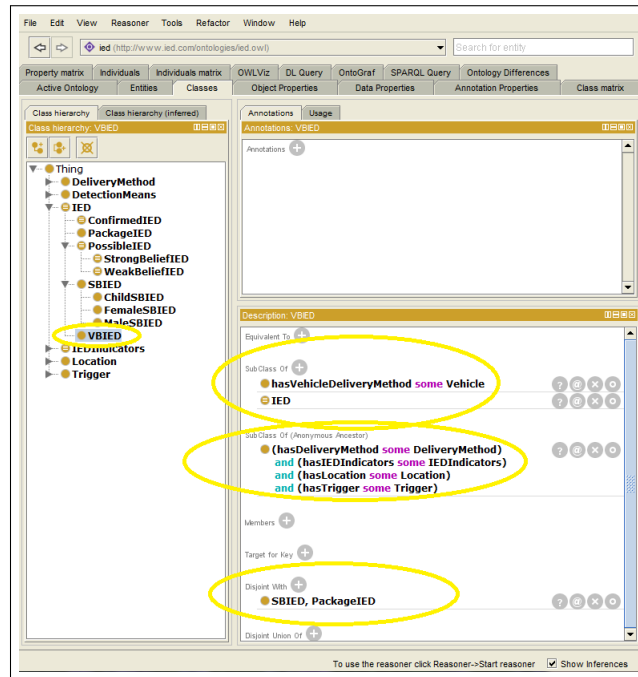


Figure 3.19: **VBIED** shown. A vehicle borne IED delivery happens by a vehicle via **hasVehicleDeliveryMethod**.

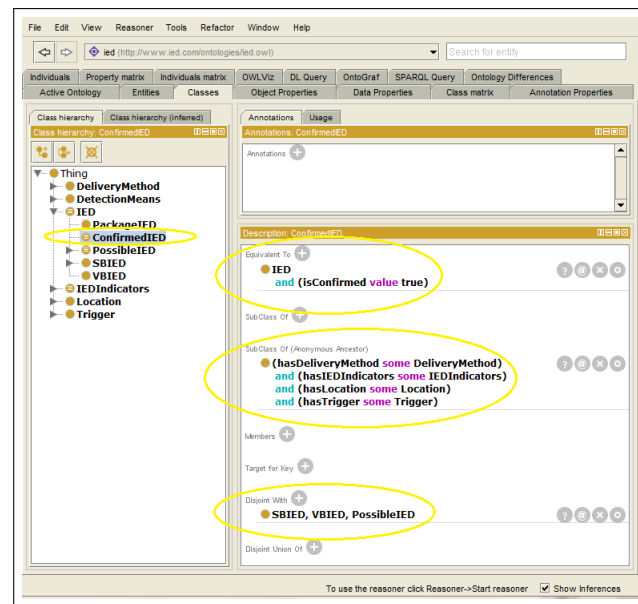


Figure 3.20: **ConfirmedIED** shown.

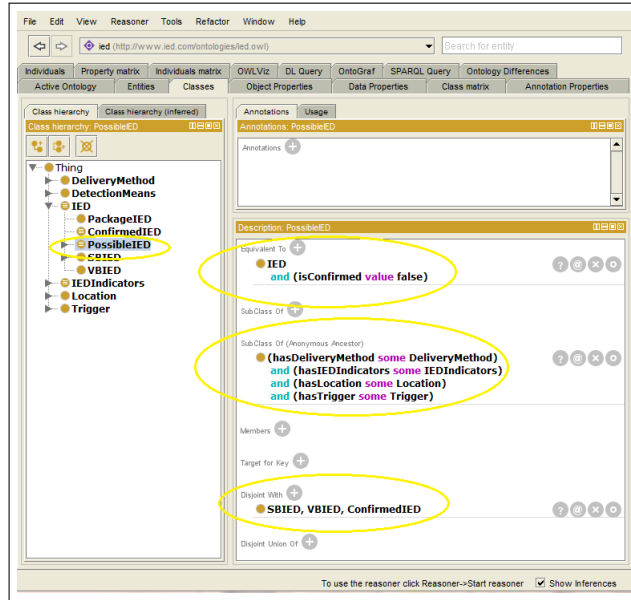


Figure 3.21: **PossibleIED** shown.

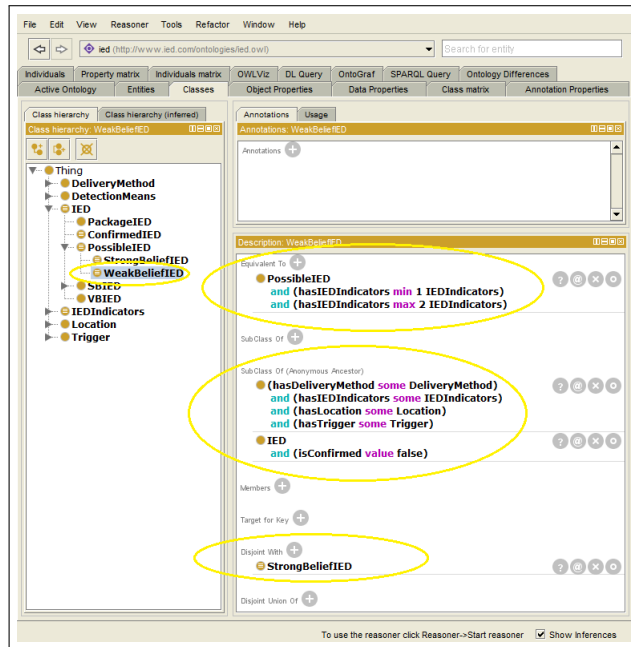


Figure 3.22: **WeakBeliefIED** shown.

The **PossibleIED** is a subclass of **IED** and has the property **isConfirmed** with a “true” value. In theory if an IED is not a “ConfirmedIED” it will always be classified as a “PossibleIED” (see Figure 3.21). A **WeakBeliefIED** inherits all of the properties of a type **PossibleIED** with the exception of its own specific requirement in “Equivalent to” box

(circled on top portion of “Description” window, see Figure 3.22). A **WeakBeliefIED** is a type of IED that has a minimum of 1 IED indicator and a maximum of 2 IED indicators. It is using a new restriction called the universal restriction *only*. The universal restriction states “*if* a relationship exists for the property then it must be to individuals that are members of a specific class” [43, p. 60]. The goal is to create an individual with a minimum of 1 and maximum 2 “IEDIndicators” and have the reasoner classify it as a “WeakBeliefIED.” The minimum and maximum are a type of cardinality restriction. The tutorial states: “A Cardinality Restriction specifies the *exact* number of P relationships that an individual must participate in” [43, p. 73]. The assumption is that you would need a few indicators such as loose soil and wires together at some specific location along a route of travel to have a UAS operator to possibly take a closer look at what they are scanning on the ground. If more than two indicators are found, the reasoner will classify the object as a “StrongBeliefIED,” which is described next. A **StrongBeliefIED** inherits all characteristics of a **PossibleIED** with the addition of a third “IEDIndicator.” Therefore in the “Equivalent To” box the addition of object property **hasIEDIndicators min 3 IEDIndicators** is necessary. The idea is to show that an operator that sees “loose soil,” “box,” and “wires” would probably think that there is a higher suspicion of being a true IED versus only seeing two or fewer indicators. The properties of a **StrongBeliefIED** can be found circled in the “Description: StrongBeliefIED” box, see Figure 3.23.

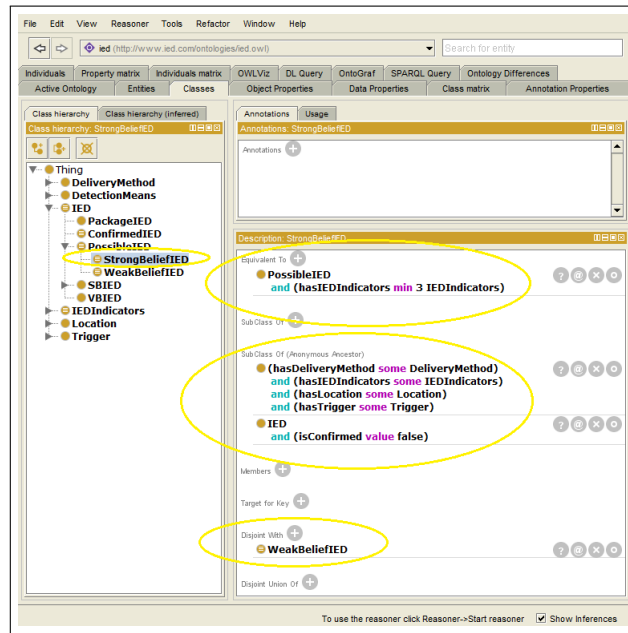


Figure 3.23: **StrongBeliefIED** shown.

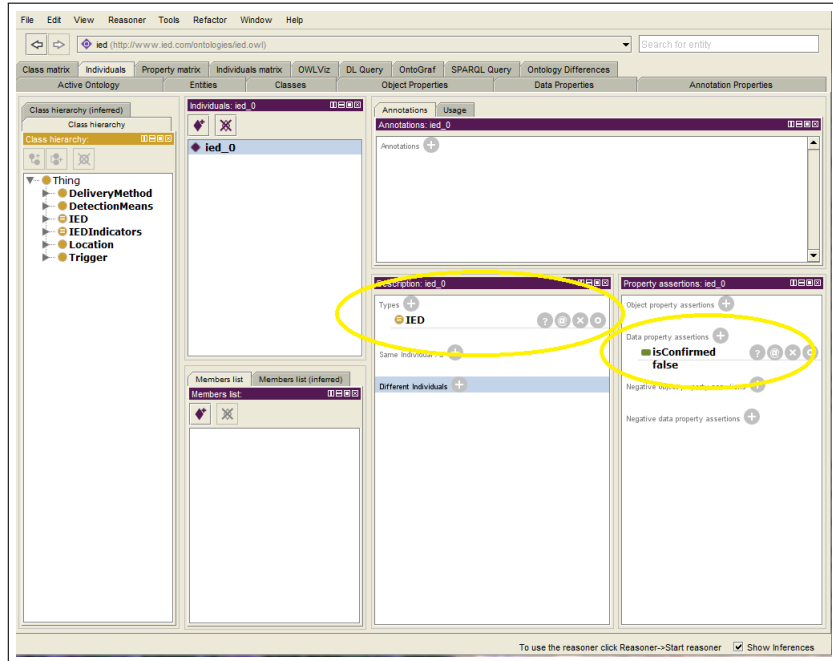


Figure 3.24: **ied\_0** individual shown.

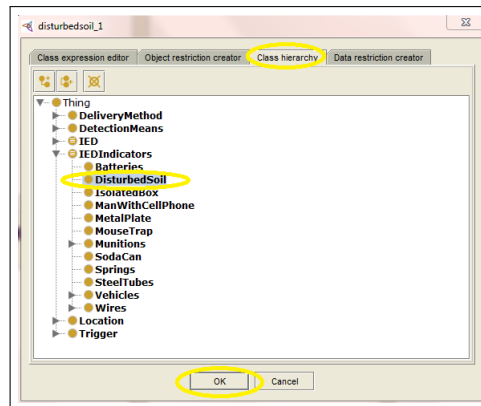


Figure 3.25: **disturbedsoil\_1** individual being assigned type **DisturbedSoil**.

### 3.5.6 Defining Individuals in IED Ontology

The study will now look at creating individuals in the ontology that will allow the reasoner to show if the created individual can be classified as a “PossibleIED,” “StrongBeliefIED,” “Weak-BeliefIED,” or “ConfirmedIED.” Individuals can be created in the “Individuals” tab. All individuals in Protégé are identified with a purple diamond and are written in lowercase. In order for the reasoner to classify (infer) an individual as being a member of a certain class, the class

of which it is a member needs to be a “defined” class, otherwise the reasoner will not properly classify (infer) its relationship to a particular class. First, the individual **ied\_0** is created, see Figure 3.24. It is given the type **IED** and Data property assertion **isConfirmed** with value “false.” This represents an object that has been detected but no indicators have been identified. The second IED individual will have with it two IED Indicators: “disturbed soil” and “wires.” These indicators will need to be created individually so we can assign them as object properties to individual **ied\_1**. First IED Indicator is named **disturbedsoil\_1**, assigned class type **DisturbedSoil** and given **hasIEDIndicatorName** “dirt” string. To assign the type **IED** click on “Types” “+” sign and select “Class Hierarchy” and select “ok.” (see Figure 3.25). Next, assign data property assertion type string “dirt” to **disturbedsoil\_1** indicator. See Figure 3.26 for assigning “dirt” to indicator **disturbedsoil\_1**. One could change “dirt” to “dog” and the reasoner would not care, assigning “dirt” makes it easier for the user to follow along. The second IED Indicator is named **wires\_1** it is assigned the **IEDIndicator**

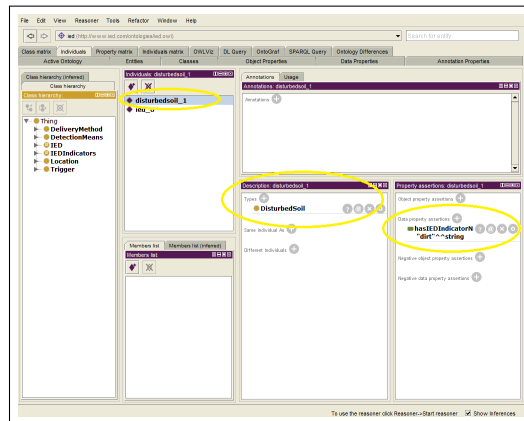


Figure 3.26: **disturbedsoil\_1** indicator being assigned string “dirt” shown.

subclass **Wire** and given the data property assertion string “wire” (see Figure 3.27). The second IED individual, **ied\_1** is like **ied\_0** except the individuals **disturbedsoil\_1** and **wires\_1** are assigned as “object property assertions,” and since this is not a “confirmedIED” it is assigned data property **isConfirmed** with value “false.” (see Figure 3.28). To assign the IED Indicators select “Object property assertions” underneath “property assertions window” and click on “+” sign. Next assign the two indicators to **ied\_1** (see Figure 3.29). The third and final indicator is defined with object property **hasIEDIndicator** with type string and individual assigned the name as **isolatedbox\_1** to represent a box alongside the road. There is no change in how you assign it a type and data property assertion. See Figure 3.30 for **isolatedbox\_1** individual screenshot. The third IED individual is **ied\_2** it has 3 IED

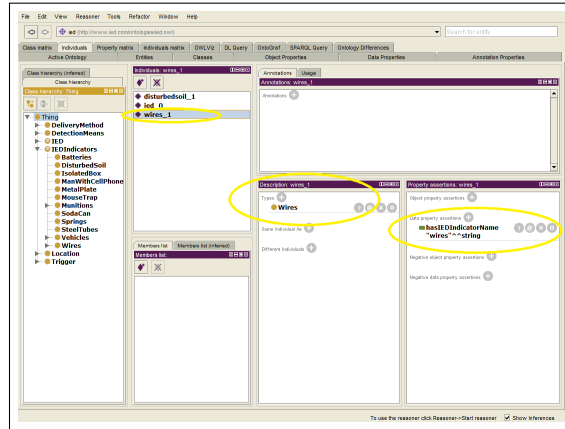


Figure 3.27: **wires\_1** individual shown.

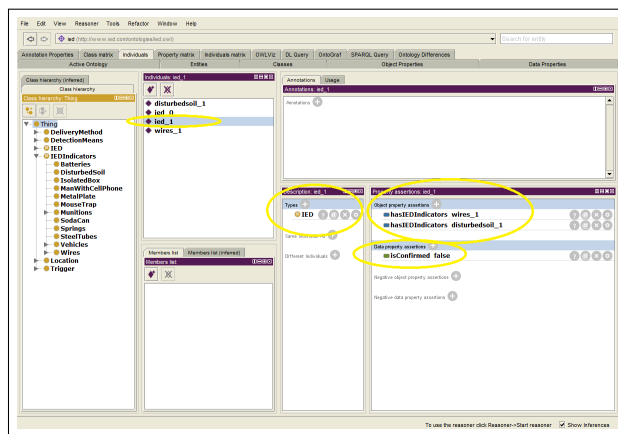


Figure 3.28: **ied\_1** individual shown.

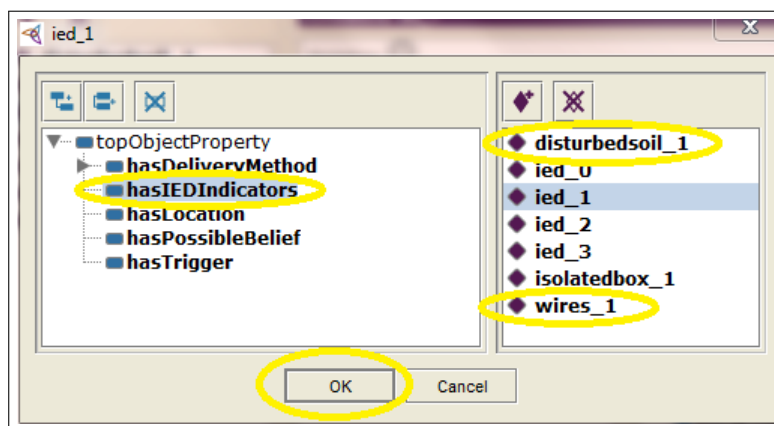


Figure 3.29: Assigning IED Indicators to **ied\_1**.

Indicators and “isConfirmed” value set to “false” (see Figure 3.31). This is an individual that an operator has been able to associate three characteristics of this “Possible IED.”

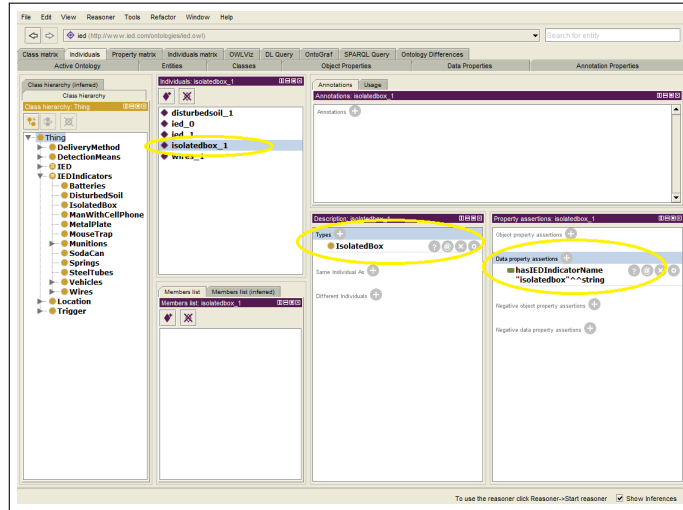


Figure 3.30: **isolatedbox\_1** individual shown.

Since it is not confirmed yet, the data property **isConfirmed** is set to “false.” The fourth and

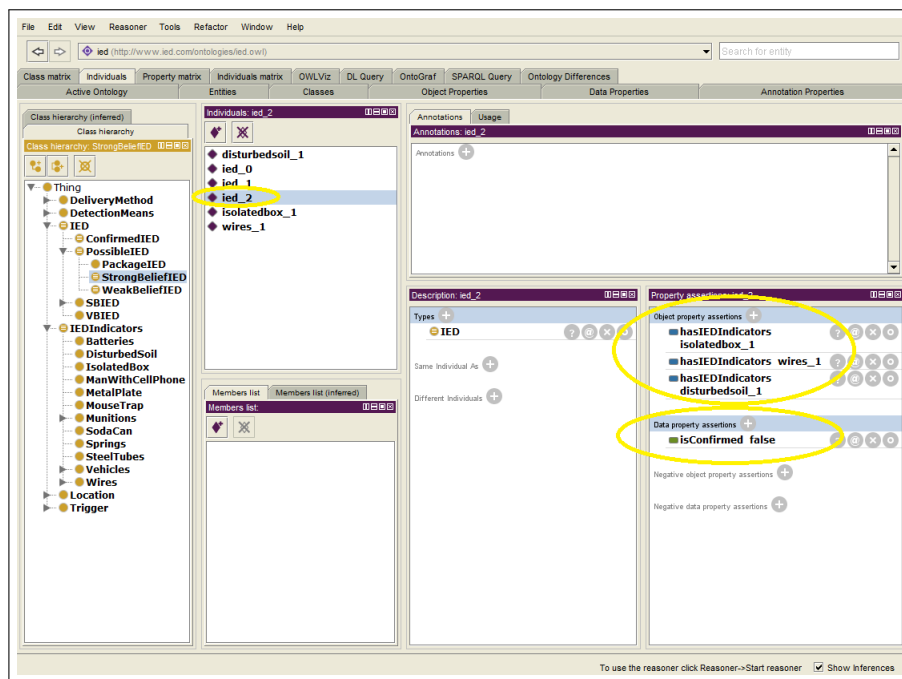


Figure 3.31: **ied\_2** individual shown.

final IED individual is **ied\_3**, which is just like **ied\_2** except this IED has been confirmed by EOD to be a legitimate IED, therefore the **isConfirmed** data property assumption is set to "true" (see Figure 3.32).



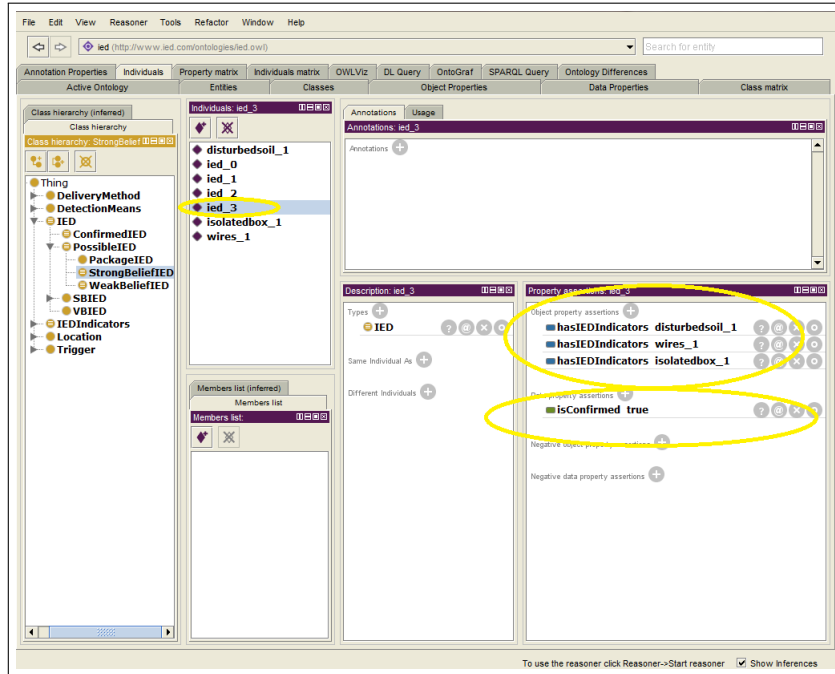


Figure 3.32: *ied\_3* individual shown.

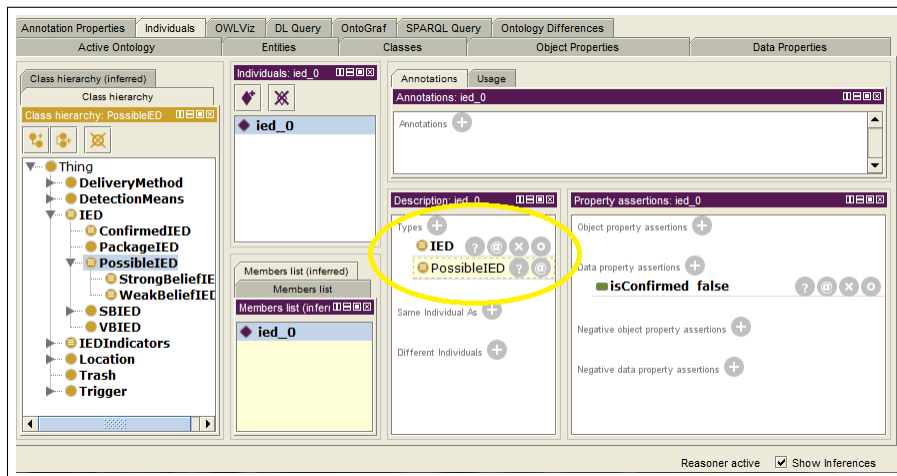


Figure 3.33: Reasoner classifies object “Possible IED.” based on its object and data properties.

### 3.5.7 Classifying an IED Using the Reasoner

The experiment will now model the behavior of a UAS’s ability to make a decision based on the information that it receives from its sensors. In this experiment, the sensor (UAV) is collecting data that falls into the categories of “IED Indicators.” Pieces of information that an operator may or may not be able to see using the UAS is represented as individuals. The reasoner is used to establish the level of confidence in what is seen based on the amount of information

the operator is able to see using his/her UAV. The UAV is flying looking for threats along the way. The operator detects an object (called **ied\_0**), however, he/she cannot identify what it is. Because **ied\_0** does not have any indicators the reasoner should and does classify this object as a "PossibleIED" (see Figure 3.33). As the operator continues to look at the object

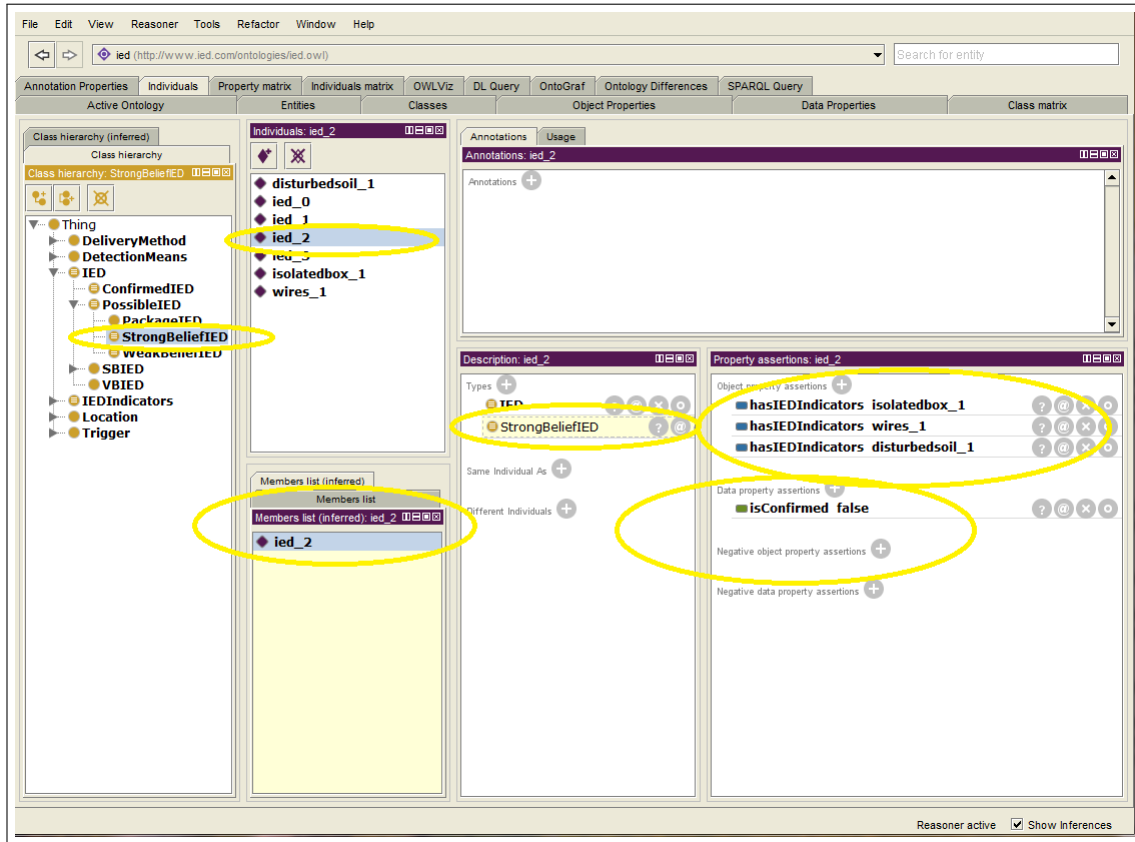


Figure 3.34: Reasoner classifies object as “Strong Belief IED.” based on properties associated with **ied\_2**.

they see IED indicators such as disturbed soil, wires, and now a box (making the user feel more confident about what is seen) causing the reasoner to classify this object (now called **ied\_2**) as a “Strong Belief IED” (see Figure 3.34) since there are three indicators. The convoy commander cordons off the area and calls for the EOD to investigate this strong belief, possible IED. Once EOD arrives, having looked at the same characteristics that the operator has, confirms that object (now called **ied\_3**) is a “real IED” and classifies it as a “Confirmed IED” (see Figure 3.35). This device is neutralized by EOD and a highly possible catastrophic kill is prevented.

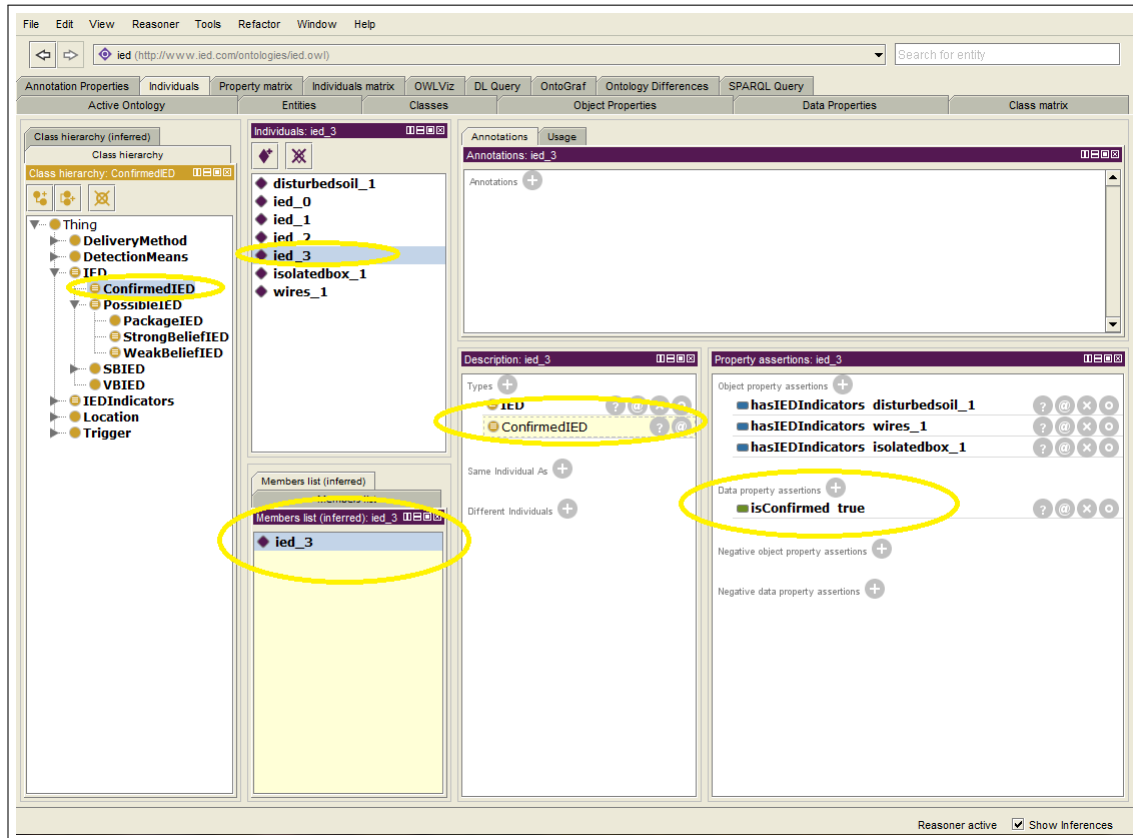


Figure 3.35: The “possible, strong belief IED” has been confirmed an IED by explosive ordnance disposal (EOD).

### 3.6 DESIGN OF EXPERIMENTS

The goal of Design of Experiments is to see if the effects of training on mission success can be shown in COMBATXXI utilizing the HTNs and UAV Sensor model. The mission objective is to arrive at Outpost X-Ray safely (as mentioned in section 3.2). The only true IED (located in NAI #2) detonates with probability 0.80 and will always kill the second vehicle in the convoy when detonated, the other three NAIs each have a single non-IED object. The mission will be over if: (1) Convoy reaches final destination without hitting IED (Mission Complete or MC) or (2) Convoy hits IED (Mission Failure). Success (MC) can only happen if IED is detected and defeated (UAS used effectively) or IED does not go off (probability 0.20) given IED is undetected (worse case). The UAV sensor [40] has a likelihood of detecting the object with probability 0.50 if the range from the sensor to the IED ( $r$ ) is 110 meters and probability 0 if  $r$  is 220 meters or greater. Time to clear an IED is 60 minutes.

The factors affecting mission success are False Positive ( $\alpha$ ) and False Negative ( $\beta$ ), with levels ranging from 0.0 to 1.0. Experiments run at six values for each (0.0, 0.2, 0.4, 0.6, 0.8, 1.0), for a total of 36 design points. Each design point is run 100 times for a total of 3600 observations (n). Below are the definitions for False Positive and False Negative:

- If False Positive ( $\alpha$ ) 0.0 then UAS ALWAYS identifies a non-IED object as a non-IED.
- If False Positive ( $\alpha$ ) 1.0 then UAS ALWAYS identifies non-IED as an IED.
- If False Negative ( $\beta$ ) 0.0 then UAS ALWAYS correctly identifies IED as an IED.
- If False Negative ( $\beta$ ) 1.0 then UAS ALWAYS incorrectly identifies IED as a non-IED.

Best case is when alpha and beta are 0.0 and 0.0. This represents an operator that both correctly classifies an object as being an IED or something that is truly not an IED (trash), reducing risk (less time exposed), and saving resources (UXO clearing team not called).

Worst case is when alpha and beta are 1.0 and 1.0. This represents an operator who always incorrectly detects an object as an IED and unnecessarily wastes resources to “clear” a non-IED, adding more time to convoy being exposed outside the wire, while at the same time always misidentifying real IEDs, leading to a high failure rate.

Metrics to measure the effects of alpha and beta are:

- Mean Mission Success Rate
- Mean Time to Complete Mission | Mission Complete
- Risk =  $1/\text{MC}\%$  x Mean Time to Complete Mission | Mission Complete

### 3.6.1 Summary

The concept of using an ontology and implementing it in order to demonstrate a different approach in measuring the sensing ability of a UAS is discussed. The DOE will determine if it is possible to model the positive and negative effects of being trained and untrained (changing values of alpha and beta) given the UAV sensor model detects the object. In the next chapter, the research will perform an analysis on data to be collected.

---

# CHAPTER 4: ANALYSIS

---

## 4.1 INTRODUCTION

We demonstrate the utility of our approach through analysis of the proof of concept scenarios. The findings show how the scenario results are sensitive to changing parameters False Positive ( $\alpha$ ) and False Negative ( $\beta$ ) in our sensor model. This proof of concept scenario would be useful to a decision maker interested in evaluating the potential impact of training on the effective use of UAVs.

## 4.2 DESIGN OF EXPERIMENTS ANALYSIS IN COMBATXXI

The analysis will investigate the effects of  $\alpha$  and  $\beta$  at six levels each (0.0, 0.2, 0.4, 0.6, 0.8, 1.0), mentioned earlier in chapter 3 “Methodology” The goal is to identify what combinations of alpha and beta give us the highest Mean Mission Success Rate, lowest Mean Time to Complete Mission | Mission Success, and to help quantify Risk.

### 4.2.1 Mean Mission Success Rate

This section is about measuring the mean success rate for the 3600 observations. A heat map is utilized to communicate results. Boxplots help to show noise and meaning to support the findings on  $\alpha$ 's and  $\beta$ 's effects on Mission Success Rate. Looking at Figure 4.1 we can see that the highest completion rate is 0.84 and occurs when  $\beta$  is 0.0, and is independent of  $\alpha$

		False Positive ( $\alpha$ ) Parameter					
		0.0	0.2	0.4	0.6	0.8	1.0
False Negative ( $\beta$ ) Parameter	0.0	0.84	0.84	0.84	0.84	0.84	0.84
	0.2	0.74	0.74	0.74	0.74	0.74	0.74
	0.4	0.57	0.57	0.57	0.57	0.57	0.57
	0.6	0.46	0.46	0.46	0.46	0.46	0.46
	0.8	0.29	0.30	0.29	0.29	0.29	0.29
	1.0	0.20	0.19	0.19	0.19	0.19	0.19

Mean Mission Success Rate per 100 trials

Figure 4.1: Heat Map of Mission Success Rate versus parameters False Positive ( $\alpha$ ) and False Negative ( $\beta$ ). Notice that the Mission Success Rate does not change as alpha increases from 0.0 to 1.0, which demonstrates that the False Positive parameter  $\alpha$  does not have an effect on mission success. Mission failures occurred even with  $\beta$  at zero because on some trials the UAV never sees the IED, and as a result the IED was not identified.

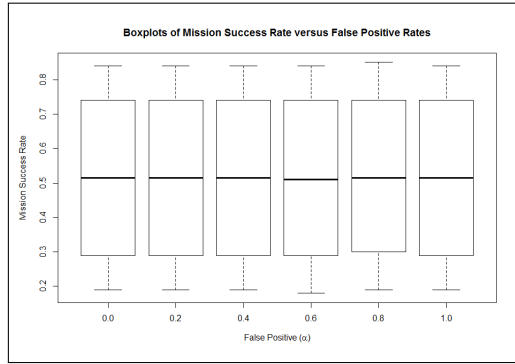


Figure 4.2: Boxplot of Mission Success Rate versus False Positive Rates. Notice that the Mission Success Rate does not change as  $\alpha$  increases from 0.0 to 1.0. This shows that  $\alpha$  does not have an effect on mission success since False Positive represents the identification of non-IED objects.

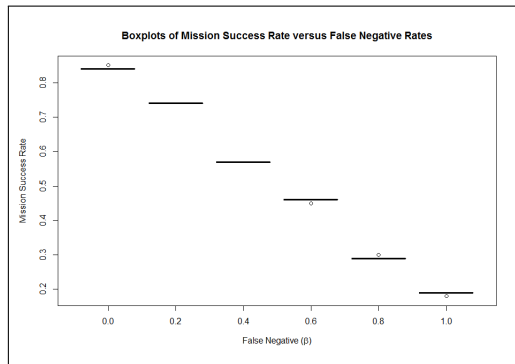


Figure 4.3: Boxplot of Mission Success Rate versus False Negative Rate ( $\beta$ ). Notice that the Mission Success Rate decreases as  $\beta$  increases from 0.0 to 1.0. There is no variance in the success rate either, which is why the boxes appear as lines instead of boxes per earlier boxplots when looking at Mission Success versus False Positive Rate. This shows that  $\beta$  has a great effect on mission success rate versus  $\alpha$ .

levels. The lowest average of completing mission is approximately 0.20 and 0.19 when  $\beta$  is 1.0, independent of  $\alpha$  levels. The decrease in mean mission success as  $\beta$  increases is caused by a higher probability of incorrectly identifying the IED in NAI #2 as a non-IED. Since the IED has a probability of 0.8 of detonating, that is why the success rate is 0.20 when  $\beta$  equals 1.0. As the False Positive value increases from left to right the success rate generally remains constant. It is only as we increase  $\beta$  that there is a difference in mission success rate, concluding that  $\beta$  has a strong influence on the likelihood of completing the mission. Looking at the first boxplot (see Figure 4.2), one can see that the False Positive ( $\alpha$ ) does not have an effect on the mission success rate since the boxplots share the same dimensions, respectively. The Mission Success

		False Positive ( $\alpha$ ) Parameter					
		0.0	0.2	0.4	0.6	0.8	1.0
False Negative ( $\beta$ ) Parameter	0.0	125.00	145.00	166.43	196.43	216.43	241.43
	0.2	123.19	141.02	163.73	192.92	214.81	239.13
	0.4	119.06	138.01	158.01	185.38	205.38	227.48
	0.6	116.00	134.26	156.43	183.82	204.69	225.56
	0.8	101.61	124.43	136.78	165.74	184.36	209.19
	1.0	69.44	91.70	113.80	139.06	161.17	186.43
			Mean Time to Complete Mission   Mission Success				

Figure 4.4: Heat Map of Mean Time to Complete Mission | Mission Success for different values of False Positive ( $\alpha$ ) and False Negative ( $\beta$ ). Mean time to complete the mission takes the longest when  $\alpha$  is 1.0 and  $\beta$  equals 0.0. Mean time for a successful mission increases as  $\alpha$  increases and decreases as  $\beta$  increases.

Rate is very close or right above 50% as  $\alpha$  increases from 0.0 to 1.0. Continuing on with the second boxplot, the False Negative Rate ( $\beta$ ), mission success decreases as  $\beta$  increases from 0.0 to 1.0 (see Figure 4.3).

This visual helps to show that in the model that False Negative Rate ( $\beta$ ) has a strong effect on mission success.

#### 4.2.2 Mean Time To Complete | Mission Success

Now the study will look at the mean time to complete the mission given mission success occurs. Looking at the heat map in Figure 4.4 it appears that the mean time to complete mission given mission success is influenced more on  $\alpha$  than  $\beta$ . Mean time to complete mission is highest at 241.43 minutes when  $\alpha = 1.0$  and  $\beta = 0.0$ . This makes sense; the operator will always identify a non-IED as an IED when  $\alpha = 1.0$ . The lowest mean time to complete mission of 69.44 minutes is at the opposite corner when  $\alpha = 0.0$  and  $\beta = 1.0$ . This number can be misleading because in this case, the UAV operator will always identify a piece of trash as trash ( $\alpha = 0.0$ ), but when  $\beta$  is equal to 1.0, the operator will always incorrectly identify an IED as a non-IED. Therefore in this case the convoy will never stop to handle a potential threat. The only reason there is any success in this case is because there is a 0.20 chance the IED will not detonate and it is in these “lucky” cases that the mission is completed. Now examining the Mission Complete Time boxplots (see Figure 4.5) it appears that the average time to complete the mission does increase as False Positive Rate ( $\alpha$ ) increases from 0.0 to 1.0.; this makes sense in our model given that we stated earlier that if  $\alpha$  is 0.0 it would always identify a non-IED object as a non-IED. Also when  $\alpha$  is 1.0 the UAS operator always identifies non-IED as an IED, leading to additional time spent on a mission and exposing oneself to more harm. Examining Figure 4.6 it appears that mission completion time does not vary as much and remains constant (at  $\beta$  value increases), proving that in the model, the False Negative Rate ( $\beta$ ) does not have as much influence for mean

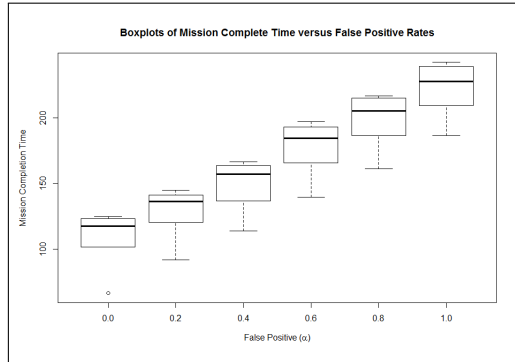


Figure 4.5: Boxplot of Mission Completion Time versus False Positive Rates. Notice that median time to complete mission increases as the False Positive Rate ( $\alpha$ ) increases. There appears to be a linear relationship; the variance for each value of  $\alpha$  appears to be equal to one another as well. This means that  $\alpha$  will always have an effect on amount of time it completes the mission.

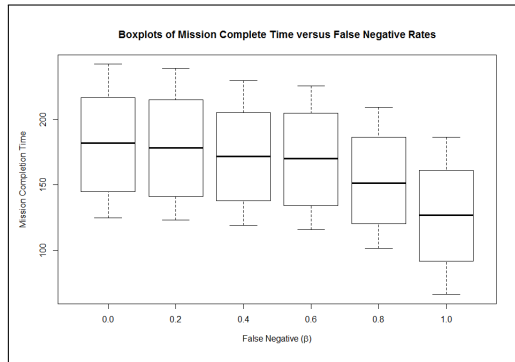


Figure 4.6: Boxplot of Mission Complete Time versus False Negative Rate ( $\beta$ ). The False Negative parameter  $\beta$  does not have as much of an effect on average time it takes the convoy to complete the mission. Notice that the Mission Completion Time is not heavily influenced on the change in  $\beta$  from 0.0 to 1.0. A time of 150 minutes could be found for each  $\beta$  value. Only when the value of 0.8 is reached is there a drop in median time of completing the mission. This is because the observer has a higher probability of incorrectly identifying the IED as a non-IED, resulting in fewer times to successfully complete the mission and have an observed time.

mission completion time as the False Positive Rate ( $\alpha$ ).

### 4.2.3 Measuring Risk

Risk is something all commanders must take when utilizing the U.S. Army’s most valuable asset: soldiers. As a means to assess risk, we can combine the considerations of mission success rate and completion time. We task risk to be:



		False Positive ( $\alpha$ ) Parameter					
		0.0	0.2	0.4	0.6	0.8	1.0
False Negative ( $\beta$ ) Parameter	0.0	148.81	172.62	198.13	233.84	257.65	287.41
	0.2	166.47	190.57	221.25	260.70	290.28	323.15
	0.4	208.88	242.12	277.21	325.22	360.31	399.09
	0.6	252.17	291.86	340.07	399.61	444.98	490.35
	0.8	350.37	414.77	471.64	571.52	635.73	721.34
	1.0	347.20	482.63	598.96	731.91	848.26	981.21
	Risk = 1 / MC% * Mean Time To Complete Mission   Mission Success						

Figure 4.7: Heat Map of Risk for different values of False Positive ( $\alpha$ ) and False Negative ( $\beta$ ). Lowest risk takes place at  $\alpha$  and  $\beta$  being 0.0, 0.0. Given the IED is detected, the user has capability to correctly identify the IED and does not waste time clearing non-IEDs. When  $\alpha$ ,  $\beta$  is 1.0, 1.0, risk is highest (less likely to correctly identify IED and more likely to incorrectly identify non IEDs as being IEDs).

$$Risk = 1 / MC\% \times (MeanTimeToCompleteMission | MissionComplete) \quad (4.1)$$

Figure 4.7 shows risk as a function of  $\alpha$  and  $\beta$ . It is the most interesting of the three heat maps. This measure of risk attempts to combine the risk of not identifying the IED with the inherent risk associated with being out in the open for an extended amount of time. Figure 4.7 confirms that this characterization of risk captures both aspects of the risk encountered by convoys. We expected the most dangerous situation to be when  $\alpha$  is 1.0 and  $\beta$  is 1.0, when the operator always misidentifies objects as being real IEDs and never identifies the real IEDs as a real IED, this is what we can see in the heat map. At the opposite corner when  $\alpha$  and  $\beta$  are 0, respectively, risk measures at 148.81. This is the best case scenario because an IED is always classified as an IED and non-IEDs (trash) are always identified as trash, minimizing the amount of time a convoy must spend on the route outside of “friendly” territory while still reducing the vulnerability to damage from IEDs.

### 4.3 SUMMARY

The results above show that it is possible that a UAS (like the Raven) may improve deployable force protection capabilities if the assumptions hold true in the real world. It is possible to measure success and time to complete a mission with different combinations of  $\alpha$  and  $\beta$ . The heat maps provided valuable insight on the ramifications of each  $\alpha$  and  $\beta$  level. The boxplots gave reassurance in the validity of the heat maps by showing differences the parameters do or do not have. Risk gave further understanding into the importance of correct versus incorrect identification of objects, and the effects of wasting time clearing non-IED threats. Next, the study will give a conclusion of using the new STA model and the idea of combining KR through

an ontology made using special software.

---

# CHAPTER 5: CONCLUSION

---

## **5.1 OUTCOMES**

We demonstrated the application of the new STA model in a tactical convoy scenario in COMBATXXI. We showed the ability to model a UAS in a working environment is fundamental to a study concerning tactical convoy operations. A design of experiments and post analysis quantifies the sensitivity of the measures of effectiveness of success to conditions contributing to variability in UAS performance.

### **5.1.1 What Does This Mean?**

Behaviors for UASs can be designed and developed using Behavior Studio Interactive Design Environment. HTNs were created to replicate the thought and action process of a convoy as it travels down a path wary of threats such as IEDs, with respect to the use of UAVs. The simulation allows for some analysis to determine if tactical convoy operations using a deployable force protection asset (Raven UAS) could be studied in COMBATXXI using the UAV sensor model, and associated dynamic behaviors without the reliance of nonadaptive scripting and compound orders currently in use. Factors such as False Positive and False Negative allows us to see how an operator's skill level can affect the outcome of a tactical convoy operations in SWA. It was found that a convoy experienced less risk and obtained a higher frequency of completing the mission when they could increase the likelihood of identifying an IED object as an IED and decreasing the likelihood of misidentifying non-IEDs.

### **5.1.2 Are There Any Advantages to UAS?**

Given the constraints, conditions, and limitations of the sensor model and COMBATXXI, it was shown that utilizing the UAS gave a higher likelihood of success. We also showed that qualitative factors such as risk could be controlled with the use of a UAS.

### **5.1.3 Have We Modeled UAS properly?**

Due to the dynamics of electro-optics (EO) sensors, the human eye and thought process, it is naive to say the Raven UAS was modeled properly, in this study. This study demonstrates that it is possible to model the complexities of a UAS in a complex simulation such as COMBATXXI.

### **5.1.4 Ontology Pros and Cons for Knowledge Representation with COMBATXXI**

The ontology showed that it is possible to express knowledge semantically and have a reasoner classify a threat based on the clues (IED indicators) that a human would detect using a UAV such as the Raven. It is important to note that the cues used in the ontology can be represented in a model like COMBATXXI.

### **5.1.5 Future Work**

The work presented in this thesis is an initial look at possible uses of combat models such as COMBATXXI. This work can be continued in many ways. Below are some possible areas that merit further investigation.

Develop extensions and support structures to allow COMBATXXI to be more amenable for use in studies that use the DOE methodology. This would allow it to be used in studies to look at complex parameter spaces that are not possible with the current structures.

Investigate the integration of ontology based KR systems into combat models such as COMBATXXI. By adding such structures, a wide range of investigation is made possible. These can range from identification of threats that take into consideration both observations and other information, such as past activity, to being able to reason about motives of opposing entities. The use of ontologies could also support more robust behaviors that can be used in a wider set of circumstances, which would facilitate building general purpose behaviors that can be reused in many scenarios.

# APPENDIX A: JOHNSON CRITERIA

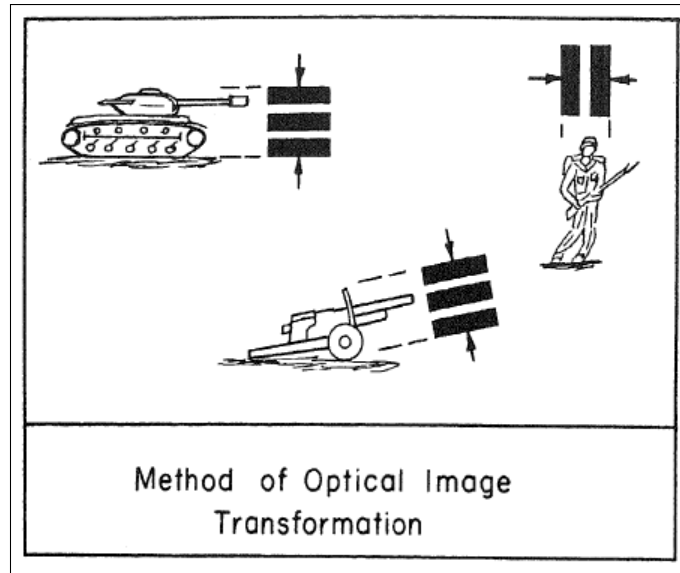


Figure A.1: Normalizing resolved line pairs for critical target dimension. After [44, p. 264]

TARGET	RESOLUTION PER MINIMUM DIMENSION				
	Broadside View	Detection	Orientation	Recognition	Identification
Truck		.90	1.25	4.5	8.0
M-48 Tank		.75	1.2	3.5	7.0
Stalin Tank		.75	1.2	3.3	6.0
Centurion Tank		.75	1.2	3.5	6.0
Half-track		1.0	1.50	4.0	5.0
Jeep		1.2	1.50	4.5	5.5
Command Car		1.2	1.5	4.3	5.5
Soldier (Standing)		1.5	1.8	3.8	8.0
105 Howitzer		1.0	1.5	4.8	6.0
Average		1.0 ± .25	1.4 ± .35	4.0 ± .8	6.4 ± 1.5

Figure A.2: Number of minimum line pairs required for the four detection, orientation, recognition, identification. From [44, p. 264]

THIS PAGE INTENTIONALLY LEFT BLANK

---

## APPENDIX B:

# CONCEPT OF OPERATIONS

---

### **Mission**

In support of 2nd Platoon (PLT), Alpha Company (Co.), 2-37 IN (Infantry), 4th squad provides security and logistic runs in partnership with the 3rd LNA Battalion to support operations for the local populace while protecting the Dam.

### **Situation**

2nd Platoon, A CO, 2-37 IN, conducted a relief in place (RIP) of the combat outpost (COP) three months ago. 2nd Platoon is scheduled to RIP with 1st Platoon within the next week and will assume FOB security at the CO HQ and serve as the CO reserve. Besides working and training the LNA troops, the platoon is also responsible for protecting the Dam workers that live on the COP. The missions are broken down into equal parts and each squad will rotate responsibilities to reduce complacency. The four tasks are (1) Entry Control Point [3 personnel x 3 eight hour shifts = 1 squad], (2) Force Protection [3 Towers /OP points x 3 eight hour shifts = 1 squad], (3) Dam Observation Post [3 personnel x 3 eight hour shifts = 1 squad], and (4) local security patrols and quick reaction force = 1 squad. For each squad there is one LNA platoon supporting to be the immediate face to the community. 4th squad is equipped with 1 x Raven 11Q B UAS. Four of the 11 soldiers in the squad are qualified Raven operators.

### **Commander's Intent**

2nd Platoon partnered with 3rd Battalion LNA will protect the population from insurgents, mentor the LNA partners, and protect the LNA partners in contact with insurgents. This partnership will maintain force protection at all times and provide over watch to the Dam to safeguard the area's power supply. 4th Squad will perform logistic runs in support of (ISO) 2nd Platoon, A CO, 2-37 IN.

### **Enemy**

Threat forces consist of 300 insurgents equipped with RPG, PKM, AK-47, HMG, AT-3, SPG-9, SA-16 and 82mm mortar. The main threat consists of 3 x PLTs with two supporting efforts. Each PLT has the ability to emplace 2 x IEDs. There are reports of 8 to 10-man insurgent teams

equipped with RPG, AK-47. There are also reports of possible IED emplacements along MSR WEST VIRGINIA.

### **B.0.6 Terrain**

Open desert to desert mountains with adjacent urban structures.

### **B.0.7 Troops Available**

Coalition and Host Nation Forces: 1 x U.S. Infantry squad equipped with organic direct fire weapons, with support from 120mm mortars. 21 soldiers from the LNA company and 16 from the LNP.

### **B.0.8 Time**

0800, day, clear.

### **B.0.9 Civilian**

Civilian populace is either controlled or intimidated by enemy operating within the area of operations (AO).



---

# APPENDIX C:

## CONVOY MOVE IN FORMATION

---

```
1=====
2Name: ConvoyMoveInFormation
3AllowMsg: true
4Node Type: DEFAULT
5Is Code File: false
6Import:
7Datamap:
8  formation=>java.lang.String
9  route=>cxxi.model.objects.features.CMPolyline
10 NAI_phaseLines=>java.util.ArrayList
11 UAS_routes=>java.util.ArrayList
12 falseNegativeRate=>[Ljava.lang.String;@7b449b1d
13 falsePositiveRate=>[Ljava.lang.String;@5523cc24
14Metadata:
15Code:
16=====
17 Name: initInfo
18 AllowMsg: true
19 Node Type: DEFAULT
20 Is Code File: false
21 Import:
22 Datamap:
23 Metadata:
24 Code:
25     if _gt_activeNode.getVar("isInited") == None:
26         _gt_activeNode.putVar("isInited", 1)
27         _htn_precon_ret=1
28=====
29 Name: isCommander
30 AllowMsg: true
31 Node Type: DEFAULT
32 Is Code File: false
33 Import:
34 Datamap:
35 Metadata:
36 Code:
```

```

37     if state.isCommander():
38         _htn_precon_ret=1
39     =====
40     Name: setup
41     AllowMsg: true
42     Node Type: DEFAULT
43     Is Code File: false
44     Import:
45     Datamap:
46     Metadata:
47     Code:
48         from cxxi.model import CombatXXIModel
49         from os import access
50         from os import F_OK
51         from os import W_OK
52         from sys import stderr
53         try:
54             logfile
55             writeOK
56         except NameError:
57             # open the log file in the same place CXXI puts the other log files
58             logfilePathName = str(CombatXXIModel.getOutputDirectory()) + "/DFP.
log"
59             writeOK = True
60             try:
61                 logfile = open(logfilePathName, "w")
62             except IOError:
63                 print >>stderr, "\n\nWarning_!!!!_Could_not_open_log_file_to_write_-
log_will_not_be_output.\n\n"
64                 print >>stderr, "Log_file_may_be_open_in_another_
application.\n\n"
65             writeOK = False
66             if writeOK:
67                 _gt_activeNode.putVar("logfile", logfile)
68             # get the rep number for this run
69             repNum = CombatXXIModel.getReplicationNumber()
70             _gt_activeNode.putVar("repNum", repNum)
71             # set up the holder in the borg to hold the non-ied names
72             name = "notIED"
73             notIED = []
74             borg.addAttribute(name, notIED, _gt_activeNode)

```

```

75  =====
76  Name: initCommander
77  AllowMsg: true
78  Node Type: DEFAULT
79  Is Code File: false
80  Import:
81  Datamap:
82  Metadata:
83  Code:
84      from HTN import UtilityFuncsExp
85      #state.addControlMeasure(nai) #adds control measure for CXXI to pay
      attention to.
86      for cm in _gt_activeNode.getParam("NAI_phaseLines"):
87          state.addControlMeasure(cm.getName())
88          speed = 2
89          # scehdule the event to get us going
90          UtilityFuncsExp.scheduleEvent(
91              info.getMyAssignedName(),
92              "GoalTracker_MoveOut",
93              0.001,
94              speed)
95  =====
96  Name: addReplanTriggers
97  AllowMsg: true
98  Node Type: INTERRUPT
99  Is Code File: false
100  Import:
101  Datamap:
102  Metadata:
103  Code:
104      goalContainer.getCurrentExecutingStack().addReplanTrigger("
      AtAControlMeasure")
105      goalContainer.getCurrentExecutingStack().addReplanTrigger("
      GoalTracker_MoveOut")
106      goalContainer.getCurrentExecutingStack().addReplanTrigger("
      GoalTracker_UASPickedUp")
107      goalContainer.getCurrentExecutingStack().addReplanTrigger("
      EmbarkComplete")
108      goalContainer.getCurrentExecutingStack().addReplanTrigger("
      DamageOccurredInMyUnit")

```

```

109     goalContainer.getCurrentExecutingStack().addReplanTrigger("
GoalTracker_ResumeMovingAtSpeed")
110     goalContainer.getCurrentExecutingStack().addReplanTrigger("
StoppedMovement")
111     =====
112     Name: doNothing
113     AllowMsg: true
114     Node Type: GOAL
115     Is Code File: false
116     Import:
117     Datamap:
118     Metadata:
119     Code:
120     '''
121     At this point only the commander does anything so if the tree is
assigned to someone who is
122     not the commander, just end the tree.
123     NOTE - if this tree is to be used if there is attrition this should
not be
124     done this way and there should be a way to keep the tree around in
case the current
125     "non-commander" is promoted to command via attrition.
126     '''
127     =====
128     Name: events
129     AllowMsg: true
130     Node Type: DEFAULT
131     Is Code File: false
132     Import:
133     Datamap:
134     Metadata:
135     Code:
136     =====
137     Name: isGoalTrackerEvent
138     AllowMsg: true
139     Node Type: DEFAULT
140     Is Code File: false
141     Import:
142     Datamap:
143     Metadata:
144     Code:

```

```

145     if state.getLastTrigger().startswith("doGoalTracker_"):
146         _htn_precon_ret=1
147     =====
148     Name: isMoveOut
149     AllowMsg: true
150     Node Type: DEFAULT
151     Is Code File: false
152     Import:
153     Datamap:
154     Metadata:
155     Code:
156         if state.getLastTrigger() == "doGoalTracker_MoveOut":
157             _htn_precon_ret=1
158             printMessage("GOAL_TRACKER_EVENT"+state.getLastTrigger(), True)
159         =====
160         Name: moveOut
161         AllowMsg: true
162         Node Type: INTERRUPT
163         Is Code File: false
164         Import:
165         Datamap:
166         Metadata:
167         Code:
168             from cxxi.model.behavior import OrderUtilities
169             from java.util import Vector
170             from mtry.cxxi.model.HierarchicalTaskNetwork import HTNUtilities
171             formation = _gt_activeNode.getParam("formation")
172             route = _gt_activeNode.getParam("route")
173             printMessage("formation_is_" + formation, True)
174             params = state.getLastTriggerParams()
175             speed = float(params[0])
176             ord = HTNUtilities._py_getMoveOrderFromRoute(route, speed, formation
177         )
178             orders.addOrder(ord)
179             startTime = state.getSimTime()
180             _gt_activeNode.putVar("startTime", startTime)
181         =====
182         Name: isGoalTracker_UASPickedUp
183         AllowMsg: true
184         Node Type: DEFAULT
185         Is Code File: false

```

```

185 Import:
186 Datamap:
187 Metadata:
188 Code:
189     if state.getLastTrigger() == "doGoalTracker_UASPickedUp":
190         _htn_precon_ret=1
191         =====
192         Name: pickedUpUAS
193         AllowMsg: true
194         Node Type: INTERRUPT
195         Is Code File: false
196         Import:
197         Datamap:
198         Metadata:
199         Code:
200             from HTN import UtilityFuncsExp
201             # we have picked up the UAS, so we should continue
202             observations = state.getLastTriggerParams()[0]
203             if observations.isEmpty():
204                 UtilityFuncsExp.scheduleEvent(
205                     info.getMyAssignedName(),
206                     "GoalTracker_ResumeMovingAtSpeed",
207                     0.001,
208                     None)
209             else:
210                 # look at what the UAV sent us
211                 # this is where the processing of the knowledge could go.  If the
info the UAS
212                 # sent back not clear other info could be used
213                 # for now we kill the IED and delay while it is destroyed
214                 whoToKill = observations.get(0).getAssignedName()
215                 #whoToKill= str((UtilityFuncs.getEntityById(IDofWhoToKill)).
getAssignedName())
216                 state.killEntity(whoToKill, "CATASTROPHIC_KILL")
217                 UtilityFuncsExp.scheduleEvent(
218                     info.getMyAssignedName(),
219                     "GoalTracker_ResumeMovingAtSpeed",
220                     600.0,
221                     None)
222         =====
223         Name: isResumeMovingAtSpeed

```

```

224 AllowMsg: true
225 Node Type: DEFAULT
226 Is Code File: false
227 Import:
228 Datamap:
229 Metadata:
230 Code:
231     if state.getLastTrigger() == "doGoalTracker_ResumeMovingAtSpeed":
232         _htn_precon_ret=1
233     =====
234 Name: resumeMoving
235 AllowMsg: true
236 Node Type: INTERRUPT
237 Is Code File: false
238 Import:
239 Datamap:
240 Metadata:
241 Code:
242     speedToResume = _gt_activeNode.getVar("speedToResume")
243     UtilityFuncsExp.changeOrderSpeed(info.getMySelf(), speedToResume)
244     =====
245 Name: modelEvents
246 AllowMsg: true
247 Node Type: DEFAULT
248 Is Code File: false
249 Import:
250 Datamap:
251 Metadata:
252 Code:
253     =====
254 Name: isAtControlMeasure
255 AllowMsg: true
256 Node Type: DEFAULT
257 Is Code File: false
258 Import:
259 Datamap:
260 Metadata:
261 Code:
262     if state.getLastTrigger()== "doAtAControlMeasure":
263         _htn_precon_ret=1
264     =====

```

```

265 Name: isAtPhaseLine
266 AllowMsg: true
267 Node Type: INTERRUPT
268 Is Code File: false
269 Import:
270 Datamap:
271 Metadata:
272 Code:
273     from cxxi.model.objects.holders import CMHolder
274     from java.util import Vector
275     from cxxi.model.behavior import OrderUtilities
276     cm = state.getLastTriggerParams()[0]
277     NAI_phaseLines = _gt_activeNode.getParam("NAI_phaseLines")
278     for i in range(NAI_phaseLines.size()):
279         aCMName = NAI_phaseLines.get(i).getName()
280         if aCMName == cm.getName():
281             printMessage("AT" + str(cm.getName()), True)
282             uasRouteToUse = _gt_activeNode.getParam("UAS_routes")[i]
283             _gt_activeNode.putVar("uasRouteToUse", uasRouteToUse)
284             _htn_precon_ret=1
285     =====
286 Name: stopAndWait
287 AllowMsg: true
288 Node Type: DEFAULT
289 Is Code File: false
290 Import:
291 Datamap:
292 Metadata:
293 Code:
294     speedToResume = state.getCurrentSpeed()
295     _gt_activeNode.putVar("speedToResume", speedToResume)
296     UtilityFuncsExp.changeOrderSpeed(info.getMySelf(),0.0001)
297     =====
298 Name: dispatchUAS
299 AllowMsg: true
300 Node Type: INTERRUPT
301 Is Code File: false
302 Import:
303 Datamap:
304 Metadata:
305 Code:

```



```

306     from HTN import UtilityFuncsExp
307     printMessage(" Start_UAS_Tree", True)
308     #airRoute =
                                     CMHolder.
retrieveControlMeasureByName(
309     "AIR_CORRIDOR_UAS_NAI_ROUTE")
310     uasRouteToUse = _gt_activeNode.getVar("uasRouteToUse")
311     # set the path to the goal
312     goalPath = "HTN/Trees/OperateUAS.xml"
313     UtilityFuncsExp.addGoal(
314         "B0000c01A1",
315         1.0,
316         goalPath,
317         [uasRouteToUse,
318         _gt_activeNode.getParam("falsePositiveRate"),
319         _gt_activeNode.getParam("falseNegativeRate")],
320         None)
321     =====
322     Name: isDamageOccured
323     AllowMsg: true
324     Node Type: DEFAULT
325     Is Code File: false
326     Import:
327     Datamap:
328     Metadata:
329     Code:
330         if state.getLastTrigger() == "doDamageOccurredInMyUnit":
331             _htn_precon_ret=1
332     =====
333     Name: stopConvoy
334     AllowMsg: true
335     Node Type: GOAL
336     Is Code File: false
337     Import:
338     Datamap:
339     Metadata:
340     Code:
341         from cxxi.model.behavior import OrderUtilities
342         from java.util import Vector
343         #if damage has occurred in this unit we need to stop and for this
scenario we are done

```

```

344     if state.isCommander():
345         _htn_precon_ret=1
346         prims=Vector()
347         prims.add(OrderUtilities.createStopPrimitive())
348         ord=orders.createOrder("HTN_Plan_Auto_Generated", 0.0, prims)
349         orders.preemptAllOrders(ord)
350         simtime = state.getSimTime()
351         elapsedTime = (simtime - _gt_activeNode.getVar("startTime"))/60.0
352         printMessage("mission_ended_after_" + str(elapsedTime) + "_minutes"
, True)
353         # log this run
354         logfile = _gt_activeNode.getVar("logfile")
355         fnr = _gt_activeNode.getParam("falseNegativeRate")
356         fpr = _gt_activeNode.getParam("falsePositiveRate")
357         repNum = _gt_activeNode.getVar("repNum")
358         logfile.write('%d\t%.2f\t'%(repNum, simtime))
359         logfile.write('Fail\t')
360         logfile.write('%.2f\t%.2f\t%.2f\n'%(elapsedTime, fnr, fpr))
361         logfile.close()
362         =====
363         Name: isStoppedMoving
364         AllowMsg: true
365         Node Type: DEFAULT
366         Is Code File: false
367         Import:
368         Datamap:
369         Metadata:
370         Code:
371         if state.getLastTrigger() == "doStoppedMovement":
372             _htn_precon_ret=1
373         =====
374         Name: doneWithMission
375         AllowMsg: true
376         Node Type: GOAL
377         Is Code File: false
378         Import:
379         Datamap:
380         Metadata:
381         Code:
382         from cxxi.model import CombatXXIModel
383         simtime = state.getSimTime()

```

```
384     elapsedTime = (simtime - _gt_activeNode.getVar("startTime"))/60.0
385     printMessage("finished_mission_at_" + str(elapsedTime), True)
386     # log this run
387     logfile = _gt_activeNode.getVar("logfile")
388     fnr = _gt_activeNode.getParam("falseNegativeRate")
389     fpr = _gt_activeNode.getParam("falsePositiveRate")
390     repNum = _gt_activeNode.getVar("repNum")
391     logfile.write('%d\t%.2f\t'%(repNum, simtime))
392     logfile.write('Success\t')
393     logfile.write('%.2f\t%.2f\t%.2f\n'%(elapsedTime, fnr, fpr))
394     #CombatXXIModel.stopModel("run_complete", 0)
395     logfile.close()
```

THIS PAGE INTENTIONALLY LEFT BLANK

---

## APPENDIX D: OPERATE THE IED HTN

---

```
1=====
2Name: OperateIED
3AllowMsg: true
4Node Type: DEFAULT
5Is Code File: false
6Import:
7Datamap:
8  sensor=>[Ljava.lang.String;@66ba60c7
9  sensorRange=>[Ljava.lang.String;@5627dd81
10 targetCount=>[Ljava.lang.String;@533f6c57
11 P_K_Kill=>[Ljava.lang.String;@68e1ee73
12Metadata:
13Code:
14  '''_Notes
15  _This_is_a_tree_to_operate_a_simple_IED.
16  '''
17 =====
18 Name: initInfo
19 AllowMsg: true
20 Node Type: DEFAULT
21 Is Code File: false
22 Import:
23 Datamap:
24 Metadata:
25 Code:
26   if _gt_activeNode.getVar("isInited") == None:
27     _gt_activeNode.putVar("isInited", 1)
28     _htn_precon_ret=1
29 =====
30 Name: isRealIED
31 AllowMsg: true
32 Node Type: DEFAULT
33 Is Code File: false
34 Import:
35 Datamap:
36 Metadata:
```

```

37 Code:
38     if _gt_activeNode.getParam("P_K_KIII") > 0.0:
39         _htn_precon_ret=1
40     =====
41 Name: startObservation
42 AllowMsg: true
43 Node Type: DEFAULT
44 Is Code File: false
45 Import:
46 Datamap:
47 Metadata:
48 Code:
49     # set the the number of entities that have entered the AOI to zero
50     _gt_activeNode.putVar("entitiesEnteredSensorAOI", 0)
51     sensor = _gt_activeNode.getParam("sensor")
52     sensorRange = _gt_activeNode.getParam("sensorRange")
53     sensorObj = obs.getSensorByName(sensor)
54     sensorObj.setEffectiveRange(sensorRange)
55     printMessage("__Turning_on_area_sensor!", True)
56     if obs.startAOISensor(sensor):
57         sensorObj = obs.getSensorByName(sensor)
58         printMessage("__AOI_sensor_turned_on", True)
59     else:
60         print info.getMyAssignedName(),":__Cannot_start_AOI_Sensor__"
61     =====
62 Name: addReplanTriggers
63 AllowMsg: false
64 Node Type: INTERRUPT
65 Is Code File: false
66 Import:
67 Datamap:
68 Metadata:
69 Code:
70     # model events
71     goalContainer.getCurrentExecutingStack().addReplanTrigger("EnteredAOI
72 ")
73     goalContainer.getCurrentExecutingStack().addReplanTrigger("
74 GoalTracker_DetonateNow")
75     =====
76 Name: setIsNotIEDInfo
77 AllowMsg: true

```

```

76 Node Type: GOAL
77 Is Code File: false
78 Import:
79 Datamap:
80 Metadata:
81 Code:
82     '''
83     Since this is not an actual IED, only something that could be
      confused as an
84     an IED, it does not need any behavior so this tree can finish.
85     '''
86     borg.notIED.append(str(info.getMyAssignedName()))
87     printMessage("_is_not_and_IED", True)
88     =====
89 Name: events
90 AllowMsg: true
91 Node Type: DEFAULT
92 Is Code File: false
93 Import:
94 Datamap:
95 Metadata:
96 Code:
97     =====
98 Name: isGoalTrackerEvent
99 AllowMsg: true
100 Node Type: DEFAULT
101 Is Code File: false
102 Import:
103 Datamap:
104 Metadata:
105 Code:
106     if state.getLastTrigger().startswith("doGoalTracker_"):
107         _htn_precon_ret=1
108     =====
109 Name: isDetonateNow
110 AllowMsg: true
111 Node Type: INTERRUPT
112 Is Code File: false
113 Import:
114 Datamap:
115 Metadata:

```

```

116 Code:
117     printMessage("GOAL_TRACKER_EVENT"+state.getLastTrigger(), True)
118     if state.getLastTrigger() == "doGoalTracker_DetonateNow":
119         _htn_precon_ret=1
120     =====
121     Name: detonate
122     AllowMsg: true
123     Node Type: GOAL
124     Is Code File: false
125     Import:
126     Datamap:
127     Metadata:
128     Code:
129         from UtilityFuncs import getUniformRandomNumber
130         # check to see if we need to kill an entity
131         rn = getUniformRandomNumber(0,1.0)
132         if rn <= _gt_activeNode.getParam("P_K_KIII"):
133             # the most recent to enter the area is the entity that will be
134             killed.
135             IDofWhoToKill = state.getLastTriggerParams()[0]
136             whoToKill= str((UtilityFuncs.getEntityById(IDofWhoToKill)).
137             getAssignedName())
138             state.killEntity(whoToKill, "CATASTROPHIC_KILL")
139             # have the IED destroy itself
140             printMessage("IED_self_destruct_RND=_"+ str(rn), True)
141             state.killEntity(str(info.getMyAssignedName()), "CATASTROPHIC_KILL")
142     =====
143     Name: modelEvents
144     AllowMsg: true
145     Node Type: DEFAULT
146     Is Code File: false
147     Import:
148     Datamap:
149     Metadata:
150     Code:
151     =====
152     Name: isEnteredAOI
153     AllowMsg: true
154     Node Type: DEFAULT
155     Is Code File: false
156     Import:

```



```

155  Datamap:
156  Metadata:
157  Code:
158      if state.getLastTrigger() == "doEnteredAOI" :
159          _htn_precon_ret=1
160          =====
161          Name: entityEnteredAOI
162          AllowMsg: false
163          Node Type: INTERRUPT
164          Is Code File: false
165          Import:
166          Datamap:
167          Metadata:
168          Code:
169              from HIN import UtilityFuncsExp
170              import UtilityFuncs
171              entitiesEnteredSensorAOI = _gt_activeNode.getVar("
entitiesEnteredSensorAOI")
172              entitiesEnteredSensorAOI += 1
173              if entitiesEnteredSensorAOI >= _gt_activeNode.getParam("targetCount
"):
174                  printMessage("Hello ,_I_am_executing_the_isEnteredAOI_node_and_the_
sim_time_is:__" +str(state.getSimTime()), True)
175                  entityID = state.getLastTriggerParams()[1]
176                  UtilityFuncsExp.scheduleEvent(
177                      info.getMyAssignedName(),
178                      "GoalTracker_DetonateNow",
179                      0.001,
180                      entityID) # pass the id of the entity that my be killed
181              else:
182                  printMessage("Not_enough_entities_im_my_AOI_yet", True)
183                  _gt_activeNode.putVar("entitiesEnteredSensorAOI",
entitiesEnteredSensorAOI)

```

THIS PAGE INTENTIONALLY LEFT BLANK

---

## APPENDIX E: OPERATE UAS HTN

---

```
1=====
2Name: OperateUAS
3AllowMsg: true
4Node Type: DEFAULT
5Is Code File: false
6Import:
7Datamap:
8  route=>cxxi.model.objects.features.CMPolyline
9  falsePositiveRate=>java.lang.Double
10  falseNegativeRate=>java.lang.Double
11Metadata:
12Code:
13 =====
14 Name: initInfo
15 AllowMsg: true
16 Node Type: DEFAULT
17 Is Code File: false
18 Import:
19 Datamap:
20 Metadata:
21 Code:
22   if _gt_activeNode.getVar("isInited") == None:
23     _gt_activeNode.putVar("isInited", 1)
24     _htn_precon_ret=1
25 =====
26 Name: addReplanTriggers
27 AllowMsg: true
28 Node Type: DEFAULT
29 Is Code File: false
30 Import:
31 Datamap:
32 Metadata:
33 Code:
34   #goalContainer.getCurrentExecutingStack().addReplanTrigger("
    TakeoffDone")
```

```

35     goalContainer.getCurrentExecutingStack().addReplanTrigger("HaveLanded"
)
36     goalContainer.getCurrentExecutingStack().addReplanTrigger("
EmbarkComplete")
37     goalContainer.getCurrentExecutingStack().addReplanTrigger("
GoalTracker_CompletedFOVScan")
38     =====
39     Name: createUASRoutes
40     AllowMsg: true
41     Node Type: INTERRUPT
42     Is Code File: false
43     Import:
44     Datamap:
45     Metadata:
46     Code:
47     from HTN import UtilityFuncsExp
48     from cxxi.model.behavior import OrderUtilities
49     from java.util import Vector
50     from java.util import LinkedHashSet
51     from mtry.cxxi.model.HierarchicalTaskNetwork import HTNUtilities
52     from cxxi.model.physics.geometry import Direction
53     from HTN.Scripts.general import SensorHelpers
54     myTransporter = myTransport.getTransporter()
55     _gt_activeNode.putVar("myTransporter", myTransporter)
56     allObs = LinkedHashSet()
57     _gt_activeNode.putVar("allObs", allObs)
58     # get off the transporter so we can fly our mission
59     myTransport.debark()
60     route = _gt_activeNode.getParam("route")
61     startPoint = state.getCurrentLocation()
62     _gt_activeNode.putVar("startPoint", startPoint)
63     prims = Vector()
64     prims.addAll(HTNUtilities._py_createMovePOsFromRoutes(route, 10.0))
65     prims.add(OrderUtilities.createMovePrimitive(startPoint, 10.0))
66     prims.add(OrderUtilities.createStopPrimitive())
67     ord = orders.createOrder("HTN_Plan_Auto_Generated", 0.0, prims)
68     orders.addOrder(ord)
69     printMessage("Ready_to_turn_on_UAS_sensor", True)
70     UtilityFuncsExp.addGoal(
71         info.getMyAssignedName(),
72         2.0,

```

```

73     "HTN/Trees/UAVSensor.xml",
74     [0.0, -45.0, 20.0, 1000.0, 3.5, 1.0],
75     None)
76     outbound = True
77     _gt_activeNode.putVar("outbound", outbound)
78     lastDistToEndPoint = 0.0
79     _gt_activeNode.putVar("lastDistToEndPoint", lastDistToEndPoint)
80     =====
81     Name: events
82     AllowMsg: true
83     Node Type: DEFAULT
84     Is Code File: false
85     Import:
86     Datamap:
87     Metadata:
88     Code:
89     =====
90     Name: isGoalTrackerEvent
91     AllowMsg: true
92     Node Type: DEFAULT
93     Is Code File: false
94     Import:
95     Datamap:
96     Metadata:
97     Code:
98     if state.getLastTrigger().startswith("doGoalTracker_"):
99         _htn_precon_ret=1
100    =====
101    Name: isCompletedFOVScan
102    AllowMsg: true
103    Node Type: INTERRUPT
104    Is Code File: false
105    Import:
106    Datamap:
107    Metadata:
108    Code:
109    if state.getLastTrigger() == "doGoalTracker_CompletedFOVScan":
110        _htn_precon_ret=1
111    =====
112    Name: checkSensor
113    AllowMsg: true

```

```

114 Node Type: INTERRUPT
115 Is Code File: false
116 Import:
117 Datamap:
118 Metadata:
119 Code:
120 #
121 # check to see if we are close to the start point in which case we
do not continue scanning
122 currentLoc = state.getCurrentLocation()
123 dist = currentLoc.distanceTo(_gt_activeNode.getVar("startPoint"))
124 outbound = _gt_activeNode.getVar("outbound")
125 lastDistToEndPoint = _gt_activeNode.getVar("lastDistToEndPoint")
126 # test if we are still outbound
127 if outbound and dist < lastDistToEndPoint:
128 # we have started back to the landing point
129 outbound = False
130 _gt_activeNode.putVar("outbound", outbound)
131 else:
132 lastDistToEndPoint = dist
133 _gt_activeNode.putVar("lastDistToEndPoint", lastDistToEndPoint)
134 if dist > 100.0 or outbound:
135 UtilityFuncsExp.addGoal(
136 info.getMyAssignedName(),
137 0.0001,
138 "HTN/ Trees /UAVSensor.xml",
139 [0.0, -45.0,30.0,1000.0,3.5,1.0],
140 None)
141 observations = state.getLastTriggerParams()[0]
142 allObs = _gt_activeNode.getVar("allObs")
143 allObs.addAll(observations)
144 =====
145 Name: modelEvents
146 AllowMsg: true
147 Node Type: DEFAULT
148 Is Code File: false
149 Import:
150 Datamap:
151 Metadata:
152 Code:
153 =====

```

```

154 Name: isHaveLanded
155 AllowMsg: true
156 Node Type: DEFAULT
157 Is Code File: false
158 Import:
159 Datamap:
160 Metadata:
161 Code:
162     if state.getLastTrigger() == "doHaveLanded":
163         _htn_precon_ret=1
164     =====
165 Name: UASLanded
166 AllowMsg: true
167 Node Type: INTERRUPT
168 Is Code File: false
169 Import:
170 Datamap:
171 Metadata:
172 Code:
173     from cxxi.model.behavior import OrderUtilities
174     from java.util import Vector
175     printMessage("I_have_landed", True)
176     myTransporter = _gt_activeNode.getVar("myTransporter")
177     prims=Vector()
178     prims.add(
179     OrderUtilities.createBoardEntityPrimitiveByName(
180     myTransporter.getAssignedName()))
181     ord=orders.createOrder("HTN_Plan_Auto_Generated", 0.0, prims)
182     orders.addOrder(ord)
183     =====
184 Name: isHaveEmbarked
185 AllowMsg: true
186 Node Type: DEFAULT
187 Is Code File: false
188 Import:
189 Datamap:
190 Metadata:
191 Code:
192     if state.getLastTrigger() == "doEmbarkComplete":
193         _htn_precon_ret=1
194     =====

```

```

195 Name: doneWithMission
196 AllowMsg: false
197 Node Type: GOAL
198 Is Code File: false
199 Import:
200 Datamap:
201 Metadata:
202 Code:
203     from HTN import UtilityFuncsExp
204     from java.util import ArrayList
205     printMessage("embarked_on_transport", True)
206     allObs = _gt_activeNode.getVar("allObs")
207     obsList = ArrayList()
208     # now process the observations and see if we have anything to report
209     for something in allObs:
210         aname = something.getAssignedName()
211         rn = getUniformRandomNumber(0,1.0)
212         printMessage("checking_obs_of_" + str(aname) + "_rnd_num_is_" + str
(
213             rn), True)
214         # check to see ground truth if this is an IED
215         if borg.notIED.count(aname) == 0:
216             # this is not in the non-IED list
217             if rn > _gt_activeNode.getParam("falseNegativeRate"):
218                 # we have correctly identified this as an IED
219                 # so add it to the observations
220                 obsList.add(something)
221             else:
222                 if rn < _gt_activeNode.getParam("falsePositiveRate"):
223                     # we think this is an IED so add it to the observations
224                     obsList.add(something)
225         # tell our transporter that we have been loaded
226         UtilityFuncsExp.scheduleEvent(
227             _gt_activeNode.getVar("myTransporter").getAssignedName(),
228             "GoalTracker_UASPickedUp",
229             0.001,
230             obsList)

```



---

# APPENDIX F:

## UAV SENSING HTN

---

```
1=====
2Name: UAVSensor
3AllowMsg: false
4Node Type: DEFAULT
5Is Code File: false
6Import:
7Datamap:
8  sensorAzimuth=>[Ljava.lang.String;@2831300d
9  sensorPitch=>[Ljava.lang.String;@3549ba18
10 FOVWidth=>[Ljava.lang.String;@7b5898fc
11  maxRange=>[Ljava.lang.String;@75a407a7
12  meanFOVTime=>[Ljava.lang.String;@63f2a147
13  standDevFOVTime=>[Ljava.lang.String;@4e3a6f94
14Metadata:
15Code:
16  '''_Notes
17  '''_This_is_a_simple_model_for_a_sensor_observing_a_single_Field_of_View.
18  '''
19 =====
20 Name: initInfo
21 AllowMsg: false
22 Node Type: DEFAULT
23 Is Code File: false
24 Import:
25 Datamap:
26 Metadata:
27 Code:
28   if _gt_activeNode.getVar("isInited") == None:
29     _gt_activeNode.putVar("isInited", 1)
30     _htn_precon_ret=1
31 =====
32 Name: initsearch
33 AllowMsg: false
34 Node Type: DEFAULT
35 Is Code File: false
36 Import:
```

```

37 Datamap:
38 Metadata:
39 Code:
40     from HTN import UtilityFuncsExp
41     from UtilityFuncs import getNormalRandomNumber
42     #
43     # compute a time to scan this FOR
44     #
45     scanTime = getNormalRandomNumber(_gt_activeNode.getParam("meanFOVTime"
),
46     _gt_activeNode.getParam("standDevFOVTime"))
47     if scanTime < 1.0:
48         scanTime = 0.0
49     _gt_activeNode.putVar("scanTime", scanTime)
50     printMessage("scan_time_is_" + str(scanTime), True)
51     UtilityFuncsExp.scheduleEvent(
52         info.getMyAssignedName(),
53         "GoalTracker_StartFOVScan",
54         0.001,
55         None)
56 =====
57 Name: addReplanTriggers
58 AllowMsg: false
59 Node Type: INTERRUPT
60 Is Code File: false
61 Import:
62 Datamap:
63 Metadata:
64 Code:
65     goalContainer.getCurrentExecutingStack().addReplanTrigger("
GoalTracker_StartFOVScan")
66     goalContainer.getCurrentExecutingStack().addReplanTrigger("
GoalTracker_FOVScanDone")
67 =====
68 Name: events
69 AllowMsg: false
70 Node Type: DEFAULT
71 Is Code File: false
72 Import:
73 Datamap:
74 Metadata:

```

```

75 Code:
76 =====
77 Name: isGoalTrackerEvent
78 AllowMsg: false
79 Node Type: DEFAULT
80 Is Code File: false
81 Import:
82 Datamap:
83 Metadata:
84 Code:
85     if state.getLastTrigger().startswith("doGoalTracker_"):
86         _htn_precon_ret=1
87     =====
88 Name: isStartFOVScan
89 AllowMsg: false
90 Node Type: DEFAULT
91 Is Code File: false
92 Import:
93 Datamap:
94 Metadata:
95 Code:
96     if state.getLastTrigger() == "doGoalTracker_StartFOVScan":
97         _htn_precon_ret=1
98     =====
99 Name: scanFOV
100 AllowMsg: false
101 Node Type: INTERRUPT
102 Is Code File: false
103 Import:
104 Datamap:
105 Metadata:
106 Code:
107     from HIN import UtilityFuncsExp
108     from HIN.Scripts.general.SensorHelpers import isInFOV
109     from UtilityFuncs import getUniformRandomNumber
110     # collect what we can see in this FOV
111     from java.util import ArrayList
112     observations = ArrayList()
113     _gt_activeNode.putVar("observations", observations)
114     maxRange = _gt_activeNode.getParam("maxRange")
115     thingsCloseBy =myIntel.getLiveEntitiesInRadius(maxRange * 1.1)

```

```

116     myLocation = state.getCurrentLocation()
117     # get the actual sensor orientation
118     sensorOrientation = state.getCurrentHeading()
119     sensorOrientation.addDegreesToHeading(_gt_activeNode.getParam("
sensorAzimuth"))
120     sensorOrientation.setPitchInDegrees(_gt_activeNode.getParam("
sensorPitch"))
121     for something in thingsCloseBy:
122         tgtLoc = something.getPhysicalState().getSynchedLocation(False)
123         side = something.getSide()
124         if str(side) == "BLUE":
125             continue
126         if isInFOV(tgtLoc, myLocation, sensorOrientation,
127             _gt_activeNode.getParam("FOVWidth"), maxRange):
128             # compute probability of seeing
129             distToTgt = myLocation.distanceTo(tgtLoc)
130             pObs = 1.0 - (distToTgt/(220))
131             rn = getUniformRandomNumber(0,1.0)
132             printMessage("In_FOV_" + str(something.getAssignedName()) +
133                 "_Prob_detect_is_" + str(pObs), True)
134             if rn < pObs:
135                 observations.add(something)
136                 printMessage("I_saw_" + str(something.getAssignedName()) +
137                     "_rn_is_" + str(rn), True)
138             UtilityFuncsExp.scheduleEvent(
139                 info.getMyAssignedName(),
140                 "GoalTracker_FOVScanDone",
141                 _gt_activeNode.getVar("scanTime"),
142                 None)
143     =====
144     Name: isFOVScanDone
145     AllowMsg: false
146     Node Type: DEFAULT
147     Is Code File: false
148     Import:
149     Datamap:
150     Metadata:
151     Code:
152         if state.getLastTrigger() == "doGoalTracker_FOVScanDone":
153             _htn_precon_ret=1
154     =====

```

```

155     Name: scanDone
156     AllowMsg: false
157     Node Type: GOAL
158     Is Code File: false
159     Import:
160     Datamap:
161     Metadata:
162     Code:
163         from HTN import UtilityFuncsExp
164         UtilityFuncsExp.scheduleEvent(
165             info.getMyAssignedName(),
166             "GoalTracker_CompletedFOVScan",
167             0.001,
168             _gt_activeNode.getVar("observations"))
169     =====
170     Name: modelEvents
171     AllowMsg: false
172     Node Type: DEFAULT
173     Is Code File: false
174     Import:
175     Datamap:
176     Metadata:
177     Code:
178     =====
179     Name: printEvent
180     AllowMsg: false
181     Node Type: INTERRUPT
182     Is Code File: false
183     Import:
184     Datamap:
185     Metadata:
186     Code:
187         printMessage("EVENT"+state.getLastTrigger(), True)

```

THIS PAGE INTENTIONALLY LEFT BLANK

---

# APPENDIX G:

## IED ONTOLOGY

---

```
1<?xml version="1.0"?>
2
3
4<!DOCTYPE Ontology [
5   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
6   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
7   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
8   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
9]>
10
11
12<Ontology xmlns="http://www.w3.org/2002/07/owl#"
13   xml:base="http://www.ied.com/ontologies/ied.owl"
14   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
15   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
16   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
17   xmlns:xml="http://www.w3.org/XML/1998/namespace"
18   ontologyIRI="http://www.ied.com/ontologies/ied.owl">
19 <Prefix name="" IRI="http://www.w3.org/2002/07/owl#" />
20 <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#" />
21 <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
22 <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#" />
23 <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#" />
24 <Annotation>
25   <AnnotationProperty abbreviatedIRI="rdfs:comment" />
26   <Literal datatypeIRI="&rdf;PlainLiteral">An IED ontology that
describes various IEDs based on different properties that make up an IED
.</Literal>
27 </Annotation>
28 <Declaration>
29   <Class IRI="#Abandoned" />
30 </Declaration>
31 <Declaration>
32   <Class IRI="#Batteries" />
33 </Declaration>
34 <Declaration>
```

```
35     <Class IRI="#BelowGround" />
36 </Declaration>
37 <Declaration>
38     <Class IRI="#Cable" />
39 </Declaration>
40 <Declaration>
41     <Class IRI="#CellPhone" />
42 </Declaration>
43 <Declaration>
44     <Class IRI="#Child" />
45 </Declaration>
46 <Declaration>
47     <Class IRI="#ChildSBIED" />
48 </Declaration>
49 <Declaration>
50     <Class IRI="#ConfirmedIED" />
51 </Declaration>
52 <Declaration>
53     <Class IRI="#DeliveryMethod" />
54 </Declaration>
55 <Declaration>
56     <Class IRI="#DetectionMeans" />
57 </Declaration>
58 <Declaration>
59     <Class IRI="#DisturbedSoil" />
60 </Declaration>
61 <Declaration>
62     <Class IRI="#Electric" />
63 </Declaration>
64 <Declaration>
65     <Class IRI="#ElectricCurrent" />
66 </Declaration>
67 <Declaration>
68     <Class IRI="#FemaleSBIED" />
69 </Declaration>
70 <Declaration>
71     <Class IRI="#Grenade" />
72 </Declaration>
73 <Declaration>
74     <Class IRI="#Human" />
75 </Declaration>
```



```
76 <Declaration>
77     <Class IRI="#IED" />
78 </Declaration>
79 <Declaration>
80     <Class IRI="#IEDIndicators" />
81 </Declaration>
82 <Declaration>
83     <Class IRI="#IsolatedBox" />
84 </Declaration>
85 <Declaration>
86     <Class IRI="#Location" />
87 </Declaration>
88 <Declaration>
89     <Class IRI="#Magnetic" />
90 </Declaration>
91 <Declaration>
92     <Class IRI="#MaleSBIED" />
93 </Declaration>
94 <Declaration>
95     <Class IRI="#Man" />
96 </Declaration>
97 <Declaration>
98     <Class IRI="#ManWithCellPhone" />
99 </Declaration>
100 <Declaration>
101     <Class IRI="#MetalPlate" />
102 </Declaration>
103 <Declaration>
104     <Class IRI="#MilitaryVehicle" />
105 </Declaration>
106 <Declaration>
107     <Class IRI="#MouseTrap" />
108 </Declaration>
109 <Declaration>
110     <Class IRI="#Munitions" />
111 </Declaration>
112 <Declaration>
113     <Class IRI="#NonMilitaryVehicle" />
114 </Declaration>
115 <Declaration>
116     <Class IRI="#NonShellMilitary" />
```

```
117 </Declaration>
118 <Declaration>
119     <Class IRI="#Occupied" />
120 </Declaration>
121 <Declaration>
122     <Class IRI="#OnGround" />
123 </Declaration>
124 <Declaration>
125     <Class IRI="#Package" />
126 </Declaration>
127 <Declaration>
128     <Class IRI="#PackageIED" />
129 </Declaration>
130 <Declaration>
131     <Class IRI="#Physical" />
132 </Declaration>
133 <Declaration>
134     <Class IRI="#PossibleIED" />
135 </Declaration>
136 <Declaration>
137     <Class IRI="#PressurePlate" />
138 </Declaration>
139 <Declaration>
140     <Class IRI="#RoundRPG" />
141 </Declaration>
142 <Declaration>
143     <Class IRI="#RoundSmallArms" />
144 </Declaration>
145 <Declaration>
146     <Class IRI="#SBIED" />
147 </Declaration>
148 <Declaration>
149     <Class IRI="#ShellArtillery" />
150 </Declaration>
151 <Declaration>
152     <Class IRI="#ShellMilitary" />
153 </Declaration>
154 <Declaration>
155     <Class IRI="#ShellMortar" />
156 </Declaration>
157 <Declaration>
```

```
158     <Class IRI="#SodaCan" />
159 </Declaration>
160 <Declaration>
161     <Class IRI="#Springs" />
162 </Declaration>
163 <Declaration>
164     <Class IRI="#SteelTubes" />
165 </Declaration>
166 <Declaration>
167     <Class IRI="#StrongBeliefIED" />
168 </Declaration>
169 <Declaration>
170     <Class IRI="#Touch" />
171 </Declaration>
172 <Declaration>
173     <Class IRI="#Trigger" />
174 </Declaration>
175 <Declaration>
176     <Class IRI="#Trip" />
177 </Declaration>
178 <Declaration>
179     <Class IRI="#VBIED" />
180 </Declaration>
181 <Declaration>
182     <Class IRI="#Vehicle" />
183 </Declaration>
184 <Declaration>
185     <Class IRI="#Vehicles" />
186 </Declaration>
187 <Declaration>
188     <Class IRI="#Visual" />
189 </Declaration>
190 <Declaration>
191     <Class IRI="#WeakBeliefIED" />
192 </Declaration>
193 <Declaration>
194     <Class IRI="#Wires" />
195 </Declaration>
196 <Declaration>
197     <Class IRI="#Woman" />
198 </Declaration>
```

```

199 <Declaration>
200     <ObjectProperty IRI="#hasDeliveryMethod" />
201 </Declaration>
202 <Declaration>
203     <ObjectProperty IRI="#hasHumanDeliveryMethod" />
204 </Declaration>
205 <Declaration>
206     <ObjectProperty IRI="#hasIEDIndicators" />
207 </Declaration>
208 <Declaration>
209     <ObjectProperty IRI="#hasLocation" />
210 </Declaration>
211 <Declaration>
212     <ObjectProperty IRI="#hasPackageDeliveryMethod" />
213 </Declaration>
214 <Declaration>
215     <ObjectProperty IRI="#hasPossibleBelief" />
216 </Declaration>
217 <Declaration>
218     <ObjectProperty IRI="#hasTrigger" />
219 </Declaration>
220 <Declaration>
221     <ObjectProperty IRI="#hasVehicleDeliveryMethod" />
222 </Declaration>
223 <Declaration>
224     <DataProperty IRI="#hasIEDIndicatorName" />
225 </Declaration>
226 <Declaration>
227     <DataProperty IRI="#isConfirmed" />
228 </Declaration>
229 <Declaration>
230     <NamedIndividual IRI="#disturbedsoil_1" />
231 </Declaration>
232 <Declaration>
233     <NamedIndividual IRI="#ied_0" />
234 </Declaration>
235 <Declaration>
236     <NamedIndividual IRI="#ied_1" />
237 </Declaration>
238 <Declaration>
239     <NamedIndividual IRI="#ied_2" />

```

```

240 </Declaration>
241 <Declaration>
242     <NamedIndividual IRI="#ied_3" />
243 </Declaration>
244 <Declaration>
245     <NamedIndividual IRI="#isolatedbox_1" />
246 </Declaration>
247 <Declaration>
248     <NamedIndividual IRI="#wires_1" />
249 </Declaration>
250 <EquivalentClasses>
251     <Class IRI="#ConfirmedIED" />
252     <ObjectIntersectionOf>
253         <Class IRI="#IED" />
254         <DataHasValue>
255             <DataProperty IRI="#isConfirmed" />
256             <Literal datatypeIRI="&xsd;boolean">true</Literal>
257         </DataHasValue>
258     </ObjectIntersectionOf>
259 </EquivalentClasses>
260 <EquivalentClasses>
261     <Class IRI="#IED" />
262     <ObjectIntersectionOf>
263         <ObjectSomeValuesFrom>
264             <ObjectProperty IRI="#hasDeliveryMethod" />
265             <Class IRI="#DeliveryMethod" />
266         </ObjectSomeValuesFrom>
267         <ObjectSomeValuesFrom>
268             <ObjectProperty IRI="#hasIEDIndicators" />
269             <Class IRI="#IEDIndicators" />
270         </ObjectSomeValuesFrom>
271         <ObjectSomeValuesFrom>
272             <ObjectProperty IRI="#hasLocation" />
273             <Class IRI="#Location" />
274         </ObjectSomeValuesFrom>
275         <ObjectSomeValuesFrom>
276             <ObjectProperty IRI="#hasTrigger" />
277             <Class IRI="#Trigger" />
278         </ObjectSomeValuesFrom>
279     </ObjectIntersectionOf>
280 </EquivalentClasses>

```

```

281 <EquivalentClasses>
282   <Class IRI="#IEDIndicators" />
283   <DataSomeValuesFrom>
284     <DataProperty IRI="#hasIEDIndicatorName" />
285     <Datatype abbreviatedIRI="xsd:string" />
286   </DataSomeValuesFrom>
287 </EquivalentClasses>
288 <EquivalentClasses>
289   <Class IRI="#PossibleIED" />
290   <ObjectIntersectionOf>
291     <Class IRI="#IED" />
292     <DataHasValue>
293       <DataProperty IRI="#isConfirmed" />
294       <Literal datatypeIRI="&xsd;boolean">>false</Literal>
295     </DataHasValue>
296   </ObjectIntersectionOf>
297 </EquivalentClasses>
298 <EquivalentClasses>
299   <Class IRI="#StrongBeliefIED" />
300   <ObjectIntersectionOf>
301     <Class IRI="#PossibleIED" />
302     <ObjectMinCardinality cardinality="3">
303       <ObjectProperty IRI="#hasIEDIndicators" />
304       <Class IRI="#IEDIndicators" />
305     </ObjectMinCardinality>
306   </ObjectIntersectionOf>
307 </EquivalentClasses>
308 <EquivalentClasses>
309   <Class IRI="#WeakBeliefIED" />
310   <ObjectIntersectionOf>
311     <Class IRI="#PossibleIED" />
312     <ObjectMinCardinality cardinality="1">
313       <ObjectProperty IRI="#hasIEDIndicators" />
314       <Class IRI="#IEDIndicators" />
315     </ObjectMinCardinality>
316     <ObjectMaxCardinality cardinality="2">
317       <ObjectProperty IRI="#hasIEDIndicators" />
318       <Class IRI="#IEDIndicators" />
319     </ObjectMaxCardinality>
320   </ObjectIntersectionOf>
321 </EquivalentClasses>

```

```

322 <SubClassOf>
323     <Class IRI="#Abandoned" />
324     <Class IRI="#Vehicles" />
325 </SubClassOf>
326 <SubClassOf>
327     <Class IRI="#Batteries" />
328     <Class IRI="#IEDIndicators" />
329 </SubClassOf>
330 <SubClassOf>
331     <Class IRI="#BelowGround" />
332     <Class IRI="#Location" />
333 </SubClassOf>
334 <SubClassOf>
335     <Class IRI="#Cable" />
336     <Class IRI="#Wires" />
337 </SubClassOf>
338 <SubClassOf>
339     <Class IRI="#CellPhone" />
340     <Class IRI="#Trigger" />
341 </SubClassOf>
342 <SubClassOf>
343     <Class IRI="#Child" />
344     <Class IRI="#Human" />
345 </SubClassOf>
346 <SubClassOf>
347     <Class IRI="#ChildSBIED" />
348     <Class IRI="#SBIED" />
349 </SubClassOf>
350 <SubClassOf>
351     <Class IRI="#ChildSBIED" />
352     <ObjectSomeValuesFrom>
353         <ObjectProperty IRI="#hasHumanDeliveryMethod" />
354         <Class IRI="#Child" />
355     </ObjectSomeValuesFrom>
356 </SubClassOf>
357 <SubClassOf>
358     <Class IRI="#DisturbedSoil" />
359     <Class IRI="#IEDIndicators" />
360 </SubClassOf>
361 <SubClassOf>
362     <Class IRI="#Electric" />

```

```

363     <Class IRI="#Wires" />
364 </SubClassOf>
365 <SubClassOf>
366     <Class IRI="#ElectricCurrent" />
367     <Class IRI="#Trigger" />
368 </SubClassOf>
369 <SubClassOf>
370     <Class IRI="#FemaleSBIED" />
371     <Class IRI="#SBIED" />
372 </SubClassOf>
373 <SubClassOf>
374     <Class IRI="#FemaleSBIED" />
375     <ObjectSomeValuesFrom>
376         <ObjectProperty IRI="#hasHumanDeliveryMethod" />
377         <Class IRI="#Woman" />
378     </ObjectSomeValuesFrom>
379 </SubClassOf>
380 <SubClassOf>
381     <Class IRI="#Grenade" />
382     <Class IRI="#NonShellMilitary" />
383 </SubClassOf>
384 <SubClassOf>
385     <Class IRI="#Human" />
386     <Class IRI="#DeliveryMethod" />
387 </SubClassOf>
388 <SubClassOf>
389     <Class IRI="#IsolatedBox" />
390     <Class IRI="#IEDIndicators" />
391 </SubClassOf>
392 <SubClassOf>
393     <Class IRI="#Magnetic" />
394     <Class IRI="#DetectionMeans" />
395 </SubClassOf>
396 <SubClassOf>
397     <Class IRI="#MaleSBIED" />
398     <Class IRI="#SBIED" />
399 </SubClassOf>
400 <SubClassOf>
401     <Class IRI="#MaleSBIED" />
402     <ObjectSomeValuesFrom>
403         <ObjectProperty IRI="#hasHumanDeliveryMethod" />

```



```

404         <Class IRI="#Man" />
405     </ObjectSomeValuesFrom>
406 </SubClassOf>
407 <SubClassOf>
408     <Class IRI="#Man" />
409     <Class IRI="#Human" />
410 </SubClassOf>
411 <SubClassOf>
412     <Class IRI="#ManWithCellPhone" />
413     <Class IRI="#IEDIndicators" />
414 </SubClassOf>
415 <SubClassOf>
416     <Class IRI="#MetalPlate" />
417     <Class IRI="#IEDIndicators" />
418 </SubClassOf>
419 <SubClassOf>
420     <Class IRI="#MilitaryVehicle" />
421     <Class IRI="#Vehicle" />
422 </SubClassOf>
423 <SubClassOf>
424     <Class IRI="#MouseTrap" />
425     <Class IRI="#IEDIndicators" />
426 </SubClassOf>
427 <SubClassOf>
428     <Class IRI="#Munitions" />
429     <Class IRI="#IEDIndicators" />
430 </SubClassOf>
431 <SubClassOf>
432     <Class IRI="#NonMilitaryVehicle" />
433     <Class IRI="#Vehicle" />
434 </SubClassOf>
435 <SubClassOf>
436     <Class IRI="#NonShellMilitary" />
437     <Class IRI="#Munitions" />
438 </SubClassOf>
439 <SubClassOf>
440     <Class IRI="#Occupied" />
441     <Class IRI="#Vehicles" />
442 </SubClassOf>
443 <SubClassOf>
444     <Class IRI="#OnGround" />

```

```

445     <Class IRI="#Location" />
446 </SubClassOf>
447 <SubClassOf>
448     <Class IRI="#Package" />
449     <Class IRI="#DeliveryMethod" />
450 </SubClassOf>
451 <SubClassOf>
452     <Class IRI="#PackageIED" />
453     <Class IRI="#PossibleIED" />
454 </SubClassOf>
455 <SubClassOf>
456     <Class IRI="#PackageIED" />
457     <ObjectSomeValuesFrom>
458         <ObjectProperty IRI="#hasPackageDeliveryMethod" />
459         <Class IRI="#Package" />
460     </ObjectSomeValuesFrom>
461 </SubClassOf>
462 <SubClassOf>
463     <Class IRI="#Physical" />
464     <Class IRI="#DetectionMeans" />
465 </SubClassOf>
466 <SubClassOf>
467     <Class IRI="#PressurePlate" />
468     <Class IRI="#Trigger" />
469 </SubClassOf>
470 <SubClassOf>
471     <Class IRI="#RoundRPG" />
472     <Class IRI="#NonShellMilitary" />
473 </SubClassOf>
474 <SubClassOf>
475     <Class IRI="#RoundSmallArms" />
476     <Class IRI="#NonShellMilitary" />
477 </SubClassOf>
478 <SubClassOf>
479     <Class IRI="#SBIED" />
480     <Class IRI="#IED" />
481 </SubClassOf>
482 <SubClassOf>
483     <Class IRI="#SBIED" />
484     <ObjectSomeValuesFrom>
485         <ObjectProperty IRI="#hasHumanDeliveryMethod" />

```

```
486         <Class IRI="#Human" />
487     </ObjectSomeValuesFrom>
488 </SubClassOf>
489 <SubClassOf>
490     <Class IRI="#ShellArtillery" />
491     <Class IRI="#ShellMilitary" />
492 </SubClassOf>
493 <SubClassOf>
494     <Class IRI="#ShellMilitary" />
495     <Class IRI="#Munitions" />
496 </SubClassOf>
497 <SubClassOf>
498     <Class IRI="#ShellMortar" />
499     <Class IRI="#ShellMilitary" />
500 </SubClassOf>
501 <SubClassOf>
502     <Class IRI="#SodaCan" />
503     <Class IRI="#IEDIndicators" />
504 </SubClassOf>
505 <SubClassOf>
506     <Class IRI="#Springs" />
507     <Class IRI="#IEDIndicators" />
508 </SubClassOf>
509 <SubClassOf>
510     <Class IRI="#SteelTubes" />
511     <Class IRI="#IEDIndicators" />
512 </SubClassOf>
513 <SubClassOf>
514     <Class IRI="#Touch" />
515     <Class IRI="#Trigger" />
516 </SubClassOf>
517 <SubClassOf>
518     <Class IRI="#Trip" />
519     <Class IRI="#Wires" />
520 </SubClassOf>
521 <SubClassOf>
522     <Class IRI="#VBIED" />
523     <Class IRI="#IED" />
524 </SubClassOf>
525 <SubClassOf>
526     <Class IRI="#VBIED" />
```

```

527     <ObjectSomeValuesFrom>
528         <ObjectProperty IRI="#hasVehicleDeliveryMethod" />
529         <Class IRI="#Vehicle" />
530     </ObjectSomeValuesFrom>
531 </SubClassOf>
532 <SubClassOf>
533     <Class IRI="#Vehicle" />
534     <Class IRI="#DeliveryMethod" />
535 </SubClassOf>
536 <SubClassOf>
537     <Class IRI="#Vehicles" />
538     <Class IRI="#IEDIndicators" />
539 </SubClassOf>
540 <SubClassOf>
541     <Class IRI="#Visual" />
542     <Class IRI="#DetectionMeans" />
543 </SubClassOf>
544 <SubClassOf>
545     <Class IRI="#Wires" />
546     <Class IRI="#IEDIndicators" />
547 </SubClassOf>
548 <SubClassOf>
549     <Class IRI="#Woman" />
550     <Class IRI="#Human" />
551 </SubClassOf>
552 <DisjointClasses>
553     <Class IRI="#Abandoned" />
554     <Class IRI="#Occupied" />
555 </DisjointClasses>
556 <DisjointClasses>
557     <Class IRI="#Batteries" />
558     <Class IRI="#DisturbedSoil" />
559     <Class IRI="#IsolatedBox" />
560     <Class IRI="#ManWithCellPhone" />
561     <Class IRI="#MetalPlate" />
562     <Class IRI="#MouseTrap" />
563     <Class IRI="#Munitions" />
564     <Class IRI="#SodaCan" />
565     <Class IRI="#Springs" />
566     <Class IRI="#SteelTubes" />
567     <Class IRI="#Vehicles" />

```

```

568     <Class IRI="#Wires" />
569 </DisjointClasses>
570 <DisjointClasses>
571     <Class IRI="#BelowGround" />
572     <Class IRI="#OnGround" />
573 </DisjointClasses>
574 <DisjointClasses>
575     <Class IRI="#Cable" />
576     <Class IRI="#Electric" />
577     <Class IRI="#Trip" />
578 </DisjointClasses>
579 <DisjointClasses>
580     <Class IRI="#CellPhone" />
581     <Class IRI="#ElectricCurrent" />
582     <Class IRI="#PressurePlate" />
583     <Class IRI="#Touch" />
584 </DisjointClasses>
585 <DisjointClasses>
586     <Class IRI="#Child" />
587     <Class IRI="#Man" />
588     <Class IRI="#Woman" />
589 </DisjointClasses>
590 <DisjointClasses>
591     <Class IRI="#ChildSBIED" />
592     <Class IRI="#FemaleSBIED" />
593     <Class IRI="#MaleSBIED" />
594 </DisjointClasses>
595 <DisjointClasses>
596     <Class IRI="#ConfirmedIED" />
597     <Class IRI="#PossibleIED" />
598     <Class IRI="#SBIED" />
599     <Class IRI="#VBIED" />
600 </DisjointClasses>
601 <DisjointClasses>
602     <Class IRI="#DeliveryMethod" />
603     <Class IRI="#DetectionMeans" />
604     <Class IRI="#IED" />
605     <Class IRI="#IEDIndicators" />
606     <Class IRI="#Location" />
607     <Class IRI="#Trigger" />
608 </DisjointClasses>

```

```

609 <DisjointClasses>
610     <Class IRI="#Grenade" />
611     <Class IRI="#RoundRPG" />
612     <Class IRI="#RoundSmallArms" />
613 </DisjointClasses>
614 <DisjointClasses>
615     <Class IRI="#Human" />
616     <Class IRI="#Package" />
617     <Class IRI="#Vehicle" />
618 </DisjointClasses>
619 <DisjointClasses>
620     <Class IRI="#Magnetic" />
621     <Class IRI="#Physical" />
622     <Class IRI="#Visual" />
623 </DisjointClasses>
624 <DisjointClasses>
625     <Class IRI="#MilitaryVehicle" />
626     <Class IRI="#NonMilitaryVehicle" />
627 </DisjointClasses>
628 <DisjointClasses>
629     <Class IRI="#NonShellMilitary" />
630     <Class IRI="#ShellMilitary" />
631 </DisjointClasses>
632 <DisjointClasses>
633     <Class IRI="#PackageIED" />
634     <Class IRI="#SBIED" />
635     <Class IRI="#VBIED" />
636 </DisjointClasses>
637 <DisjointClasses>
638     <Class IRI="#ShellArtillery" />
639     <Class IRI="#ShellMortar" />
640 </DisjointClasses>
641 <DisjointClasses>
642     <Class IRI="#StrongBeliefIED" />
643     <Class IRI="#WeakBeliefIED" />
644 </DisjointClasses>
645 <ClassAssertion>
646     <Class IRI="#DisturbedSoil" />
647     <NamedIndividual IRI="#disturbedsoil_1" />
648 </ClassAssertion>
649 <ClassAssertion>

```

```

650     <Class IRI="#IED" />
651     <NamedIndividual IRI="#ied_0" />
652 </ClassAssertion>
653 <ClassAssertion>
654     <Class IRI="#IED" />
655     <NamedIndividual IRI="#ied_1" />
656 </ClassAssertion>
657 <ClassAssertion>
658     <Class IRI="#IED" />
659     <NamedIndividual IRI="#ied_2" />
660 </ClassAssertion>
661 <ClassAssertion>
662     <Class IRI="#IED" />
663     <NamedIndividual IRI="#ied_3" />
664 </ClassAssertion>
665 <ClassAssertion>
666     <Class IRI="#IsolatedBox" />
667     <NamedIndividual IRI="#isolatedbox_1" />
668 </ClassAssertion>
669 <ClassAssertion>
670     <Class IRI="#Wires" />
671     <NamedIndividual IRI="#wires_1" />
672 </ClassAssertion>
673 <ObjectPropertyAssertion>
674     <ObjectProperty IRI="#hasIEDIndicators" />
675     <NamedIndividual IRI="#ied_1" />
676     <NamedIndividual IRI="#disturbedsoil_1" />
677 </ObjectPropertyAssertion>
678 <ObjectPropertyAssertion>
679     <ObjectProperty IRI="#hasIEDIndicators" />
680     <NamedIndividual IRI="#ied_1" />
681     <NamedIndividual IRI="#wires_1" />
682 </ObjectPropertyAssertion>
683 <ObjectPropertyAssertion>
684     <ObjectProperty IRI="#hasIEDIndicators" />
685     <NamedIndividual IRI="#ied_2" />
686     <NamedIndividual IRI="#isolatedbox_1" />
687 </ObjectPropertyAssertion>
688 <ObjectPropertyAssertion>
689     <ObjectProperty IRI="#hasIEDIndicators" />
690     <NamedIndividual IRI="#ied_2" />

```

```

691     <NamedIndividual IRI="#wires_1"/>
692 </ObjectPropertyAssertion>
693 <ObjectPropertyAssertion>
694     <ObjectProperty IRI="#hasIEDIndicators"/>
695     <NamedIndividual IRI="#ied_2"/>
696     <NamedIndividual IRI="#disturbedsoil_1"/>
697 </ObjectPropertyAssertion>
698 <ObjectPropertyAssertion>
699     <ObjectProperty IRI="#hasIEDIndicators"/>
700     <NamedIndividual IRI="#ied_3"/>
701     <NamedIndividual IRI="#wires_1"/>
702 </ObjectPropertyAssertion>
703 <ObjectPropertyAssertion>
704     <ObjectProperty IRI="#hasIEDIndicators"/>
705     <NamedIndividual IRI="#ied_3"/>
706     <NamedIndividual IRI="#isolatedbox_1"/>
707 </ObjectPropertyAssertion>
708 <ObjectPropertyAssertion>
709     <ObjectProperty IRI="#hasIEDIndicators"/>
710     <NamedIndividual IRI="#ied_3"/>
711     <NamedIndividual IRI="#disturbedsoil_1"/>
712 </ObjectPropertyAssertion>
713 <DataPropertyAssertion>
714     <DataProperty IRI="#hasIEDIndicatorName"/>
715     <NamedIndividual IRI="#disturbedsoil_1"/>
716     <Literal datatypeIRI="&xsd:string">dirty</Literal>
717 </DataPropertyAssertion>
718 <DataPropertyAssertion>
719     <DataProperty IRI="#isConfirmed"/>
720     <NamedIndividual IRI="#ied_0"/>
721     <Literal datatypeIRI="&xsd:boolean">>false</Literal>
722 </DataPropertyAssertion>
723 <DataPropertyAssertion>
724     <DataProperty IRI="#isConfirmed"/>
725     <NamedIndividual IRI="#ied_1"/>
726     <Literal datatypeIRI="&xsd:boolean">>false</Literal>
727 </DataPropertyAssertion>
728 <DataPropertyAssertion>
729     <DataProperty IRI="#isConfirmed"/>
730     <NamedIndividual IRI="#ied_2"/>
731     <Literal datatypeIRI="&xsd:boolean">>false</Literal>

```



```

732 </DataPropertyAssertion>
733 <DataPropertyAssertion>
734   <DataProperty IRI="#isConfirmed" />
735   <NamedIndividual IRI="#ied_3" />
736   <Literal datatypeIRI="&xsd;boolean">true</Literal>
737 </DataPropertyAssertion>
738 <DataPropertyAssertion>
739   <DataProperty IRI="#hasIEDIndicatorName" />
740   <NamedIndividual IRI="#isolatedbox_1" />
741   <Literal datatypeIRI="&xsd;string">isolatedbox</Literal>
742 </DataPropertyAssertion>
743 <DataPropertyAssertion>
744   <DataProperty IRI="#hasIEDIndicatorName" />
745   <NamedIndividual IRI="#wires_1" />
746   <Literal datatypeIRI="&xsd;string">wires</Literal>
747 </DataPropertyAssertion>
748 <SubObjectPropertyOf>
749   <ObjectProperty IRI="#hasHumanDeliveryMethod" />
750   <ObjectProperty IRI="#hasDeliveryMethod" />
751 </SubObjectPropertyOf>
752 <SubObjectPropertyOf>
753   <ObjectProperty IRI="#hasIEDIndicators" />
754   <ObjectProperty abbreviatedIRI="owl:topObjectProperty" />
755 </SubObjectPropertyOf>
756 <SubObjectPropertyOf>
757   <ObjectProperty IRI="#hasPackageDeliveryMethod" />
758   <ObjectProperty IRI="#hasDeliveryMethod" />
759 </SubObjectPropertyOf>
760 <SubObjectPropertyOf>
761   <ObjectProperty IRI="#hasVehicleDeliveryMethod" />
762   <ObjectProperty IRI="#hasDeliveryMethod" />
763 </SubObjectPropertyOf>
764 <FunctionalObjectProperty>
765   <ObjectProperty IRI="#hasLocation" />
766 </FunctionalObjectProperty>
767 <ObjectPropertyDomain>
768   <ObjectProperty IRI="#hasDeliveryMethod" />
769   <Class IRI="#IED" />
770 </ObjectPropertyDomain>
771 <ObjectPropertyDomain>
772   <ObjectProperty IRI="#hasHumanDeliveryMethod" />

```

```

773     <Class IRI="#IED" />
774 </ObjectPropertyDomain>
775 <ObjectPropertyDomain>
776     <ObjectProperty IRI="#hasIEDIndicators" />
777     <Class IRI="#IED" />
778 </ObjectPropertyDomain>
779 <ObjectPropertyDomain>
780     <ObjectProperty IRI="#hasLocation" />
781     <Class IRI="#IED" />
782 </ObjectPropertyDomain>
783 <ObjectPropertyDomain>
784     <ObjectProperty IRI="#hasPackageDeliveryMethod" />
785     <Class IRI="#IED" />
786 </ObjectPropertyDomain>
787 <ObjectPropertyDomain>
788     <ObjectProperty IRI="#hasPossibleBelief" />
789     <Class IRI="#IED" />
790 </ObjectPropertyDomain>
791 <ObjectPropertyDomain>
792     <ObjectProperty IRI="#hasTrigger" />
793     <Class IRI="#IED" />
794 </ObjectPropertyDomain>
795 <ObjectPropertyDomain>
796     <ObjectProperty IRI="#hasVehicleDeliveryMethod" />
797     <Class IRI="#IED" />
798 </ObjectPropertyDomain>
799 <ObjectPropertyRange>
800     <ObjectProperty IRI="#hasDeliveryMethod" />
801     <Class IRI="#DeliveryMethod" />
802 </ObjectPropertyRange>
803 <ObjectPropertyRange>
804     <ObjectProperty IRI="#hasHumanDeliveryMethod" />
805     <Class IRI="#Human" />
806 </ObjectPropertyRange>
807 <ObjectPropertyRange>
808     <ObjectProperty IRI="#hasLocation" />
809     <Class IRI="#Location" />
810 </ObjectPropertyRange>
811 <ObjectPropertyRange>
812     <ObjectProperty IRI="#hasPackageDeliveryMethod" />
813     <Class IRI="#Package" />

```

```

814 </ObjectPropertyRange>
815 <ObjectPropertyRange>
816     <ObjectProperty IRI="#hasPossibleBelief"/>
817     <Class IRI="#PossibleIED"/>
818 </ObjectPropertyRange>
819 <ObjectPropertyRange>
820     <ObjectProperty IRI="#hasTrigger"/>
821     <Class IRI="#Trigger"/>
822 </ObjectPropertyRange>
823 <ObjectPropertyRange>
824     <ObjectProperty IRI="#hasVehicleDeliveryMethod"/>
825     <Class IRI="#Vehicle"/>
826 </ObjectPropertyRange>
827 <FunctionalDataProperty>
828     <DataProperty IRI="#hasIEDIndicatorName"/>
829 </FunctionalDataProperty>
830 <FunctionalDataProperty>
831     <DataProperty IRI="#isConfirmed"/>
832 </FunctionalDataProperty>
833 <DataPropertyDomain>
834     <DataProperty IRI="#hasIEDIndicatorName"/>
835     <Class IRI="#IEDIndicators"/>
836 </DataPropertyDomain>
837 <DataPropertyDomain>
838     <DataProperty IRI="#isConfirmed"/>
839     <Class IRI="#IED"/>
840 </DataPropertyDomain>
841 <DataPropertyRange>
842     <DataProperty IRI="#hasIEDIndicatorName"/>
843     <Datatype abbreviatedIRI="xsd:string"/>
844 </DataPropertyRange>
845 <DataPropertyRange>
846     <DataProperty IRI="#isConfirmed"/>
847     <Datatype abbreviatedIRI="xsd:boolean"/>
848 </DataPropertyRange>
849</Ontology>
850
851
852
853<!-- Generated by the OWL API (version 3.4.2) http://owlapi.sourceforge.net
-->

```

THIS PAGE INTENTIONALLY LEFT BLANK

---

---

## LIST OF REFERENCES

---

- [1] U. S. Army, *COMBATXXI Users Guide*, U.S. Army WSMR, n.d.
- [2] T. Mauer et al., “Search and detection modeling of military imaging systems,” in *Proc. SPIE 5784, Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XVI, 201*, Orlando, Florida, 2005, pp. 201–215.
- [3] I. Balogh et al., “Using hierarchical task networks to create dynamic behaviors in combat models,” unpublished.
- [4] S. C. for Biomedical Informatics Research. *What Is Protege? The National Center for Biomedical Ontology*. (2013). [Online]. Available: <http://protege.stanford.edu/aboutus/aboutus.html>
- [5] U. Staff, *Eyes of the Army, U.S. Army Unmanned Aircraft Systems Roadmap 2010-2035*, U.S. Army UAS Center of Excellence, Fort Rucker, Alabama, n.d.
- [6] J. Gertler, “U.S. unmanned aerial systems,” Congressional Research Service, CSR Report for Congress R42136, Jan. 2012.
- [7] Raven overview, (2013). [Online]. Available: [http://www.avinc.com/downloads/Raven\\_Gimbal.pdf](http://www.avinc.com/downloads/Raven_Gimbal.pdf)
- [8] Raven UAS launch, (2013). [Online]. Available: [http://www.avinc.com/img/media\\_gallery/UAS\\_Raven\\_Field\\_Launch.jpg](http://www.avinc.com/img/media_gallery/UAS_Raven_Field_Launch.jpg)
- [9] Kestrel - land mti for small unmanned aircraft systems, (2013). [Online]. Available: <http://www.avinc.com/uas/kestrel/>
- [10] U. A. Force, “U.S. Air Force scan eagle fact sheet,” (2013). [Online]. Available: <http://www.af.mil/AboutUs/FactSheets/Display/tabid/224/Article/104532/scan-eagle.aspx>
- [11] AAI unmanned aircraft systems. Shadow RQ-7B System Data Sheet. Hunt Valley, MD, 2012.
- [12] Shadow 200 rq-7 tactical unmanned aircraft system, United States of America. [Online]. Available: <http://www.army-technology.com/projects/shadow200uav/shadow200uav3.html>
- [13] General atomics aeronautical. Gray Eagle Armed Persistence Data Sheet. Poway: CA, G.A.A.S.I., 2012.
- [14] General atomics awarded \$30m gray eagle contract. (2012). [Online]. Available: <http://www.unmanned.co.uk/unmanned-vehicles-news/unmanned-aerial-vehicles-uav-news/general-atomics-awarded-30m-gray-eagle-contract/>

- [15] U.S. Air Force mq-9 reaper fact sheet. (2010). [Online]. Available: <http://www.af.mil/AboutUs/FactSheets/Display/tabid/224/Article/104470/mq-9-reaper.aspx>
- [16] MQ-9 reaper. photo: Air force, (2012). [Online]. Available: <http://www.wired.com/dangerroom/2012/04/killer-drone-upgrade/training-to-hunt/>
- [17] U.S.J.C. of Staff, *Joint Publication 3-07.2 Antiterrorism*, Department of Defense, Washington, D.C.
- [18] Faces of the fallen. (2013). [Online]. Available: <http://apps.washingtonpost.com/national/fallen/causes-of-death/ied/>
- [19] U. A. Training and Doctrine, *FM 3-24.2 Tactics in Counterinsurgency*, Headquarters Department of the Army, Washington, D.C.
- [20] C. Shepherd, "Methods for ied reconnaissance and detection," vol. 113, p. not cited, Sep./Oct. 2004.
- [21] R. H. Vollmerhausen and E. Jacobs, "The targeting task performance (ttp) metric a new model for predicting target acquisition performance," Modeling and Simulation Division Night Vision and Electronic Sensors Directorate, VA, Tech. Rep. AMSEL-NV-TR-230, Aug. 2004.
- [22] O. H. Schade, "Optical and photoelectric analog of the eye," *J. Opt. Soc. Am.*, vol. 46, no. 9, pp. 721–738, Sep 1956. [Online]. Available: <http://www.opticsinfobase.org/abstract.cfm?URI=josa-46-9-721>
- [23] Night vision and electronic sensors directorate history. (2013). [Online]. Available: <http://www.nvl.army.mil/history.html>
- [24] Thermal imaging: how far can you see it? data sheet. (2013). [Online]. Available: [http://www.flir.com/uploadedfiles/eng\\_01\\_howfar.pdf](http://www.flir.com/uploadedfiles/eng_01_howfar.pdf)
- [25] B. O’Kane et al., "Cycle criteria for detection of camouflaged targets," DTIC Document, Tech. Rep., 2005.
- [26] C. J. Darken, "Computer graphics-based models of target detection: Algorithms, comparison to human performance, and failure modes," *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, vol. 4, no. 3, pp. 262–276, 2007. [Online]. Available: <http://dms.sagepub.com/content/4/3/262.abstract>
- [27] F. Baez, "Target acquisition methods in combat simulations," unpublished.
- [28] E. D. Sacerdoti, "A structure for plans and behavior," DTIC Document, Tech. Rep., 1975.
- [29] S. S. J. A. Baier and S. A. McIlraith, "Htn planning with preferences."

- [30] I. Balogh et al., "Use of hierarchical task networks to model complex operational tasks in combatxxi(PowerPoint Presentation)," 80th Military Operations Research Society Symposium, Mission Area Analysis Branch, 3300 Russell Road Quantico, VA, June 11-14, 2012.
- [31] R. Brachman and H. Levesque. (2003.) "Knowledge representation and reasoning". [Online]. Available: <http://www.cin.ufpe.br/~mtcfa/files/in1122/Knowledge%20Representation%20and%20Reasoning.pdf>
- [32] R. Davis et al., "What Is a Knowledge Representation," *AI Magazine*, vol. 14, no. 1, pp. 17–33, 1993.
- [33] V. Mehta, D. Patel, and N. Reddy. History of knowledge representation. (2008). [Online]. Available: <http://www.knowledgerep.info/history.html>
- [34] K. Klement. Propositional logic: History. (2005). [Online]. Available: "<http://www.iep.utm.edu/prop-log/#H2>"
- [35] G. Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Boston, MA: Addison-Wesley, 2009.
- [36] A. Collins and M. Quillian, "Retrieval time from semantic memory," *Journal of Verbal Learning and Verbal Behavior*, vol. 8, no. 2, pp. 240–247, 1969.
- [37] Merriam-Webster. Merriam-webster. (2013). [Online]. Available: <http://www.merriam-webster.com/dictionary/ontology>
- [38] C. Childers, "Applying semantic web concepts to support net-centric warfare using the tactical assessment markup language TAML," M. S. thesis, Department of Computer Science, Naval Postgraduate School, Monterey, CA, 2006.
- [39] TRAC-Monterey, "Deployable force protection scenario," unpublished.
- [40] I. Balogh, "UAV sensor model," PowerPoint Slide, Naval Postgraduate School, Monterey, CA, Aug 2013.
- [41] W.W.W. Consortium. Owl web ontology language overview, introduction. (2004). [Online]. Available: <http://www.w3.org/TR/2004/REC-owl-features-20040210/>
- [42] Semantic reasoner. (2013). [Online]. Available: "[http://en.wikipedia.org/wiki/Semantic\\_reasoner](http://en.wikipedia.org/wiki/Semantic_reasoner)"
- [43] M. Horridge. *A practical guide to building owl ontologies using protege 4 and co-ode tools edition 1.3. (2011)*. [Online]. Available: "[http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4\\_v1\\_3.pdf](http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf)"

[44] J. Johnson, "Analysis of image forming systems," presented at the Image Intensifier Symposium, Warfare Vision Branch, Electrical Engineering Dept., U.S. Army Engineer Research and Development Laboratories, 1958.



---

---

# INITIAL DISTRIBUTION LIST

---

1. Defense Technical Information Center  
Ft. Belvoir, Virginia
2. Dudley Knox Library  
Naval Postgraduate School  
Monterey, California