USING AGENT-BASED MODELING TO EVALUATE UAS BEHAVIORS IN A TARGET-RICH ENVIRONMENT

THESIS

Joseph A. Van Kuiken, Captain, USAF

AFIT/GOR/ENS/09-16

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

USING AGENT-BASED MODELING TO EVALUATE UAS BEHAVIORS IN A TARGET-RICH ENVIRONMENT

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Masters of Science in Operations Research

Joseph A. Van Kuiken

Captain, USAF

March 2009

USING AGENT-BASED MODELING TO EVALUATE UAS BEHAVIORS IN A TARGET-RICH ENVIRONMENT

Joseph A. Van Kuiken, BS
Captain, USAF

Approved:

Dr. J. O. Miller (Chairman)	Date
Maj. Shane N. Hall, Ph.D. (Member)	Date

Abstract

The trade-off between accuracy and speed is a re-occurring dilemma in many facets of military performance evaluation. This is an especially important issue in the world of ISR. One of the most progressive areas of ISR capabilities has been the utilization of Unmanned Aircraft Systems (UAS). Many people believe that the future of UAS lies in smaller vehicles flying in swarms. We use the agent-based System Effectiveness and Analysis Simulation (SEAS) to create a simulation environment where different configurations of UAS vehicles can process targets and provide output that allows us to gain insight into the benefits and drawbacks of each configuration. Our evaluation on the performance of the different configurations is based on probability of correct identification, average time to identify a target after it has deployed in the area of interest, and average time to identify all targets in an area.

Acknowledgements

First and foremost, I would not be where I am today without God. He makes all things possible, and I owe my accomplishments to Him. I am grateful for the many blessings he has given me in my life.

Secondly, I thank Dr. J.O. Miller. I greatly appreciate his continuous support and professional guidance. His patience and dedication to my success made it possible to overcome many challenges along the way.

To my parents, your support and guidance is priceless. Thank you for giving me encouragement through it all. You will always have my love, respect, and admiration.

To my classmates and friends I have made while at AFIT, thank you for your support as well. I wish you all the best and continued success in the future!

Table of Contents

List of Figures

List of Tables

USING AGENT-BASED MODELING TO EVALUATE UAS BEHAVIORS IN A TARGET-RICH ENVIRONMENT

I. Introduction

Background

A UAV (Unmanned Aerial Vehicle) is defined as an unpiloted aircraft capable of controlled, sustained, level flight and powered by a jet or reciprocating engine. This refers to either an unmanned aircraft that flies while being remotely controlled by an operator or an aircraft that flies autonomously without human direction. Today, the term UAS (Unmanned Aircraft System) is more common to reflect the fact that these vehicles are not stand-alone aircraft, but rather use a network of supporting elements such as ground stations. There are several benefits to using UAS aircraft in today's military. Perhaps most importantly is that there is no risk of loss of life when using a UAS. Secondly, because they are unmanned, they are no longer confined to restrictions related to humans. This means that UAS aircraft can fly at much higher levels, can perform much riskier CONOPS, and can in general perform much riskier missions than manned aircraft.

The roots of the UAS concept come from the same place as that of the cruise missile; they were developed as remote aerial directed munitions. The first successful flight of one of these unmanned vehicles was in 1918 by the Curtiss/Sperry Aerial Torpedo. Since then, there have been several models developed, but the biggest explosion of UAS development came during the Second World War The UAS aircraft developed during this time were primarily used for anti-aircraft gun training and to fly attack missions. It wasn't until after World War II that people started applying jet

engines to UAS vehicles. In 1955, the US Navy bought a UAS developed by Beechcraft. However, none of these were used like the UAS vehicles of today. In the 1980s and 90s, the US military's interest in UAS vehicles grew and the technology followed. Although roles traditionally focused around surveillance, some models (Predator MQ-1) were equipped with weapons (AGM-114 Hellfire Air to Ground Missiles). As interest in UAS vehicles grow, it is inevitable that its roles will expand as well. (Newcome, 2004)

Currently, UAS vehicles support twelve missions; reconnaissance, signals intelligence, mine countermeasures, target designation, battle management, chemical/biological reconnaissance, counter cam/con/deception, electronic warfare, combat search and rescue, communications/data relay, information warfare, and digital mapping (Gooden, 2000). It is not unreasonable to expect that UAS vehicles may one day replace all roles currently supported by manned aircraft.

Before any further detail is discussed, it is important to understand how a model is defined. According to the Air Force Modeling and Simulation Resource Repository, a model is "A physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process." (Gansler, 1998). Another definition is offered by Law & Kelton, who say that a model is a representation of a system to gain insight into how the system behaves or to predict future behavior (Law, 2000). They also note that models can be either physical or mathematical. Physical models represent the target system in form and are used primarily to convey information about a system's physical aspects. Mathematical models are models that represent a system by characterizing certain parameters of the system using equations.

Mathematical models are further broken-down into either analytical solutions or simulations. Analytical solutions can be calculated without computer aid using statistical techniques. Simulations, on the other hand, are typically used for problems that involve many interactions making them too complicated to calculate using simple techniques. Another benefit of simulations is that knowing only the input parameters, you can often observe behaviors to better understand why the system is behaving the way it is. This is where the real value in simulations lies; gaining insight into the behavior of a system using observable patterns, not just output values.

Combat models are one way in which analysts can observe the behavior of military systems. Since the focus of our research will be on the behavior of UAS vehicles, using combat models is appropriate. There are many different classifications of combat models depending on the scope of the model's purpose. Some models focus on modeling the detailed interaction of agents. These models are typically used to describe the strengths and weaknesses of single players interactions and are referred to as engagement models. One layer up in complexity (and one layer down in fidelity) are mission level models. These models are typically a little broader than engagement models, focusing more on force vs. force scenarios where the details of single agent engagements aren't so important, and can be sacrificed for scenario efficiency, considering the increased complexity. Above mission level models are campaign level models, which focus even less on individual interactions and more on broad results. Figure 1.1 shows the relationship between the different levels. As you move down the pyramid, the more detailed the model becomes (higher resolution), but the fewer interactions that are actually modeled.

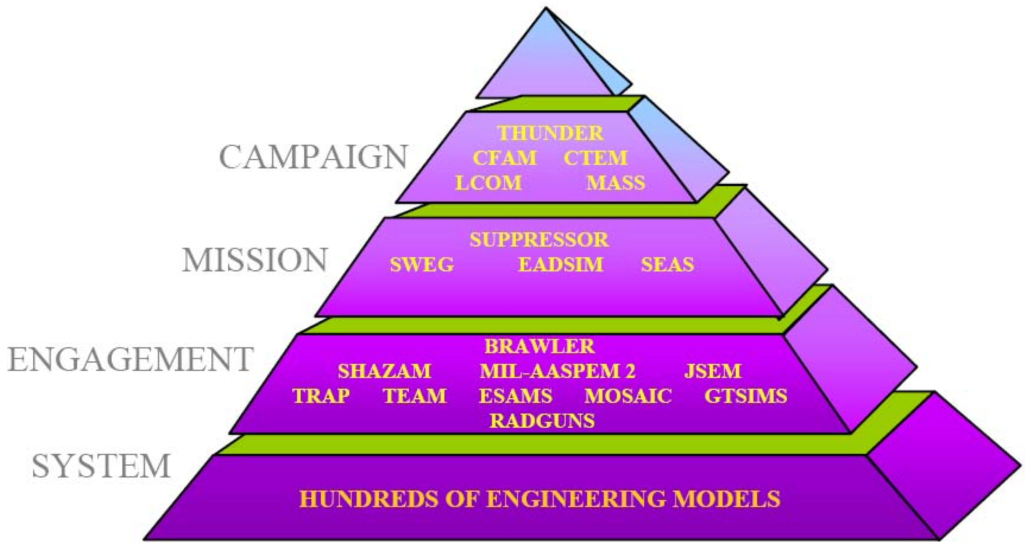

Figure 1.1 Hierarchy of Models (Miller, 2008)

Since we're interested in observing UAS interactions/swarming methods, a mission-level model is most appropriate for our endeavors.

Problem Statement / Research Objective

The objective of our efforts is to better understand which queing/swarming techniques would be most beneficial for various UAS vehicles to utilize in different scenarios. Depending on how many UAS vehicles are available, and how large/target-rich the environment is, different CONOPS may be better than others. Our work is made in an effort to provide some better understanding of what governs these artifacts.

Research Focus

The focus of our research is to provide a guideline for what CONOPS are most appropriate for different situations (defined by area, threat, targets, etc). Once we get a better understanding of this (based on simulations), we can apply our results to analyze past conflicts where UAS vehicles have been utilized and evaluate the CO-NOPS used in those operations.

System Effectiveness Analysis Simulation (SEAS) will be the combat model of choice for our efforts. There are several reasons SEAS is preferred. First, it is a user-friendly simulation program. Parameters and orders are logically named and easy to distinguish, making it easy to decipher the coding language. SEAS also offers an extensive help file which is useful in both coding and debugging. Secondly, it is a mission-level agent-based model. This is appropriate when dealing with this kind of problem, because we are interested in the swarming behavior of groups of cooperative UAS vehicles, and we require coding and observation of the individual aircraft. Third, SEAS has a simple graphical display that is useful in conveying information visually without being too flashy. This is desired for our kind of analysis. Fourth, SEAS has the processing capability to model the kind of scenario that will be required for analyzing UAS behavior. It adequately models relationships between air, sea, ground, and space agents and their interactions. Fifth, SEAS recently introduced new features into its new release (SEAS 3.7.1) that we would like to examine. These features include a confusion matrix useful in assigning a probability of misidentification and road networks for modeling traffic.

Overview of Thesis

Chapter II will be comprised of reviews of existing literary works on related subjects. These subjects include agent-based modeling, SEAS, and UAS CONOPS. Agent-based modeling has been used since the 1940s but didn't become widely accepted until the 1990's (Eamonn, 2008). Since simulation and agent-based modeling have recently become popular within the operations research community, there is an abundance of information on agent-based modeling and its applications. We can use lessons learned from previous applications to ensure proper execution of our study. SEAS is a specific agent-based model used by many Department of Defense agencies to evaluate systems effectiveness. In Chapter II, we will look at some previous applications of SEAS, specifically in autonomous aerial vehicle based scenarios, and use these applications to better model our scenario. We will also research traditional UAS CONOPS to ensure accurate representation within our model.

Chapter III describes how we modeled our scenario. This includes what agents were present, what capabilities these agents had, what orders the agents had, and what assumptions we made. Since combat modeling is subjective by nature, Chapter III will be highly detailed to ensure that the details of the model are conveyed adequately.

Chapter IV will discuss not only what statistics we chose to pull from our model, but also how we analyzed that data to make our eventual conclusions. SEAS has a post-processor designed to easily manipulate output from the model, so we will use that extensively. If needed, it is also possible to extract the data to an excel file and manipulate them there using some simple VBA code since SEAS output files are comma delimited.

Chapter V will outline our conclusions as dictated by the results outlined in Chapter IV. We will also discuss ideas for future research and suggestions for applications of our results.

II. Literature Review

Introduction

Modeling and Simulation has been used by analysts since the 40s. Currently it is widely used by military branches, social sciences, biological sciences, and environmental sciences. Later in this chapter, we will begin with the definition of agent-based modeling (ABM) and discuss both how it was used in the past, and how it is used today. The next part will delve into some of the specific strengths and weaknesses of the particular model chosen for the analysis. Lastly, past studies that have been focused on ABM, UAS behaviors, or both will be discussed and applicable aspects will be expounded upon.

Agent-based Modeling

ABM has been defined many different ways by many different analysts. The creators (SPARTA, Inc.) of the ABM (SEAS) we will use for our study define it as an environment in which

> "…complex, real-world systems are modeled as a collection of autonomous decision making entities, called agents. Each agent individually assesses its situation and makes decisions based upon its own set of rules. Agents may execute various behaviors appropriate for the system they represent - for example, sensing, maneuvering, or engaging" (SPARTA, 2005)

It is also important to understand how an agent is defined. In his book, J. Ferber defines an agent as a physical or virtual entity

- that is capable of acting in an environment;
- that can communicate directly with other agents;
- that is driven by a set of tendencies (has autonomy);
- which possesses resources of its own;
- that is capable of perceiving its environment;
- that has only a partial representation of this environment;

- which possesses skills and can offer services;
- that may be able to reproduce itself; and

- whose behavior tends towards satisfying its objectives, taking account of the resources and skills available to it and, depending on its perception, its representations and the communications it receives.

Multi-agent systems must include:

- an environment;
- a set of objects that can be perceived, created, destroyed and modified by the agents;
- an assembly of agents;
- an assembly of relations linking the agents;
- an assembly of operations enabling agents to perceive, produce, consume, transform, and manipulate objects; and
- operators whose task is to represent the application and reaction to these operations.

Perhaps the most important characteristic of an ABM is the existence of emergent behavior. We have already noted that ABM's are comprised of agents that can interact with each other. Emergent behavior is when many (even simple) interactions take place, often times complex patterns or behaviors not explicitly programmed into the agents appear in the model as a result of the interactions and resource usage. This is considered one of ABM's greatest strengths because it helps us understand complex system characteristics that are often indistinguishable using other mathematical methods. "This ability to possess alternate behaviors enables the agents to adapt over time beyond their initial state and may result in unexpected system behavior" (Price, 2003). Because of this phenomenon, the agent's eventual behavior may not have even initially been thought feasible by the programmer or analyst.

So far emergent behavior has been made possible because of three things; agent interaction, environment interaction, and resource interaction. It is important to

understand that not all of these need to be present in order for emergent behavior to exist. Any one of these three interactions can lead to emergent behavior.

Overview and History

In the late 1940s, mathematician John von Neumann conceptualized a machine that could in effect "clone" itself. The idea was that this single machine could be programmed to construct another machine identical to itself. Upon completion, the machine would give the newly created machine the same set of instructions that it had used to create it, enabling the new machine to create a replica of itself also. These were called Von Neumann machines, and although impossible to physically create, von Neumann drafted the concept down as a grid in which one cell had the capability to "live" and behave according to its instructions (i.e. reproducing in adjacent cells). The resulting device became known as cellular automata. (Boccara, 2003)

As interest in cellular automata grew, people began considering what other complex system behaviors this kind of simple simulation could represent. The interest eventually lead Craig Reynolds to examine emerging behavior practices in a model that represented the flocking of birds (he called them "boids"). This model was the first to make use of actual individual agents. The model was expected to demonstrate that the boids would eventually flock together, but what was surprising was the resulting movement of the flock of boids. In this way, Reynold's model is considered the first to demonstrate emergent behavior. (Reynolds, 1987)

Applications of Agent-based Models

Agent-based Modeling has been used across several fields of study, as it is flexible enough to represent many different kinds of systems. Some of these fields include military application, social sciences, biological sciences, oceanography, and traffic management. The fact that an agent can be modeled in so many ways makes it useful in different ways to each of these fields. This review of some applications of ABM will consist primarily of military applications, since that is the field in which we will be conducting our analysis. (Sanders, 2003)

System Effectiveness and Analysis Simulation

System Effectiveness and Analysis Simulation (SEAS) is a government owned ABM developed by SPARTA, Inc and will be used as the primary tool in this study. SEAS is powerful in that joint war fighting scenarios can be represented on a variety of different scales to evaluate the effectiveness of various system designs, architectures, and concept of operations (CONOPS) (SPARTA, 2005).

Figure 2.1 shows how many of the agents interact with each other and what kind of objects agents can use to effect their environments.

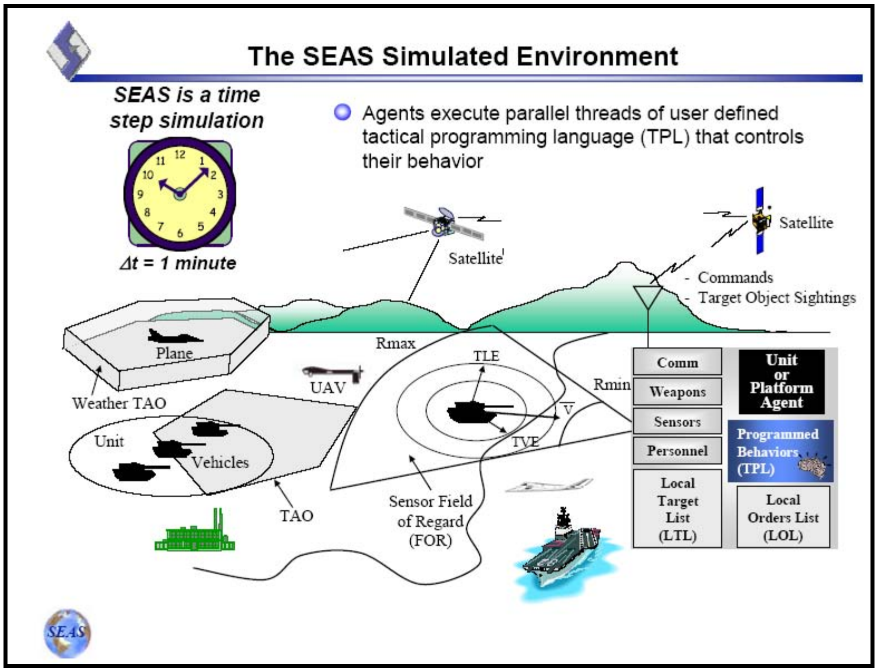

Figure 2.1 SEAS Simulated Environment

There are several reasons that SEAS will be used for our analysis. First, it is an agent based model and has the capability to model agent behaviors that are accurate enough for the systems being studied. Secondly, SEAS has a user-friendly post-processor that can be used to quantify the measures of performance demonstrated in our model. Thirdly, SEAS is an accepted model included in the Air Force's Standard Analysis Toolkit. This means it has been verified and validated as a model deemed acceptable for Air Force use.

Benefits and Limitations of Agent-based Models

One of the major benefits of using agent-based models is that they cost very little to create and execute. When performing analysis on major systems (especially combat scenarios), physically representing the pieces of the system of interest can be difficult and

costly. Exercises or wargames both require sufficient equipment and manpower to represent the system. Simulations, on the other hand, are 100% constructed, meaning that all manpower and equipment are virtually represented by agents and objects within the model.

Another benefit is that often times the model is easily modified for multiple case runs. In this way, a model can be tested several times for accurate representation of the system before final runs are executed.

Another benefit of ABM is the capability to represent emergent behavior. Emergent behavior is patterns or behaviors that aren't explicitly modeled in the agents that make up the scenario. Often times, these emergent behaviors represent system characteristics that would otherwise be missed by other analysis techniques.

One of the major criticisms of ABM is that although a programmer can design an agent to behave in any kind of manner that they wish, they can not program an agent to "think" on its own. Therefore the analyst must be extremely robust in how they model behaviors as to ensure that important interactions are simulated realistically. Modeling only those interactions that are important is a method of abstraction. Abstraction is defined as the process of reducing the details of a concept to only those important to a particular purpose.

Although it is important to choose a tool that is designed to answer the kind of problem that the analyst is facing, agent-based modeling is becoming more and more popular as a robust tool that can answer a variety of problems. As agent-based models become more powerful and more robust, ABMs will continue to become more popular. (SPARTA Inc., 2005)

Previous Research

Since we have an idea of what tool we will be using for our research, we will first examine previous studies using this tool (SEAS). USAF Captain Jeffrey Rucker's thesis involved a SEAS scenario with Satellites queuing UAS aircraft to provide constant surveillance until F16s could be queued to destroy them. Special attention is paid to how targets are generated. Rucker's sponsor specified that targets were to be able to be generated by 2 separate methods (the particular method to be specified before the runs); one being that targets were randomly generated, and another that targets were read and generated from locations listed on an external txt file. Rucker found a way to implement both, using SEAS' programming language and flags for which particular method to use. However, it was determined that using random generation for all runs would introduce too much variance into the data, and using targets generated from a file would introduce correlation in the runs and questions on randomness. Rucker's solution was the use of SEAS' TPL (Tactical Programming Language) to specify that on the first run, targets should be randomly generated within the TAO (Tactical Area of Operation) which means redrawing a random location when one is generated outside the TAO, then written to a file and re-used for the remainder of the runs. This reduces the variance between runs, and ensures randomness in the generated locations of the targets. (Rucker, 2006)

There have been many papers, journal articles, and thesis's published concerning both agent-based models and complex cooperative UAS behavior programming. When addressing this subject, there are a couple topics necessary for discussion. One of these is swarm intelligence.

Swarm intelligence is a behavior characteristic that is expected to be observed in our research. Swarm Intelligence can be defined as “the study of collective intelligence, originally exhibited by large groups (or *swarms*) of animals, usually insects such as ants and bees.” (Karim, 2004). In swarms of non-living agents, the individual members of the swarm are unaware of their position in the swarm or that the macro-behavior is even happening at all. The group’s behavior is not directly coded in the individual agents’ often simplistic behavioral code. However, this is not to say that the agents can not be programmed to be aware of certain environmental conditions that may influence their behavior within the swarm. The cognitive level of the agents is often limited only to the time constraints and patience of the programmer. As a result, emergent behavior can be taken advantage of by the individual agents (if their behaviors are detailed enough) to perform their tasks better and faster.

Ryan, Zennaro, Howell, Sengupta, and Hendrick write a comprehensive overview of emerging results in cooperative UAS control. They note that cooperative behavior in UAS vehicles allows for more agents to be controlled by a single user. They also note that cooperative behavior is beneficial in areas such as collision and obstacle avoidance. Obstacle avoidance is done in one of two ways; either each agent reports its location and path discrepancies are de-conflicted by the controller, or in the case of autonomous UAS vehicles, an object is detected and behaviors are implemented to avoid collision. In each case, some kind of forward detection is beneficial, especially in areas where flight-plans may include low-level canyon navigation, where there isn’t a second object with similar programmed behaviors. (Ryan et al., 2004)

Ryan, Zennaro, Howell, Sengupta, and Hendrick also discuss communication issues, both in air-to-ground cases and in air-to-air cases. In air-to-ground cases, the coverage area is generally limited by line-of-sight, long-range communications. Aircraft-to-aircraft communications run into problems in that it has been difficult to find an omni-directional antenna with lower power requirements and high bandwidth capabilities. (Ryan et al., 2004)

One of the cases that Ryan, Zennaro, Howell, Sengupta, and Hendrick mention deals with two different ways of modeling flocking formation flying. In the first case, Murray and Olfati-Saber use graph theory as a method to simulate a group of cooperating agents. When these agents encounter an object, a structural net represents the interaction between all agents, behaving to ensure obstacle avoidance. In the second case, a genetic algorithm is used to evaluate parameters for neighbors, obstacles, and the target. Then the two best designs are evaluated in order to avoid degradation of performance in the next generation. (Ryan et al., 2004)

Data Fusion is a term used to reference "… a multilevel, multifaceted process dealing with the automatic detection, association, correlation, estimation, and combination of data and information from multiple sources." (U.S. DoD, 1991) Advantages of data fusion include improving signal-to-noise ratio in order to improve detection accuracy and to acquire more data than is obtainable from a single sensor. In our case, this means using the information acquired by multiple agents and multiple sensors in an automated algorithm for sorting and classifying the targets. We will have data being shared between all local UAS vehicles in the scenario. Then an algorithm will be used to determine the optimal queuing of the UAS vehicles based upon the data fusion

results. There are also different levels at which data fusion can be done. We will model it as an automated process of decision-level fusion. Decision level fusion is the process in which data is shared regarding the confidence that each system maintains regarding a target's identification (as opposed to raw sensor data fusion or feature level fusion).

There are several methods for data fusion. Four methods discussed in Dongseob, Shin, Shim and Hung's paper are Bayes Theory, Dempster-Shafer Theory, Heuristical methods, and Fuzzy Set Theory. According to the authors, it was determined that Fuzzy Set Theory worked the best in correctly identifying airborne targets. Their data fusion involved two stages. The first stage was identifying whether the aircraft was a friend or foe. To determine this, they used known flight scheduling data and classified based on whether or not the aircraft had a schedule or not. The second stage was to identify the particular aircraft. To do this, they used data on the aircraft's speed, size, and whether or not it made a sudden appearance. Although they found Fuzzy Theory to be the best in identifying the correct type of aircraft, they noted that the next step was to use the outputs from each method and the best assignment of values for each method in order to make them more accurate. (Dongseob et al., 2005)

Godfrey, Cunningham, and Knutsen do a study on the negotiation mechanisms for autonomous, distributed coordination of surveillance tasks. In their research, they use several methods of target swapping for UAS vehicles. These methods include varying the algorithm for swapping (greedy vs. cooperative) and varying the amount of information shared by the agents. First, Godfrey, Cunningham, and Knutsen examine the Greedy Algorithm. This dictates that a proposing UAS goes through his list and picks a target, evaluates the benefit to the proposing UAS for a target swap with each target on

every other local UAS, then sorts the swaps in order of benefit to the proposing UAS. After a sorted list is generated, the proposing UAS goes down the list, proposing each swap with the other UAS vehicles in the scenario. Each UAS receiving a proposal evaluates whether the swap is beneficial to them. If it is, that UAS accepts the swap and the proposing UAS stops processing its list. This is a "greedy" algorithm because both the proposing UAS and the receiving UAS have to benefit in order for there to be a swap. Because of this, swaps that are more beneficial to the overall system may be rejected for a less beneficial swap. Also, this algorithm requires a large amount of processing (since proposing UAS vehicles seek acceptance from every other UAS in the scenario). Lastly, this is a one-to-one swapping method (meaning a UAS target list doesn't grow or shrink over time). (Godfrey et al., 2005)

The other three methods mentioned by Godfrey, Cunningham, and Knutsen are considered "cooperative target swapping" in that the value of a swap is judged on the benefit to the system as opposed to being evaluated on each end of the swap. Because of this, a UAS may accept a swap that is not beneficial to it, because an overall benefit to the system exists.

The first cooperative target swapping method is the Cooperative Even strategy. This method is a minor alteration of the Greedy Algorithm, except for the receiving UAS decides to accept or reject based on the benefit to the system instead of the receiving UAS.

The second cooperative target swapping method is the Basic Push strategy. This strategy dictates that the proposing UAS tries to push targets that are furthest from it to the UAS that is closest to the target. This strategy is not one-to-one; the pushing UAS

requires nothing in return for its push. Potential receiving UAS vehicles accept or reject the new target based on benefit to the system.

The third cooperative target swapping method is the Advance Pull strategy. In this method, a receiving UAS vehicle proposes to take another UAS vehicle's target based on calculated system benefit. This is another method that is not one-to-one; the pulling UAS isn't required to sacrifice one of its targets for the newly acquired one. This method is considered "advanced" because the pulling UAS calculates beforehand whether or not the target's UAS would accept or reject the donation based marginal benefits before it decides to propose the transfer.

Godfrey, Cunningham, and Knutsen end their study by concluding that local agent optimization can result in satisfactory results from UAS vehicles. They also show that when UAS vehicles are cognitive of system goals (and not just local goals) and share a greater amount of information, they can quickly converge to a much more efficient system solution. They note that "…cooperative strategies that allow uneven swaps enable UAS vehicles to exploit target clustering to further improve the system solution, to preserve solution quality as the number of targets increases, and to adapt quickly to changes in the number of UAS vehicles and targets in the environment."

Conclusion

As seen through reviewing the previous work on the subjects of both ABM and UAS, there has been much progress in both areas. In regards to ABM, we plan to examine aspects that are new to the latest version of SEAS. In regards to UAS, our research will go down a different path in relation to those described above. Our research

will be focused on the identification accuracy vs. time trade-off compared to swarming vs. independent UAS configurations. Because we explore a new area in both agent-based modeling using SEAS and ways to examine swarming configurations, our research is valuable to both subjects.

III. Methodology

Introduction

In this section, we go through the steps in achieving a complete scenario capable of providing the results we require using the SEAS agent based model. Explaining the code and the logic behind each agent will provide a basic understanding of how the model serves as an abstraction of the real world. We will begin by discussing some background information on the scenario, followed by some discussion on our chosen Measures of Performance (MOP), verification, validation, and ending with some pseudo-code for some of the more complicated parts of the scenario.

A "scenario" is a theoretical description of circumstances including who, what, when, where, why, and how. We will use the agent-based model SEAS to represent a specific scenario in a combat model. We create the modeled representation of the scenario using SEAS-specific programming code referred to as tactical programming language (TPL). This code defines all aspects of the scenario and tells the model how to perform. The file containing the TPL is referred to as the warfile. SEAS reads in the warfile (and any associated Include files specified by the warfile) to determine the model. When completed, our warfile will be used to determine the tradeoff of two single UAS vehicles vs. four less-capable swarming UAS vehicles for a specific scenario of interest in terms of both accuracy and identification time.

Scenario Background

In 2006, Captain Jeffrey Rucker completed a thesis sponsored by Air Combat Command (ACC) in which SEAS was used to model the significance of several pieces of a communication chain between area satellites and theater UAS vehicles. His scenario involved agents associated with either a red side or a blue side. Red SCUD agents in the scenario would periodically come out of hiding, move within Iraq, and eventually attempt to fire a missile at the blue side's AOC. Blue satellites would detect these red SCUDs, send their target information to the blue side's UAS vehicles in the area and the ground station. The blue UAS vehicles would fly to the detected target, and watch it until the blue ground station could queue a blue aircraft in the appropriate area to destroy the target. Capt Rucker ran a series of cases in which different agents were left out of the scenario. Capt Rucker used the following as MOPs:

1. The number of initial target detections, by any sensor, within two hours of the first target detection.
2. The proportion of targets destroyed before being able to fire.
3. The proportion of targets destroyed by the end of the 24-hour period of operations.
4. Which Blue Force agent detected which target and at what time did the detection occur?

Captain Rucker's design of experiments is shown below.

Table 3.1 Captain Rucker's Design of Experiments

	Baseline Case (All Agents)	Test Case 1 (No *F-16C*s)	Test Case 2 (No UAVs)	Test Case 3 (No Satellites)
MOP 1	X	X	X	X
MOP 2	X		X	X
MOP 3	X		X	X
MOP 4	X		X	

In his thesis, Captain Rucker describes in detail both how he constructed his scenario and where he obtained TPL code that he modified to fit his scenario. This includes sample warfiles included in the distribution of SEAS and prior work done by AFRL. Specifically, the UAS vehicles in Captain Rucker's thesis were modified from a previous AFRL study. We used Captain Rucker's scenario as a basis in constructing the warfile for this study.

Measures of Performance

In this section we discuss our plan to establish quantifiable values associated with certain aspects of our scenario. The aspects of our scenario that we wish to evaluate must be decided upon before the completion of the warfile to ensure appropriate data collection. The quantifiable values we are discussing are referred to as Measures of Performance (MOP). MOPs are specific and quantifiable. This is important because they are typically used in an aggregated form to support Measures of Effectiveness (MOE) and answer higher-level critical issues.

Since our scenario involves UAS vehicle configurations and their effect on the identification of targets, we should ensure that we have a sufficient understanding of the

identification process. Taylor (2000) describes the target identification process in four distinct steps.

1. **detection** (target detected at such a level of resolution/discrimination that observer can distinguish an object of military interest that is foreign to the background in its field of view, e.g. distinguish a vehicle from a bush)

2. **aimpoint** (target detected at such a level of resolution/discrimination that observer can distinguish an object by its class, e.g. a tracked vehicle versus a helicopter or a wheeled vehicle; observer can thus establish an aimpoint on the object)

3. **recognition** (observer can categorize targets discriminated at aimpoint within a given class, e.g. recognize a tank versus an armored personnel carrier in the tracked vehicle class)

4. **identification** (observer can distinguish between specific recognized target models, e.g. a T-72 tank versus a T-80) (Taylor, 2000: 929)

These phases are distinguished by how well you can discriminate between the target and non-targets. In our abstraction, the identification is a much more simplified process. Since we are trying to minimize the amount of stochastic variables, the time of detection is equivalent to the time of identification (as soon as an object is seen by the sensor, an identification decision is made).

For our scenario, the main interest was the trade-off between identification time and accuracy. To evaluate accuracy, we constructed several MOPs to help us quantify the worth of different sensor configurations. Specifically, how much accuracy we lose/gain when we have identifications being made based on single UAS vehicle data vs. having a swarm of four less-capable UAS vehicles.

One of the new upgrades to SEAS v3.7.1 is the Probability of Identification (PID) listing. The PID listing in the TPL specifies to what probability a sensor mistakes a

target that it has already sensed for something else. It's important to distinguish between a Probability of Detection (PD) list and a PID list. Below is an example of a PD list.

Figure 3.2 Example PD Table

```
Pd_Table
     Entry   "B_Sat_Sensor"      "SCUD_TEL1"    1.00
     Entry   "UAV_SAR1"          "SCUD_TEL1"    1.00
     Entry   "UAV_SAR2"          "SCUD_TEL1"    1.00
     Entry   "UAV_SAR3"          "SCUD_TEL1"    1.00
     Entry   "UAV_SAR4"          "SCUD_TEL1"    1.00
     Entry   "UAV_SAR5"          "SCUD_TEL1"    1.00
```

A PD table defines the probabilities for a specific sensor against a specific target. In the above examples, the sensor named "B_Sat_Sensor" detects an agent named "SCUD_TEL1" 100% of the time (as does "UAS vehicle_SAR1", "UAS vehicle_SAR2", "UAS vehicle_SAR3", etc). As defined here, these sensors act as definite range law devices. A definite range law device is one that detects with 100% certainty any object of interest inside the range of its sensor, but cannot detect any object outside of this range.

One reason for defining these sensors as range law devices in our scenario is to narrow the stochastic elements down to the identification process. Our study is focused on time vs. identification, and more random elements would add variance to our results. In a real-world study, this may be desired as real world sensors are not range law devices. However, for the sake of our study, and to narrow the variation outside of the areas of interest, our sensors will be defined as range law devices.

Identification probability is very different from detection probability. As above, we have set our sensor/target parameters as such to ensure 100% detection as long as the target is within the range distance. However, we wish to specify specific values for identification probabilities. A new upgrade in SEAS v3.7.1 allows us to do this. Below is an example of a PID listing from a warfile.

Figure 3.3 Example PID Table

```
PId "UAV_SAR3" "Decoy_TEL"
      Entry "Decoy_TEL"   .80
      Entry "Friend_TEL"  .05
      Entry "SCUD_TEL1"   .05
      Entry "SCUD_TEL2"   .05
      Entry "SCUD_TEL3"   .05
End
PId "UAV_SAR3" "Friend_TEL"
      Entry "Decoy_TEL"   .025
      Entry "Friend_TEL"  .90
      Entry "SCUD_TEL1"   .025
      Entry "SCUD_TEL2"   .025
      Entry "SCUD_TEL3"   .025
End
PId "UAV_SAR3" "SCUD_TEL1"
      Entry "Decoy_TEL"   .15
      Entry "Friend_TEL"  .15
      Entry "SCUD_TEL1"   .65
      Entry "SCUD_TEL2"   .025
      Entry "SCUD_TEL3"   .025
End
```

The TPL list above shows that for the sensor “UAV_SAR3” targeting a target named “Decoy_TEL”, there is an 80% chance that it identifies it correctly (i.e. as a “Decoy_TEL”), a 5% chance that it identifies it as a “Friend_TEL”, a 5% chance that it identifies it as a “SCUD_TEL1”, etc. Because all of our PD’s are set to 1.0 (i.e. 100%), the stochastic elements of our model are narrowed down to only include identification error. Because our study involves using identification as an MOP, this new feature is something we will use extensively in our study.

One of the MOP's we are interested in is time. In a scenario such as the one we're working with (with time-sensitive targets), it is important to process (i.e. identify) targets as quickly as possible. Therefore we wish to create MOPs that quantify the time aspect of each case. For our scenario, we will create 6 time-related MOPs. The first three time-related MOPs describe how long it takes for the UAS vehicles to make a identification decision on a target from the time the target deployed (became visible) in the scenario. We will evaluate the minimum time, the average time, and the maximum time for these targets.

We also wish to evaluate the time to make identification decisions on all targets in the scenario. Many of our cases have no variation in the targets' deployment direction or location, making scenario completion time a useful MOP. We will evaluate the minimum, average, and maximum times to process all targets in the scenario for each case.

The other MOP is the accuracy of target identification. We described earlier how SEAS allows us to establish values for the probability of identification. Our MOP will evaluate the identification output to both compare with the values we specified and to evaluate the trade-off with time. This MOP categorizes the identification decisions made by target and assesses what percent were correct vs. incorrect vs. undetermined. The threshold for identification decision in the second case is at least two of the four identifications are the same. Because we are dictating the probability of correct vs. incorrect identification in our warfile, the statistics for the output should be close to the values we specify. Furthermore, analysis using the input values is easily done using a

Binomial distribution. However, we are also interested in comparing how the simulation performs given the input probabilities.

Now that we've established some MOPs, we will discuss about how we extract these from the combat model. SEAS has built in output file formats that the user can enable to obtain reports on such things as detections, communication, and victims. However, for the MOPs we are interested in, the data provided by the built-in output file generation capability are insufficient. Fortunately, one of the benefits of SEAS is the capability to write TPL code that dumps variables of interest into a text file. An example of TPL code that takes advantage of this capability is shown below.

```
Open, Write "debug.joe"
Write "debug.joe" j + 1 ", " me->Name ", " me->dec_data[j][2] - depdelay
```

Figure 3.4 Example Code to Write to Custom File

The first line of the above code creates an empty file named "debug.joe." The second line writes a string of desired variables separated by commas. The purpose of separating the data using commas is to make post-processing (using Microsoft Excel) easier. Because SEAS provides us with this capability, we can use SEAS to custom write files containing the data we are interested in while the scenario runs.

Cases

Because we removed so much of the variance in our scenario, we had to decide what aspects of the scenario we wished to change. The obvious variable was the UAS vehicle configuration. We had two options for vehicle configuration. First, was a swarm

of four UAS vehicles that flew together and tracked a single target simultaneously. No member of the swarm could track another target until every member of the swarm had made an identification decision on the current target. The other option was that a pair of UAS vehicles could fly independently (but still deconflicting if they happened to track the same target).

Our second variable was whether the targets were consistent in their deployment locations and movements across all runs for a single case. One option was to choose random locations and movements at the beginning of each set of runs, and use that recorded data for every following run. This would be beneficial in reducing variance across runs, but also establishes a risk of introducing a bias in the runs if the locations happened to be chosen in locations that would benefit one configuration over another. Therefore, we will run both options and compare to see if the constant cases are biased.

Our third variable is the size of the theater. One option is to evaluate the northern 1/3 of Iraq. The other option is to evaluate the northern 2/3 of Iraq. The two TAO sizes are shown below.

Figure 3.5.1 Small TAO

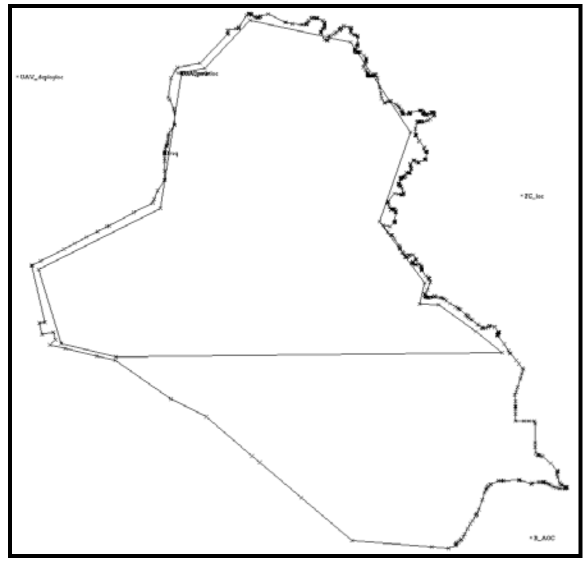

Figure 3.5.2 Large TAO

Our fourth variable was to vary the number of targets that deploy (15 vs. 30). This was one way to determine how both the number of targets and the size of the area affect scenario processing times. The complete list of the cases we ran is shown below.

Table 3.6 List of Cases

CASE	Swarm/Independent Configuration	Random/Consistent Target Locations	Small/Large TAO	Number of Targets
1	S	R	S	15
2	S	R	L	30
3	S	R	L	15
4	S	C	S	15
5	S	C	L	30
6	I	R	S	15
7	I	R	L	30
8	I	R	L	15
9	I	C	S	15
10	I	C	L	30
11	S	R	S	30
12	I	R	S	30

From this point forward, we will use a notation with a single letter or number per option to represent a particular case. For example, case 1 would be referred to as SRS15. The first letter represents whether the case is for a swarm or individual UAS configuration (S or I). The second letter represents whether the case represents one where deployment locations were held constant for every run or whether it was randomly chosen at the beginning of every run (C or R). The third letter represents whether we are using a small or large theater (S or L). The fourth and fifth characters (number) represents how many targets are deployed for that particular case (15 or 30).

Verification

Verification means ensuring the model was built to the specifications expected by the author (i.e. did we build the model correctly). This step was accomplished through rigorous debugging/testing to ensure that what was happening within the model was what we expected to happen. The SEAS software provides an extensive help file that was referenced many times in the design of the model to ensure that code was written correctly. Often times an error would be displayed when syntax was incorrectly used, making verification easier. In addition to the help file, the previously-mentioned SEAS capability to custom-write variables to an external file was crucial in the verification process of the model. By writing variable values to an output file, the model has a transparent characteristic whereas the user can monitor why the agents are behaving as they are. The debug function was also helpful, especially when monitoring an agent's target list.

One way to verify that our model is performing the way we expect it to is to compare PID value input vs. identification matrix output. One of the cases we input in our model was one in which four separate UAS vehicles tracked objects, identified them as targets, and made identification decisions based upon their perception of the object. All of the cases we ran were examined to see if a significant difference existed between the expected identification matrix (based on PID input) and actual identification matrix. Below is one example of output where four separate UAS vehicles with probability of correct identification is .85 (and .0375 for each of the four remaining types of objects).

Table 3.7 Output Identification Matrix

		IDENTIFIED AS				
		DECOY	FRIEND	SCUD1	SCUD2	SCUD3
PERCENT	DECOY	0.77	0.04	0.10	0.07	0.02
	FRIEND	0.02	0.89	0.07	0.01	0.01
	SCUD1	0.04	0.03	0.81	0.03	0.08
	SCUD2	0.01	0.01	0.04	0.89	0.04
	SCUD3	0.06	0.08	0.02	0.02	0.82

The above shows probabilities that average an .836 probability of correct identification. If we examine the values per run, we can use a student's T test to evaluate whether this value is significantly different from the expected input value of .85. The formula for the T statistic is $t = \frac{\overline{X}_D - \mu_0}{s_D/\sqrt{N}}$. Using this formula, the resulting t statistic is .0483. We are basing our t statistic off of 30 values (runs), so our threshold value is determined using 29 degrees of freedom. Since $t_{10,29} = 1.70$, we fail to reject that the two values are significantly different.

Another method of verification is our case comparisons between cases where targets deploy independently each run vs. cases where the targets deploy from the same location and move in the same directions every run. If we determine that the two do not yield significantly different results, we can use the data from the consistent location cases since they have less inherent variance in them.

Validation

The process of validating a model involves ensuring that the model adequately imitates the real life scenario it is meant to represent. The purpose of validating a model is to ensure that the output is useful. If a model fails to adequately represent a real life system, the output is not sufficient for use in analyzing the real-life scenario. Banks (2004) describes two distinct purposes for the validation process:

> 1. To produce a model that represents true system behavior closely enough for the model to be used as a substitute for the actual system for the purpose of experimenting with the system, analyzing system behavior, and predicting system performance;
>
> 2. To increase to an acceptable level the credibility of the model, so that the model will be used by managers and other decision makers.

Since our study is more focused on the tradeoff between time and accuracy in UAS vehicles without naming specific platforms/sensors, ensuring that the parameters used for our systems are comparable enough to those found in operational systems is sufficient for the purpose of our study.

Model Design

Before we begin examining the different sections of the warfile code in detail, we will go over big-picture purposes of the model. In our combat model, UAS vehicles deploy from a location outside of IRAQ and fly to a pattern location. Once they arrive, they begin flying the perimeter of the theater of interest (northern Iraq). While the UAS vehicles are flying the perimeter of the theater, five different types of objects (decoy scuds, friendly objects, and three different types of scuds) deploy from random locations in the theater. Once they deploy, they move to different locations within the theater. These locations are determined randomly on the first run of the model, then that data is recorded and used for all subsequent runs. Nine satellites are also present in the scenario. When a satellite is over the theater and a target is in view, the satellite takes the target sighting and passes it to the UAS vehicles. The closest UAS vehicle takes the target sighting and moves to the target location. When the UAS vehicle gets to the target's location, it makes an identification decision on the target and removes the target from the scenario. When all 15 targets (3 of each kind) have deployed, been identified, and removed from the scenario, the current run ends and the next one begins. Some images depicting the identification process are shown below.

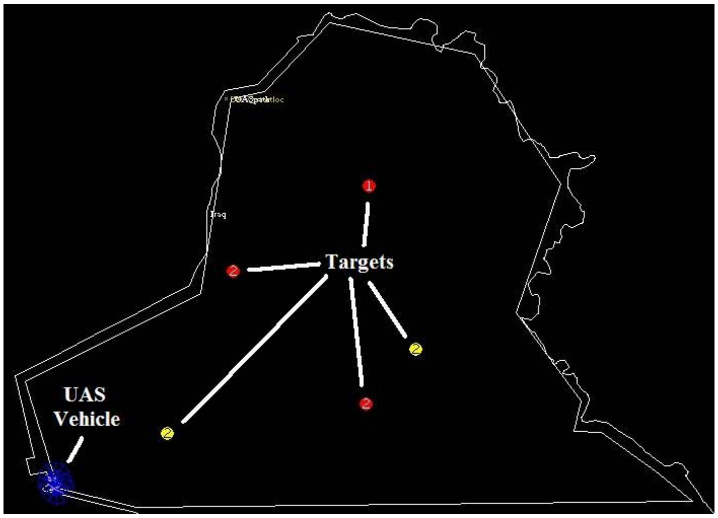

Figure 3.8.1 Targets in the Scenario, UAS Vehicles Unaware of Them

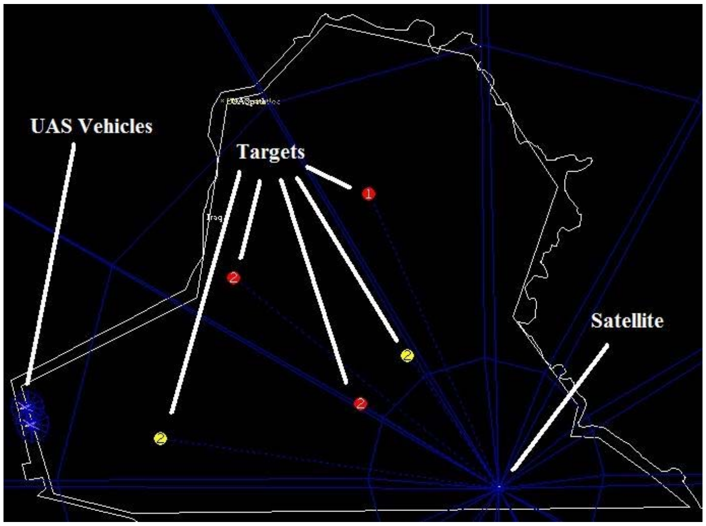

Figure 3.8.2 Satellite Flies Overhead and Detects Targets in Theater

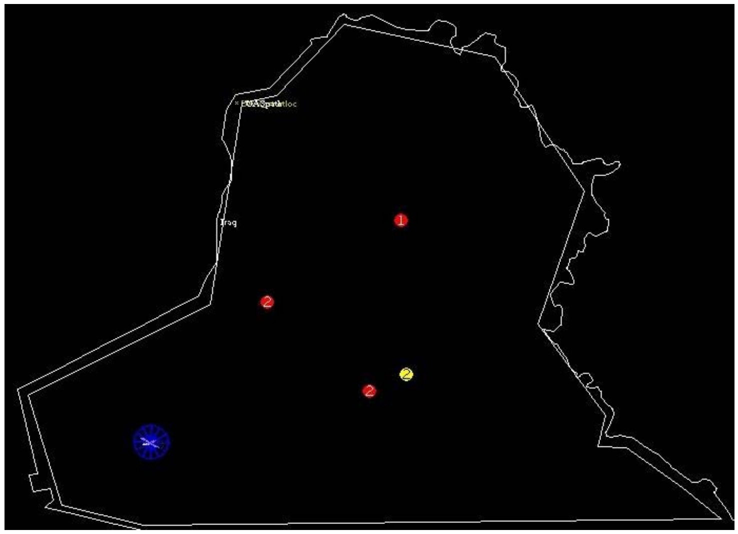

Figure 3.8.3 UAS Vehicles Fly to, Identify, and Remove Closest Target

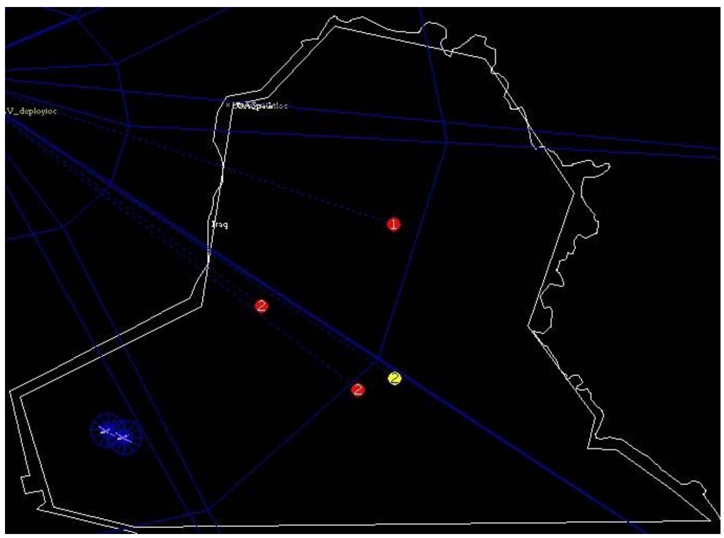

Figure 3.8.4 UAS Vehicles Return to Path, Awaiting the Next Target

The two cases for our study differ in the configuration/capabilities of the UAS vehicles. In the first case, two UAS vehicles circle the perimeter of the theater. When a potential target shows up on a UAS vehicle's Local Target List (LTL), it checks if the other UAS vehicle is tracking the object. If the object is already being tracked by the other UAS vehicle, the first UAS vehicle goes to the next target in its LTL. If no non-

targeted agent of interest is found in the area, it continues circling the perimeter of the theater. If it finds an agent of interest, it flies to the agent's location, makes an identification decision, and removes the target from the scenario. Once it removes the agent from the scenario, the UAS checks for any other potential targets in the area, and appropriately either returns to the flight path or moves to the next target.

The second case differs in that four UAS vehicles stick together in formation to represent swarming behavior. They perform as one UAS vehicle would in the first case, but when they get to a target, four independent identification decisions are made. This is also different in that the capabilities (i.e. correct PID values) of these UAS vehicles range in values.

To determine capabilities, we'll use a probability equation to determine an arbitrary set of grouped UAS vehicle capabilities. Once we determine the overall probability of a successful identification for any given target, we can set our PID of the individual UAS vehicles in case 1 to that probability and we should get comparable results. This will give us a starting point for our cases.

Setting the probabilities for correct identification of a given target type by the swarming UAS vehicles arbitrarily at {.62, .62, .62, .62} yields ~83% probability that at least two of the four UAS vehicles will correctly identify the object.

$$P(\text{Correct ID}) = (.62)(.62)(.62)(.62) + [4 * (.62)(.62)(.62)(1-.62)] + [6 * (.62)(.62)(1-.62)(1-.62)] = {\sim}.8431$$

However, we also have to account for the possibility that two UAS vehicles correctly identify the object, and the other two identify it as the same incorrect thing. The possibility of that happening is

$$P(\text{Undetermined}) = 4 * [(.62)(.62)(1-.62)(1-.62)]$$

$$= \sim.0139$$

Therefore, the probability of a correct identification without confusion (i.e. the other two or less identifications are not the same) is given by the equation below.

$$P(\text{Confirmed ID}) = \sim.8431 - .0139$$

$$= \sim.8292$$

Therefore, setting the independent probabilities of successful identification for each of the UAS vehicles in case 1 to .83 should yield comparable results in terms of identification. Running the above-described cases supports this. For our research, we will set each UAS vehicle in the swarm to have a probability of correct identification equal to .62. However, for the independent UAS vehicle cases, each UAS vehicle will have a probability of correct identification value of .75. We base this on an assumption that because there are fewer vehicles in the independent cases (two vs. four), it is rational to compare two more capable independent UAS vehicles to four less capable swarming UAS vehicles.

Next we will examine the logic for the UAS agents. Below is a shortened version of the orders for the UAS vehicles for case 1.

```
Orders
     Deploy "FC_loc"
     While me->Status == 2
         Locate m_enemy, l_enemy, -400, namescuds, mytarg
         While m_enemy > 0
            If mytarg->Target->chaser == "noone"
                 mytarg->Target->chaser = me->Fullname
            EndIf
            If mytarg->Target->chaser != me->Fullname
                 i = 0
                 numtarg = lock(1)
                 While i < numtarg & mytarg->Target->chaser != "noone"
                      mytarg = gettarget(i)
                      i = i + 1
                 EndWhile
                 If mytarg
                      If mytarg->Target->chaser != "noone"
                           Break
                      EndIf
                  Else
                      Break
                  EndIf
            EndIf
            If distance(me->Location, mytarg->Target->Location) >= 20
                 If mytarg->Target->chaser == me->Fullname
                      SubMove mytarg->Location
                 Else
                      mytarg->Target->chaser = "noone"
                 EndIf
            EndIf
            If mytarg->Target->Status != 0 & distance(me->Location, mytarg->Target->Location) < 20
                 If mytarg->Target->chaser == me->Fullname
                      mytarg->Target->dec_data[mytarg->Target->dec_index][0] = me->Fullname
                 EndIf
            Else
                 If distance(me->Location, mytarg->Target->Location) >= 20
                      mytarg->Target->chaser = "noone"
                      mytarg = 0
                 EndIf
            EndIf
         EndWhile 1
     EndWhile
EndOrders
```

Figure 3.9 Case 1: Orders for UAS vehicles

The above code has three main sections within its code. The line that states "While me->Status == 2" is the main loop of the code. Status is a parameter describing whether the agent is alive or dead. If an agent's status is 2, that agent is alive in the model. All code within the loop described above executes as long as the agent has not been killed. The three main sections within the "alive" code can be described as finding a target, moving to the target, and identifying/removing the target.

The “finding a target” part of the code within the UAS vehicle “alive loop” uses the local sensor as well as information passed from the satellites to find the closest candidate target. If a candidate is found, the UAS vehicle evaluates whether the target has already been assigned to another UAS vehicle. If it has, the UAS vehicle cycles through all targets on its list until it finds a candidate target that has not been assigned to another UAS vehicle. Once an unassigned candidate target is found the next part of the code executes.

The “moving to the target” part of the code tells the UAS vehicle to move to the perceived location of the target. If a target was spotted by a satellite, but left the satellite’s field of view (FOV) before the UAS vehicle could get to the target, the perceived location may not be the same as the actual location. If the UAS vehicle gets to the perceived location of the target, but the target is not there, the target drops off the UAS vehicle’s target list after five minutes.

The “identifying/removing the target” part of the code specifies that if the UAS vehicle has reached the actual location of the target, it makes a determination of the target’s type, documents it, and removes the target from the scenario. It is possible that the UAS vehicle gets to the target and loses it. In this case, the UAS vehicle unassigns itself from the target (to make it a possible candidate for other UAS vehicles,), and documents that it has lost a target.

Below is the code for the UAS vehicles in case 2. Recall that in this case, the UAS vehicles stay in formation and simultaneously make identification decisions.

```
Orders
     Deploy "UAV_deployloc"
     If me->Ident != 1
          While me->Status == 2
              SubMove leader->Location
              Locate m_enemy, l_enemy, -30, namescuds, mytarg
              If mytarg
                   If distance(me->Location, mytarg->Target->Location) < 20 & mytarg->Target->IDd[me->Ident - 1] == 0
                        If mytarg->Target->IDd[me->Ident - 1] != 1
                           mytarg->Target->dec_data[mytarg->Target->dec_index][0] = me->Fullname
                           mytarg->Target->dec_data[mytarg->Target->dec_index][1] = mytarg->Name
                           mytarg->Target->dec_data[mytarg->Target->dec_index][2] = war->Time
                           mytarg->Target->dec_index = mytarg->Target->dec_index + 1
                           mytarg->Target->IDd[me->Ident - 1] = 1
                        EndIf
                   EndIf
              EndIf
          EndWhile
     ElseIf me->Ident == 1
          While me->Status == 2
              Locate m_enemy, l_enemy, -400, namescuds, mytarg
              While m_enemy > 0
                    If mytarg
                              If distance(me->Location, mytarg->Target->Location) >= 18
                                   SubMove mytarg->Location
                              EndIf
                    EndIf
                    If mytarg
                         While mytarg->Target->Status == 2
                         If distance(me->Location, mytarg->Target->Location) < 20 & mytarg->Target->IDd[me->Ident - 1] == 0
                                   mytarg->Target->dec_data[mytarg->Target->dec_index][0] = me->Fullname
                                    mytarg->Target->dec_data[mytarg->Target->dec_index][1] = mytarg->Name
                                    mytarg->Target->dec_data[mytarg->Target->dec_index][2] = war->Time
                                    mytarg->Target->dec_index = mytarg->Target->dec_index + 1
                                    mytarg->Target->IDd[me->Ident - 1] = 1
                         Else
                                   SubMove mytarg->Location
                         EndIf
                         EndWhile 1
                    Else
                    EndIf
              EndWhile
          EndWhile
     EndIf
EndOrders
```

Figure 3.10 Case 2: Orders for UAS vehicles

The above code can be examined in two separate parts. Each agent only executes one of the two parts. The first part of the code is defined as all code contained by the "If me->Ident != 1" statement. Each UAS vehicle in this case has a separate identification number (one through five). The first part is executed by three of the four UAS vehicle agents. This part of the code is pretty simple. Like the code described from case 1, the main part of this code is contained in a while loop. While the three UAS vehicle agents with an identification value not equal to 1 are alive, they follow the leader UAS vehicle. If they're at a target's location, they make an identification decision on the target.

The second part of the code is executed by only the UAS vehicle agent with an identification value equal to 1. It has two main sections. The first involves finding a target and moving to it. The second involves making an identification decision if the UAS vehicle is at the target's location. The second case doesn't require as much code as the first case because no de-confliction code is needed (since all UAS vehicles are flying in a swarm).

Next we will examine the code for the targets. There are five different types of targets, but they all execute the same code. The code for these agents is shown below.

```
Orders
     Deploy SCUDArray[me->Ident - 1][1]
     k = 2
     While me->Status == 2
         Move SCUDArray[me->Ident-1][k]
         k = k + 1
         While me->Location != me->Goal
               If me->dec_index == numneeded
                    previndex = me->dec_index
                    j = 0
                    dec = 1
                    While j < me->dec_index
                         If me->dec_data[j][1] == "SCUD_TEL1"
                              numtel1 = numtel1 + 1
                              numtotal = numtotal + 1
                         ElseIf me->dec_data[j][1] == "SCUD_TEL2"
                              numtel2 = numtel2 + 1
                              numtotal = numtotal + 1
                         ElseIf me->dec_data[j][1] == "SCUD_TEL3"
                              numtel3 = numtel3 + 1
                              numtotal = numtotal + 1
                         ElseIf me->dec_data[j][1] == "Decoy_TEL"
                              numdecoy = numdecoy + 1
                              numtotal = numtotal + 1
                         ElseIf me->dec_data[j][1] == "Friend_TEL"
                              numfriend = numfriend + 1
                              numtotal = numtotal + 1
                         EndIf
                         j = j + 1
                    EndWhile
                    If numtel1 / numtotal >= 1/2
                         numcor = numcor + 1
                    ElseIf numtel2 / numtotal >= 1/2
                         numcor = numcor + 1
                    ElseIf numtel3 / numtotal >= 1/2
                         numcor = numcor + 1
                    ElseIf numdecoy / numtotal >= 1/2
                         numcor = numcor + 1
                    ElseIf numfriend / numtotal >= 1/2
                         numcor = numcor + 1
                    EndIf
                    me->Status = 0
               EndIf
         EndWhile
     EndWhile
EndOrders
```

Figure 3.11 Orders for Potential Targets

The code above is very simple. The majority of it is keeping track of how many identifications have been made and documenting them. Within the "alive loop" there are two main parts. The first part executes when the agent has reached its goal. It looks for the next location and moves to it. The second part of the code is executed when a specified number of identifications for the target is reached (i.e. one for the first case, four for the second case). This part of the code tallies the number of identifications for each type. If 1/2 of the identifications are of the same type, the code documents that as a definite macro-level decision. For Case 1, only one identification is needed, so there is always a definite macro-level identification decision made. For Case 2, if two UAS vehicles identified the target as one thing, and two other UAS vehicles identified it as something else, the macro-level identification decision is "undetermined." The last part of the code writes the full array of identifications to a custom output file.

Some people may find the unconventional way of keeping track of identifications (i.e. having the targets keep track of them) controversial, but none of the processes are affected by this method and the coding is much more efficient. The only concern in creating the code was that it may take extra time to process the record-keeping portion. However, the extra processing time was minimal and the timelines were unaffected since after making an identification, a UAS vehicle disassociates itself with the target and moves on.

There are a few other pieces of code that are important to mention. The first is the way that the targets deploy. The unit "Location_Generator" is code that was borrowed

from Captain Rucker's thesis and modified to fit my own. The TPL code for this unit is shown below.

```
Orders
    If $REAL_TARGET_FILE == 0  & war->Iteration == 1          !! only do this on first Monte Carlo trial
        Open Write, "30TargetLocs.txt"                        !! open external text file to write to
        While i < ($NUM_REAL_TARGETS1 + $NUM_REAL_TARGETS2 + $NUM_REAL_TARGETS3 + $NUM_DECOY_TARGETS + $NUM_FRIEND_TARGETS)
            SCUDArray[i][0] = uniform() * 1000
            Write "30TargetLocs.txt", SCUDArray[i][0]
            k = 1
            While k < 10
                While Inside("UASpath", rnloc) != 1
                    !!rnlong = 43.5  !! central longitude
                    !!rnlat = 33     !! central latitude
                    !!rnloc = location(rnlong, rnlat, 0) + 500*uniform() * vector(360*uniform())
                    rnlong = 35.6 + 16.6*uniform()      !! theater lonmin = 35.6, lonmax = 52.2, delta = 16.6
                    rnlat = 28.8 + 8.9*uniform()        !! theater latmin = 28.8, latmax = 37.7, delta = 8.9
                    rnloc = location(rnlong, rnlat, 0) !!+ 500*uniform() * vector(360*uniform())
                EndWhile 1
                SCUDArray[i][k] = rnloc
                Write "30TargetLocs.txt", SCUDArray[i][k]
                k = k + 1
                rnloc = location(0,0,0)
            EndWhile
            i = i + 1
        EndWhile 1
        Close "30TargetLocs.txt"
    Else
        Open Read, "30TargetLocs.txt"     !! open target file
        j = 0
        While j < ($NUM_REAL_TARGETS1 + $NUM_REAL_TARGETS2 + $NUM_REAL_TARGETS3 + $NUM_DECOY_TARGETS + $NUM_FRIEND_TARGETS)
            Read "30TargetLocs.txt", SCUDArray[j][0]      !! read target locations from file
            Read "30TargetLocs.txt", SCUDArray[j][1]      !! read target locations from file
            Read "30TargetLocs.txt", SCUDArray[j][2]      !! read target locations from file
            Read "30TargetLocs.txt", SCUDArray[j][3]      !! read target locations from file
            Read "30TargetLocs.txt", SCUDArray[j][4]      !! read target locations from file
            Read "30TargetLocs.txt", SCUDArray[j][5]      !! read target locations from file
            Read "30TargetLocs.txt", SCUDArray[j][6]      !! read target locations from file
            Read "30TargetLocs.txt", SCUDArray[j][7]      !! read target locations from file
            Read "30TargetLocs.txt", SCUDArray[j][8]      !! read target locations from file
            Read "30TargetLocs.txt", SCUDArray[j][9]      !! read target locations from file
            j = j + 1
        EndWhile 1
        Close "30TargetLocs.txt"
    EndIf
EndOrders
```

Figure 3.12 Orders for Location Generating Unit

The above code creates an array on the first run (i.e. if war->iteration = 1). If the current run is the first run, the unit creates an array to document the randomly created deployment location and subsequent movements. In this way, we minimize variability on all aspects except for those we wish to examine (i.e. identification times). We also maintain an unbiased scenario by assigning all deployment locations and movements to match the first run which is randomly assigned. However, this makes the results absolutely specific to the scenario. If we allow the runs to mimic a single run that is

randomly assigned, there is the possibility that the single run may be an extreme point. If this happens, all runs will be based off of extreme points. Therefore, in presentation of results in the next section, we will be careful to caveat all results to the specific scenario we ran.

Another code-related point of interest is that all of our potential targets (SCUD types 1, 2, 3, decoy SCUDS, and friendly objects) are grouped under the same unit. This is done for the sake of efficiency. Whether we define the friendly objects as the child of a separate force or a red force makes no difference on how the combat model performs nor does it affect the output.

Lastly, it is important to distinguish our assumptions. In this scenario we have made a couple of assumptions that are important to clarify. The following are a list of our assumptions:

1. There is no "human in the loop" process time. When a UAS vehicle is closer than 20 meters, it immediately makes an identification decision. This assumption was established to minimize variability.
2. Because both deploy locations and subsequent movements are randomly chosen at the first iteration (and used for every other run), the results are particularly specific to the scenario ran.
3. ½ is an adequate threshold for macro-level identifications. In reality, the probability of correct identification should be taken into consideration (since the UAS vehicles have different probabilities).

4. Defining the friendly forces as a sub-unit of the red forces makes no difference in the scenario play nor the output. Therefore, it is ok (for the sake of efficiency) to put friendly forces under red command.

Conclusion

Most of the analysis occurs after the model is developed. However, without adequate model development, the integrity of the results are questionable. Special attention has been paid to this study to ensure that all behavior code and assumptions are rational. This chapter describes in detail the logic and TPL code behind the major players in our scenario, and thus is crucial in understanding our intentions.

The next chapter will describe our output and evaluate any discrepancies we find between the output and what we expect to find.

IV. Analysis and Results

Introduction

This chapter describes in detail the results of our analysis, including quantifiable data as well as qualitative scenario-specific output. We ran a number of cases and will now go through the measures of performance to compare cases of interest. We conclude the chapter with some insight concerning the output provided by the model.

Once our cases were determined, our first decision was how many runs would be sufficient to be confident in the integrity of the results we were getting. Our initial guess at an adequate number of runs was 30. We tested this hypothesis by running a scenario 30 times and examining the output for normality. We had several options for output values to do normality testing on (scenario processing time, target processing time, variation between input probability of correct identification and actual correct identification, etc). We chose to test average scenario processing time (i.e. the time it took from time 0 to the time when the last target of the scenario was processed for normality. However, because we had cases where targets deployed from different locations at different times depending on the run, we had to choose a case where the deployment locations and times were constant. Case number four seemed like a viable candidate (case four was a “swarming” case with consistent target deployment locations, a small TAO, and 15 targets). The output appeared approximately normal, leading us to conclude that 30 runs were sufficient for the purposes of our analysis. The absence of run completions in the second and fifth bin is explained by the fact that we have reduced so much variance in the scenario, that a few critical events ultimately determine the difference between the first or sixth bin and the third or fourth bin.

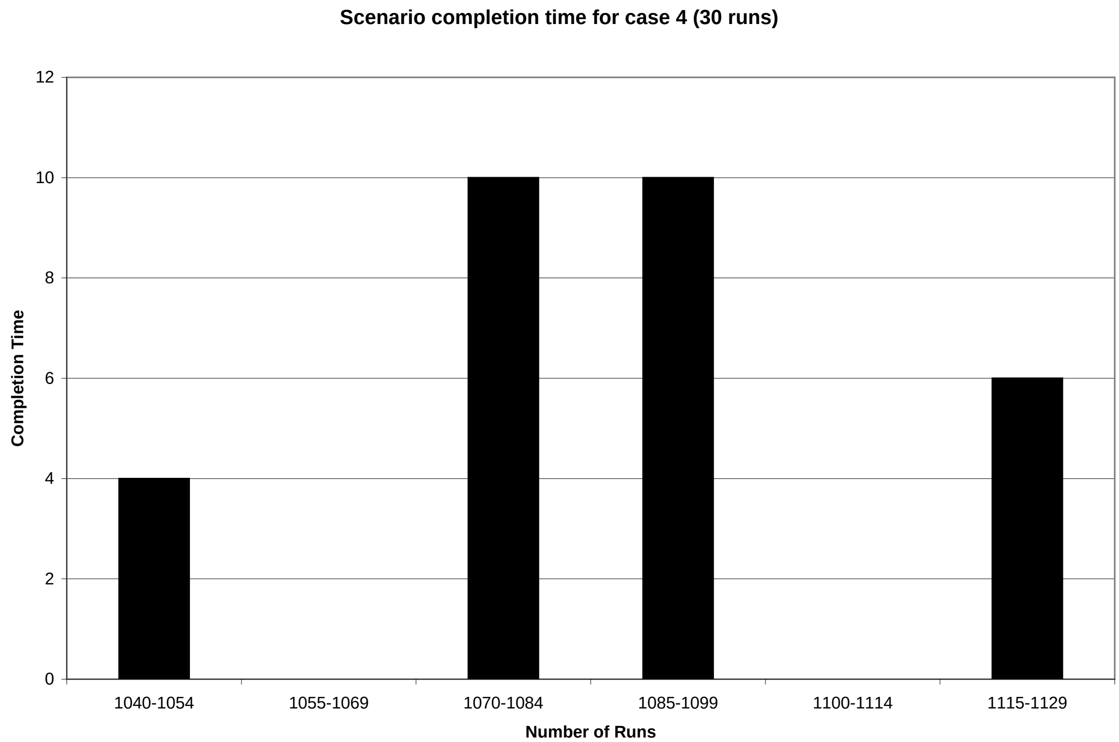

Figure 4.1 Distribution of Scenario Completion Time for Case 4 (SCS15)

Another important issue to consider once our cases were determined is how fast it would take to run the cases. Our scenario is both efficient and compact enough so that runs were completed quickly. Our runs were completed using SEAS 3.7.1 on a custom desktop running an AMD Athlon 64 X2 Dual Core Processor 6000+ CPU with 2GB of RAM.

Our simplest case, Case 4 with settings (SCS15 – this follows column designations for cases in Tables 3.6 and 4.2), took approx 3 minutes and 57 seconds with graphics for a set of 30 runs. However, for the sake of our analysis, graphics were only turned on during the development of our scenario. Once we had our cases set up the way

we wanted, the runs were processed without graphics (total run time for all 30 replications of approximately 10 seconds).

Our most complicated case, Case 7 with settings (IRL30), took approximately 4 minutes and 49 seconds with graphics for 30 replications. When graphics were turned off, it only took approximately 14 seconds for 30 replications.

Recall from the previous chapter that our run matrix is comprised of twelve separate cases and is as shown below.

Table 4.2 Case Matrix

CASE	Swarm/Independent Configuration	Random/Consistent Target Locations	Small/Large TAO	Number of Targets
1	S	R	S	15
2	S	R	L	30
3	S	R	L	15
4	S	C	S	15
5	S	C	L	30
6	I	R	S	15
7	I	R	L	30
8	I	R	L	15
9	I	C	S	15
10	I	C	L	30
11	S	R	S	30
12	I	R	S	30

There are four different options that make up this run matrix. The first is whether or not a group of UAS vehicles moves in a swarm (i.e. a 4-vehicle formation that simultaneously tracks the same target at any given point in the scenario) or follow individual behaviors (i.e. track targets not already tracked by other UAS vehicles in the scenario). Since we designed identification probabilities so that the swarming behavior would provide a more accurate identification decision, the trade-off is scenario

processing time. Because the independent pair of UAS vehicles can process separate targets at the same time, we expect to see that the cases where UAS vehicles can process targets independently process targets significantly faster than the cases where the UAS vehicles move in a swarm.

The second option is whether or not target deployment locations and subsequent movement directions are consistent for all runs in the set or if they are independently chosen each time a different run is initiated. The cases with constant target locations provide us with a better starting point for case analysis because the amount of variance between the runs is minimized. However, this can be bad if the initial locations are biased one way or the other. Therefore, runs are done with both to ensure that the values examined in the constant location cases are representative of values from the cases where each run is independent of the previous one.

The third option is how big the TAO is. Recall from Chapter III that we have cases where the area of interest (AOI) is only the northern third of Iraq. The other cases consist of a TAO roughly double the size (northern two-thirds of Iraq). The purpose in varying this option is to examine what kind of an effect area has on processing time. We expect to see processing time increase for the larger TAO since there is more area to cover. More importantly though, we expect to see the differences in processing times get larger when the area increases due to an increased amount of stress on the swarming cases where one formation of vehicles is responsible for the whole area.

The fourth option is the number of targets. One of the things we were worried about when setting up our scenario was that fifteen targets may not be enough to adequately stress the systems. Therefore, we determined that we would use the number

of targets as another (in addition to TAO size) variable to show the influence on processing times.

Overview of Analysis

The purpose of our study is to examine the relationship between identification accuracy and identification time when comparing independent vs. swarming behavior. One of the reasons we varied whether or not deployment locations and directions were constant through every replication or whether they were re-determined at the beginning of each replication was to establish that there was not a significant difference between the two values. Ideally, we would like to use the cases with a consistent deployment location for the reduced variance, but the cases with a random location and direction for each run is useful to compare times to ensure non-biased initial locations for the constant deployment location cases. The cases where deployment method is the only difference are the following pairs: {(1,4), (6,9), (7,10), (2,5)}. Case 1 and case 4 are both swarming cases with a smaller TAO and 15 targets. They differ in that case 1 uses consistent location/directions and case 4 uses random locations/directions for each target at the beginning of every replication. The following table summarizes the output for both cases.

Table 4.3 Comparison of Random vs. Consistent SXS15 Cases

Swarm/Independent	Random/Consistent	Small/Large TAO	Number of Targets	CASE	TARGET IDENTIFICATION TIME			SCENARIO TIME		
					MINIMUM	AVERAGE	MAXIMUM	MINIMUM	AVERAGE	MAXIMUM
S	R	S	15	1	7.91	115.89	660.38	837.00	1044.50	1286.00
S	C	S	15	4	4.86	109.40	392.81	1041.00	1087.50	1131.00

We can create confidence intervals for both average target times and scenario times on each pair of cases to test whether or not the values are significantly different. If the confidence interval contains the value zero, we can fail to conclude that the pair of cases are significantly different. To calculate these values we will use a paired t test. The formula for calculating a paired T statistic is shown below where d-bar is the average of the differences between each value.

$$t = \frac{\bar{d} - 0}{s_D/\sqrt{n}}$$

In our case, we have 30 differences because we have 30 runs. S_D is the standard deviation of the differences, and n is the number of runs

Using the above formula, the threshold for a 2-sided t statistic with an alpha of .10 is 1.70. We can calculate a 90% confidence interval using the formula below.

$$\overline{D} \pm t_{\alpha/2}\left(\frac{S_D}{\sqrt{n}}\right)$$

The results of the confidence interval calculations are shown below.

Table 4.4 Confidence Intervals for Random vs. Consistent SXS15 cases

		Target Times			Scenario Time		
	Case	Lower Bound	Upper Bound	Result	Lower Bound	Upper Bound	Result
S X S 15	1 vs. 4	-8.01540723	11.10875007	FTR	-64.4719813	-2.39468539	REJ

Our conclusion is that the times for target identification may be the same, but the times for scenario completion are different. Because we found a significant difference, we can go back to the scenario to look at the deployment locations for the constant case to see if a bias is obvious. Below is a map showing the deployment locations for the constant cases.

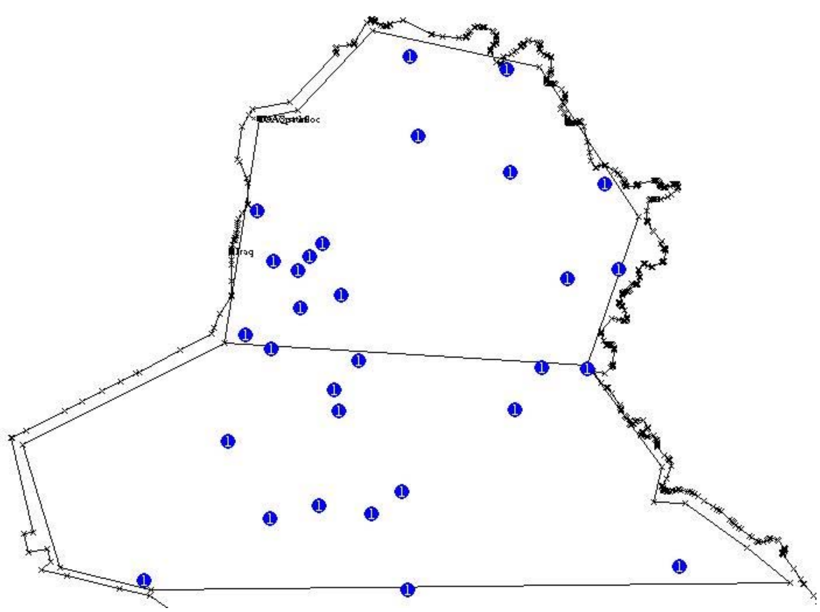

Figure 4.5 Distribution of Locations for all Consistent Cases

Keeping in mind, that this picture represents all random locations, and since this case (case 4) was a small TAO case, only the northern targets apply, the initial locations

do appear to be a little biased. One influential factor that may explain the difference could be the direction of the target's movements (especially when the satellites fly overhead and see them). Regardless, our constant location case provides significantly different scenario completion times than the case where locations and directions are picked randomly. However, we see that the confidence interval is not far from encompassing zero. Therefore, we can conclude that the difference is due to scenario timing.

Case 6 and case 9 are both independent UAS vehicle cases with the smaller TAO and 15 targets. Case 6 uses random location draws for each run. For case 9, only the first run has random draws and every following run starts with the same locations and targets follow the same movements. The following table summarizes the output for both cases. Another hypothesis test is done with the same null hypothesis for these cases.

Table 4.6 Comparison of Random vs. Consistent Independent/Small/15 Cases

Swarm/Independent	Random/Consistent	Small/Large TAO	Number of Targets	CASE	TARGET IDENTIFICATION TIME			SCENARIO TIME		
					MINIMUM	AVERAGE	MAXIMUM	MINIMUM	AVERAGE	MAXIMUM
I	R	S	15	6	4.33	76.59	351.71	828.00	1004.20	1073.00
I	C	S	15	9	17.24	111.49	390.17	910.00	921.17	929.00

The results of the paired t test are shown below.

Table 4.7 Confidence Intervals for Random vs. Consistent IXS15 Cases

					Target Times			Scenario Time		
				Case	Lower Bound	Upper Bound	Result	Lower Bound	Upper Bound	Result
I	X	S	15	6 vs. 9	-37.6109477	-23.8681456	REJ	66.93256526	104.2007681	REJ

The differences for these cases are more difficult to explain. We notice that case 6 has a lot lower values for target identification times, but higher times for scenario times. This can be explained in that case 9 may have deployment times that have more variance, but end well before the deployment times for case 6.

The next two cases to compare for deployment method are cases 7 and 10. These cases are identical in that they are independent UAS vehicle configurations in a large TAO with 30 targets. Case 7 is a random case and case 10 is the consistent. The following table summarizes the output for both cases.

Table 4.8 Comparison of Random vs. Consistent IXL30 Cases

Swarm/Independent	Random/Consistent	Small/Large TAO	Number of Targets	CASE	TARGET IDENTIFICATION TIME			SCENARIO TIME		
					MINIMUM	AVERAGE	MAXIMUM	MINIMUM	AVERAGE	MAXIMUM
I	R	L	30	7	5.78	172.17	1055.50	1024.00	1235.83	1470.00
I	C	L	30	10	4.00	135.32	846.02	1062.00	1233.57	1403.00

The results for the t tests are shown below

Table 4.9 Confidence Intervals for Random vs. Consistent IXL30 Cases

					Target Times			Scenario Time		
				Case	Lower Bound	Upper Bound	Result	Lower Bound	Upper Bound	Result
I	X	L	30	7 vs. 10	25.94457758	46.92044652	REJ	-67.2994431	19.49944307	FTR

This pair of runs has a very small difference in values. The test results suggest that there is a significant difference in target identification times. It does not conclude that there is a difference in scenario processing times. We also see much longer times compared to the previous pair of runs (for cases 6 and 9) since this pair has both a larger TAO and more targets.

The last pair of scenarios to compare with regards to consistent vs. random deployment locations is cases 2 and 5. These were swarming cases with a large TAO and 30 targets.

Table 4.10 Comparison of Random vs. Consistent SXL30 Cases

Swarm/Independent	Random/Consistent	Small/Large TAO	Number of Targets	CASE	TARGET IDENTIFICATION TIME			SCENARIO TIME		
					MINIMUM	AVERAGE	MAXIMUM	MINIMUM	AVERAGE	MAXIMUM
S	R	L	30	2	7.60	415.97	2090.46	1420.00	1689.40	2474.00
S	C	L	30	5	5.00	330.95	1931.61	1325.00	1652.93	2145.00

The results of the hypothesis tests are shown below.

Table 4.11 Confidence Intervals for Random vs. Consistent SXL30 Cases

	Case	Target Times			Scenario Time		
		Lower Bound	Upper Bound	Result	Lower Bound	Upper Bound	Result
S X L 30	2 vs. 5	15.37073998	66.3060383	REJ	-92.6965184	64.16318511	FTR

These results suggest that a significant difference exists in the target times, but not in scenario times. This can be explained by deployment times potentially being more uniform for case 5 (the consistent case), but having later deployment times.

In summary, we fail to reject target times for one of the comparisons, and scenario times for two comparisons.

Looking at the deployment times for cases 4 and 9 provide insight into where the difference comes from. Deployment times are calculated in the code as a random variable uniformly distributed between 0 and 1000. To determine whether deployment times are unbiased, graphs of the deployment times and averages are examined. The expected value of the output should be 500 minutes. The graph for both cases is provided below.

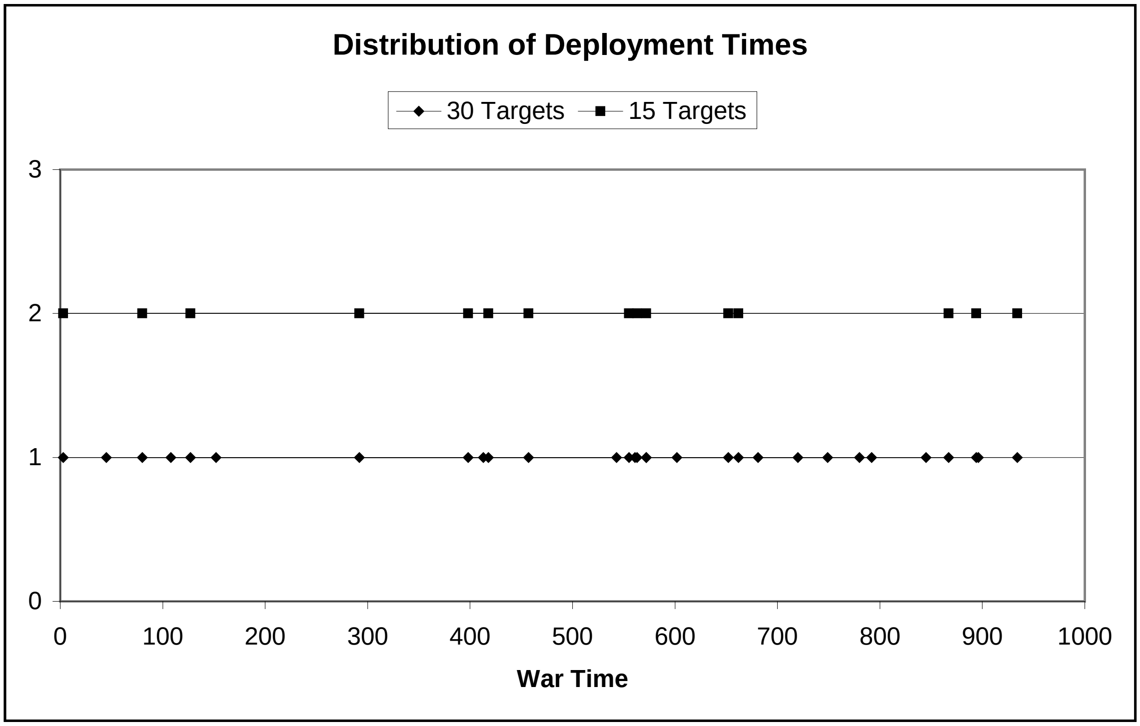

Figure 4.12 Deployment Time Distributions for Cases 4 and 9

Both cases appear to be uniformly distributed. The averages for the deployment times for the 15 and 30 targets are 498 and 531 respectively. Both of these values are close to what we would expect the average value to be (500 minutes). We hypothesized earlier that the deployment times may be later than expected, and the largest value we see is 934 minutes for both cases. This is what we would expect for the 15 target cases, but approximately 30 minutes earlier than we expect to see from the 30 target case.

Comparing Identification Statistics with Identification Probabilities

Because our testing cases (described in the previous chapter) did not show a significant difference between the expected output of identification accuracy (based on input values) and the actual output values, we concluded that our estimations based on

inputs were sufficient for analysis. However, the results were reviewed to ensure nothing seemed out of the ordinary. Those results are shown below.

Table 4.13 Statistics for Correct Identification

Case	probability of correct identification expected value	30 runs	difference
1	0.82	0.847	0.027
2	0.82	0.837	0.017
3	0.82	0.796	0.024
4	0.82	0.818	0.002
5	0.82	0.833	0.013
6	0.75	0.780	0.030
7	0.75	0.778	0.028
8	0.75	0.776	0.026
9	0.75	0.791	0.041
10	0.75	0.793	0.043
11	0.82	0.838	0.018
12	0.75	0.778	0.028

One interesting thing to note is that the cases with the individual UAS vehicles (cases 6-9) consistently yielded better results than was estimated (by an average difference of ~ 3.4%). To see if there is a significant difference in the differences of the number of correct identifications between the individual cases and the swarming cases, we run a paired t test. The results of the test are shown below.

Table 4.14 Confidence Intervals for Differences in Biases for Correct Identification (Swarm vs. Independent)

	Lower Bound	Upper Bound	Test Result
Cases 1 and 6	-0.04	0.18	FTR
Cases 4 and 9	-0.04	0.18	FTR
Cases 3 and 8	-0.03	0.17	FTR
Cases 2 and 7	-0.02	0.16	FTR
Cases 5 and 10	-0.01	0.15	FTR
Cases 11 and 12	-0.03	0.17	FTR

The results suggest that although the swarming cases seemed to be biased one way, the difference between the biases was not significant. To re-check the swarming cases, we ran 100 runs on all of those cases. The results are shown below.

Table 4.15 Expected, 30 Run, and 100 Run Correct Identification Statistics

	probability of correct identification		
Case	expected value	30 runs	100 runs
1	0.82	0.85	0.82
2	0.82	0.84	0.83
3	0.82	0.80	0.81
4	0.82	0.82	0.82
5	0.82	0.83	0.83
6	0.75	0.78	0.78
7	0.75	0.78	0.78
8	0.75	0.78	0.80
9	0.75	0.79	0.78
10	0.75	0.79	0.80
11	0.82	0.84	0.83
12	0.75	0.78	0.78

The results of the 100 runs support the idea that there is an existing bias in the swarming cases, but not in the individual cases (on average there is a bigger difference between the expected value and the 100 run output value for the swarming cases). We predicted that one possible explanation for this is that the swarming cases process targets faster, meaning their identification from the satellites is passed to the UAS vehicle and an identification from the UAS vehicle was made based on information from the satellites instead of a perceived identification from the UAS vehicle.   However, the “threat hold” values were specifically set to keep this from happening. Threat hold was set to 5 time steps (minutes) for UAS vehicles, meaning 5 minutes after the satellite passes the target

sighting to the UAS vehicle, the UAS vehicle has to re-acquire the target from either another satellite sighting or its local sensor. The fact that the average target sightings were significantly smaller for the swarming cases supports this hypothesis, however the cases where we rejected that there was no difference were not significantly smaller than the others.

We test this hypothesis by re-running the scenarios with modified code with lower threat hold times. The table below compares our old results with the new results (with modified code).

Table 4.16 Table of Old vs. New PID Output Statistics

				Case	Lower Bound	Upper Bound	Significantly Different
X	R	S	15	1 vs. 6	-0.03	0.17	NO
X	C	S	15	4 vs. 9	-0.05	0.19	NO
X	R	L	15	3 vs. 8	-0.05	0.19	NO
X	R	L	30	2 vs. 7	-0.01	0.15	NO
X	C	L	30	5 vs. 10	-0.03	0.17	NO
X	R	S	30	11 vs. 12	-0.01	0.15	NO

Running a t test on these results provides conclusions that better fit our expectations. All values fit our test, and our expected value is closer to our actual average for correct identification for each case.

Comparing Cases with Swarming vs. Independent UAS Vehicles

The following pairs of cases are identical with the exception of the UAS configuration (four swarming UAS vehicles vs. two independent UAS vehicles): {(1,6),(2,7),(3,8),(4,9),(5,10)}.

Table 4.17 Case List (Swarming vs. Independent)

Swarm/Independent	Random/Consistent	Small/Large TAO	Number of Targets	CASE	TARGET IDENTIFICATION TIME			SCENARIO TIME		
					MINIMUM	AVERAGE	MAXIMUM	MINIMUM	AVERAGE	MAXIMUM
S	R	S	15	1	7.91	115.89	660.38	837.00	1044.50	1286.00
I	R	S	15	6	4.33	76.59	351.71	828.00	1004.20	1073.00
S	R	L	30	2	7.60	415.97	2090.46	1420.00	1689.40	2474.00
I	R	L	30	7	5.78	172.17	1055.50	1024.00	1235.83	1470.00
S	R	L	15	3	8.90	187.99	839.90	980.00	1197.80	1568.00
I	R	L	15	8	5.07	116.48	501.17	968.00	1060.10	1152.00
S	C	S	15	4	4.86	109.40	392.81	1041.00	1087.50	1131.00
I	C	S	15	9	17.24	111.49	390.17	910.00	921.17	929.00
S	C	L	30	5	5.00	330.95	1931.61	1325.00	1652.93	2145.00
I	C	L	30	10	4.00	135.32	846.02	1062.00	1233.57	1403.00
S	R	S	30	11	6.78	229.12	1247.50	1190.00	1297.80	1410.00
I	R	S	30	12	3.64	94.80	458.68	980.00	1053.27	1119.00

We focus on the cases where the deployment locations are the same (i.e. {(4,9),(5,10)} because variance is reduced in those cases. For all cases we see a difference in both the target identification time and scenario time between the swarm cases and the independent vehicle cases. Only for the XCS15 target identification times is the average for the swarm smaller than the independent. We can attribute this to the lack of stress (small area and few targets) on the competing systems, as the values are very close together. We can perform a paired-t test on the data to confirm that is a significant difference. The table below shows the confidence intervals associated with the statistics for the different combinations.

Table 4.18 Confidence Intervals for Consistent Cases (Swarming vs. Independent)

					Target Times			Scenario Time		
				Case	Lower Bound	Upper Bound	Result	Lower Bound	Upper Bound	Result
X	C	S	15	4 vs. 9	-7.53	4.92	FTR	158.36	174.30	REJ
X	C	L	30	5 vs. 10	176.80	216.15	REJ	356.92	481.82	REJ

We conclude that there is a significant difference between the pairs of cases except for target times for the XCS15 case. This is not unexpected, because that was the pair that was counter-intuitive, with the average swarming target identification time smaller than the average independent time.

To check for a difference in the rest of the cases (i.e. the cases where deployment locations were not consistent), we can perform paired t-tests. The cases being considered are {(1,6),(2,7),(3,8),(11,12)}. The results of the t-tests are shown below.

Table 4.19 Confidence Intervals for Random Cases (Swarming vs. Independent)

					Target Times			Scenario Time		
				Case	Lower Bound	Upper Bound	Result	Lower Bound	Upper Bound	Result
X	R	S	15	1 vs. 6	23.16	38.80	REJ	18.18	76.49	REJ
X	R	L	30	2 vs. 7	176.59	225.17	REJ	364.65	493.35	REJ
X	R	L	15	3 vs. 8	99.63	168.81	REJ	57.89	157.24	REJ
X	R	S	30	11 vs. 12	120.23	149.38	REJ	256.70	316.90	REJ

The results of these tests also support our hypothesis that the individual configuration cases have significantly lower times than the swarming cases. What's interesting in these cases is that even the XRS15 cases (the ones that didn't show a significant difference for consistent deployment locations) do show a significant difference in these pairs.

Comparing Scenarios with Small TAOs vs. Large TAOs

The following pairs are identical with the exception of the size of the TAO: {(1,3), (6,8),(11,2),(12,7)}. The values for these pairs are shown below.

Table 4.20 Case List of Small TAOs vs. Large TAOs

Swarm/Independent	Random/Consistent	Small/Large TAO	Number of Targets	CASE	TARGET IDENTIFICATION TIME			SCENARIO TIME		
					MINIMUM	AVERAGE	MAXIMUM	MINIMUM	AVERAGE	MAXIMUM
S	R	S	15	1	7.91	115.89	660.38	837.00	1044.50	1286.00
S	R	L	15	3	8.90	187.99	839.90	980.00	1197.80	1568.00
I	R	S	15	6	4.33	76.59	351.71	828.00	1004.20	1073.00
I	R	L	15	8	5.07	116.48	501.17	968.00	1060.10	1152.00
S	R	S	30	11	6.78	229.12	1247.50	1190.00	1297.80	1410.00
S	R	L	30	2	7.60	415.97	2090.46	1420.00	1689.40	2474.00
I	R	S	30	12	3.64	94.80	458.68	980.00	1053.27	1119.00
I	R	L	30	7	5.78	172.17	1055.50	1024.00	1235.83	1470.00

As expected, the average times get larger when the size of the TAO increases. As expected, the differences are much more obvious in the swarming UAS vehicle cases. This makes sense because the individual UAS vehicle cases have an advantage in that they can simultaneously process a scenario. Calculating confidence intervals for all pairs yields the following results.

Table 4.21 Confidence Interval for Swarming cases (Small vs. Large TAOs)

				Case	Target Times			Scenario Time		
					Lower Bound	Upper Bound	Result	Lower Bound	Upper Bound	Result
S	R	X	15	1 vs. 3	-183.48	-117.45	REJ	-174.91	-86.16	REJ
I	R	X	15	6 vs. 8	-59.11	-35.34	REJ	-108.00	-32.60	REJ
S	R	X	30	2 vs. 11	-169.29	-112.00	REJ	-362.32	-230.88	REJ
I	R	X	30	7 vs. 12	-84.39	-64.7460579	REJ	-192.31	-116.49	REJ

As expected, all pairs show significant differences, leading us to conclude that TAO size has a significant effect on both average time to identify a target and average time to complete a scenario.

Comparing Scenarios with 15 Targets vs. 30 Targets

The following pairs are identical with the exception of the number of targets being deployed: {(11,1),(2,3),(12,6),(7,8)}. The values for these pairs are shown below.

Table 4.22 Case List of 15 vs. 30 Targets

Swarm/Independent	Random/Consistent	Small/Large TAO	Number of Targets	CASE	TARGET IDENTIFICATION TIME			SCENARIO TIME		
					MINIMUM	AVERAGE	MAXIMUM	MINIMUM	AVERAGE	MAXIMUM
S	R	S	30	11	6.78	229.12	1247.50	1190.00	1297.80	1410.00
S	R	S	15	1	7.91	115.89	660.38	837.00	1044.50	1286.00
S	R	L	30	2	7.60	415.97	2090.46	1420.00	1689.40	2474.00
S	R	L	15	3	8.90	187.99	839.90	980.00	1197.80	1568.00
I	R	S	30	12	3.64	94.80	458.68	980.00	1053.27	1119.00
I	R	S	15	6	4.33	76.59	351.71	828.00	1004.20	1073.00
I	R	L	30	7	5.78	172.17	1055.50	1024.00	1235.83	1470.00
I	R	L	15	8	5.07	116.48	501.17	968.00	1060.10	1152.00

As expected, the more targets in a scenario, the longer it takes for the UAS vehicles to process the scenario. However, it is important to remember that targets are deployed at a time randomly chosen between 0 and 1000 minutes from the beginning of the scenario. Also as expected, the differences are much more apparent in the swarm case since the UAS vehicles are at a disadvantage in that they have to stick together. The table below summarizes the confidence intervals for these comparisons.

Table 4.23 Confidence Intervals for 15 vs. 30 Targets

					Target Times			Scenario Time		
				Case	Lower Bound	Upper Bound	Result	Lower Bound	Upper Bound	Result
S	R	S	X	1 vs. 11	103.03	137.49	REJ	251.05	324.95	REJ
S	R	L	X	2 vs. 3	63.51	157.37	REJ	391.53	516.61	REJ
I	R	S	X	6 vs. 12	10.38	22.49	REJ	23.38	73.68	REJ
I	R	L	X	7 vs. 8	29.76	57.80	REJ	98.50	166.76	REJ

As expected, we reject the null hypothesis that the swarm case is the same as the individual case using an alpha of .10.

Overall Implications of Results

The main point of our analysis was to evaluate the tradeoffs between accuracy and scenario completion time (defined as how long it takes to identify all targets in an area). Because we were only using target deployment differences for case-bias evaluation purposes, we are left with eight cases with random deployment locations (varying by UAS vehicle configuration, area size, and number of targets). The following is a graph of scenario completion times using all cases. The cases with consistent

deployment locations (4, 5, 9, and 10) are included in the graph providing point estimates to compare to the random location points.

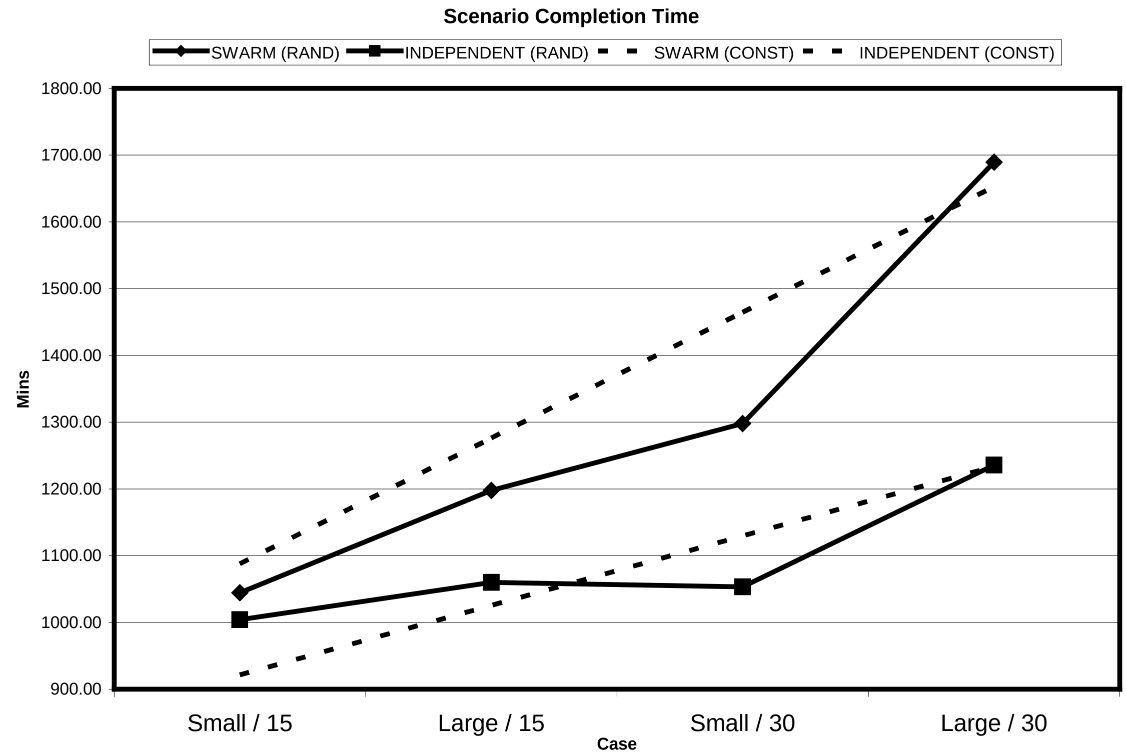

Figure 4.24 Comparison of Scenario Completion Times for Different TAO Sizes / Number of Targets

The above graph demonstrates how the difference grows between the different UAS vehicle configurations depending on the scenario circumstances. Recall that the probability of correct identification was 82% for the swarming cases and 75% for the individual cases. At what point the tradeoff for these values is acceptable is a subjective call to be made by the decision-maker depending on the applicability of the scenario.

The below graph demonstrates a similar comparison, but for average target identification times. As with the graph above, all cases were used for these comparisons.

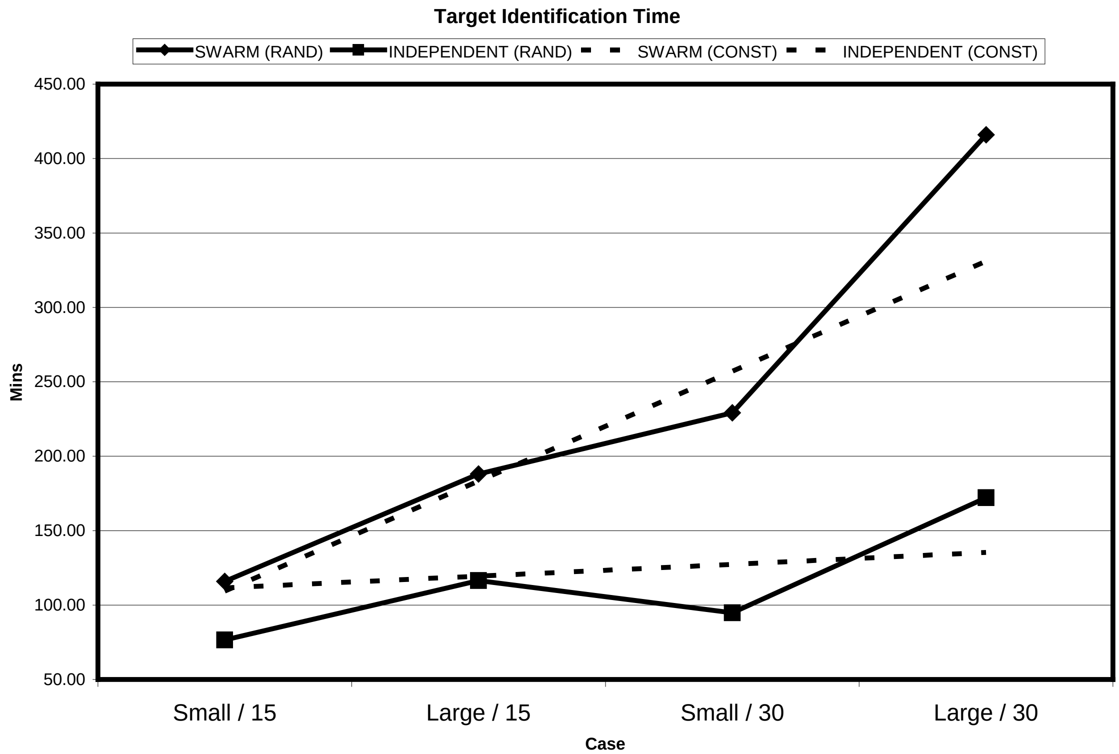

Figure 4.25 Comparison of Target Identification Times for Different TAO Sizes / Number of Targets

As with the previous graph, at which point the trade-off between time difference and identification accuracy is a subjective call to be made by the decision-maker.

Another important thing to understand when considering the difference in identification probabilities is that the swarming cases have an associated probability of "no identification" associated with them. Because our decision rules dictated that a positive identification would only be made in cases where at least two of the four

vehicles' identifications were the same, there was a possibility that either no two identifications were the same or that two were the same and the other two were the same. This should be taken into consideration when determining an acceptable level of trade-off.

One last thing to consider is that although we went through great strides to reduce variability in our cases, ultimately satellite windows provided a huge part of the results. Being in the "right place" at the "right time" could be considered a significant contributing aspect, especially in a stochastic study such as this one. Below is a graph of satellite windows. It is important to keep in mind that having a satellite overhead means that the satellite field-of-view can see at least some part of the TAO. These results are for the smaller TAO, for the larger TAO, the windows may be larger.

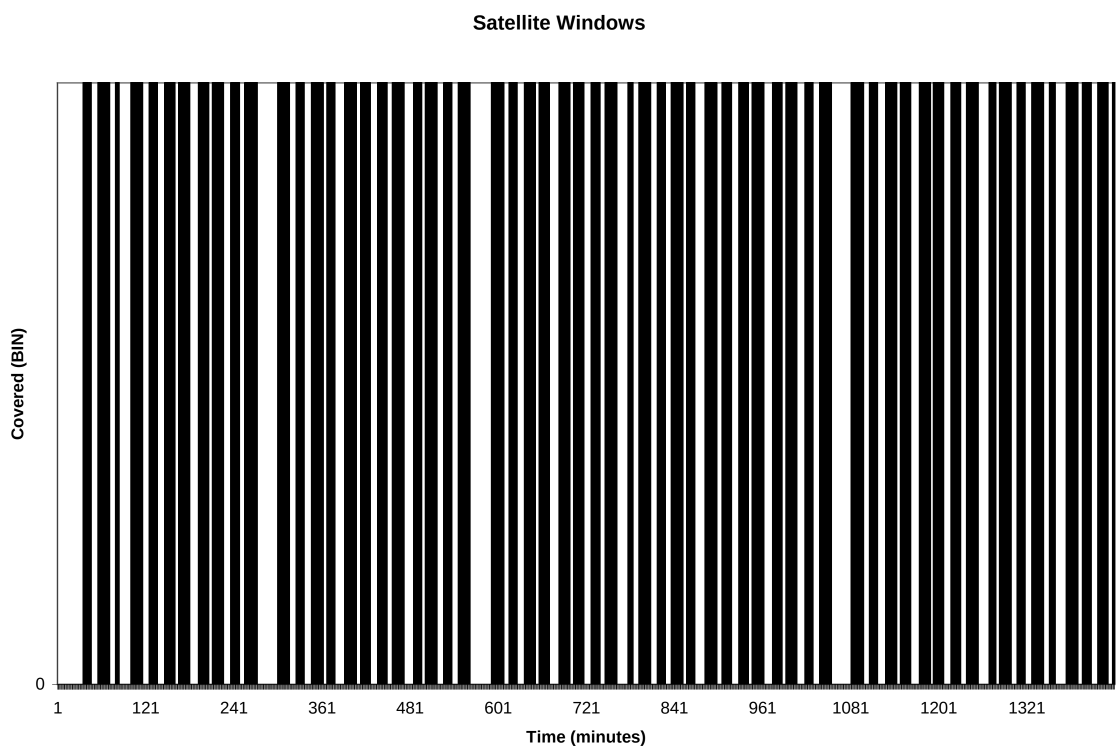

Figure 4.26 Satellite Windows Over 24 Hours (1440 Minutes)

Notice that there are significant windows where satellites are not available (4 windows of over 25 minutes). As always, these are important aspects to consider when evaluating stochastic agent-based models.

V. Conclusion

Overview

The intention of our research was to use an agent-based model (SEAS) to examine UAS configuration tradeoffs between accuracy and speed in a target-rich environment. We were familiar with a previous study done by Captain Jeffrey Rucker that used UAS vehicles to examine SCUD hunting missions. By using his warfile and modifying it to fit our needs, we saved a significant amount of effort. Our study examines the benefits and drawbacks of configuring a swarm of UAS vehicles to identify a target vs. having the vehicles independently identify targets. By examining how different environmental aspects (number of targets, size of theater) influenced the difference between the two configurations, we were able to quantify the advantages and disadvantages for several different situations. This chapter concludes our research by reviewing the MOP data we collected, reviewing modeling issues and assumptions, and making suggestions for potential follow-on research.

Output Comparisons

The list of things we wanted to examine can be found below:

1. The effect of random deployment locations every run vs. constant deployment locations every run (to potentially minimize variance).
2. Whether a significant difference exists between UAS vehicles in a swarming behavior vs. an independent configuration.
3. Whether a significant difference exists when TAO size changes.
4. Whether a significant difference exists when number of targets increases

A summary of our output results for times is shown below. Whether or not a significant difference existed was determined using a 2-sided paired-t test with an alpha

of .10 with a run size of 30 for all cases. All confidence interval differences are for the first setting minus the second (Random minus Consistent, Swarming minus Independent, Small minus Large, 15 minus 30).

Table 5.1 Summary of Target/Scenario Time Results

						Target Identification Time			Scenario Processing Time		
					CASES	Lower Bound	Upper Bound	Significantly Different	Lower Bound	Upper Bound	Significantly Different
Random vs. Consistent	S	X	S	15	1 vs. 4	-8.02	11.11	NO	-64.47	-2.39	YES
	I	X	S	15	6 vs. 9	-37.61	-23.87	YES	66.93	104.20	YES
	I	X	L	30	7 vs. 10	25.94	46.92	YES	-67.30	19.50	NO
	S	X	L	30	2 vs. 5	15.37	66.31	YES	-92.70	64.16	NO
Swarming vs. Independent	X	C	S	15	4 vs. 9	-7.53	4.92	NO	158.36	174.30	YES
	X	C	L	30	5 vs. 10	176.80	216.15	YES	356.92	481.82	YES
	X	R	S	15	1 vs. 6	23.16	38.80	YES	18.18	76.49	YES
	X	R	L	30	2 vs. 7	176.59	225.17	YES	364.65	493.35	YES
	X	R	L	15	3 vs. 8	99.63	168.81	YES	57.89	157.24	YES
	X	R	S	30	11 vs. 12	120.23	149.38	YES	256.70	316.90	YES
Small vs. Large TAO	S	R	X	15	1 vs. 3	-183.48	-117.45	YES	-174.91	-86.16	YES
	I	R	X	15	6 vs. 8	-59.11	-35.34	YES	-108.00	-32.60	YES
	S	R	X	30	11 vs. 2	-169.29	-112.00	YES	-362.32	-230.88	YES
	I	R	X	30	12 vs. 7	-84.39	-64.75	YES	-192.31	-116.49	YES
15 vs. 30 Targets	S	R	S	X	11 vs. 1	103.03	137.49	YES	251.05	324.95	YES
	S	R	L	X	2 vs. 3	63.51	157.37	YES	391.53	516.61	YES
	I	R	S	X	12 vs. 6	10.38	22.49	YES	22.38	73.68	YES
	I	R	L	X	7 vs. 8	29.76	57.80	YES	98.50	166.76	YES

Our first comparison was between cases where deployment location and subsequent movements of the targets were consistent between every run and cases where they were chosen randomly for every run. Our tests suggest that some cases were biased (showing a statistical difference) and some weren't. To diagnose why we saw a significant difference between some cases, we examined the deployment locations for the

consistent cases and determined that they appeared random. To further diagnose the problem, we examined deployment times for these cases and found they weren't uniformly distributed as expected. That explained why our results for a number of our consistent cases were significantly different.

Our second comparison was to determine whether there was a significant difference between swarming cases and independent cases. We expected that there would be because two independent UAS vehicles could process two different targets simultaneously while four swarming vehicles would have to process the same target simultaneously. Our output supports our hypothesis, as all differences were statistically significant and positive (suggesting that independent times were smaller than swarming times for both targets and scenario) except for consistent deployment in small TAO with 15 targets which showed no significant difference. These results also help quantify the difference in the two configurations for different scenario specifics. As expected, the smallest difference was observed in the small TAO case with 15 targets and the largest difference in the large TAO with 30 targets. This indirectly suggests that both larger TAOs and more targets increase the differences between scenario and target processing times (which we expect).

Our third comparison was a direct (as opposed to the previously indirect comparison) between large and small TAO sizes. The results show that the differences in both target identification and scenario times between independent and swarming UAS vehicles becomes more apparent as area increases. This was expected.

Our last comparison was between cases of fifteen and thirty targets. The results suggest that as the number of targets increase, the amount of time necessary to process these targets also increases. These results were expected.

Overall the results provided a way to quantify the influence of certain factors on our scenario. One interesting aspect of the results is that the difference in both scenario and target identification times is influenced more by TAO size than number of targets for the independent cases. However, the swarming cases were influenced more by the number of targets than the size of the TAO. This makes sense because the independent vehicles are better equipped to handle an increase in number of targets, whereas an increase in number of targets would stress the swarming vehicle cases.

We also evaluated the correct identification statistics for each case by comparing the difference in values to the difference in probabilities expected based on input. Below are the confidence interval results of the paired-t test with none of the results different from what we would expect.

Table 5.2 Summary of Identification Results

				Case	Lower Bound	Upper Bound	Significantly Different
X	R	S	15	1 vs. 6	-0.03	0.17	NO
X	C	S	15	4 vs. 9	-0.05	0.19	NO
X	R	L	15	3 vs. 8	-0.05	0.19	NO
X	R	L	30	2 vs. 7	-0.01	0.15	NO
X	C	L	30	5 vs. 10	-0.03	0.17	NO
X	R	S	30	11 vs. 12	-0.01	0.15	NO

Assumptions and Model Development

We made several assumptions in our study that are important to understand for our research. The first assumption is that "human in the loop" is negligible as long as it is removed from all cases. When a target is close enough to be detected by a UAS vehicle using it's own sensor (20 meters), an identification decision is immediately made. This was done to minimize variability in the cases. Realistically, there would be a delay between detection and identification.

Our second assumption is that for the consistent deployment location cases, the randomly chosen locations on the first run are adequately random to not bias the case to be better for either the swarming UAS configuration or the independent UAS configuration.

Our third assumption is that 1/2 of the four swarming vehicles is an adequate threshold for macro-level identification decisions. Ideally the capability of each UAS vehicle would be considered and some kind of weighted formula would be used to make a decision on identification. However, for our cases each swarming UAS vehicles had the same probability of correct identification.

Our fourth assumption concerns modeling decisions that don't necessarily reflect reality, but don't affect any results and therefore can be modeled differently for efficiency sake. One example of this is considering "friendly" targets as part of the red force. Because we didn't differentiate between different kinds of targets in our output analysis, it was acceptable for us to model all targets as children agents of the red force. This made modeling much more efficient.

Recommendations for Future Research

Our warfile was created specifically for the cases we set up. While some aspects are very easily modified to fit other cases (i.e. increasing the amount of targets, changing probabilities for identification, and changing the flight path of the UAS vehicle), other aspects that other users may be interested in using as a variable aren't as easy to change (i.e. differentiating between friendly/enemy targets, adding different orders for UAS vehicles, etc). For our purposes, the warfile was kept as efficient as possible. It is robust enough for our purposes, but for future research may not be easy to modify.

Another aspect that may be expounded on is having the UAS vehicles have sensors of different types. This would be more realistic, but would also require an extensive amount of additional code if the user wanted the swarming UAS vehicles to be able to share information and base micro-level identification decisions on previous identification decisions from other vehicles.

One last area that may be improved is making the identification procedures more accurate to real life. In our scenario we found that some of our statistics for identification didn't match the expected values based upon our inputs. Through further inspection, we deduced that identification decisions from the satellites (which were perfect) were influencing UAS vehicle decisions if there wasn't sufficient time between the last sighting by satellite and identification time of UAS vehicles. The UAS vehicles were designed to have a threat hold of five minutes. This value was set so that the UAS vehicles could make their own identification decisions after 5 minutes. Any more and the satellites made the decisions for the targets. Any less and UAS vehicles would lose more targets than was reasonable due to taking more than five minutes to arrive at the

perceived target location. Ideally we would write code so that the satellites would pass along detections only (not identifications) to the UAS vehicles, who would use their own sensors for decision making. However, we modeled the effect by setting the satellite sensor capability identical to the UAS vehicle (depending on case).

We have demonstrated in our research that SEAS (and agent-based modeling in general) is a very powerful tool in performing mission-level analysis. Our model in particular demonstrates an ability to accurately represent an operation environment, while also being robust enough to serve many different interests. With UAS being such a hot topic recently, there is no doubt that agent-based modeling can and will be beneficial in determining aspects of CONOPS.

Appendix A. List of Acronyms

ABM	**Agent-based Modeling**
ACC	**Air Combat Command**
AOI	**Area of Interest**
CONOPS	**Concept of Operations**
FOV	**Field of View**
LTL	**Local Target List**
MOE	**Measure of Effectiveness**
MOP	**Measure of Performance**
PD	**Probability of Detection**
PID	**Probability of Identification**
SEAS	**System Effectiveness and Analysis Simulation**
TAO	**Tactical Area of Operations**
TPL	**Tactical Programming Language**
UAS	**Unmanned Aircraft System**
UAV	**Unmanned Aerial Vehicle**

Appendix B. Blue Dart

Evaluating UAS Employment Configurations Using an Agent Based Combat Model

Capt. Joseph Van Kuiken
Student / Air Force Institute of Technology
joseph.vankuiken@afit.edu
Word Count: 560
Keywords: simulation, combat modeling, Agent Based Modeling (ABM), Unmanned Aerial Vehicles (UAV), Unmanned Aircraft Systems (UAS)

Over the last few years, Unmanned Aircraft Systems (UAS) have proven themselves through the support provided in areas such as Iraq and Afghanistan. Of particular interest to the US military is the aircrafts' ability to loiter for hours at altitudes where they are safe from threats and still maintain the ability to attack ground targets when needed. As the DoD develops new UAS capabilities and employment concepts, it will need to evaluate trade-offs such as target identification speed vs. identification accuracy in several UAS scenarios – a perfect application area for the use of Agent Based Modeling (ABM) as the combat modeling tool of choice (the Systems Effectiveness Analysis Simulation (SEAS) is an example). For such an analysis, the UAS vehicles would be modeled as individual agents that interact with each other, with enemy agents, and with the environment by following a set of programmed rules based upon real world operational employment.

Some would argue that one of the most likely pursuits of the DoD is a swarm of smaller (perhaps independently less capable) unmanned aircraft to provide several independent identification "looks" to ultimately make a macro-level identification decision. A study that evaluates the benefits and draw-backs to swarming UAS vehicles compared to independent vehicles would be of benefit to the DoD and can be effectively represented in an ABM through use of different sets of rules for different UAS configurations.

Independent identification assessments made by swarming UAS vehicles combined by the ground station to make an ultimate determination on the target's type would ideally be more accurate than a single UAS vehicle (especially if the sensors

equipped on the swarm were of different types). This would be beneficial in scenarios where correct identification is important to scenario processing (i.e. when an attack is queued as a result of the identification). However, independent UAS vehicles processing targets individually (less accurate than a swarm) could be considered better if a theater was large enough to the point that speed was more important than accuracy. This configuration (independent non-swarming UAS vehicles) would be applicable in scenarios where targets might be more time-sensitive. Both of these situations exist in today's operations, and an analysis capability to help the commander decide which configuration is appropriate is important.

UAS vehicles provide many advantages to manned aircraft that are making high-level decision makers re-look at how we play the game. How we utilize these UAS vehicles will undoubtedly be important to our future military. One of the biggest perceived controversies in future UAS identification is the scenario process speed vs. identification accuracy trade-off when deciding how to configure and utilize UAS vehicles. Evaluation of these aspects is important when investment in one capability over another may result in significantly enhanced performance in theater. The DoD would benefit from having the capability to effectively evaluate many different competing capabilities and employment options in a timely manner, along with the ability to model anticipated behavior of future unmanned systems. The use of an Agent Based combat model such as SEAS, provides such a tool.

"Captain Joseph Van Kuiken is a student at the Air Force Institute of Technology."

Bibliography

Banks, Jerry, et al. Discrete-Event System Simulation. New Jersey: Pearson Prentice Hall, 2004.

Boccara, Nino. Modeling Complex Systems. Verlag: Springer, 2003.

Dongseob, Jang, et al. “A Comparison on Information Fusion Methods for Air Target Identification.” Proceedings of the 2005 ACM Symposium on Applied Computing. 2005: 45-46.

Ferber, J. Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Harlow, England: Addison-Wesley, 1999.

Gansler, J. S. “DoD Modeling and Simulation (M&S) Glossary.” Dtic.mil. 1 march 2009 <http://www.dtic.mil/whs/directives/corres/pdf/500059m.pdf>.

Godfrey, G. A. et al. “Negotiation mechanisms for coordinated unmanned aerial vehicle surveillance.” Integration of Knowledge Intensive Multi-Agent Systems. 2005: 324-329.

Gooden, Clarence Dr. “U.S. Department of Defense Unmanned Aerial Vehicles Master Plan” www.dtic.com 11 Oct. 2000 Department of Defense. 14 Mar. 2008 <http://www.dtic.mil/ndia/targets/gooden1.pdf>.

Karim, S., et al. Agent-based mission management for a UAV. “International Conference on Intelligent Sensors, Sensor Networks & Information Processing.” 2004: 481-486.

Lavery, Eamonn. “Introduction to Agent Based Simulation in Flexim.” Flexim.com. 1 March 2009 <http://www.flexsim.com/products/ds/FlexsimABS.pdf>.

Law, Averill M. and W. David Kelton. Simulation Modeling and Analysis, 3rd Ed. Boston: McGraw-Hill, 2000.

Miller, J.O. OPER 672: Combat Modeling II. Classroom Lecture Slides. 2008

Newcome, Laurence R. Unmanned Aviation: A Brief History of Unmanned Aerial Vehicles. Reston: American Institute of Aeronautics and Astronautics, 2004.

Price, Joseph C. *Game Theory and U-Boats in the Bay of Biscay*. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH: MS thesis, AFIT/GOR/ENS/03-18, March 2003.

Reynolds, Craig W. "Flocks, Herds, and Schools: A Distributed Behavioral Model." Computer Graphics, Vol. 21, No. 4: 25 – 34, July 1987.

Rucker, Jeffrey. *Using Agent-Based Modeling to Search for Elusive Hiding Targets*.. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH: MS thesis, AFIT/GOR/ENS/06-16, March 2006.

Ryan, A., et al. "An overview of emerging results in cooperative UAS control." 43rd IEEE Conference on Decision and Control. 2004: 602–607.

Sanders, T. Irene and Judith A. McCabe. The Use of Complexity Science. Washington Center for Complexity & Public Policy: Washington D.C., 2003.

Sauter, J. A., et al. Distributed Pheromone-Based Swarming Control of Unmanned Air and Ground Vehicles for RSTA. Forthcoming in Procedings of SPIE Defense & Security Conference. 2008.

SPARTA, Inc.: Home Page. Retrieved Nov. 12, 2008, from SEAS Web site: http://www.teamseas.com/ (2005).

Stockton, R. G. "Models Versus Analysis." Discussion Paper distributed within the Center for Army Analysis. 1987.

Taylor, James G. "Hierarchy-of-Models Approach for Aggregated-Force Attrition." Proceedings of the 2000 Winter Simulation Conference. 2000: 925–932.

U.S. Department of Defence. "Data Fusion Subpanel of the Joint Directors of Laboratories." Data Fusion Lexicon. 1991.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
26-03-2009	**Master's Thesis**	Jul 2007 - Mar 2009

4. TITLE AND SUBTITLE
USING AGENT-BASED MODELING TO EVALUATE UAS BEHAVIORS IN A TARGET-RICH ENVIRONMENT

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Van Kuiken, Joseph A., Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Street, Building 642
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GOR/ENS/09-16

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

10. SPONSOR/MONITOR'S ACRONYM(S)
AF/A9

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

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT The trade-off between accuracy and speed is a re-occurring dilemma in many facets of military performance evaluation. This is an especially important issue in the world of ISR. One of the most progressive areas of ISR capabilities has been the utilization of Unmanned Aircraft Systems (UAS). Many people believe that the future of UAS lies in smaller vehicles flying in swarms. We use the agent-based System Effectiveness and Analysis Simulation (SEAS) to create a simulation environment where different configurations of UAS vehicles can process targets and provide output that allows us to gain insight into the benefits and drawbacks of each configuration. Our evaluation on the performance of the different configurations is based on probability of correct identification, average time to identify a target after it has deployed in the area of interest, and average time to identify all targets in an area.

15. SUBJECT TERMS Agent Based Modeling (ABM), Unmanned Aerial Vehicles (UAV), Unmanned Aircraft Systems (UAS)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			J. O. Miller, PhD, ENS
					19b. TELEPHONE NUMBER *(Include area code)*
U	**U**	**U**	**UU**	92	(937) 255-3636 x4326 john.miller@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18