NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

SYSTEMS ENGINEERING CAPSTONE PROJECT REPORT

UAV SWARM OPERATIONAL RISK ASSESSMENT SYSTEM

by

Team CQ Alpha
Cohort 311-141A

September 2015

Project Advisors: Gregory Miller
Mark Rhoades

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2015	**3. REPORT TYPE AND DATES COVERED** Capstone Project Report

4. TITLE AND SUBTITLE
UAV SWARM OPERATIONAL RISK ASSESSMENT SYSTEM

5. FUNDING NUMBERS
N/A

6. AUTHOR(S) Cohort 311-141A/Team CQ Alpha

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

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

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
N/A

11. SUPPLEMENTARY NOTES: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____NPS.2015.0040-IR-EM2-A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

This paper examines the need for a UAV Swarm Risk Assessment Tool and how it can assist the Navy's decision makers in assessing risk of UAV swarm threats in littoral environments, near potentially hostile countries, based on the latest intelligence.

Human-centered design principles help determine the needs of experienced battle commanders. These needs form the basis of requirements and functional analysis. The system design concept consists of several parts: discrete-event simulation of UAV swarm attacks using ExtendSim, statistical analysis of the simulation data using Minitab, and a graphical user interface (GUI) that evolved as a web-app prototype written in MATLAB.

Data from the simulation is analyzed and used to generate equations that calculate the effect of critical factors: physical environment, number of UAVs, distance from land, and the ship's defensive weapons. The GUI uses these equations to provide users with the capability to vary those critical factors and analyze different courses of action and risk. The physical GUI web-app can be used as-is, tailored or expanded. The paper concludes with an analysis of the actual GUI prototype built for UAV swarm risk assessment and how it meets user needs.

14. SUBJECT TERMS
unmanned aerial vehicle, littoral environment, area of responsibility, counter UAV, risk assessment, graphical user interphase, GUI, decision makers, assessing risk, intelligence, physical constrained environments, ships maneuverability, swarm attack, gap, assess risk, interactive.

15. NUMBER OF PAGES
151

16. PRICE CODE
N/A

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

UAV SWARM OPERATIONAL RISK ASSESSMENT SYSTEM

Cohort 311-141A/Team CQ Alpha

Sariyu Marfo	Jamaries Benitez Negron	John Junek
Shane Ehler	Shane Skopak	Justin Zarzaca
Ryan Fields		Robert Perrotta

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING
and
MASTER OF SCIENCE IN ENGINEERING SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 2015

Lead editor: Jamaries Benitez Negron

Reviewed by:

Gregory Miller	Mark Rhoades
Project Advisor	Project Advisor

Accepted by:
Ronald Giachetti
Systems Engineering Department

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This paper examines the need for a UAV Swarm Risk Assessment Tool and how it can assist the Navy's decision makers in assessing risk of UAV swarm threats in littoral environments, near potentially hostile countries, based on the latest intelligence.

Human-centered design principles help determine the needs of experienced battle commanders. These needs form the basis of requirements and functional analysis. The system design concept consists of several parts: discrete-event simulation of UAV swarm attacks using ExtendSim, statistical analysis of the simulation data using Minitab, and a graphical user interface (GUI) that evolved as a web-app prototype written in MATLAB.

Data from the simulation is analyzed and used to generate equations that calculate the effect of critical factors: physical environment, number of UAVs, distance from land, and the ship's defensive weapons. The GUI uses these equations to provide users with the capability to vary those critical factors and analyze different courses of action and risk. The physical GUI web-app can be used as-is, tailored or expanded. The paper concludes with an analysis of the actual GUI prototype built for UAV swarm risk assessment and how it meets user needs.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AOR	area of responsibility
APM	Assistant Program Manager
CAPT	Captain
CIWS	close-in weapons system
CO	Commanding Officer
COCOMO	Constructive Cost Model
CONOPS	concept of operations
COSYSMO	Constructive System Engineering Cost Model
COTS	commercial off the shelf
CMM	Capability Maturity Model
DARPA	Defense Advanced Research Projects Agency
DoE	Design of Experiments
DDG	guided missile destroyer
DOD	Department of Defense
EOSS	Electro-Optical Sensor System
ESLOC	estimated software lines of code
FAC	Fast Attack Craft
GPS	global positioning system
GUI	graphical user interface
HCD	human-centered design
HMW	how might we
ICAM	integrated computer aided manufacturing
IDEF0	ICAM Definition for Function Modeling
IRB	Institutional Review Board
IWS	Integrated Weapon Systems
JIAMDO	Joint Integrated Air and Missile Defense Organization
KSLOC	thousands of software lines of code
kts	knots
lb	pound
LOS	line-of-sight

LT	Lieutenant
M&S	modeling and simulation
m/s	meters per second
MALE	medium altitude long endurance
MATLAB	Matrix Laboratory
MGS	machine gun system
MH	Multipurpose Helicopter
MIT	Massachusetts Institute of Technology
MOE	measures of effectiveness
MSL	Mean Sea Level
NAVSEA	Naval Sea Systems Command
nm	nautical miles
NPS	Naval Postgraduate School
NSWC	Naval Surface Warfare Center
P_D	probability of detection
PDF	portable document format
PEO	Program Executive Office
P_k	probability of kill
POV	point of view
rds/min	rounds per minute
ROE	Rules of Engagement
SE	System Engineering
SIPRNET	Secret Internet Protocol Router Network
SLOC	software lines of code
SME	subject matter expert
SPY	surface ship, radar, surveillance
TTP	tactics, techniques, and procedures
UAV	unmanned aerial vehicle
WEZ	weapons engagement zone

EXECUTIVE SUMMARY

Many of the United States Navy's operational areas of responsibility are located in littoral environments near countries hostile to the United States. These operational areas consist of physically constrained environments that limit surface ship maneuverability. These areas have a higher likelihood of an unmanned air vehicle (UAV) swarm attack than typical blue water operations because of the ship's close proximity to land. Also, most countries now possess the technology to produce mass quantities of UAVs that can be weaponized.

UAV threats are unlike traditional air threats such as manned aircraft and cruise missiles. UAVs have flight envelopes that make them challenging to defeat. The littoral environment makes it difficult to detect and identify UAVs in such a cluttered environment. Warfighters will be challenged to make time-critical decisions, possibly with insufficient threat information, while determining an appropriate response.

This report identifies a gap in the ability to assess the risk of these swarm attacks from the operational, tactical and planning level. Current commanders have to take the intelligence provided to make an assessment without tangible data outputs. The intent of this paper is to build and examine a tool that will bridge the gap between intelligence and tangible outputs in order to assist decision makers in assessing risk. This software tool will primarily be used by surface ship commanding officers, carrier task force commanders, fleet commanders, operational planning teams, or area of responsibility commanders to conduct UAV swarm attack risk assessments based on real-time intelligence. The system was designed to assess the interactions of controllable and uncontrollable factors of a UAV swarm attacks. Key attributes include:

1. Providing the stakeholders a way to rapidly assess risk of a potential UAV swarm attack
2. Allowing criterion to be manually entered for automatic outputs
3. Providing a visual representation of the outputs in terms of reaction time and probability of attack outcome

4. Providing a rapid risk assessment based on available data to provide an analysis traceable to a high fidelity model

The primary output of this project was an interactive operational risk assessment system that included a graphical user interface (GUI). This GUI is a preliminary design prototype. This paper examines the development of a system using the human-centered design process. This process relied heavily on the definition of user needs of surface ship commanding officers, task force commanders and planning team members. These user needs were determined using interviews of the stakeholders. The operational context was developed using a combination of user needs and literature research. Needs and operational context were examined in order to determine the high-level system requirements of both the GUI and simulation. The users primarily needed a tool with visual representation of the outcome of a swarm attack. The user wanted to be able to enter UAV type and location, as well as their own ship's defense capabilities. The output had to be unambiguous, and it had to provide distance and time from threat. This allows the decision maker to determine the ship route of travel during planning to reduce the risk and understand the time to react after initial detection.

This paper presents a high-level system design that is used to understand the subsystem interactions required for the implementation of a working prototype. The system components include a high-fidelity discrete-event simulation of a UAV swarm attack on a ship, a statistical analysis of the discrete-event simulation outputs, and a computer-based GUI. Given the complexity of the discrete-event simulation and the statistical analysis of its outputs, it was determined that the GUI should be a standalone subsystem that utilizes response equations based on the simulation output rather than including the simulation itself to produce outputs that fulfill user requirements. The response equations were written in MATLAB GUI based on the statistical analysis of the discrete-event simulation outputs and included in the GUI software code to enable immediate responses. By making the discrete-event simulation and the statistical analysis separate from the GUI, task-specific commercial software could be used to perform the required work.

Using the high-level system design and user defined requirements, a functional analysis of the system was preformed to ensure that all defined and derived requirements were met. System architecture was presented based on this functional analysis. The functional analysis was performed using a top-down method of all required functions and their interactions.

This paper presents the simulation that addresses the interaction of factors relevant to UAV swarm attacks. Using ExtendSim 9, the authors created a discrete-event simulation of a basic UAV swarm attack on a U.S. Navy surface ship. The simulation allowed the authors to explore and analyze the trade-space in order to identify those input factors that would have the greatest effect on the scenario outcome. Controls for those factors were then featured prominently in the GUI design.

Minitab 17 was used to analyze the simulation output (step one) and to fit a regression equation (step two) representing the relationships between factors. This equation was used in the MATLAB GUI code (step three) to enable immediate responses to user inputs, without the need to re-run the simulation after each change. The primary factors that affect ship success during a UAV swarm attack were distance, number of UAVs, UAV speed, and environmental factors. Ultimately, these three steps enabled the authors to create a prototype that was complete, realistic and adaptable using the data available and making the tool manageable for future improvements by employing Minitab17 and MATLAB GUI.

This paper presents the development of multiple low fidelity prototypes. These prototypes were designed based on user needs and simulation results. The GUI gives the user a visual representation of the UAV swarm attack risk under various conditions and lets users explore the trade space for risk using a number of inputs. This paper examines a series of prototypes with increased fidelity designed in MATLAB GUI emulating a web-based application. The prototype was updated to a final iteration using stakeholder feedback after testing. The final GUI presents two visual representations: absolute position, drawn on a map, and relative position, indicated by range rings around the ship. Heat maps and contour lines visually represent the level of threat to the ship and the available time to react based on distance detected.

Following the design of the prototype, a cost estimate was presented for the risk assessment system using System Cost Model Suite provided by the Naval Postgraduate School. The System Cost Model Suite performed estimates for various engineering disciplines including system engineering and software development. Total cost for a fleet development would be in excess of $1.74M.

This report concludes with an examination of the recommendations for future designs based on user feedback. Future applications would need to be web-based in order to remove the responsibility of updating the application from the user. Nonkinetic weapons will be a consideration for UAV attack response in the future and will require the GUI to have inputs for enhanced capabilities. Kill radius will need to become a controllable factor based on the type of UAV selected.

I. INTRODUCTION

Chapter I include background information to allow the reader to get familiar with the topics to be discussed. The problem statement, goal and objectives are discussed. In addition, the system engineering model approached used in this project is explained in detail.

A. BACKGROUND

With the increased investment in unmanned system development in countries hostile to the United States, the likelihood of an unmanned air vehicle (UAV) swarm attack on a U.S. flagged warship is becoming a reality. The operational areas that the risk is highest are in narrow passages that require either limited ship maneuverability with territorial waters on each side or a physically constrained operating environment. Despite the dangers of operating in these constrained littoral environments, these areas of the world are and will continue to be major shipping lanes for worldwide commerce, thus requiring a military presence (U.S. Energy Information Administration 2014).

UAVs are unlike other air threats to a surface ship. When compared to traditional air threats, both manned aircraft and cruise missiles, UAVs have characteristics that make them challenging to defeat. In general, UAVs fly at slower speeds than traditional air threats and have relatively smaller radar cross-sections (Davis et al. 2014). Given that many modern, deployed air defense radar systems were not designed to detect threats with such characteristics, UAVs remain a formidable foe (Button et al. 2008).

The littoral environment adds an additional layer to the current detection problem given the possibility for radar terrain masking until a low-altitude air threat has transitioned to the water environment. Even when the threat transition to water, “sea surface can limit the observation of low-altitude targets by surface or low altitude radars” (Curry 2011). Radar terrain masking refers to the radar horizon range were the line-of-sight (LOS) of the radar is blocked by terrain or limited by sea surface (Curry 2011). The observation of low-altitude targets by surface or low-altitude radars can be masked by the terrain or be limited by the sea surface (Curry 2011). When taken as a whole, these

characteristics provide UAVs with a distinct advantage over war ships that need to establish robust procedures in order to protect against their inherent vulnerability in these operating conditions.

Even when they are detected, clear monitoring is required to track and identify the possible intentions of inbound UAVs. And when a target is identified, enough data may not be available given current shipboard systems to estimate the size or payload of the threat (Button et al. 2008). As such, battle commanders could find themselves in a time-critical situation where insufficient information exists to determine an appropriate response.

In addition to the detection and identification difficulties that UAVs pose, they also have potential strength in numbers. Given their relatively low construction and operating cost when compared to traditional air threats, countries can target ships using a large number of UAVs. Since many of the current attack UAVs are meant to self-detonate, they are designed as an unrecoverable asset (Davis et al. 2014). When compared to traditional air threats, UAVs also carry significantly smaller payloads that not only make attacking in large numbers a strategic advantage, but also a necessity (Button et al. 2008). These increases in potential UAVs make the detection and identification of such attacks that much more difficult in terms of time-critical decisions for battle commanders.

Examination of the effectiveness of the ships in these attacks can be broken down into four categories: detection and tracking, identification, kill chain response time, and weapons engagement (kinetic and nonkinetic). This research was in support of creating a simulation that examines the relationship between the primary factors affecting the effectiveness against an unmanned swarm. It was envisioned that a high-fidelity simulation model could be used to create outputs of UAV swarm scenarios. These outputs would then be used to design a preliminary operational risk management system that determines a risk output file. The risk would be displayed with all the pertinent information that the battle commanders need in order to make the time-critical decisions required during such an attack.

B. PROBLEM STATEMENT

A UAV swarm attack risk assessment tool is not currently available for surface ship commanding officers, carrier task force commanders, fleet commanders, operational planning teams or area of responsibility commanders. Current commanders have to take the intelligence provided to make an assessment without tangible data outputs. That is to say, intelligence exists about the potential for specific UAV threats, however, there is no appreciable information provided to make decisions if an attack actually occurs. The determination of possible engagement timelines and outcomes is information that battle commanders do not currently receive prior to littoral operations.

C. GOALS AND OBJECTIVES

The authors' intent is to provide a new tool that will provide tangible data outputs that will assess the risk for an operation. This tool will solve the problem at hand by providing the decision makers a way to quantitatively predict risk to operations via a user-friendly output.

The goal of this project was to develop a system that includes a software tool for use by surface ship commanding officers, carrier task force commanders, fleet commanders, operational planning teams or area of responsibility commanders to conduct UAV swarm attack risk assessments. The system was designed to collect information about the systems and interactions involved in UAV swarm attacks.

Key attributes of the system are:

1. Provide the stakeholders a way to rapidly assess risk of a potential UAV swarm attack.
2. Allow for the manual entry of criterion that produce automatic outputs.
3. Provide a visual representation of the outputs in terms of reaction time and probability of attack outcome.
4. Provide a rapid risk assessment based on available data to provide an analysis traceable to a high fidelity model.

The preliminary system was developed in order to inform strategic and tactical commanders about the risk of UAV swarm attacks in constrained areas. This system is intended to enable leaders to make a more informed decision on whether or not to accept the risk of encountering a UAV swarm attack with the information provided by the tool. The tool allows the stakeholders to have data-driven results that take into account current intelligence and presents the user with a preliminary operational risk assessment. This system is not intended to be a complete solution for all possible situations; however, this system is designed to provide users the means to make educated decisions if a UAV attack is encountered.

The system displays risk with all the pertinent information that the stakeholder needs in order to make a relevant decisions to accept risk or to take action in considering other relevant alternatives that may be available. The tool allows the stakeholders to propose new measures of operation based on the risk assessment given by the tool and have data to support changes in the operations or improvements necessary to reduce the risk of a UAV swarm attack in constrained environments.

The primary output of this project was an interactive operational risk assessment system that includes a graphical user interface (GUI). This GUI is a preliminary design prototype. In order to design the GUI, a modeling and simulation tool was developed to provide outputs to the GUI. The GUI then calculates the risk assessment outputs required by the battle commanders using the modeling and simulation data outputs. As such, an in-depth study was required to ensure that the problem space was accurately depicted and that appropriate data was generated. The proposed study was intended to answer the following research questions:

1. What are the types, capabilities, and flight envelopes for unmanned aerial "attack" vehicles of hostile foreign countries?
2. What tangible information, systems, and interactions are needed by surface ship commanding officers, carrier task force commanders, fleet commanders, operational planning teams or area of responsibility commanders to make a risk decision for an operation?

3. What are the standard and nonstandard navy tactics for UAV swarm attack?
4. What are the gaps in technology and processes for UAV swarm response and what are the current work-arounds being used by the fleet?

D. SYSTEM ENGINEERING PROCESS

This project involved the development of a UAV Swarm Operations Risk Management Software system GUI. The authors held discussions with surface warfare commanding officers, carrier task force commanders, fleet commanders, operational planning teams and area of responsibility commanders. The primary goal of these discussions was to identify the needs of the battle commanders and utilize their professional inputs to refine the requirements of the system. Following the initial refinement of requirements, the authors developed the modeling and simulation tool and GUI prototype. After initial GUI development, the authors demonstrated the GUI and collected subject matter expert (SME) feedback and input for iterative upgrades.

The system engineering process model approach used is a tailored version of the V-Model. A pictorial representation of the Capstone System Engineering (SE) model with each phase deliverables is shown in Figure 1.

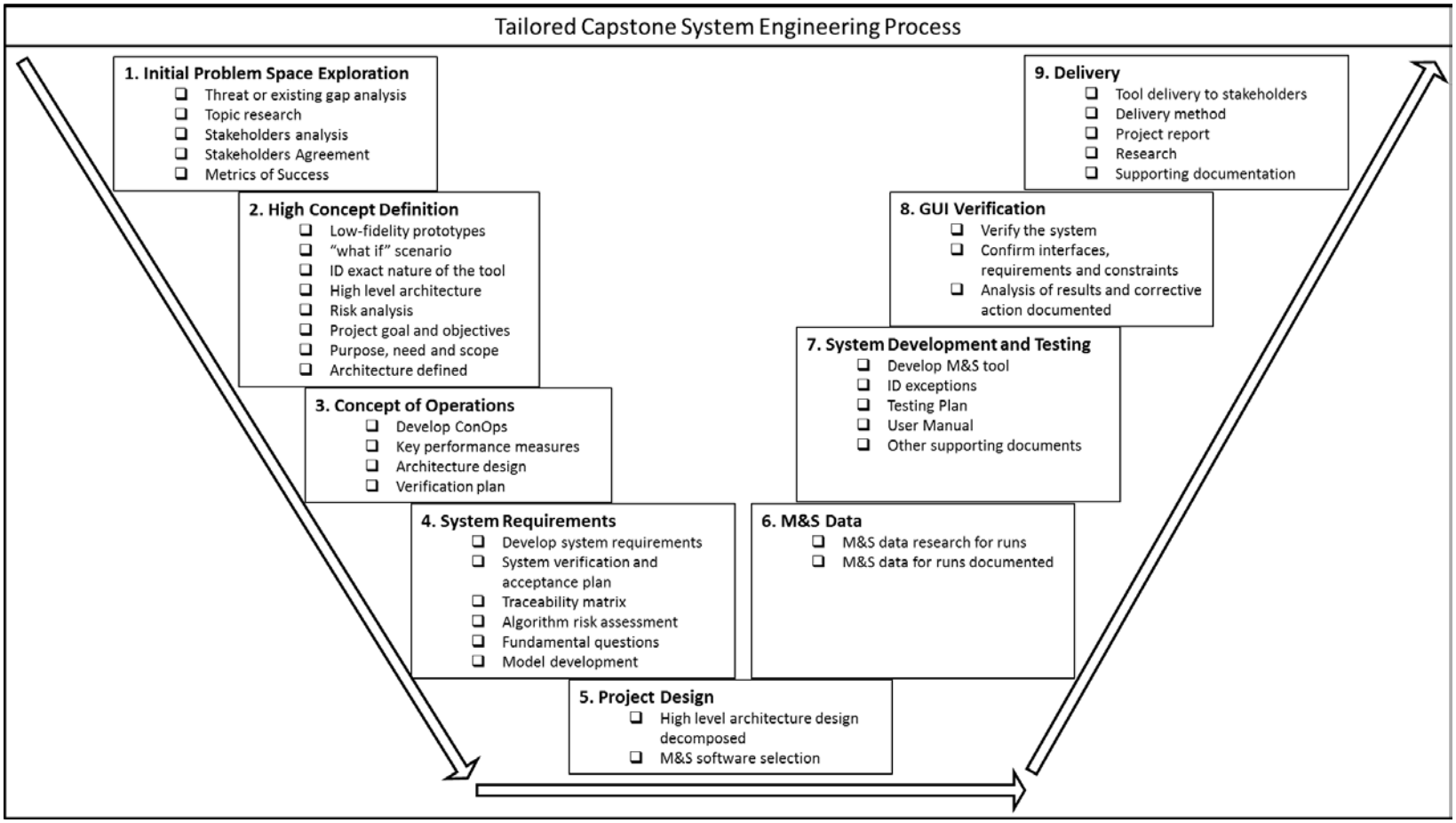

Figure 1. Tailored Capstone System Engineering Process

Elements one through nine define each SE phase and deliverables.

1. Initial Problem Space Exploration: Phase One consists of a threat analysis and existing gap analysis to explore the problem space. This phase included in-depth topic research along with an initial stakeholder analysis. The goal was to identify the stakeholders, engage them in collaborative discussions to identify their needs and use their professional inputs to refine the requirements for the system. The phase output provided the authors with a solid understanding of stakeholders' perspective on gaps, roles, responsibilities and expectations.
2. High Concept Definition: Phase Two focused on the human user. The focus was to use early low-fidelity prototypes based on the research to discover and validate concepts, and user needs. The high concept design will be based on "what if?" scenario that will act as a catalyst for the UAV swarm events in the GUI prototype risk management tool. Key risks were identified. Alternative concepts for meeting the project purpose and needs were explored. The exact nature of the anticipated functioning prototype risk management tool was identified based on the stakeholders' needs and documented research. Additionally, the authors verified project feasibility, project scope and identified preliminary risks. By this stage the goal was to have the exact nature of the project goals and objectives, purpose and need, scope and initial high level architecture defined.
3. Concept of Operations: Phase Three documented the concept of operations (CONOPS) as a foundation for more in-depth analysis. The authors identified high level user needs and systems capabilities in terms that all stakeholders can understand. The authors also identified the: who, what, why, where, and, how of the system GUI. The focus was to engage stakeholders in collaborative discussions to define and create the initial CONOPS, review with stakeholders and iterate. The primary output was a CONOPS describing who, what, why, where, and, how of the project including stakeholders needs, constraints and identified risks. The authors

also had a system verification plan defining the approach that was used to verify the functioning prototype risk management tool GUI.

4. System Requirements: During Phase Four, the authors developed a validated set of system requirements that meet the stakeholder needs based on the CONOPS. The authors elicited requirements that were analyzed, documented, validated and managed according to the stakeholders needs.
5. Project Design: During Phase Five, the authors focused on the project design using simulation capabilities based on the system requirements. The requirements were allocated to the system components and interfaces specified. This phase produced a high-level architecture design with detail design specifications.
6. Modeling & Simulation (M&S) Data: Phase Six focused on the M&S data that was used in the simulation. The simulation provided the input data that was used in the GUI for verification, test and evaluation.
7. System Development and Testing: Phase Seven focused on the development of a GUI. This was a preliminary design prototype. The solution was modified as needed to meet the design specifications. The output in this phase was a GUI model that was tested in iterative phases for verification and acceptance.
8. Model Verification: Phase Eight verified the system model in accordance with the high level design, requirements, verification plans and procedures.
9. Delivery: Phase Nine focused on the delivery of the interactive operational risk assessment system preliminary design prototype GUI. The GUI was delivered to the stakeholders. The project final deliverables were arranged accordingly and delivered by the set due dates. This included the tool, final project capstone, research and any supporting project related documentation.

II. PROBLEM SPACE EXPLORATION

In Chapter II, the authors begin with an explanation of the system operational concept diagram. Then the threat analysis was broken down to three UAVs of interest. Current Navy surface ships counter UAV capabilities were identified. Tactics, techniques, and procedures (TTP) with existing UAV simulations and related tools were explored. Chapter II ends with a concept description and the path used for the development of this project.

As shown in Figure 2, system development was a multi-pronged effort that encompassed multiple areas of concern with respect to the problem space. The authors started with a literature research that included but was not limited to: DOD intelligence reports, UAV threat assessment, U.S. weapons systems and mission planning. Throughout the report there is reference to a modeling and simulation portion. The modeling and simulation was used to create the background information in order to generate the data, evaluate the threats and create a data set to be provided as an input in the GUI. The modeling and simulation is not the focus of this report, but it was an integral part of the GUI development. Therefore, significant rigor was required to ensure that all possible factors and their interactions were captured in the development, refinement, and subsequent application of discrete-event simulation of the UAV swarm attack model.

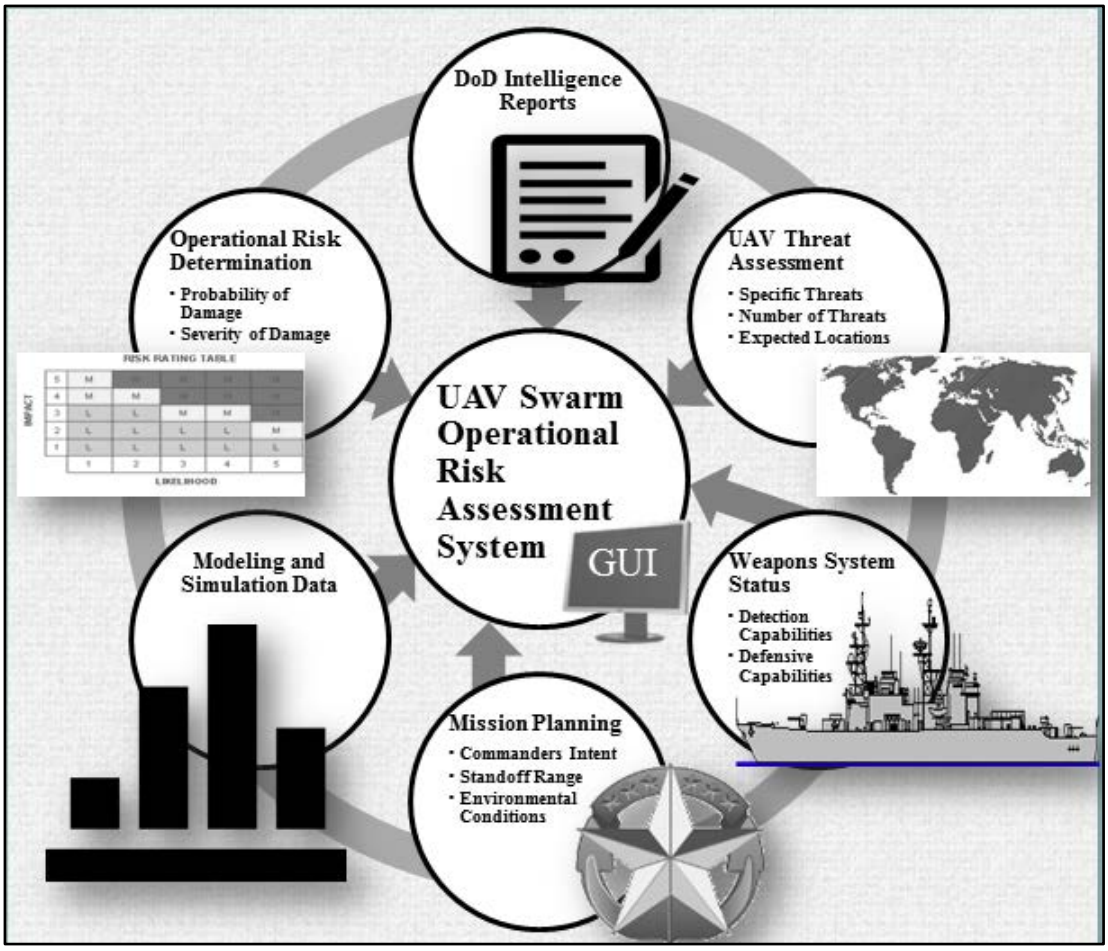

Figure 2. System Operational Concept Diagram

A. THREAT ANALYSIS

Recent developments in precision navigation, satellite communication, and lightweight materials have paved the path for the widespread development and attractiveness of armed UAVs (Davis et al. 2014). Although manufacturers in the U.S. and Israel dominate the global UAV market (approximately 75 percent share between them), the availability of commercial of the shelf (COTS) technology has provided a host of other countries with the means to develop, test, and field UAV technology of their own (Jane's Information Group 2014d). The primary focus of the most recent UAV development has been in support of medium altitude long endurance (MALE) platforms (Jane's Information Group 2014d). Nations such as Turkey, Pakistan, Iran, and the United Arab Emirates are all actively engaged in the development of their own MALE

UAVs (Jane's Information Group 2014d). Although, the LOS two-way communication range usually limits the operating range of MALE platforms, many of these platforms can be used for one-way fully autonomous strike missions (Davis et al. 2014). In addition to using MALE UAVs as a strike platform, nonrecoverable, strike-specific UAVs have also been developed and widely proliferated. Research has shown that the UAV inventories of many of the countries hostile to the U.S. contain a combination of MALE and nonrecoverable strike-specific UAVs. Therefore, for the purpose of this system development, it was determined that these two types of UAVs comprise the most realistic threat to U.S. flagged warships operating in the littoral environment.

There are several key characteristics of UAVs that make them a credible threat to warships in the littoral environment. These characteristics include, but are not limited to:

1. Strength in numbers
2. Difficult to detect and differentiate
3. Relatively low cost
4. Mobile launch sites
5. Fire and forget capability
6. Technology is accessible

UAVs are difficult to detect given their relatively slow speed and small radar cross section, especially in a cluttered overland environment. Furthermore, a singular UAV is even harder to differentiate in an attacking group of multiple UAVs. UAVs are also significantly cheaper than the high value units that they are designed to attack. Mobile launch sites allow for the rapid deployment of UAVs in remote areas during an attack and make it very difficult to track and gain detailed intelligence on UAV threats. Fire and forget capability can be used via preprogrammed points for an attack; a UAV(s) can be launched without continuous monitoring.

The following UAVs were used for the purpose of this system development project. This project is limited to examining only three UAVs due to the scope of the project and the schedule constraints associated with the academic timeline. These three UAVs were chosen based upon their capabilities and the countries that operate them.

1. Yasir

Pictures and Iranian media reports indicate that the Yasir, shown in Figure 3, was developed by reverse-engineering a Boeing-Insitu Scan Eagle UAV obtained by the Iranian government (Jennings 2013). Assuming performance capabilities similar to the Scan Eagle, the Yasir has a 10-foot 2.5-inch wingspan with an operational payload plus fuel of 16 pounds. The Yasir is launched using a portable, pneumatic launcher with a propulsion system comprised of a single piston engine that is capable of a maximum level speed of 80 knots (92 miles per hour) (Jane's Information Group 2015b). According to Iranian defense authorities, the Yaris has a ceiling of 15,000 feet and operational range of 108 nautical miles (nm) (124 miles) (Jennings 2013).

Figure 3. Iranian Yasir UAV (from Cenciotti 2013)

2. Ababil-T

The Ababil-T, shown in Figure 4, is the short to medium range attack variant of the Iranian-built Ababil UAV. It is a twin-tailed, swept wing design capable of carrying a payload of 100 pounds of high explosive warhead and is launched from a portable, pneumatic launcher. The Abibal-T propulsion system consists of a single two-stroke

piston engine that drives two pusher propellers and is capable of 200 knots (230 miles per hour) at a ceiling of 10,820 feet and operational range of 27 nm (31 miles). The Ababil-T is highly dependent upon LOS communication; however, it is capable of global positioning system (GPS) navigation for striking both fixed and moving targets (Jane's Information Group 2014a).

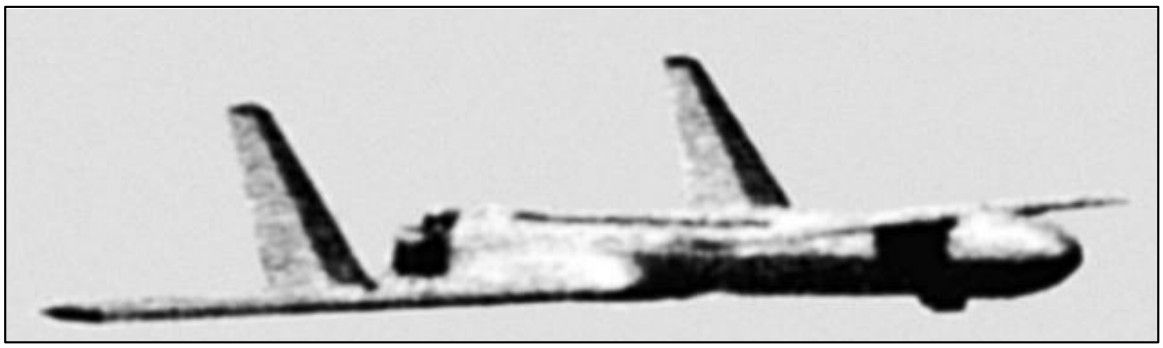

Figure 4. Iranian Ababil-T UAV (from Jane's Information Group 2014a)

3. Harpy

The Harpy, shown in Figure 5, is an Israeli-built strike-specific UAV. It has mid-mounted delta wings with full span elevons capable of carrying a payload of 70 pounds. The Harpy is launched by booster rocket from a ground or truck-mounted 18-round container. The propulsion system consists of a single two-stroke piston engine that drives two-blade pusher propeller and is capable of 135 knots (155 miles per hour) at a ceiling of 9,840 feet with an operational range of 270 nm (311 miles). The Harpy is fully autonomous and flies a preprogrammed flight profile until its radar acquires a target. Once commit altitude is reached, side force panels are deployed to stabilize the UAV during a terminal dive. In other words, this type of UAV could be used as a one-way or kamikaze style attack where the UAV dives into a target and explodes with no intent of returning to base (Jane's Information Group 2013b).

Figure 5. Israeli Harpy UAV (from Jane's Information Group 2013b)

For comparison purposes, UAV specifications are shown in Table 1.

Table 1. Threat UAV Specifications (after Jennings 2013; Jane's Information Group 2013; Jane's Information Group 2013b, 2014a, 2015b)

UAV Operated By	Payload (lb)	Max Speed (kts)	Ceiling (ft MSL[(2)])	Operational Range (nm)	Launch Method
Yasir Iran	16[(1)]	80	15,000	108	Portable, pneumatic launcher with a propulsion system comprised of a single piston engine
Ababil-T Iran	100	200	10,820	27	Runway using its single two-stroke piston engine that drives two pusher propellers
Harpy China Chile India Israel Turkey	70	135	9,840	270	Booster rocket from a ground or truck-mounted 18-round container

Notes: (1) Yasir UAV payload of 16 lb does not include fuel load; therefore, available payload is impacted by the amount of fuel required for the mission.

(2) MSL=Mean Sea Level

B. CURRENT NAVY SURFACE SHIP COUNTER UAV CAPABILITIES

In order to explore the problem space for weapons engagement against UAVs, the type of platform that they would be attacking had to be assessed. The idea behind the project was to work with U.S. ships that were in constrained environments. The type of ships that were evaluated needed to be able to conduct operations as either an individual ship or as part of a larger fighting group.

Based on the identification of U.S. ships able to work in constrained environments, it led to the selection of the Arleigh Burke Class Destroyers and the Ticonderoga Class Cruiser. The project effort concentrates on the evaluation of these two classes of ships. They conduct missions as both a single ship and as part of larger task forces or strike groups. Both of these ships have numerous detection and engagement systems that could be used to counter UAV attacks. The different ship capabilities can be seen in Table 2.

Table 2. Ship Type for Model Development (after Jane's Information Group 2015a, 2015b, 2015c)

	Arleigh Burke Class Destroyer	**Ticonderoga Class Cruiser**
Specifications	509 ft	567 ft
	31 kts (4300 nm range)	30 kts (6000 nm range)
	278-282 Crewmembers	330 Crewmembers
Detection	SPY-1D	SPY-1D
	EOSS	EOSS
Weapons (Counter UAV)	1 Mk45- 5in Gun	2 Mk45- 5in Gun
	1-2 Mk15 CIWS	2 Mk15 CIWS
	Mk38 Machine Gun System	Mk38 Machine Gun System
	Standard Missile (Out of Scope)	Standard Missile (Out of Scope)

Both ships are comparable in size, manpower, armament, and speed. They also look similar to the enemy as seen in Figure 6 and Figure 7. Their weapons engagement capabilities are the primary reason that the ship classes were selected. The five-inch gun, close-in weapons system (CIWS), and the machine gun system (MGS) have been tested against unmanned systems (Jane's Information Group, 2014b). The effectiveness of these

weapon systems against unmanned threats is at a classification level outside the scope of this project. The data used for modeling and simulation was based on generalities acquired through open source documentation. A detail discussion on how the two classes of ships were modeled based on unclassified weapons engagements effectiveness is located in the modeling and simulation chapter of this report.

Figure 6. Ticonderoga Class Cruiser (from Jane's Information Group 2015c)

Figure 7. Arleigh Burke Class Destroyer (from Jane's Information Group 2015a)

The kinetic weapons were assessed at a completely unclassified level. Information on their effectiveness was acquired from unclassified and open source material. The

capabilities of the five-inch gun, CIWS, and MGS are presented in Table 3. Currently there is no unclassified source that speaks to the probability of kill for each system against a UAV.

Table 3. Counter UAV Weapons (after Jane's Information Group 2013a, 2014b, 2014c)

	5 in Gun	**CIWS**	**Machine Gun System**
Designator	Mk 45	Mk 15	Mk 38
Caliber	127mm	22mm	25mm
Muzzle Velocity	808 m/s	1,030 m/s	1,100 m/s
Rate of Fire	16-20 rds/min	3,000 rds/min	180 rds/min
Effective Range	23 km	1.47 km	2.47 km

Images of the weapons seen in Figure 8 and Figure 9 give an idea of the size and magnitude of the systems being used to engage a UAV. Even as large as these systems appear to be, they are fairly agile and quick to react. CIWS is actually designed to defend against aerial attacks and missiles. Both the CIWS and MGS have very high rates of fire, but they lack the lethal range of the five-inch gun. CIWS and MGS would not be available to engage until the attacking UAV is very close to the ship. This limits the engagement capabilities against multiple UAVs because the CIWS and MGS will not be able to engage at the maximum range of UAV detection. The five-inch gun has a large effective range but lacks a high rate of fire. With the rate of fire less than one shot every three seconds, it will take longer for each UAV to be engaged.

Figure 8. Mk 15-Close-In Weapons System (from Jane's Information Group, 2014b)

Figure 9. Mk 45-5in Gun (from Jane's Information Group, 2013a)

Along with educated assumptions, probability of kill was based on the UAV envelope, UAV radar cross-section, sea state and other environmental factors. An in-depth assessment to study the actual effects on probability of detection and kill based on the environment and UAV parameters was outside the scope of this report. Information to their relevance is available but at higher classification level than available in this document. This project provides the building blocks to analyze intelligence data to evaluate the UAV swarm attacks.

In addition, research into nonkinetic engagement systems was limited based on the scope of the project. It was decided to concentrate on kinetic systems only because of scope limitations. It is recommended as an area for further study to include nonkinetic systems in the Risk Assessment System.

C. TACTICS, TECHNIQUES, PROCEDURES

A U.S. Navy surface ship sailing into a constrained environment such as a straits transit presents several well documented challenges. Challenges include restricted maneuvering, a potential for high air and surface traffic to include civilian, commercial and military all within close proximity of each other as shown in Figure 10. During a straits transit a surface ship or any ship for that matter, is required to obey the Law of the Sea. The Law of the Sea applied to warships and military aircraft mandates that:

> Ships and aircraft must avoid "any threat or use of force against the sovereignty, territorial integrity or political independence of the State bordering the strait" (Alexander 1991, 92)

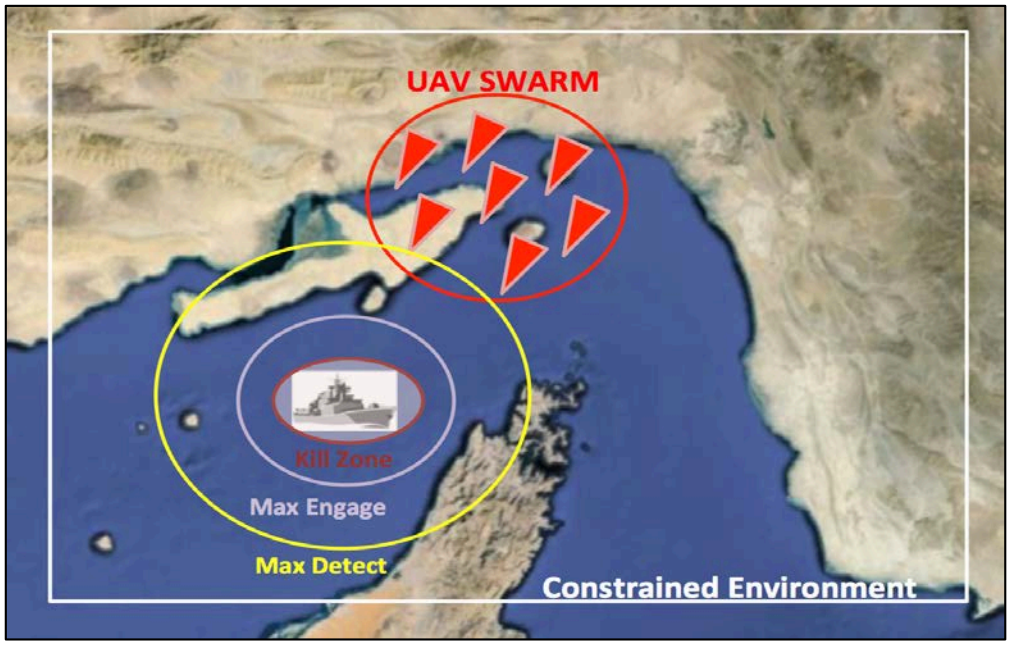

Figure 10. Straits Transit UAV Swarm Attack Scenario

In addition to avoiding the appearance of being threatening through the use of force in a strait transit scenario, the Law of the Sea also requires that ships move expeditiously through the strait. It has and will continue to be the policy of the U.S.:

> That warships, operating in their normal mode through international straits overlapped by territorial seas, may undergo formation steaming, and launch and recover aircraft. (Alexander 1991, 92)

The tactics employed by U.S. Navy surface ships during a straits transit scenario include having one or two helicopters airborne, typically MH-60S or MH-60R. The helicopters are not allowed to go over land and are not allowed to violate the airspace of the countries that boarder the strait. The rules of engagement (ROE) also are another driver of tactics. For instance, consider a single ship; a typical ROE when it has been fired upon or is under attack is that the crew has the right to defend itself with the proportional amount of force, or to maneuver out of the area. A typical ROE for a ship according to ship commanders interviewed is to not engage a UAV if the UAV is observing or capturing data of the ship at a distance.

The tactics for a surface ship to defend against a swarm of UAVs in a constrained environment is a tough challenge to solve as UAV technology is always evolving. An example of an evolving drone technology that is being done by the Defense Advanced Research Projects Agency (DARPA) is the development of a neuromorphic chip that is designed to be capable of learning like a human (Reagan 2014). According to MIT Technology Review during a simple three-room test for the drone host that had the chip installed:

> The first time the drone was flown into each room, the unique pattern of incoming sensor data from the walls, furniture, and other objects caused a pattern of electrical activity in the neurons that the chip had never experienced before. That triggered it to report that it was in a new space, and also caused the ways its neurons connected to one another to change, in a crude mimic of learning in a real brain. Those changes meant that next time the craft entered the same room, it recognized it and signaled as such. (Simonite 2014)

If this type of technology is able to be matured to the point where UAVs can operate autonomously, learn and adjust to Navy tactics while enroute, and in a networked fashion, then that ability poses a potentially significant threat to the ship.

Formally adopted Navy TTPs for responding to UAV swarm style of attacks against a ship operating in a constrained environment, such as a straits transit scenario, have not yet been promulgated. Additionally, once tactics are established, they will need to change to meet the ever evolving threat posed by advances in UAV technology.

D. EXISTING UAV SIMULATIONS AND RELATED TOOLS

The authors surveyed available references covering the current capabilities relevant to the project. These fall into two categories: UAV swarm attack simulations and operational planning user tools. There are many existing UAV swarm attack simulations whose results have been captured in reports and theses. These results, along with available model implementation insight, guided the development of the modeling and simulation input data to create the simulation and GUI. The simulation was used to guide the development of the prototype user interface.

Two UAV defense simulations were applicable to this project. These were documented in the theses "Adaptive Discrete Event Simulation for Analysis of Harpy Swarm Attack" (Cobb 2011) and "UAV Swarm Attack: Protection System Alternatives for Destroyers" (Pham et al. 2012).

Cobb (2011) developed a Java-based discrete-event simulation for the evaluation of the sensitivity of impact rate of Harpy-like UAVs on a ship to UAV and ship characteristics. The Harpy UAVs were modeled as loitering overhead searching for radar signals to lock on to. Modeling of the ship's defenses was limited to a static probability of impact for each UAV. In all simulation runs, 54 UAVs were loitering in the ship's path. Cobb's analysis indicated that the probability of each UAV detecting the ship and ship speed significantly affected the UAV impact rate by changing how many UAVs pursued the ship — meaning a "big" UAV swarm has a higher probability of detection (2011). A faster ship will have lower probability of response time available. Also significant was the UAV dive speed, though it was shown not to strongly interact with

other parameters. The limited simulation of ship defenses and the lack of variation in the number of UAVs present made this model and results unsuitable for the development of the application.

Pham et al. (2012) used a model developed in Microsoft Excel to explore the sensitivity analysis of UAV impact rate on a guided missile destroyer (DDG) to detailed UAV and ship characteristics. In Pham et al. (2012), the approaching UAVs were a mix of remote controlled and fully autonomous types. The UAV and ship characteristics were selected based on publicly available specifications on known UAV and ship configurations. Pham et al. (2012) provided guidance on the information flow through a DDG following threat detection that is useful in the accurate modeling of delay between detection and engagement. According to Pham et al. (2012), the number of CIWS systems aboard the DDG affects the impact rate more strongly than any of the other factors assessed. Most notably, the improvement of detection systems did not greatly affect the impact rate because the ship defensive abilities were limited by its weapons, not by its sensors. Because of the sensitivity to CIWS capability, a high fidelity model of the CIWS was critical to properly modeling this threat and response.

Applications and user interfaces number beyond counting. It was not possible to survey all such tools or capture all applicable lessons learned. Instead, the project focused on collecting a few useful but very relevant examples to guide the development of the prototype. Three publications were identified that covered similar applications and user interfaces. None of these redefine general guidelines of good user interface design, but rather provide useful examples of well-implemented interfaces.

The first publication focuses in map overlay software for discrete-event simulation data (Mack 2000). Discrete-event simulation:

> A general framework, build around the idea of “discrete events,” has been developed to help one follow a model over time and determine the relevant quantitates of interest. (Ross 2013)

In Mack (2000), the application is called THORN, and it visualizes data from discrete-event simulations on an open-source mapping tool. The author emphasizes Shneiderman’s (1998) eight golden rules of interface design:

1. Strive for consistency
2. Enable frequent users to use shortcuts
3. Offer informative feedback
4. Design dialogs to yield closure
5. Offer error prevention and simple error handling
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

The THORN application provides a good example of the concept of layered information, allowing users to selectively enable and disable visuals to reduce clutter as needed. This concept is in line with the user needs expressed by former DDG commanding officers with experience in this field.

The second publication is a path-planning application for the navigation of minefields (Piatko et al. 2001). Investigation into this report was prompted by the likening of loitering UAVs to “mines in the sky” by some stakeholders with extensive experience in the field. The mine path-planning application provided examples of simple, clear spatial information communicated through gray scale plotting overplayed on a map.

The third publication detailed the development of a decision support system for maritime operational planning (Grasso et al. 2012). Grasso et al. (2012) proved to be one of the most relevant literature sources for the project showing numerous examples of maps filled with color-coded data. Especially of interest was their implementation of perceptual redundancy using a color map background to augment a vector field map overlay.

Based on the literature, it was determined that an application was not currently available to cover the efforts of the intended capstone project goal. It was determined to use the relevant literature and stakeholder feedback to create a suitable GUI prototype.

E. CONCEPT DESCRIPTION

The concept was to fill the information gap between specific UAV capabilities and the potential impact of those specific UAVs should an attack occur. Currently battle commanders have intelligence reports regarding UAVs in their respective areas of operation. These intelligence reports can include such information as possible type, location, number, and overall threat status of the country. The battle commander must take this information into consideration when making operational decisions. By providing the battle commanders with a tool that transforms pre-existing intelligence reports into situation-specific information, the battle commanders are provided more pertinent information with which to make informed decisions.

As shown in Figure 11, the concept was to use a discrete-event simulation approach (referred as model in Figure 11) of a UAV attack on a specific ship and then utilize the outputs of the model to provide the battle commanders with the tactical information they required.

> Simulating a probabilistic model involves generating the stochastic mechanisms of the model and then observing the resultant flow of the model over time. Depending on the reasons for the simulation, there will be certain quantities of interest that we will want to determine. However, because the model's evolution over time often involves a complex logical structure of its elements, it is not always apparent how to keep track of this evolution so as to determine these quantities of interest. A general framework, built around the idea of "discrete events," has been developed to help one follow a model over time and determine the relevant quantities of interest. The approach to simulation based on this framework is often referred to as the *discrete event simulation approach*. (Ross 2013)

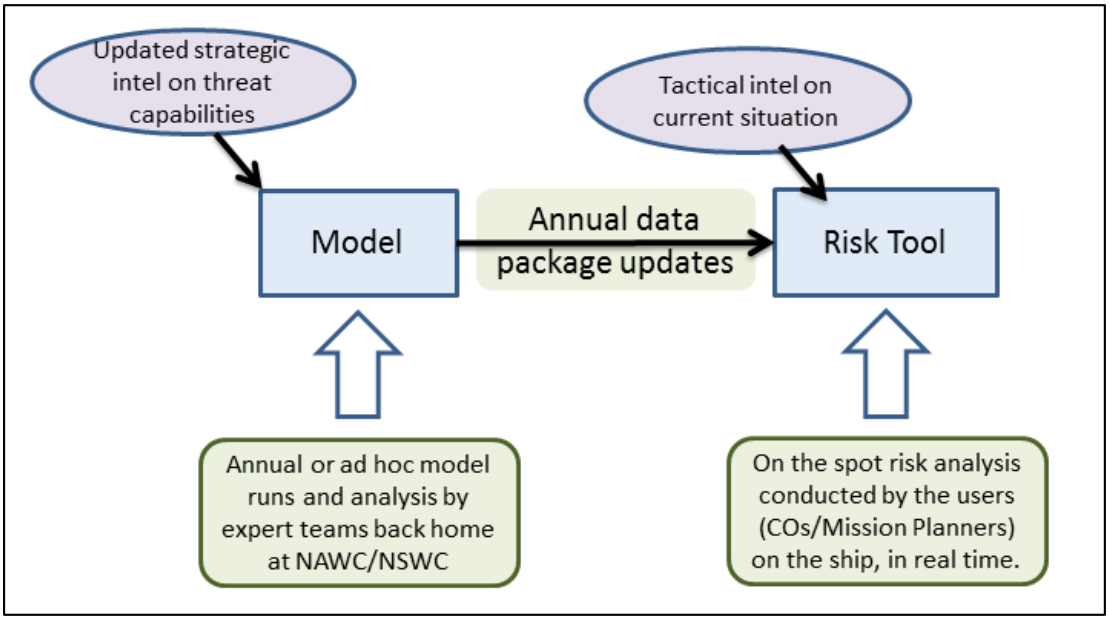

Figure 11. System Concept Diagram

The concept allowed for future growth in both threat and own ship capabilities due to its reliance on modeling and simulation data. It was also the desire to ensure that the concept system did not require additional hardware components for its use by the battle commanders. The authors determined that the actual simulation runs and analysis of the outputs would require computing power and time that may not be available to users aboard the ships. As such, separating the discrete-event simulation from the user interface and only using the output of the model within the application was determined to be the preferred preliminary design concept. Additionally, this concept was adopted because of the flexibility it allowed with regards to the actual risk tool outputs.

Following completion of problem space exploration, the authors learned what made a UAV swarm attacks unique, what ship board defensive systems existed, how a UAV attack might be defended against, and what information was currently available for battle commanders. The concept provided the authors with the opportunity to interview actual battle commanders and determine what information they needed to make more informed decisions.

THIS PAGE INTENTIONALLY LEFT BLANK

III. USER-BASED NEEDS

Chapter III discuss the human-centered design approach used in the project. The stakeholder analysis is explained in detail. Requirements are outlined and described. At the end of the chapter the functional analysis is presented.

A. HUMAN CENTERED DESIGN

Human-centered design (HCD) was used in conjunction with traditional systems engineering processes in order to develop rapidly a product for stakeholders. HCD is a way to develop a product form the design to prototype phase based on user needs. This is all done at a relatively low cost to the early phases of design.

One of the major keys to HCD is design thinking. It is a critical relationship that allows HCD to be an effective design tool. Design thinking has multiple characteristics to aid in system development shown in the list below. These characteristics are essential in bounding the solution and scoping the problem during early stages of development. Critical to design thinking is that the development of a solution must be centered on human needs. Solutions must be based on user needs using a multi-disciplinary approach (Salem 2015). Once the user needs are defined, initial low fidelity prototypes must be created and tested. These early low fidelity prototypes will help drive requirements. Prototype-test-prototype method drives this iterative process. The following list describes the characteristics of design thinking (Salem 2015):

1. Innovation – Solutions come from a creative space where human intentions, business viability and technology feasibility intersect. Nontraditional approaches and thinking outside the box are highly encouraged.

    a. Bounded – Focused problem solving. Understanding the scope of the problem space.

    b. Solution Oriented – Focused on the solution and not the analysis. Come up with the solution quickly.

2. Human Centered – Make the human user the primary source for requirements, prototype acceptance, and all aspects of the solution. This is an attempt to reduce the requirement creep from the user in the final stages of development.

    a. Empathy – The problem should be understood from the user's point of view.

    b. Empirical – Use the user's experience and observation to determine the solution instead of theory and data analysis.

    c. Contextual – The solutions should always be related to the problem.

    d. User Need – The actual user needs should drive the requirements and functionality of the solution designed.

    e. Multi Discipline – All engineering disciplines should be a part of the initial design process to reduce integration issues during the final development.

3. Prototype – Build multiple cheap prototypes that attempt to solve the problem early and often. Prototypes will help to define the requirements.

    a. Low Fidelity – Prototypes that are quickly and easily made in order to test the solution should not use a large amount of resources.

    b. Disposable – Multiple cheap prototypes will be built and tested before a final design is decided.

4. Iterative – Build a prototype and then test it. Redesign the prototype based on the test results and test again. Continue this process until all requirements are teased out of the design.

a. Early and often – The more low fidelity prototypes that are built and tested, the better the solution will meet all the requirements of the user.

b. Model Test Model – Redesign the model based on testing feedback.

c. Evolution – The solution will evolve as more iteration occurs.

These characteristics lead to a lot of benefits in the development of a system. It leads to a lot of ideas at the early stages. This allows for a large problem space to be explored. This problem space defines what the user needs are. Then the actual user is queried for his need based on the problem space exploration. This leads to multiple cheap prototypes that can be tested without the time and money needed to develop a single high fidelity prototype that might need to be changed after testing. The benefit of this iterative process is that there will be less requirements creep when a final prototype is designed, which saves the program a large sum of money.

This project tailored the human-centered design process to design the Risk Assessment System GUI. As seen in Figure 12, the project started with the problem statement where efforts focused on identifying the problem. The next step was to identify and engage stakeholders. Users were interviewed in order to understand the problem further. They described the tactics, systems, kill chain, and thought processes that are relevant to the problem statement. This was used to identify the functionality and requirements required for the tool. Next step was to build and share the low fidelity prototype to the stakeholders. Based on feedback from stakeholders, requirements and user needs were updated. This was an iterative process, and feedback was essential in the human-centered design. The requirements shown later were updated multiple times because requirements fell out of prototype testing.

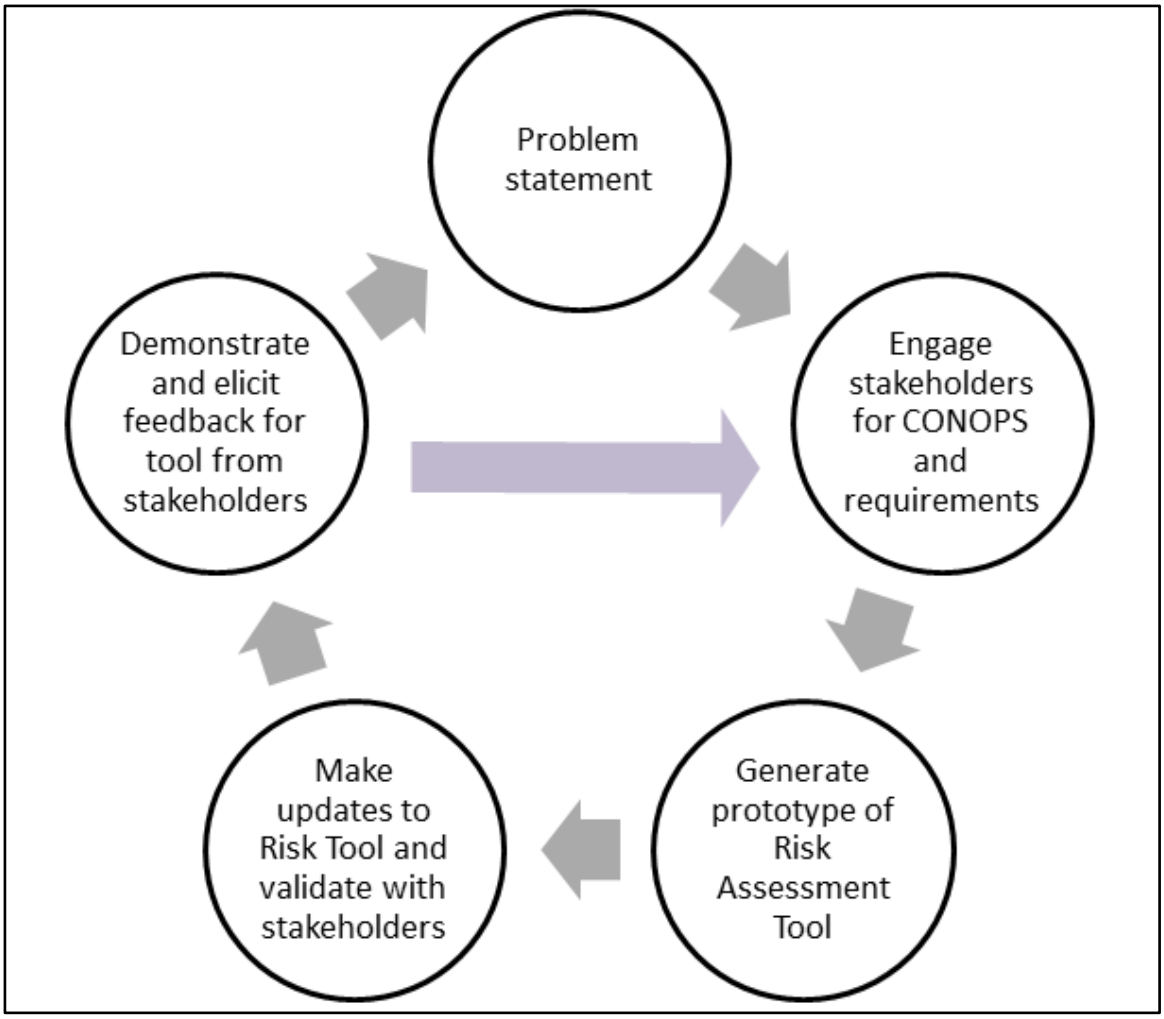

Figure 12. Tailored Human-Centered Design Process

B. STAKEHOLDER ANALYSIS

Section B describes stakeholder analysis. This section includes all stakeholder interaction from identification of stakeholders to affinity diagrams of user needs. Careful examinations of the interviews helped to refine the limitations to scope as well as defined the user needs that drove the prototypes.

1. Stakeholder Identification

Once the problem statement was defined, the stakeholders had to be identified. The primary purpose is to give the warfighter the best information available to assess the risk of a UAV swarm attack. No one stakeholder could do all that is needed in order to

provide the context and user needs in order to accomplish this. This led to a need for multiple stakeholders over a spectrum of skillsets. Table 4 shows an initial list of stakeholders and the needs relevant to their position. Surface ship commanding officers, task force commanders, and fleet commanders are responsible for the safe execution of missions on a daily basis. The areas of responsibility (AOR) of these missions are ever changing, and these commanders have to reassess the risk with new intelligence information. They are the final decision makers on risk at the relative levels and would be the primary benefactors of the tool. They have to understand the inputs and outputs explicitly to make the best decision. Therefore, if the system does not have the proper input selections or the output does not present the information in a useful manner the system would be of no use. Planning teams and intelligence teams would most likely use this for planning purposes before executing a mission. This system would be used to consolidate the information that pertained to the risk of a UAV swarm attack against a ship. They would use the output to present to the decision makers so that they could make their determination.

Table 4. Stakeholder and Needs

Stakeholder (User)	**Needs**
Surface Ship Commanding Officers, Carrier Task Force Commander, Fleet Commander	To get a consolidated report on possible threats to make informed decisions
Operational Planning Teams, Area of Responsibility Commanders, Air Warfare Commanders	To get damage assessment and make sense of data (intelligence of types of threats, how they are collected, the number of systems, damage evaluation based on data)
Office of Naval Intelligence, JIAMDO	To implement the tool solution and integrate other data and tool sets
Shipboard Weapon and Detection System Program Offices, Navy Operational Test and Evaluation	To understand how to improve the systems and to validate that the system meets the need

In order to refine the problem and bound the solution, the authors associated the initial needs based on the normal functions the stakeholder preforms, as described in Table 4. These needs are at a high level and were modified based on interaction with the stakeholders. These interactions were crucial to the development because they determined the initial requirements for the simulation and user interface. Stakeholders tested an initial prototype and their feedback was integrated into both the model and simulation (see Figure 13) to drive out new requirements.

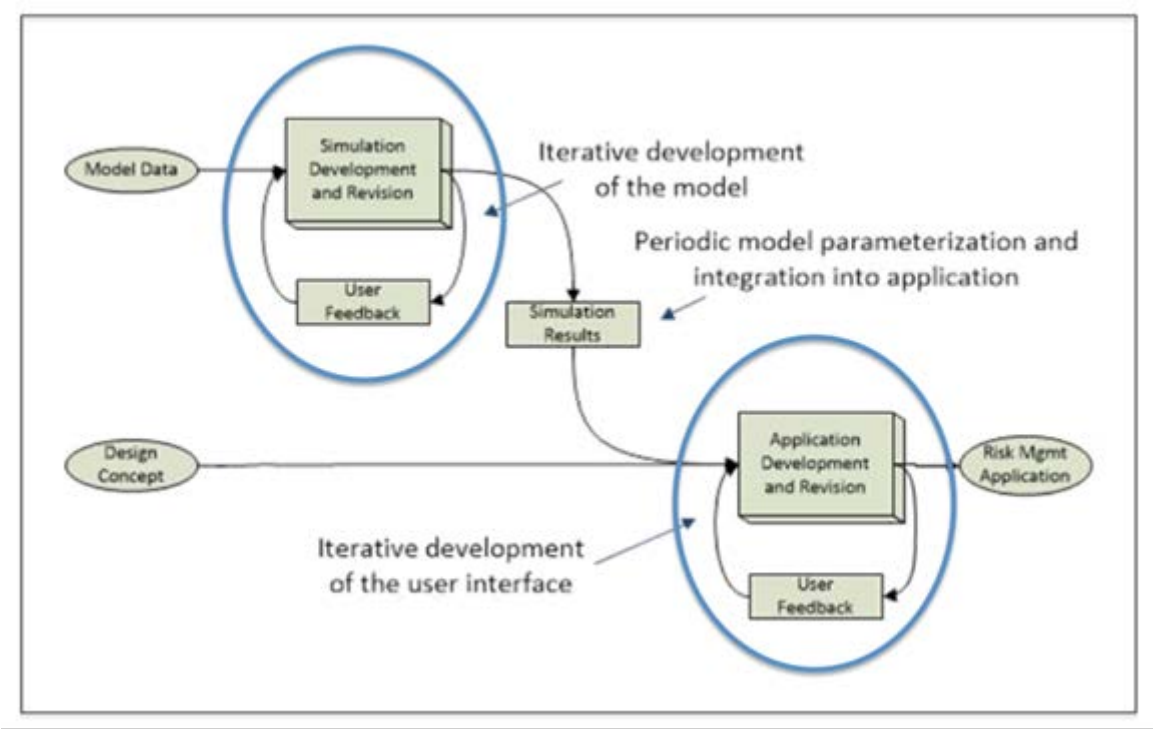

Figure 13. User Feedback in Development

2. Limitations to Scope

Before the stakeholders could be engaged, an initial attempt to limit the scope of the simulation and GUI was conducted. This was the initial step in bounding the solution. Figure 14 is based on the problem statement and CONOPs and represents the initial boundary of the system. This boundary was used to help define the scope of questioning of the users; scoping the system space allowed for more solution-oriented interview questions and testing.

This context diagram included the GUI, model/simulation, and the external factors that are relevant to the system. The far left of the diagram is the external inputs into the system. These are some of the proposed factors that would be entered by the user, such as number of ships, number of UAVs, ship and UAV types, and geographical location. Based on these inputs, the center box of the diagram is where the system will process the information to produce the output at the far right.

The center box contains both the GUI and Model context. The GUI will process the input and deliver the output based on the response equations. It will provide the output that will be a visual representation to risk. This cannot occur without information that was provided by the model context. The model will provide the data needed to build the surface response equations based on the relevant factors to UAV swarm attack. There are limitations to the Model based on project scope. These scope limitations in the model will affect the allowable inputs by the user to the GUI.

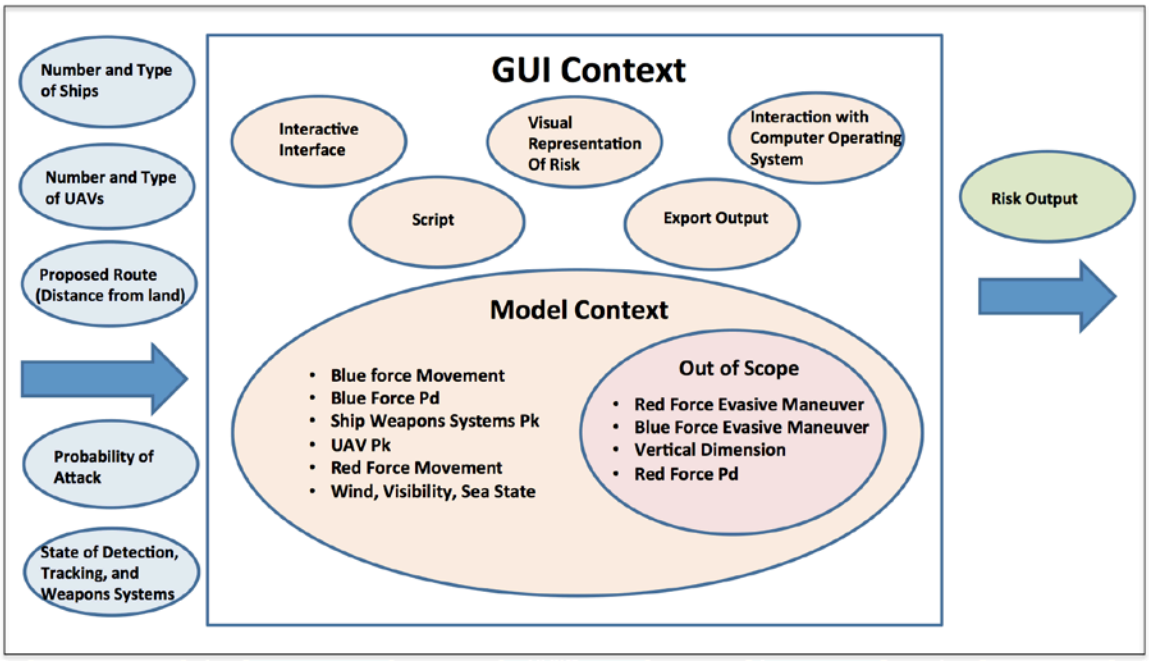

Figure 14. Overall System Context Diagram

3. Interviews

The stakeholders were identified, initial needs were addressed, and the solution was scoped; then the interaction with stakeholders started in order to understand their needs. Human interaction is the crux of HCD. The first part of human interaction is the interview stage.

The authors engaged in exercises of design thinking in order to develop the right questions. The first was to develop point of view (POV) statements and the second was to develop: “how might we” (HMW) statements. POV statements were used to gain insight into the user and his needs (Salem 2015). Simply said, User + need=insight (Salem 2015). An example of the POV statements that were developed is seen in Figure 15. Multiple POVs were developed for each user.

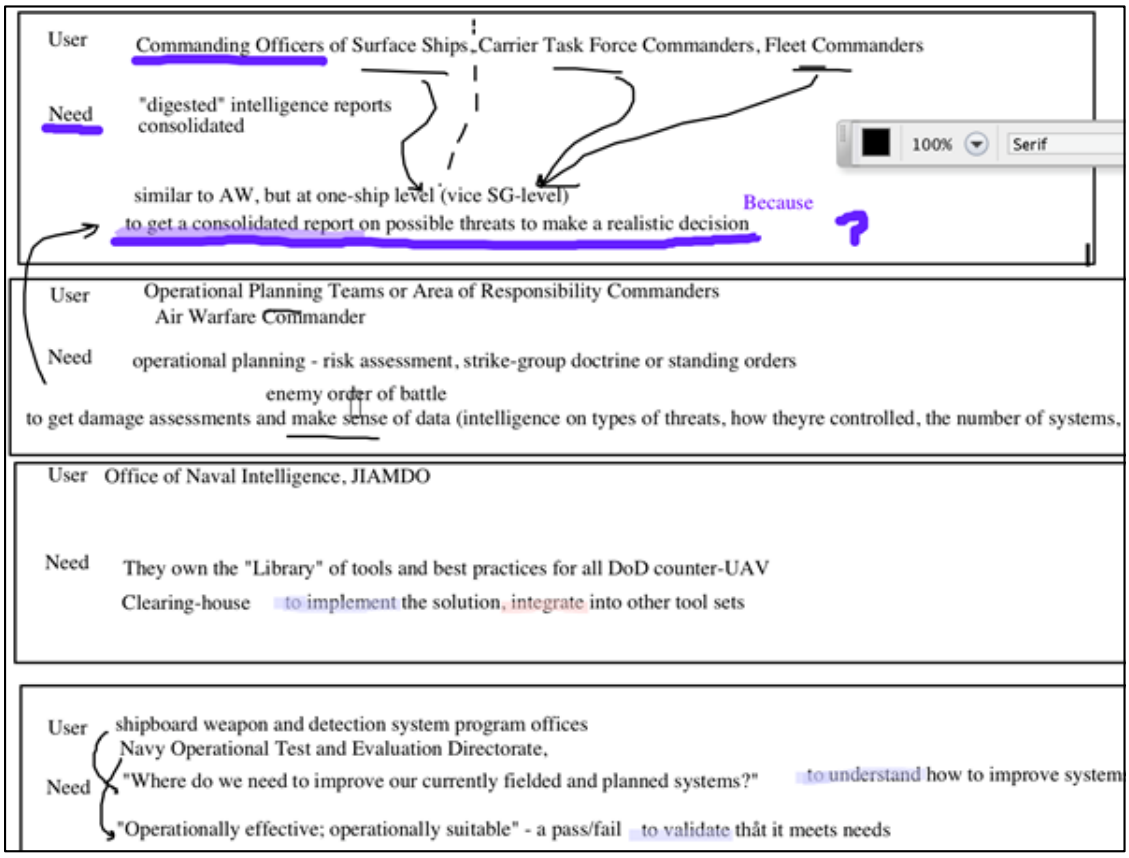

Figure 15 Point of View Example

"How might we" statements are questions that enhance the understanding of the problem space (Salem 2015). This allows for a better understanding of the impact of the users need. An example of the HMW exercise used on this project is seen in Figure 16. This HMW exercise also lead to the initial development of the visual organization of UAV threats and their risk output, Figure 17.

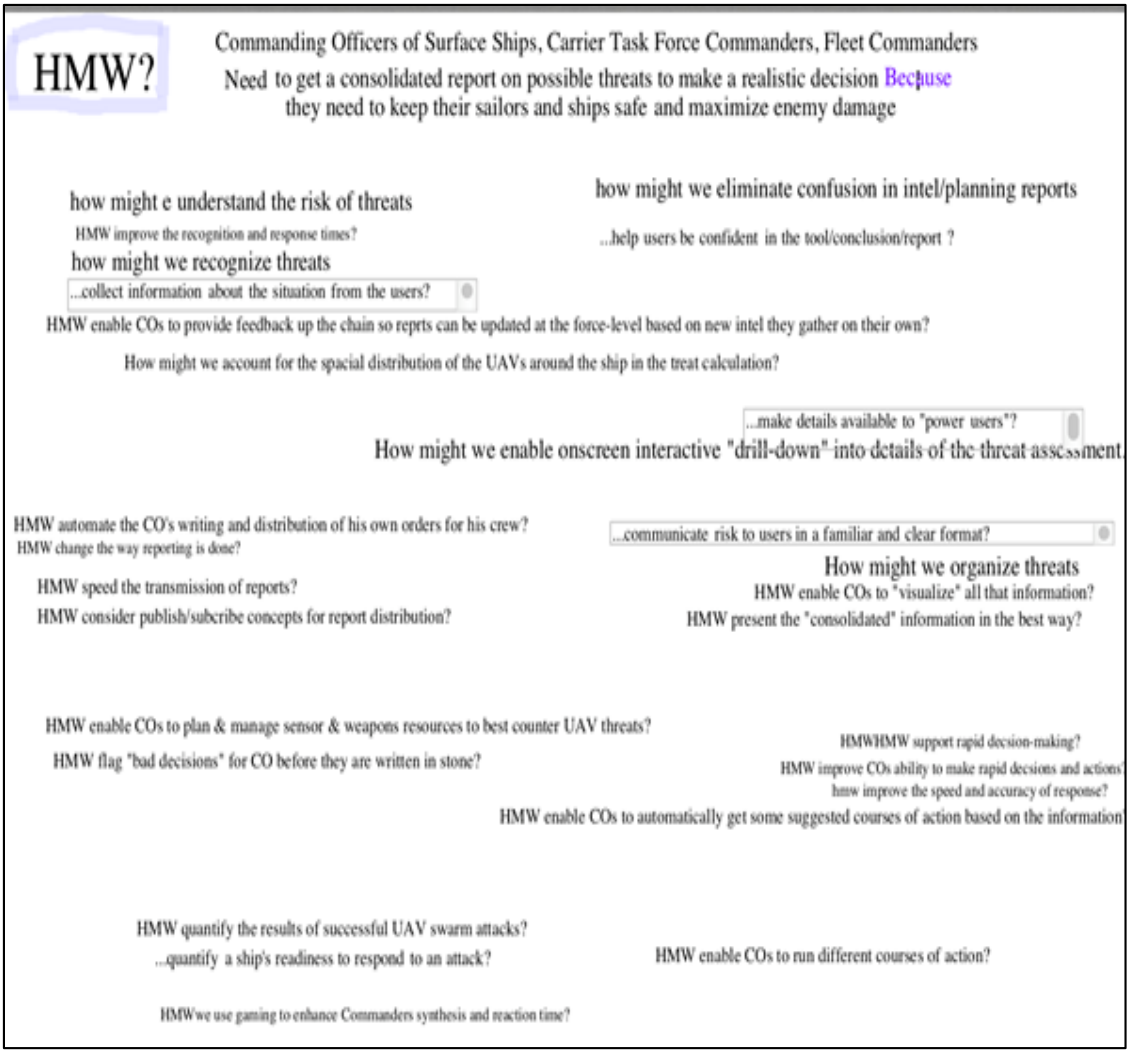

Figure 16. Example of "How Might We" Questions

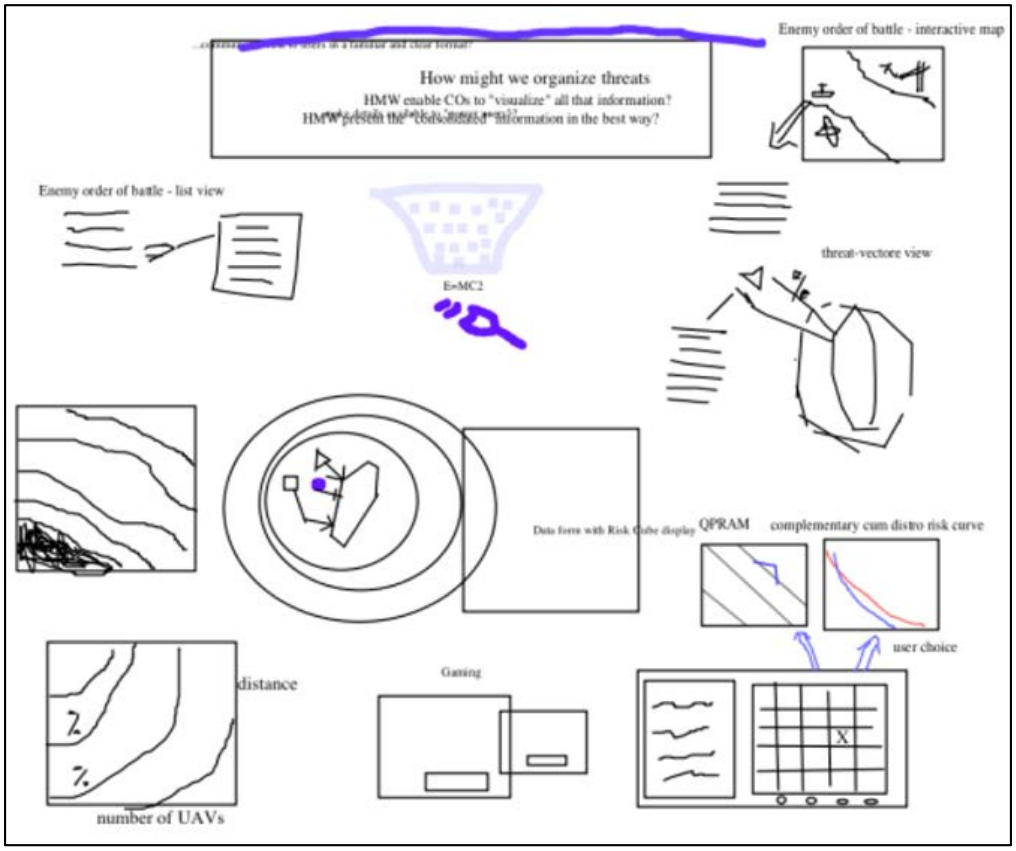

Figure 17. HMW Initial Visual Representation of Output

The visual representations of how the authors would organize threats in Figure 17 were actually some of the earliest examples in the prototyping process. For example, the upper right corner is an early example of how one would geographically represent the user and threat on the same map. The lower right hand corner is the development of how the interface would show the risk using risk cubes or text. One of the major purposes of this was to get as many low fidelity examples of threat organization as possible so that the authors could develop the visual representation interview questions.

The POV, HMW, and CONOPS were finally used to develop the questions seen in Figure 18. This set of questions was written to allow for an open discussion of the stakeholder needs based on their position. The authors solicited the stakeholders opinions based on the system intended use.

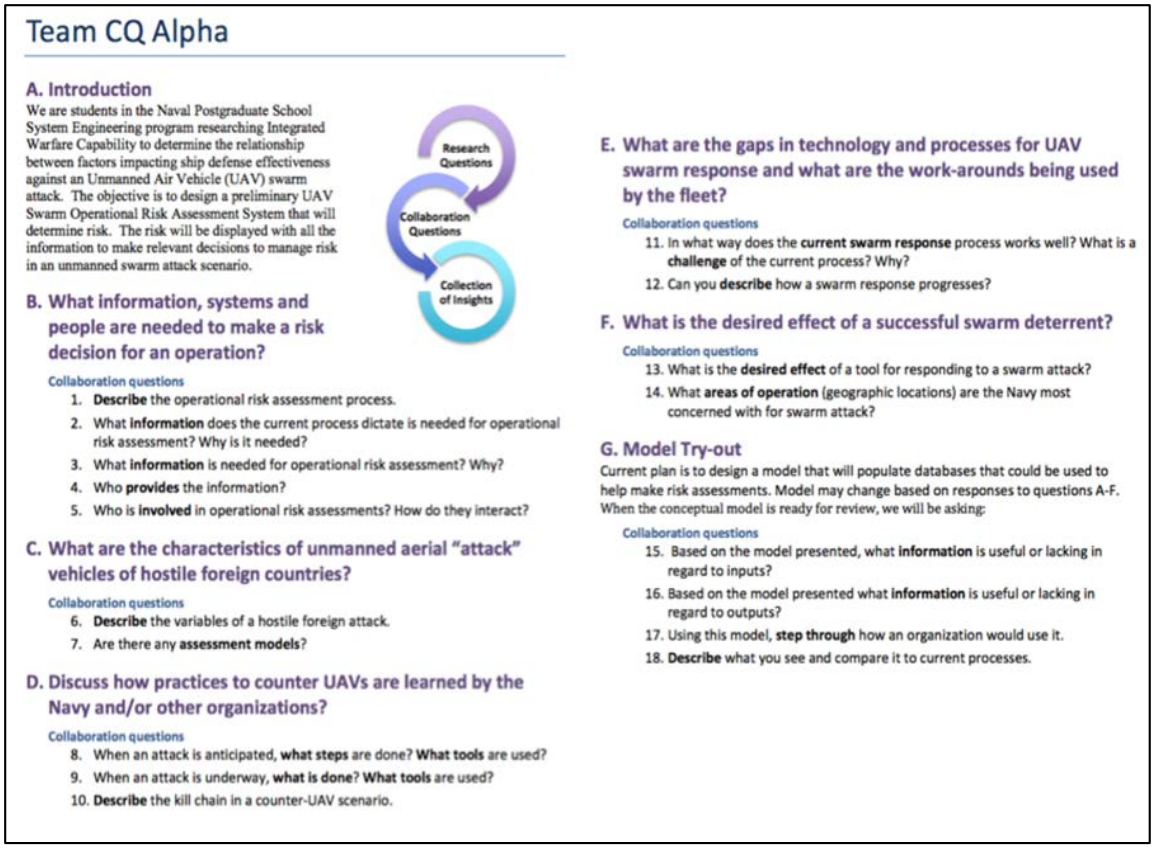

Figure 18. IRB Approved Questions

4. User Needs

The authors interviewed former surface ship commanding officers, task force commanders and planning team members over a series of multiple interviews. Interviewees included multiple ranks within the kill chain from CAPTs to LTs to gain a wide variety of surface warfare perspective. Both the tactical and strategic areas of responsibility were examined.

Interviews were conducted on the phone and in person with multiple groups of users. The authors collected notes on the users' answers. The interview results were dissected into user needs. Those user needs were placed into like groups. A full affinity diagram of the user needs based on interviews can be seen in Figure 19. The reader is not intended to be able to read each note in Figure 19 but instead should have an idea of the contributions and iterations the authors had during the project to capture, define and group user needs from the interviews.

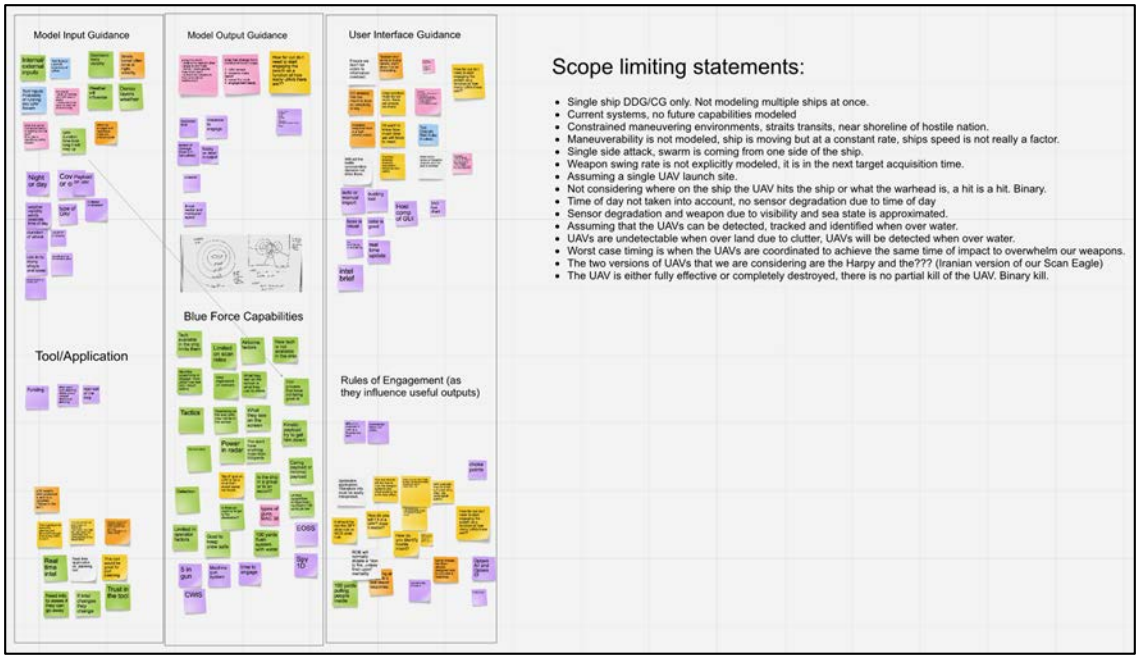

Figure 19. User Need Affinity Diagram

This affinity helped organize the user needs for each section of the system. The groupings were defined for model input, model output, tool/application, user interface, blue force capabilities, and rules of engagement. Based on these groupings and the interviews, more scope limiting statements were determined.

Blue force capabilities (Figure 20) and model inputs (Figure 21) could be further combined in order to describe the user needs for the model. The major influencing factor of blue force capability was the weapons systems available to counter the UAV swarm. The range, availability, and readiness of the CIWS, five-inch gun, and MGS will determine the weapon system the Commanding Officer (CO) will chose to engage with. The other major factor from the blue force was how the UAVs are detected. According to the COs, the detection system may have an issue with detecting and identifying a UAV in the clutter of a littoral environment. This littoral environment's impact on the capability to detect and track was some of the primary needs for user input. The environmental factors such as time of day, visibility, sea state, weather and clutter were of the greatest concern to the user. Weather and clutter were too difficult to quantify for the purposes of the model and were deemed out of scope.

Figure 20. Blue Force Capabilities User Needs

Figure 21. Model Input Guidance User Needs

The other aspect of model input that was of concern to the stakeholder was the type, number and location of UAVs. They did not want to have to enter into the software the flight envelopes and capabilities of the UAV. Therefore, the simulation/model would have to be run for a large range of capabilities and flight envelopes. This will allow for a single input of type of UAV. This is the same for the selection of ship type. However, they would like the ability to indicate what systems are functioning. The model will have to be run for a variety of different weapon system configurations.

User interface, Figure 22, and tool/application, Figure 23, can be combined in order to address the needs of the GUI. In order to trust the tool, users wanted to keep the man out of the loop as much as possible. They saw UAVs similar to being mines in the sky. There are needs to have a tool to navigate safely around these “sky mines.” This tool would be used for planning purposes not for necessarily real time deployment. COs are more concerned with countering the attack than using a tool if the attack is underway. This tool would aid in battle commander decisions but would not drive the decision. COs already have too much to look at, and they need a simplified output. Users said the decision makers liked visual representation over text, but want ability to see the text that backs up the picture if needed. Use of color is good as long as it is not overwhelming and open to multiple interpretations.

Figure 22. User Interface Guidance User Needs

Figure 23. Tool Application User Needs

Time to react was one of the most unexpected results of the interviews. They were uncomfortable with attacking a UAV when they first detected it. It is very difficult to identify a UAV. It is even more difficult to identify a hostile act of a UAV. Because of this, the CO wants to know how much time he has to engage before he is vulnerable. This also gives the ship the ability to maneuver out of the threat zone or to get sailors safely inside, under cover. Rules of engagement (ROE), Figure 24, influence every decision that the CO makes. The CO would rather maneuver out of the threat zone or get sailors safely inside than to engage the target as a first response. Therefore, the tool has to aid in making decision that is in accordance with the ROE.

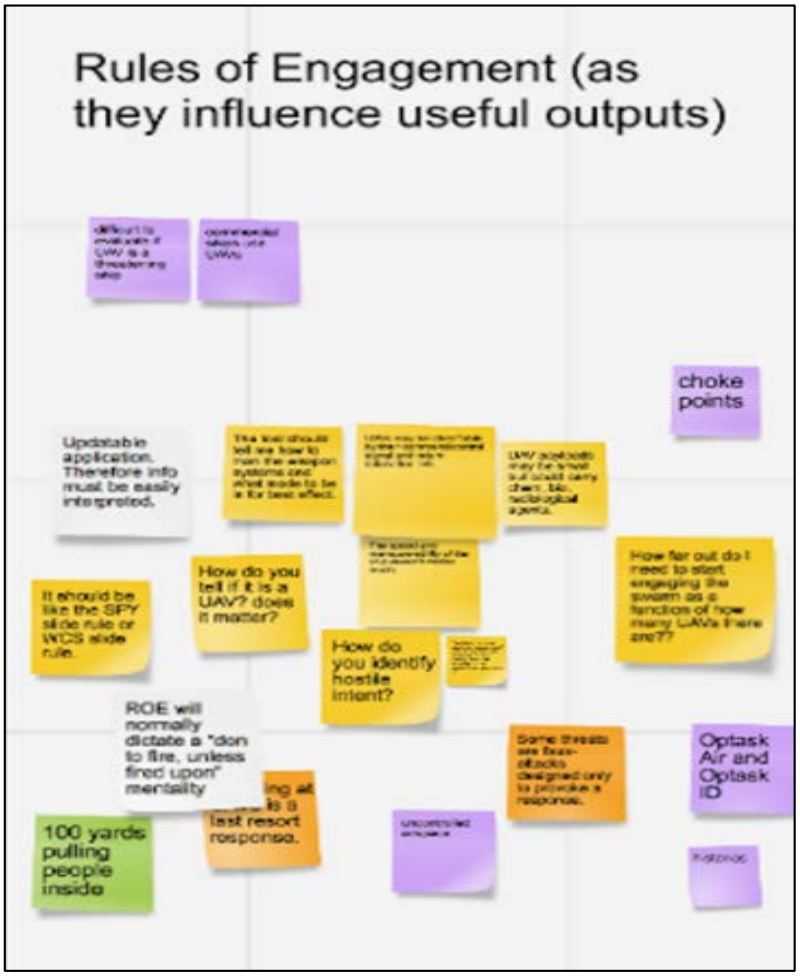

Figure 24. Rules of Engagement User Needs

In summary, there were a lot of user needs that were expected and not expected. Maps with visual representation of hostile threat were thought to be a great way to quickly assess the risk geographically, as expected. Inputs such as type, number, and location of UAVs are needed along with types of ships and combinations of ships in order to produce a unique risk output to their situation. Some of the things not expected were the use of visual representation of time to react. This gave the ship the ability to maneuver instead of engage. The user did not see the product as a real-time tool. Users were also very concerned about how to identify correctly a hostile act, which would make them hesitant to engage. High level decision makers do not trust probabilities of occurrence to be a function of the tool. There is too much human bias in probability.

C. REQUIREMENTS

Section C discusses the requirement development process used for the system. The GUI input, functional and output requirements used to create the tool; a decomposition of discrete-event output simulation requirements. In addition, a complete list of the system requirements was included.

1. Requirements Development Process

As discussed earlier in the User Needs portion of this report, a series of interviews were conducted in order to determine the needs of potential users and stakeholders. Once all of the interviews were completed, the authors had several meetings to group those needs into a series of affinity diagrams. Requirements for the system were derived from the needs, and traced to the functions that accomplish them. Decomposition and traceability of the requirements were developed and documented using CORE. CORE from Vitech, Inc., is a model-based system engineering tool that provides comprehensive traceability from need definition through requirements and analysis to architecture and test. Using CORE, the requirements were broken down into five main sections to include GUI Input, GUI Functional, GUI Output, Support and Other, and, Discrete Output Simulator. Figure 25 shows the high level requirements decomposition.

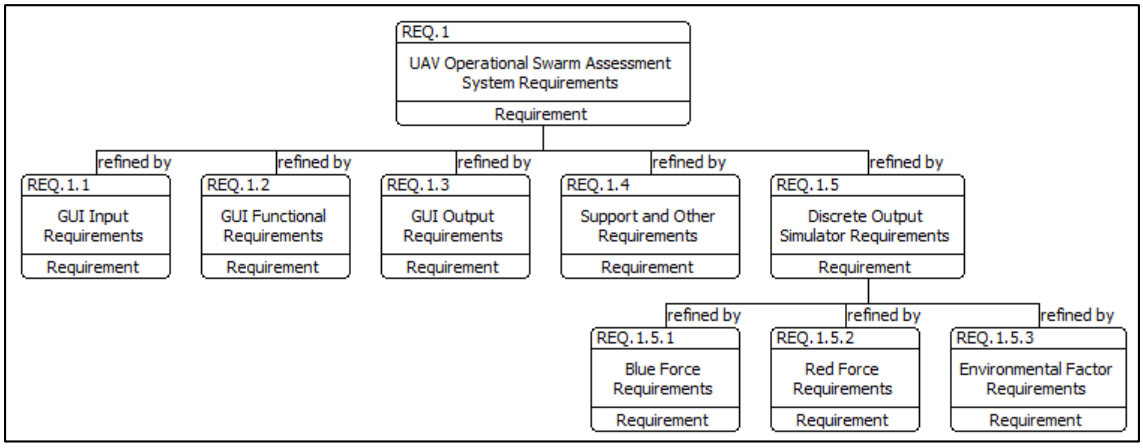

Figure 25. UAV Operational Swarm Assessment System Requirements Decomposition

2. GUI Input Requirements

The input requirements for the GUI (REQ.1.1) were decomposed or refined into nine separate requirements. All nine requirements form the basis of the Accept User Inputs Function. The input requirements for the GUI trace back to user needs for what they would like to see as inputs into the GUI. Further traceability has been established by allocating the Accept User Inputs Function, which is the software function that processes user inputs. Also in this case the Accept User Inputs Function has been allocated to the Computer Peripherals as shown in Figure 26.

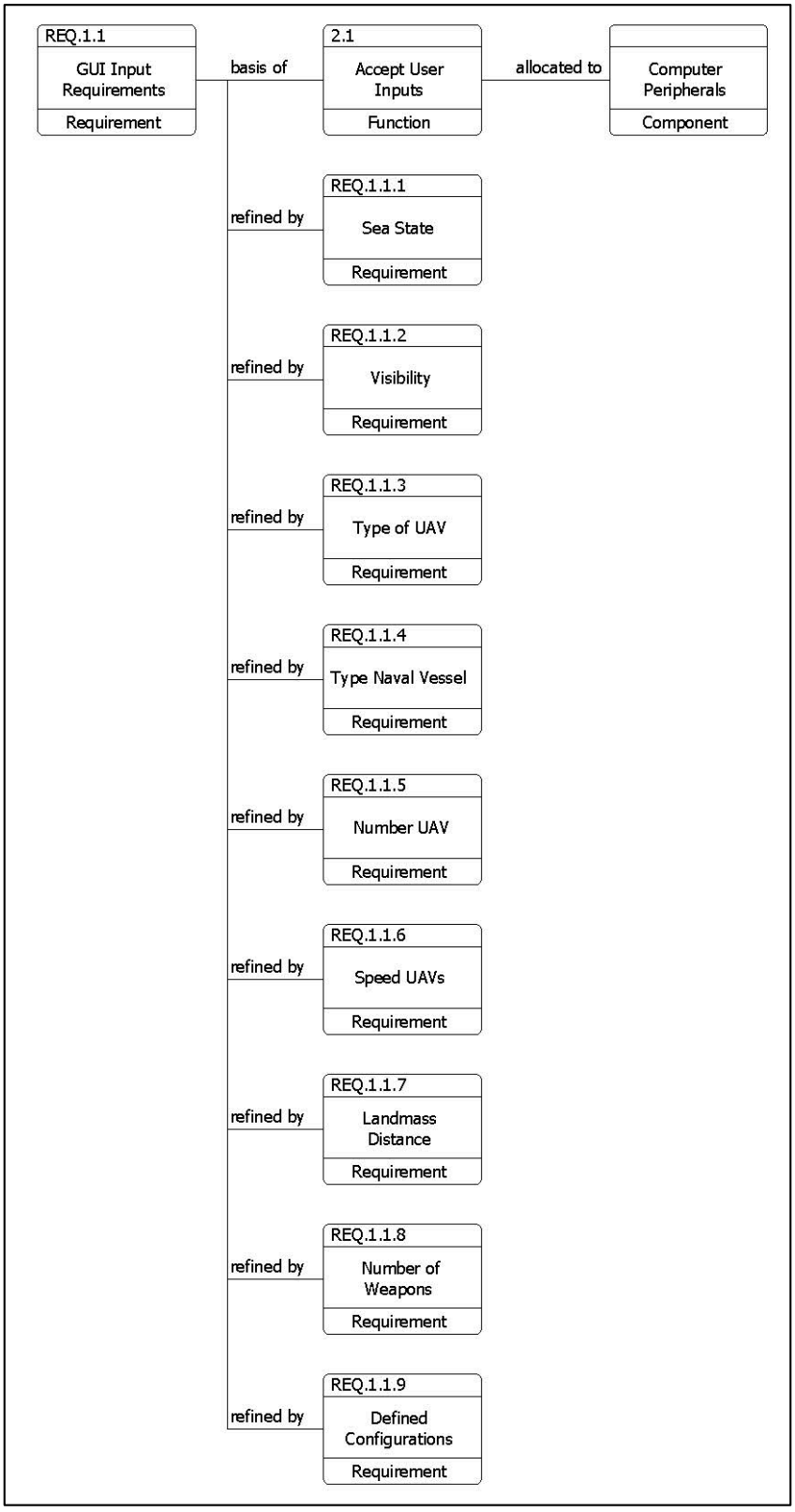

Figure 26. GUI Input Requirements Decomposition and Traceability

3. GUI Functional Requirements

There are five functional requirements for the GUI that are can be traced to five functions required by the users. These functions are the ability for the user to enter inputs for the GUI independently, perform input validation as every software application should, respond to keyboard shortcuts, save inputs, and load inputs from a file, which in this case would be the output from the Discrete Event Simulator. Figure 27 demonstrates requirements traceability as each functional requirement form the basis of a base function, which performs that action.

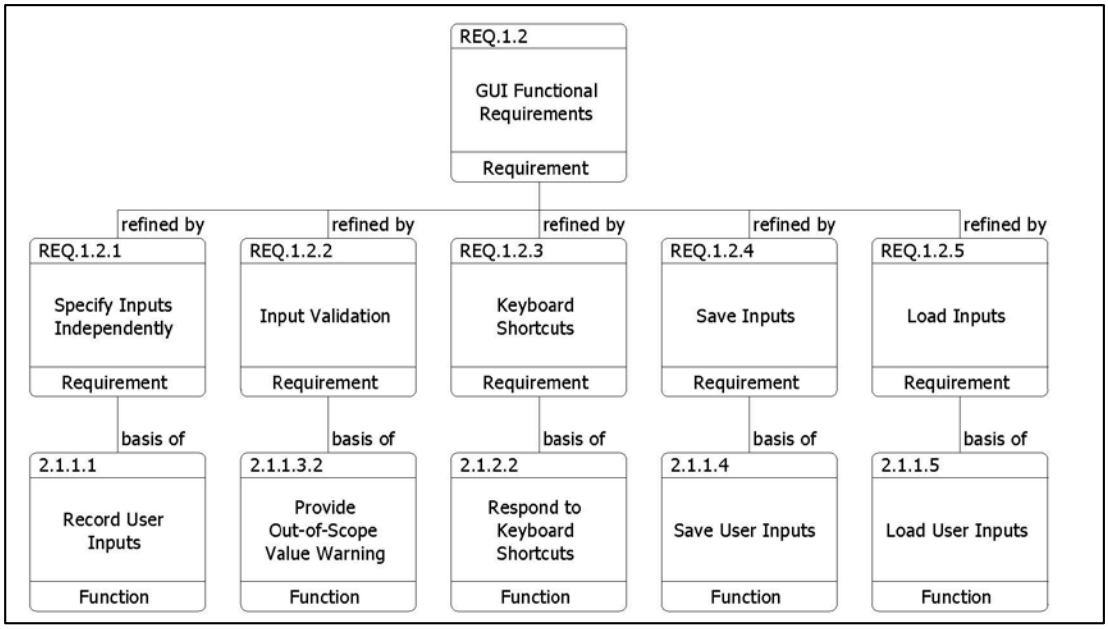

Figure 27. GUI Functional Requirements Decomposition and Traceability

4. GUI Output Requirements

How the GUI presents the results to the user is very critical for this application. As discussed earlier in, Chapter III Section B 3 Interviews, section the users requested several things that the GUI needed to be capable of providing output for. Outputs for the GUI include visually presenting reaction time versus range, displaying threat areas based on user input, producing outputs in a certain timeframe, text formatted, and in different file formats to include .ppt(x), .doc(x), and .pdf as shown in Figure 28. There are seven

output requirements that form the basis of the Provide User Outputs Function. The Provide User Outputs Function is allocated to the Software Program and Visual Display Components, which perform that function.

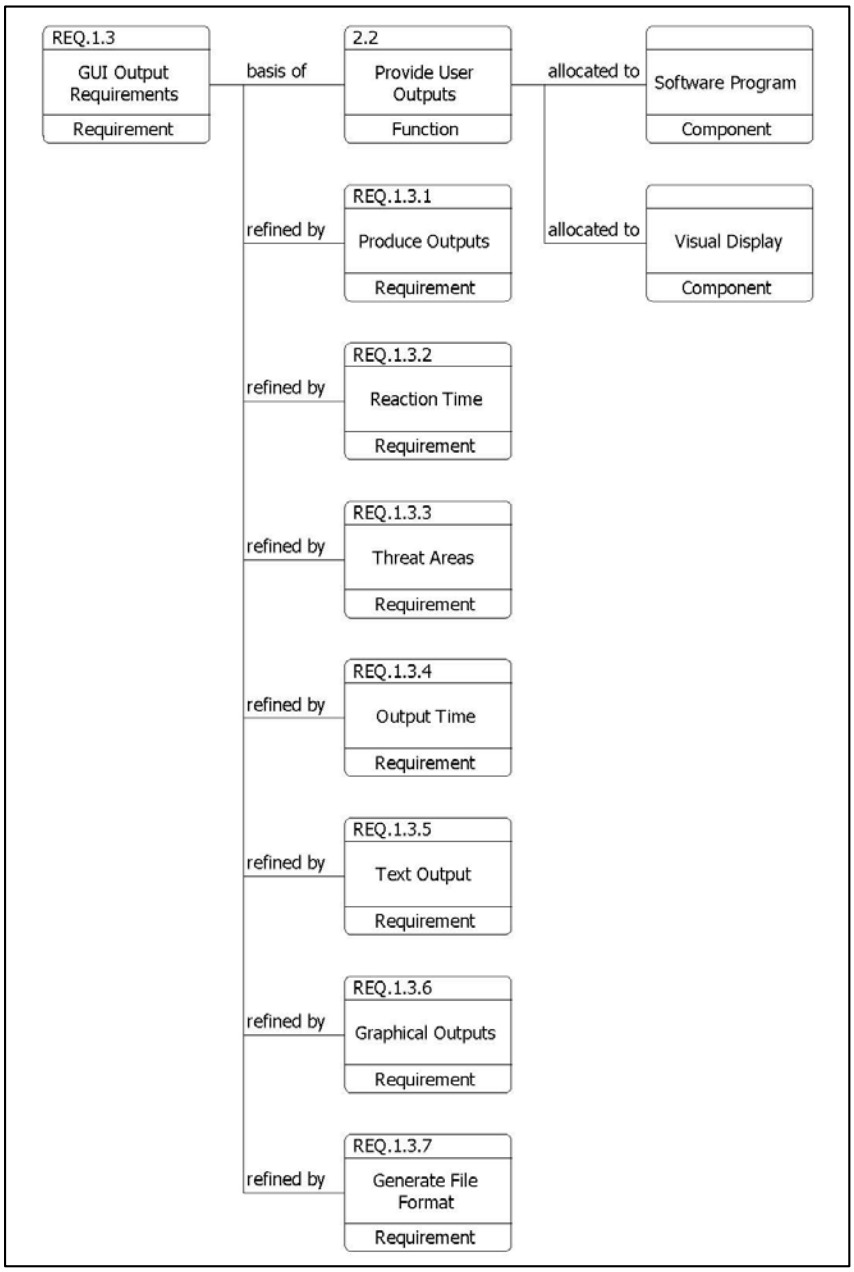

Figure 28. GUI Output Requirements Decomposition and Traceability

5. Support and Other Requirements

The Support and Other Requirements for the GUI include being able to run on a Windows® 7 or higher operating system as that is the enterprise operating system that is in used across the Navy. There is also a requirement that the GUI needs to have a modular architecture that has high cohesion and low coupling that allows for future updates to be inserted easily. Finally, the GUI needs to be simple enough to operate that a user with an intermediate level skillset could use it. During the interviews the users did not seem too concerned with the support requirements for the GUI, but these sorts of requirements are needed. Figure 29 shows requirements decomposition for the Support and Other Requirements.

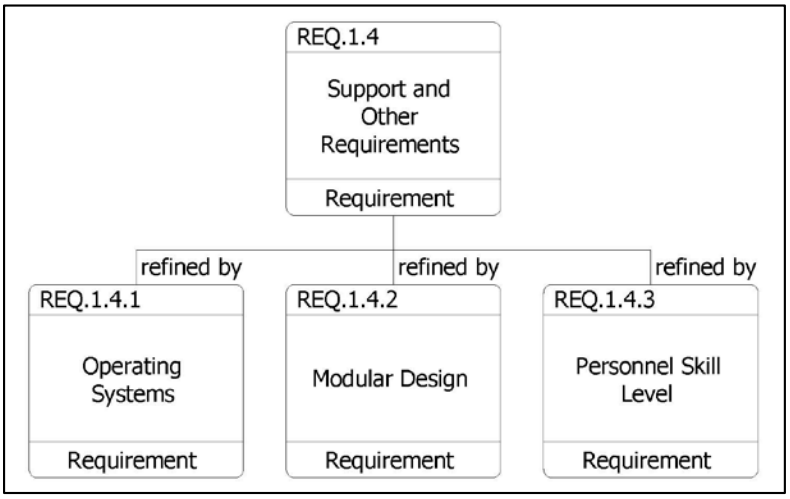

Figure 29. GUI Support and Other Requirements Decomposition

6. Discrete Output Simulator Requirements

The requirements for the Discrete Output Simulator, also known as ExtendSim model, were broken down into three high level requirements to include Blue Force Requirements, Red Force Requirements, and Environmental Factor Requirements. The Blue Force was further decomposed into five requirement sections as shown in Figure 30. The requirements under these five sections came from the user interview process.

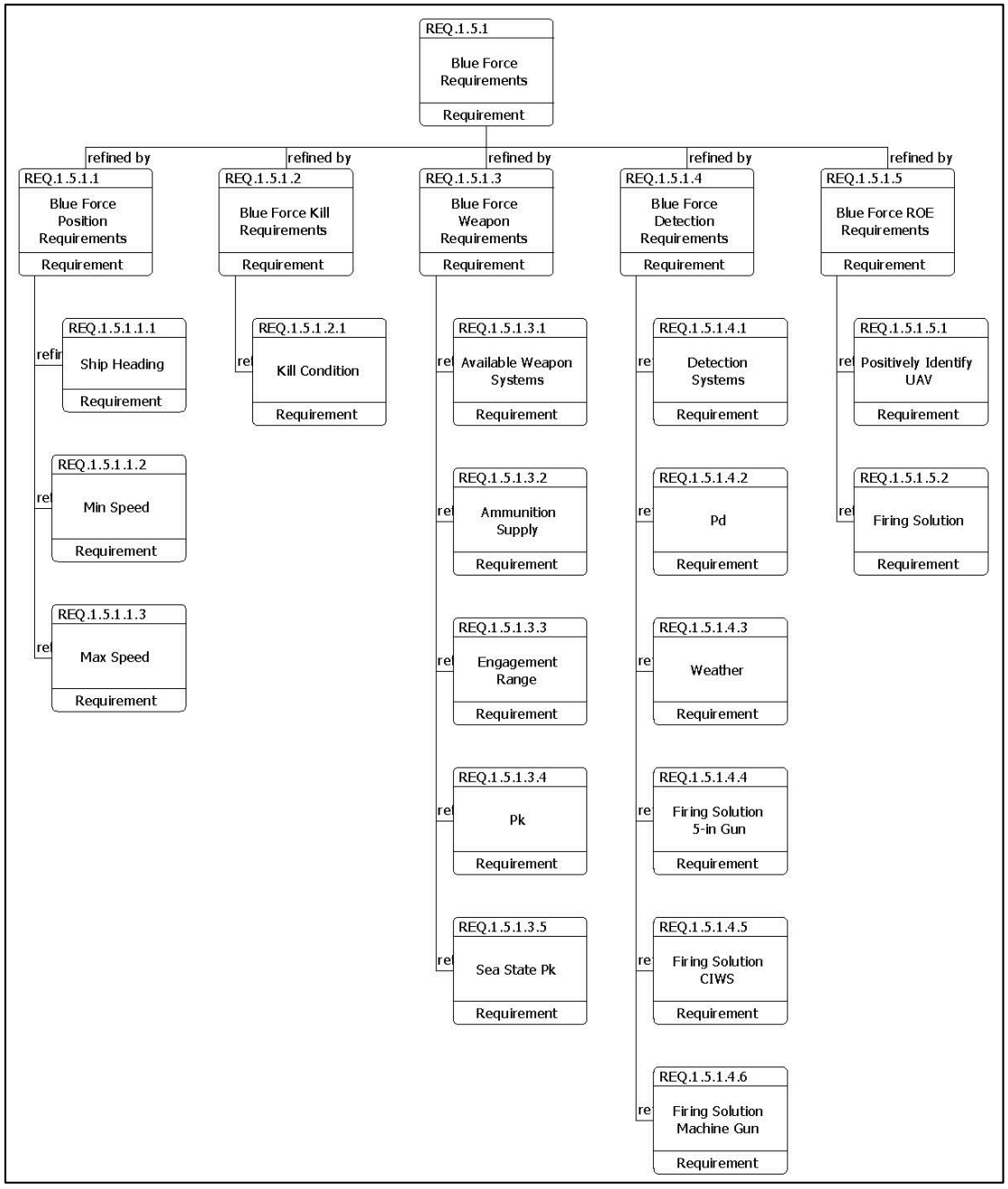

Figure 30. Blue Force Requirements Decomposition

Similarly, decomposition was also performed for the Red Force Requirements as shown in Figure 31. In this case there were also five sections that came as a result of the user interview process. In the Modeling and Simulation portion of this report, there is a more detailed description as to how these parameters are incorporated into the Discrete Output Simulator as shown in Figure 32.

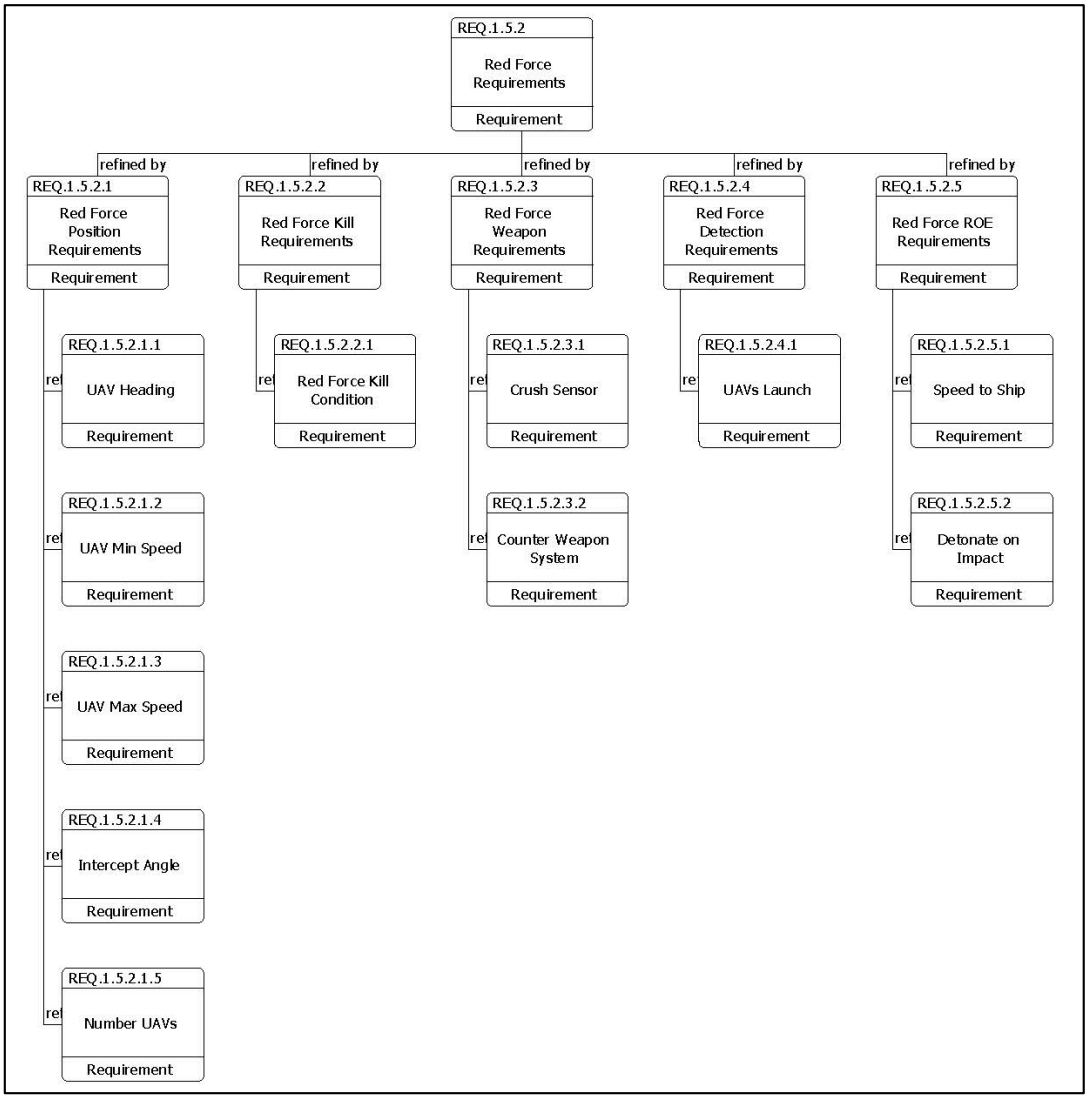

Figure 31. Red Force Requirements Decomposition

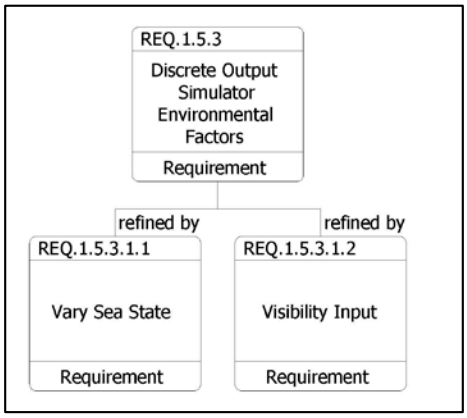

Figure 32. Discrete Output Simulator Environmental Factors Decomposition

The following is an all-inclusive list of the system requirements. The list is divided by the GUI inputs, functionality, output and other requirements. In addition, the discrete-event simulation requirements are included for the benefit of the reader and for future areas of study to improve this system.

1. Requirements

1.1. GUI Input Requirements

1.1.1. The GUI shall accept sea state as an input.

1.1.2. The GUI shall accept visibility as an input.

1.1.3. The GUI shall accept the type of UAV as an input.

1.1.4. The GUI shall accept the type of U.S. Naval Vessel as an input.

1.1.5. The GUI shall accept the number of UAVs as an input.

1.1.6. The GUI shall accept the speed of the UAV as in input.

1.1.7. The GUI shall accept the distance to the nearest landmass on the suspected threat axis.

1.1.8. The GUI shall accept the number of operational weapon systems as an input.

1.1.9. The GUI shall auto-populate the attributes for the specific ship class and UAV.

1.2. GUI Functionality Requirements

1.2.1. The GUI shall allow the user to specify all discrete-event simulator inputs independently.

1.2.2. The GUI shall perform input validation of user inputs.

1.2.3. The GUI shall respond to common keyboard shortcuts.

1.2.4. The GUI shall allow the user to save inputs to a file.

1.2.5. The GUI shall allow the user to load saved inputs from a file.

1.3. GUI Output Requirements

1.3.1. The GUI shall use outputs from the discrete-event output simulator to produce outputs.

1.3.2. The GUI shall be capable of visually presenting reaction time versus range, where the reaction time is equal to the time required to prevent a single UAV from hitting the ship.

1.3.3. The GUI shall be capable of visually displaying threat areas based upon user inputs.

1.3.4. The GUI shall be capable of producing outputs within 30 seconds of user initiation.

1.3.5. The GUI shall format all outputs in text form.

1.3.6. The GUI shall format the results graphically.

1.3.7. The GUI shall be able to print the results to a .pdf, .doc(x), or .ppt(x) file.

1.4. Support and Other Requirements

1.4.1. The system shall be capable of operating on Windows® 7 or higher operating system.

1.4.2. The system shall have a modular design to allow for updates that enhance the functionality based on future capabilities.

1.4.3. The GUI shall be useable by personnel with a minimum Intermediate Skill Level.

1.5. Discrete Output Simulator Requirements

1.5.1. Blue Force Requirements

1.5.1.1. Blue Force Position Requirements

1.5.1.1.1. The blue force ship heading will remain in a constant direction.

1.5.1.1.2. The blue force ship shall have a minimum speed 5 knots.

1.5.1.1.3. The blue force ship shall have a maximum speed of 15 knots.

1.5.1.1.4. The blue force ship shall be modeled as a 570-foot circle.

1.5.1.2. Blue Force Kill Requirements

1.5.1.2.1. A blue force kill condition occurs when any UAV enters the 570-foot ship modeled circle.

1.5.1.3. Blue Force Weapon Requirements

1.5.1.3.1. The blue force ship shall have three weapon systems available (5-in Gun, CIWS, and MGS).

1.5.1.3.2. The blue force ship ammunition shall be unlimited.

1.5.1.3.3. Each weapon system on the blue force ship shall have its own engagement range.

1.5.1.3.4. Each weapon system on the blue force ship shall have its own P_K.

1.5.1.3.5. P_K for each weapon system on the blue force ship shall decrease with an increase in sea state.

1.5.1.4. Blue Force Detection Requirements

1.5.1.4.1. The blue force ship shall have two different detection systems that are based on range (SPY-1D and EOSS).

1.5.1.4.2. The blue force ship EOSS probability of detection P_D shall decrease as visibility decreases.

1.5.1.4.3. The weather shall not affect the ability of the blue force ship SPY-1D to detect.

1.5.1.4.4. The firing solution of the blue force ship 5-in Gun shall be based on the P_D for the SPY-1D.

1.5.1.4.5. The firing solution of the blue force ship CIWS shall be based on the P_D for the EOSS.

1.5.1.4.6. The firing solution of the blue force ship machine gun shall be based on the P_D for the EOSS.

1.5.1.5. Blue Force Rules of Engagement Requirements

1.5.1.5.1. The blue force ship shall positively identify a UAV before engagement.

1.5.1.5.2. The blue force ship shall have a firing solution for the individual UAVs before weapon release.

1.5.2. Red Force Requirements

1.5.2.1. Red Force Position Requirements

1.5.2.1.1. The red force UAVs shall always be headed in a direction to intercept the blue force ship.

1.5.2.1.2. The red force UAVs minimum speed shall be 80 knots.

1.5.2.1.3. The red force UAVs maximum speed shall be 200 knots.

1.5.2.1.4. The red force UAVs intercept angle shall vary.

1.5.2.1.5. The red force UAVs shall vary in number.

1.5.2.2. Red Force Kill Requirements

1.5.2.2.1. A red force UAV kill condition occurs when any UAV is shot by any blue force ship weapon system.

1.5.2.3. Red Force Weapon Requirements

1.5.2.3.1. The red force UAVs shall only have a crush sensor (has to hit ship).

1.5.2.3.2. The red force UAVs shall have no counter weapon systems.

1.5.2.4. Red Force Detection Requirements

1.5.2.4.1. The red force UAVs shall have the blue force ship position upon launch.

1.5.2.5. Red Force Rules of Engagement Requirements

1.5.2.5.1. The red force UAVs shall fly at maximum speed to the blue force ship.

1.5.2.5.2. The red force UAVs shall detonate on impact with the blue force ship.

1.5.3. Discrete Output Simulator Environmental Factors

1.5.3.1.1. The sea state shall be an input to the discrete-event output simulator that will vary.

1.5.3.1.2. The visibility shall be an input to the discrete-event output simulator that will vary.

D. FUNCTIONAL ANALYSIS

User requirements were used to influence the high-level system design and functional analysis. A detailed breakdown of this analysis is provided below.

1. High-Level System Design

Once the high-level system requirements were determined using the stakeholder interviews, a high-level system design was determined to help better understand the subsystem interactions required for the implementation of a working prototype. As shown in Figure 33, the system components include a high fidelity discrete-event simulation model of a UAV swarm attack on a ship, statistical analysis of the discrete-event model outputs, and a computer based GUI. The authors developed a discrete-event simulation in Extend Sim 9, top left of Figure 33, using controllable factors (discussed in Chapter IV) and problem space research (discussed in Chapter II) as inputs. Then a statistical analysis was performed with Minitab 17, top right of Figure 33, with the measure of effectiveness (MOE) results of the discrete-event simulation. The statistical analysis goal was to develop response equations to be used as the building blocks of the GUI dashboard created in MATLAB (bottom of Figure 33). The GUI dashboard (discussed in Chapter V) used the problem space research (discussed in Chapter II), user inputs and stakeholder requirements (discussed in Chapter III), and the response equations (discussed in Chapter IV) as inputs to create the risk assessment tool outputs.

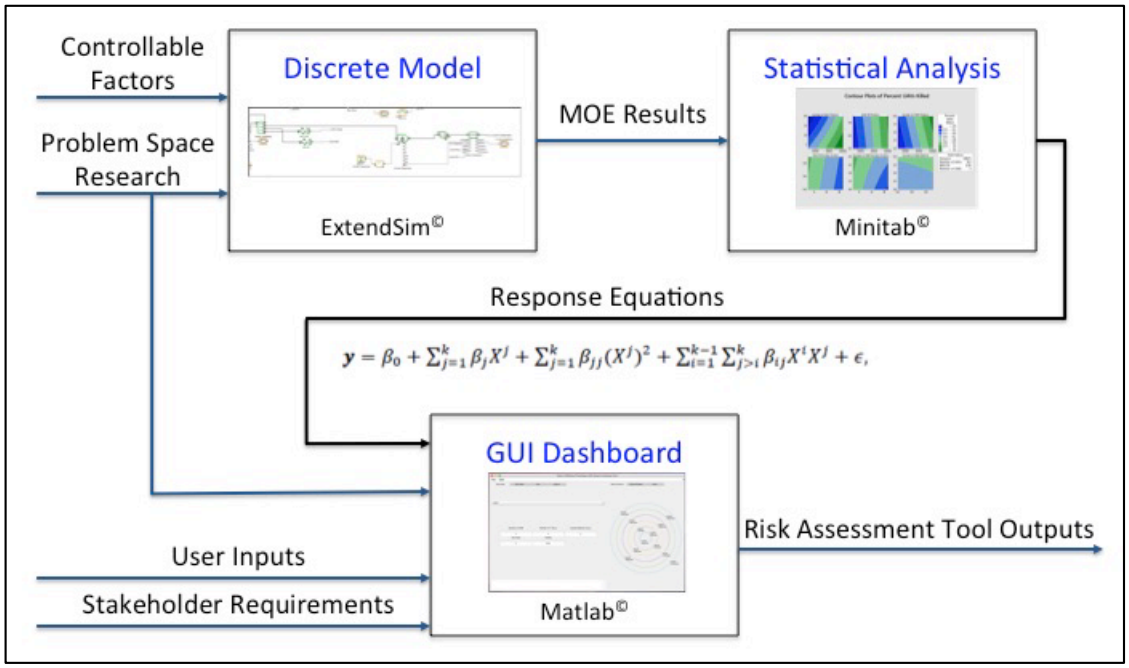

Figure 33. High Level System Design

Given the complexity of the discrete-event simulation and the statistical analysis of its outputs, it was determined that the GUI should be a stand-alone subsystem that utilizes response equations in order to produce outputs that fulfill user requirements. By making the discrete-event simulation and the statistical analysis separate from the GUI, task-specific commercial software could be used to perform the required work. For the purpose of this project, ExtendSim was used to create and run the discrete-event simulation, Minitab was used to create the design of experiments (DoE) and statically analyze the results, and MATLAB was used to create the GUI dashboard with which the user ultimately interacts. Ultimately, the decision to use these specific software packages was based upon previous experience and ease of use.

2. Functional Breakdown

Using the high-level system design and user defined requirements, a functional analysis of the system was preformed to ensure that all defined and derived requirements were met. For the purpose of this project, the CORE software program was used to perform the functional analysis and track requirements. In general, the functional analysis was performed using a top-down method of all required functions and their interactions. Upon completion of the functional analysis for one level, a functional breakdown was then completed for the next lower level. Due to the functional requirements of the system, a functional breakdown was performed down to the third level. A very simplistic design architecture was used for the design of the system since the interactions amongst the subsystem functions was singular in nature.

The primary goal of the top-level functional analysis of the system was to ensure that the scope of the project and the required mechanisms were identified. As shown in Figure 34, the entire system is comprised of two main functions: Model UAV Swarm Attacks and Perform UAV Risk Assessment (GUI). As previously discussed, the scope of the system and classification level of the project was important controls for the overall system; however, a similar functional breakdown could be used for future development with these controls removed.

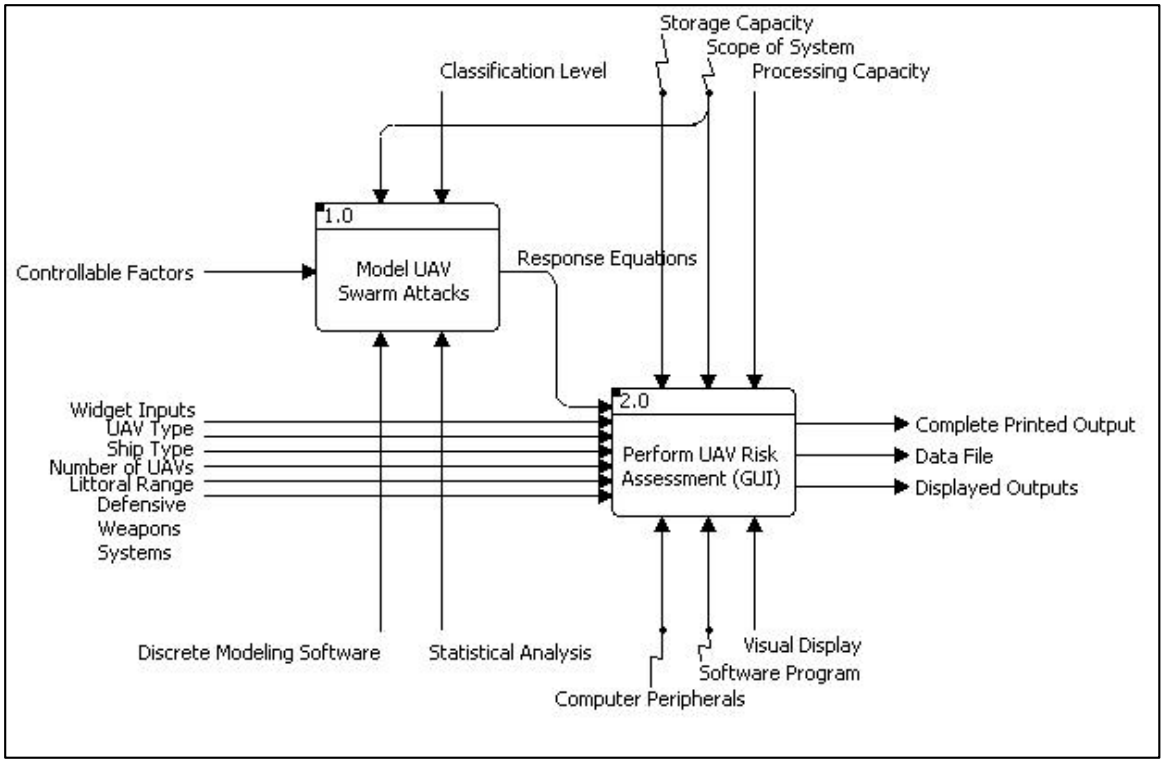

Figure 34. System IDEF0

Function 1.0, Model UAV Swarm Attacks, encompassed the discrete-event modeling and simulation including the statistical analysis of the results. As such, two mechanisms were required for the function, Discrete Modeling Software and Statistical Analysis. An IDEF0 and functional hierarchy of Function 1.0 are shown in Figure 35 and Figure 36. Research performed during the problem space exploration and stakeholder interviews were used to determine the controllable factors.

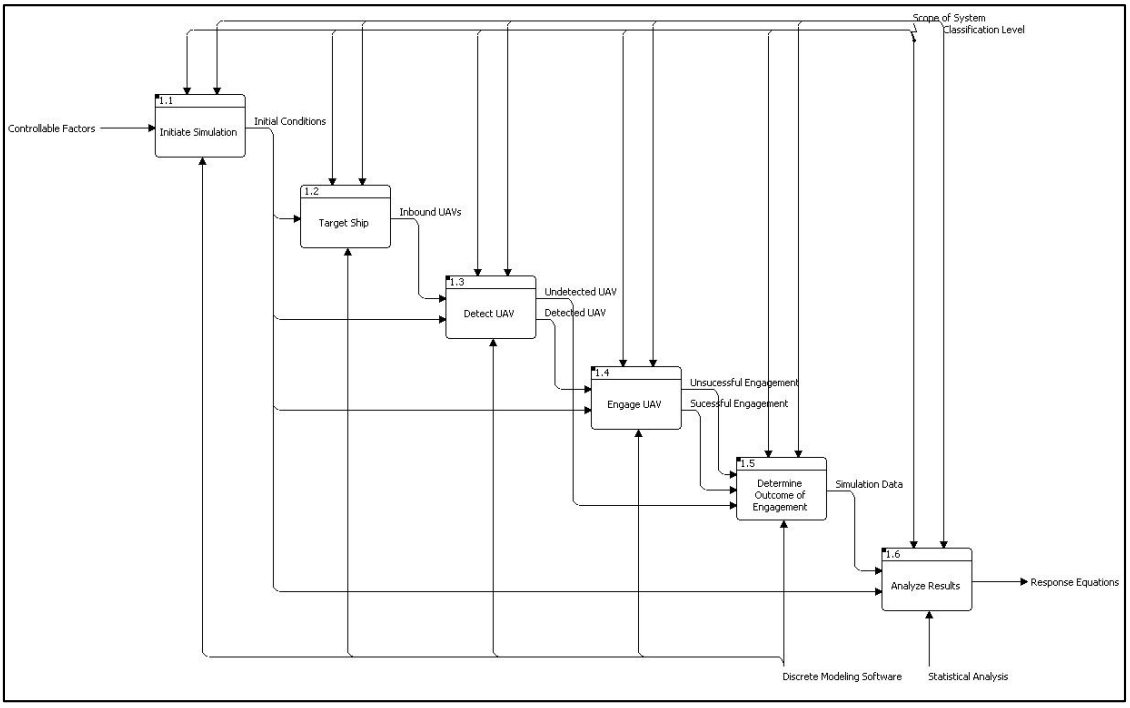

Figure 35. Discrete Model IDEF0

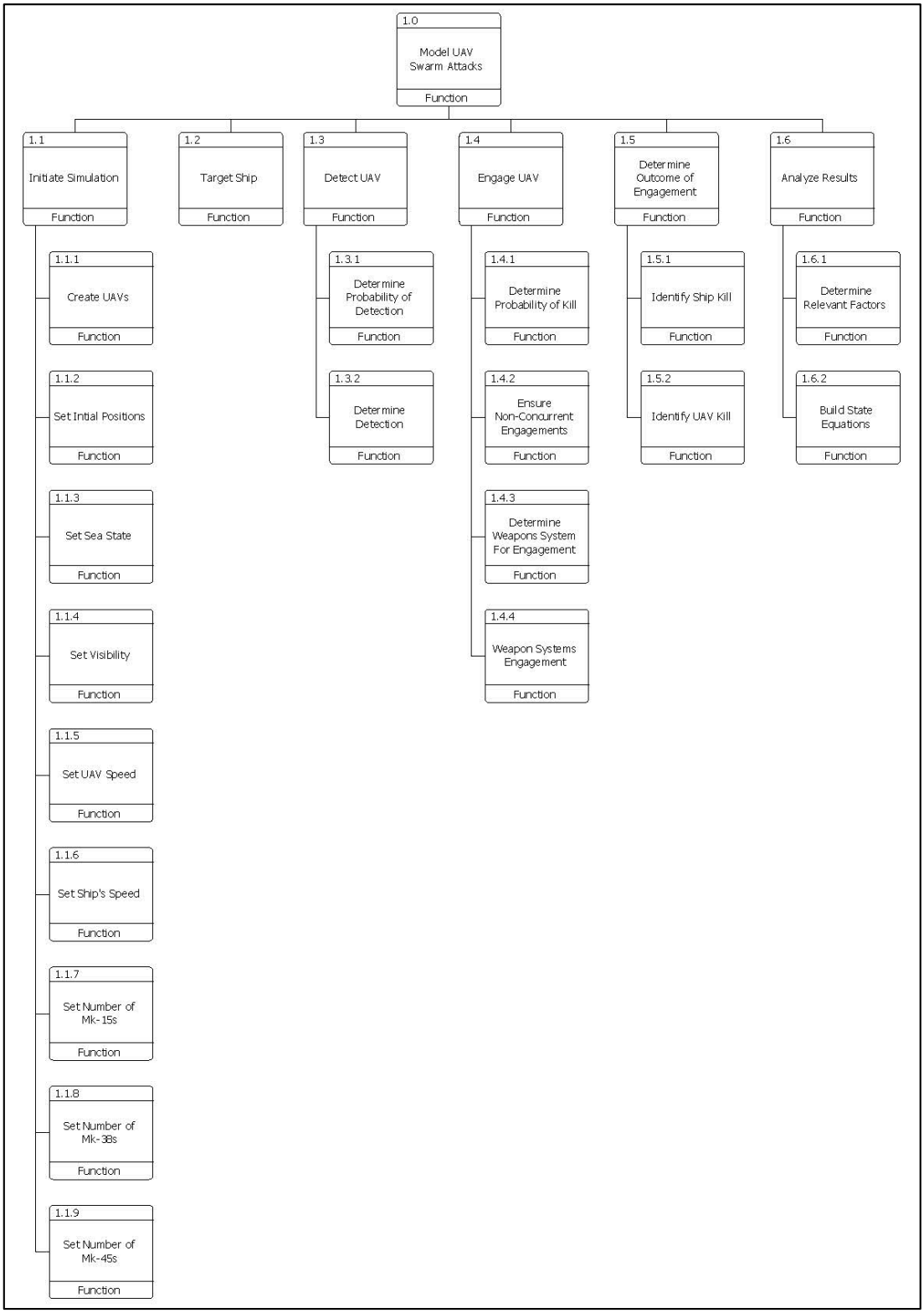

Figure 36. Discrete Model Functional Hierarchy

Once the statistical analysis was completed, the only output of the Function 1.0 was the Response Equations that were ultimately used by the GUI to create required outputs. Further discussion on the implementation of this functional breakdown, controllable factors used, and the statistical analysis performed on the discrete-event simulation results is located in Chapter IV.

Function 2.0, Perform UAV Risk Assessment (GUI), shown in Figure 37, was the main focus of the functional analysis, shown in Figure 38, with respect to the user defined functional requirements. Based upon stakeholder interviews, it was determined that user GUI interaction was a key requirement. To be more specific, user input requirements were UAV type, Ship Type, Number of UAVs, Littoral Range, and Defensive Weapons Systems. In addition, these inputs were required to be text based with limited capability for auto-filling data inputs based upon an internal library of fixed values. As discussed earlier, the scope of the system was a significant control; however, given that all data used for the inputs is open-source, classification level was not a control for the GUI.

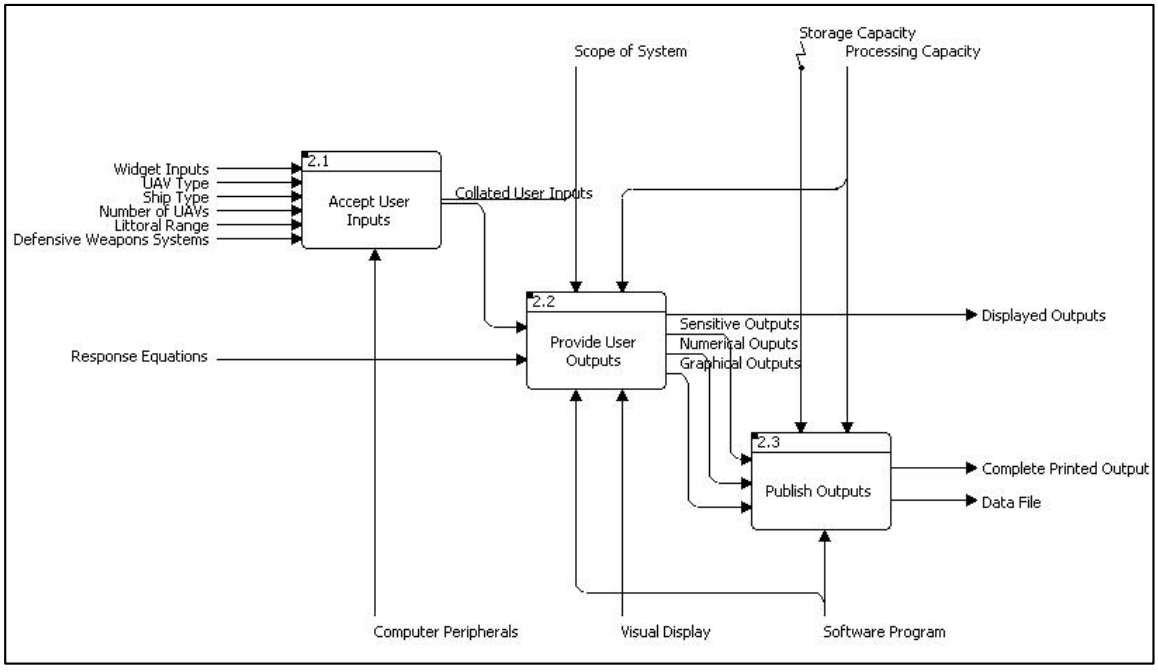

Figure 37. GUI IDEF0

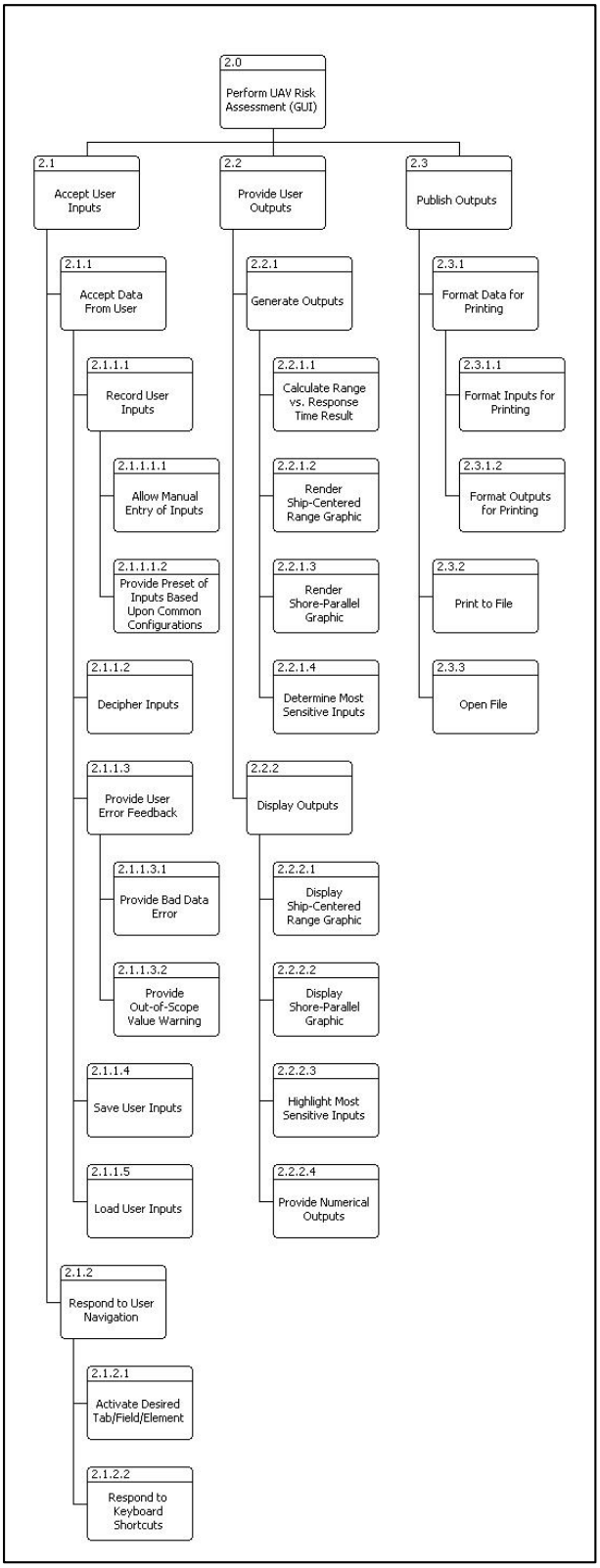

Figure 38. GUI Functional Hierarchy

Once Function 2.1 accepted all the user inputs, the Response Equations from Function 1.0 were used to create the required outputs. The primary interaction within Function 2.0 occurred within Function 2.2. It was at this point where the Response Equations and User Inputs were both required. Since the system was required to both display the outputs and save them for future use, Function 2.3, Publish Outputs, was required. Again, the different categories of outputs were based on user requirements whereas Sensitive Outputs refers to significant outputs that the user is made aware of and Numerical and Graphical Outputs refer to the different ways that the user required the data to be displayed.

Similar to Function 1.0, Scope of the System was a significant control for Function 2.0. As before, this control was mainly driven by the scope of the project. Further discussion on the implementation of this functional breakdown in the GUI prototype can be found in Chapter V.

IV. MODELING AND SIMULATION

Chapter IV introduces the modeling and simulation portion of the system. The chapter discusses the scope, assumptions and, design of the simulation. The design of experiments, factor analysis and output analysis is discussed in detail. The integration with the GUI is explained with a high level summary and recommendations.

A. MODELING AND SIMULATION INTRODUCTION

The authors created a discrete-event simulation of a basic UAV swarm attack on a U.S. Navy surface ship using ExtendSim 9. Although many changes have been made by several of the authors before and during this capstone project, the original basis for this model can be traced all the way back to Dr. Rama Gehris', professor of practice at the NPS, original "Zombies Model" used in SE3250 Capabilities Engineering class at NPS. The purpose of this effort was to create a way for the authors to explore and analyze the trade-space primarily in order to identify those input factors that would have the greatest effect on the scenario outcome. Human factors engineering principles for the design of effective user interfaces dictate that

> The most important elements should be easily perceived. Non-critical elements should be de-emphasized and clutter should be minimized so as not to hide critical information. (Martine 1995)

So it was necessary for the authors to identify those important elements in order to design an effective GUI. Another reason for creating the simulation was to generate a realistic equation representing the relationships between factors. This equation was then used by the GUI to provide outputs that respond to inputs in a generally correct manner so that they make sense to the user interviewees during demonstration. The authors determined that this extra step using human-centered design yielded insights into the GUI design that ultimately resulted in a higher fidelity prototype.

The input data set was created based on open source references on UAV and U.S. Navy surface ship performance and capabilities discussed in detail in Chapter II. The design of experiments was set up using Minitab 17. The response equation and the

graphical representations of the simulation outputs were generated using Minitab 17. The data flow relationships between the three tools used in this project (ExtendSim 9, MATLAB GUI and Minitab 17) are shown in Figure 39, but the two components of this relationship discussed in detail in this chapter are the ExtendSim model and Minitab statistical analysis.

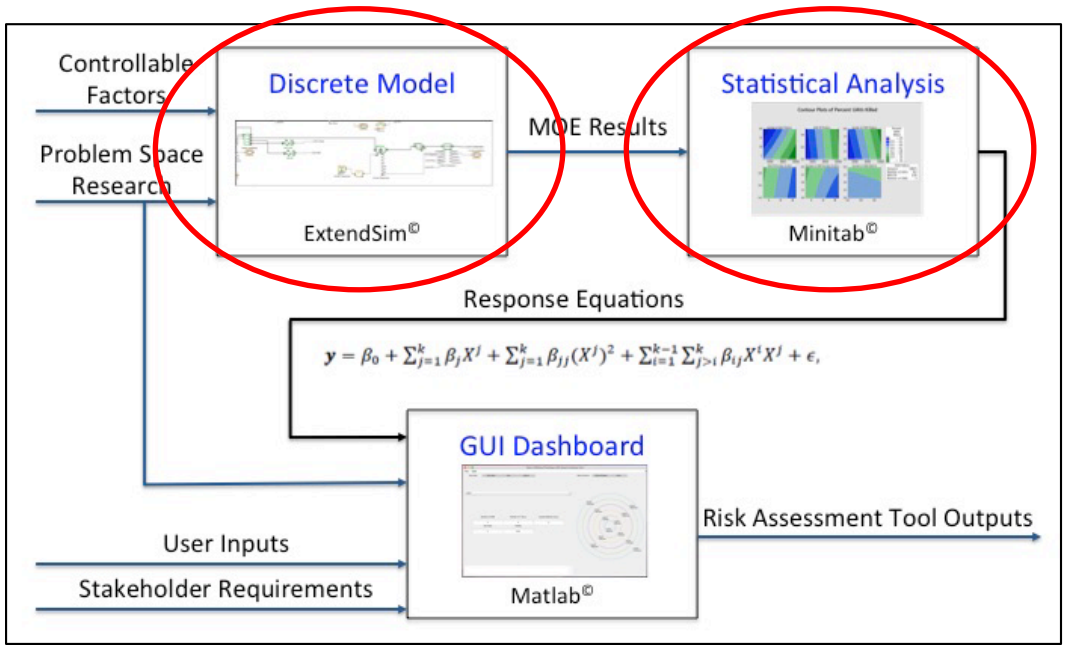

Figure 39. Data Flow Architectural Layout

B. SIMULATION SCOPE AND ASSUMPTIONS

In order to make the project feasible with the amount of time and manpower available, while maintaining the unclassified nature of this effort, it was necessary to make several simplifying assumptions in the design of the simulation. These simplifications allowed the authors to create an initial simulation that meets the project requirements of identifying the important input factors and providing a realistic equation to drive the MATLAB GUI. Thus these assumptions gave the authors the ability to go further in the design iterations of the project and enable them to collect customer

feedback by presenting a proposed product that accurately represents the design concept. Below are listed the assumptions used in the simulation:

1. Blue Force Characteristics
    1.1. The authors are modeling a single ship only and do not address any cooperative interaction of multiple U.S. ships operating close together.
    1.2. The authors are only modeling the unclassified, publicly available data on capabilities of current U.S. weapon systems. The potential effects of theoretical enhanced future weapon system capabilities are out of scope for this project.
    1.3. The ship's weapon systems represented in the simulation include the Mk45 Five-Inch Gun, Mk15 CIWS, and the Mk38 MGS.
    1.4. Blue force weapons have an unlimited ammunition supply in the simulation.
    1.5. The simulated ship is operating in tightly constrained maneuvering environments, such as a straits transit, where it is forced to sail near the shoreline of a hostile nation. The ship does not have the freedom to simply leave the high threat area.
    1.6. The maneuverability of the ship is not modeled. Although the ship is moving at a constant speed and direction during the engagement, the ship does not execute any defensive maneuver in response to the attack.
    1.7. At the start of each run, the ship's speed is randomly set between 0kts and 31kts and will remain constant throughout that simulation run.
    1.8. Use of the Mk45 five-inch gun requires UAV detection and track by the SPY-1D radar before it can fire.
    1.9. Use of the Mk15 Close-In Weapon System and Mk 38 Machine Gun System requires a UAV detection and track by the Mk 20 EOSS, which is affected by weather.
    1.10. The detection range of the SPY-1D is unaffected by weather and is greater than the distances between the ship and the shore.
2. Attack Characteristics

2.1. A single side attack is simulating. The UAV swarm has not surrounded the ship but instead has come directly from hostile launch locations ashore and is proceeding directly towards the ship.

2.2. Weapons cutouts are not considered because the authors assumed a beam attack where all the weapons are able to shoot in the direction of the approaching swarm.

2.3. Weapons swing rate is not explicitly modeled, however, it is accounted for in the random additional time required to acquire the next target. This effect also implies that, in simulation space, the UAVs are slightly spread out in azimuth, altitude or both.

2.4. The simulation does not consider precisely where on the ship the UAV might impact. There are no damage effects or ship's survivability considerations. The assumption is that, due to a possible chemical, biological or radiological warhead, having a single UAV reaching the ship is unacceptable.

2.5. If, or when, one UAV reaches the ship, the simulation run ends with the ship defeated and proceeds to the next run.

2.6. If all of the UAVs are shot down without one reaching the ship, the simulation run ends with a blue victory and proceeds to the next run.

3. Environment

3.1. The time of day is not taken into account in the simulation. There is no modeled effect on the ship's systems due to sun position, thermal crossover, or any other time of day related issue that might cause sensor degradation in the real world.

3.2. Sensor degradation and weapon effectiveness due to visibility and sea state are approximated with a decreased probability of detection (P_D) factor.

3.3. Sea state affects the weapon systems probability of kill (P_K).

4. Threat UAV Characteristics

4.1. The example threat UAVs that were modeled are the Yasir, Ababil-T and Harpy. The tactical scenario that was modeled involved relatively short ranges, which are sufficiently less than the range capabilities of all three of these UAVs. Therefore UAV range is not an issue. The key aspects of UAV performance to be represented in the model are the minimum and maximum speeds. Using publicly available information, the authors have chosen a speed range from 70 knots up to 200 knots. When the UAV swarm is generated, the speed is set randomly between these values. The UAV swarm will fly at that constant speed for the duration of the run.

4.2. The UAV will fly directly at the ship without any evasive maneuvers or counter measures.

4.3. The UAV is either fully effective or completely destroyed. There is no damage, soft kill or partial kill of the UAV.

4.4. UAVs can be detected, tracked and identified only when over water. This stems from an assumption that they would be operated at low altitude by the enemy in an attempt to hide in the ground clutter over land.

C. SIMULATION DESIGN

The general architecture of the simulation is represented graphically in Figure 40. Starting on the left side, the number of UAVs to be generated during a run, and their distance from the ship as they cross the shoreline, is read from the input database. The simulation references the range and visibility for each run and determines at what range the UAVs are detected. The swarm of UAVs passes through the weapons engagement zone (WEZ) of each of the three modeled weapon systems. Each weapon continues to fire at the UAVs for as long as the UAV is within the weapon's WEZ. The simulation determines when, or if, each UAV is shot down based on the weapon's Probability of Kill (P_K), which is related to range. The simulation continues until all UAVs are destroyed, or until one UAV reaches the ship. When each run ends, the simulation logs the ending conditions in the output file and proceeds to the next run.

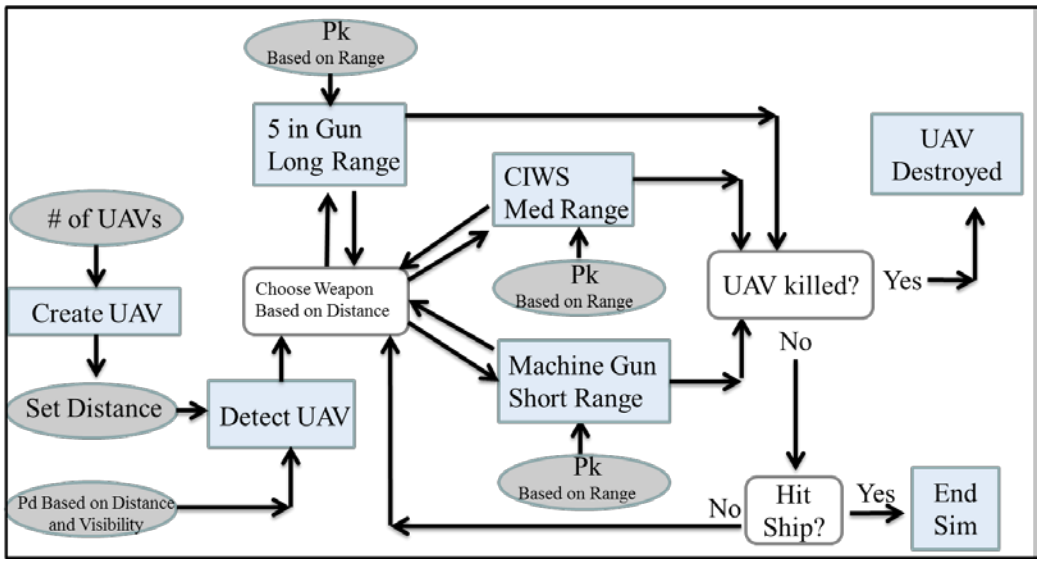

Figure 40. Simulation Process Flowchart

Using ExtendSim 9, the simulation was laid out following the basic architecture and data flow shown in Figure 40. All of the major process steps are visible in the ExtendSim screen shots, shown in Figure 41 and Figure 42 with annotations identifying the major components of the model. The model generates the UAVs and sets their attributes before tracking them as they fly out toward the ship. The model determines when the UAVs can be detected and passes them to the Engagement Loop. In the Engagement Loop the Probability of Kill for each weapon system, modified by range, sea state, and visibility, is used to determine if each UAV is hit or missed by each shot attempt. After each pass through the Engagement Loop, the UAV is routed according to its status either back through the Engagement Loop again or to one of the exit blocks. There is an exit block for UAVs that are duds, that hit the ship and that are shot down by each weapon. A detailed step by step description of the model design can be found in Appendix A.

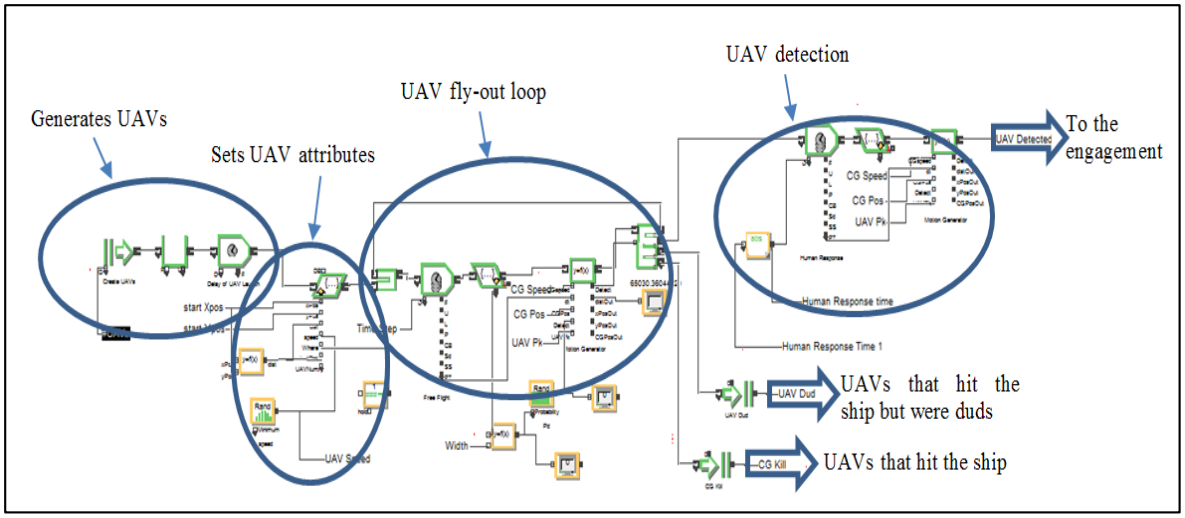

Figure 41. Screenshot of ExtendSim UAV Swarm Simulation, First Half

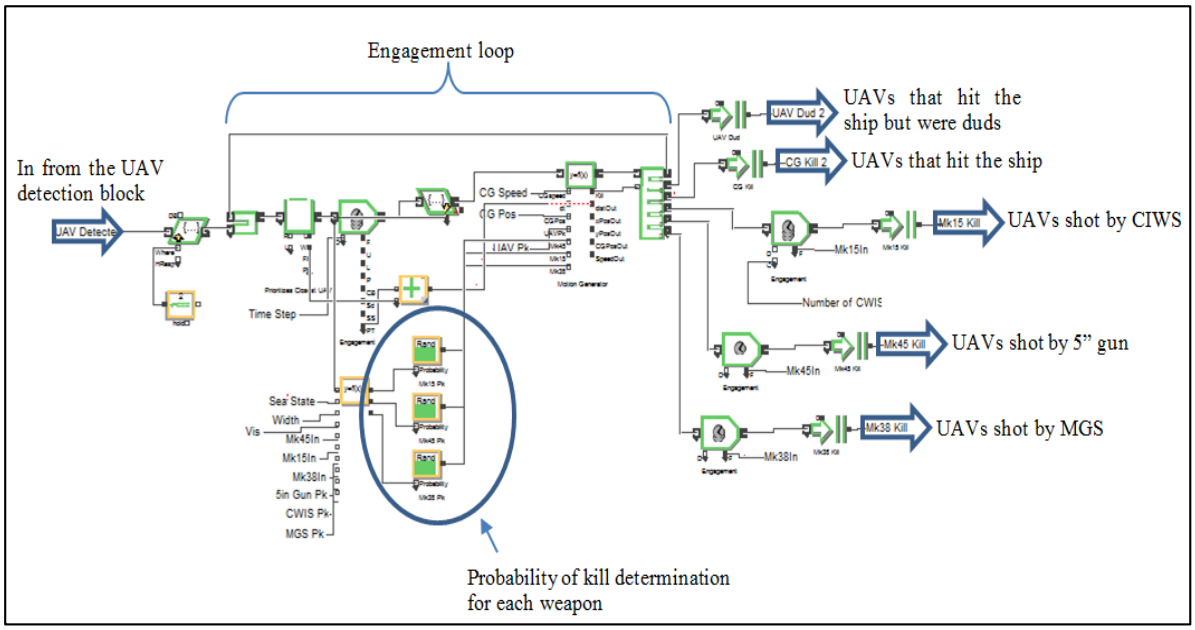

Figure 42. Screenshot of ExtendSim UAV Swarm Simulation, Second Half

The simulation was tested for stability and functionality without issue. The simulation typically took about 33 seconds to complete 1000 runs using a Windows laptop with an Intel Pentium CPU.

D. DESIGN OF EXPERIMENTS AND FACTOR ANALYSIS

Initially, the list of controllable factors in this simulation design was as inclusive as possible. The authors did not know which factors would be significant without analyzing them first, and they wanted to ensure an exhaustive factor exploration. The initial controllable factors included distance, number of UAVs, UAV speed, ship speed, visibility, sea state, P_K for each weapon system, human response time, and number of CIWS systems. These factors were determined during the early phases of the project. They were extensively researched and believed to be potentially relevant by the authors.

An initial design of experiments data table was created in Minitab 17 for a two level, 11 factor DoE requiring 2048 runs for a single replication. This simple two level DoE is used initially in order to identify the factors that are most significant to the scenario outcome. All 11 factors were included for completeness and only one replication was necessary at this point because the authors were not interested in statistical significance yet. The high and low values from each controllable factor were used to define the two levels. Minitab 17 was set to randomize the run order to minimize any potential unforeseen effects from external factors that are not part of this study. The DoE table of input data was copied from Minitab 17 and pasted directly into the ExtendSim 9 read database. When the simulation runs were complete, the ExtendSim output database was cut and pasted back into Minitab 17 for analysis.

The purpose of this initial set of runs was to identify the important factors that influence the output. The outcome permitted the authors to separate the important factors from those that were not with the purpose of focusing on the top few factors that affected the outcome. Using Minitab 17 a factorial analysis was conducted and the Pareto chart of standardized effects was created shown in Figure 43. This chart graphically indicates the magnitude of the influence each input has on the outcome of the simulated scenario. On the x-axis, the chart displays the absolute value of the standardized effect and includes a reference line at the Pareto-value of 1.97, which corresponds to a 95 percent confidence level. Any factor that extends to the right of this reference line passes the statistical hypothesis test and it can be concluded with 95 percent confidence, that the factors are statistically significant and not just irregularities in the data.

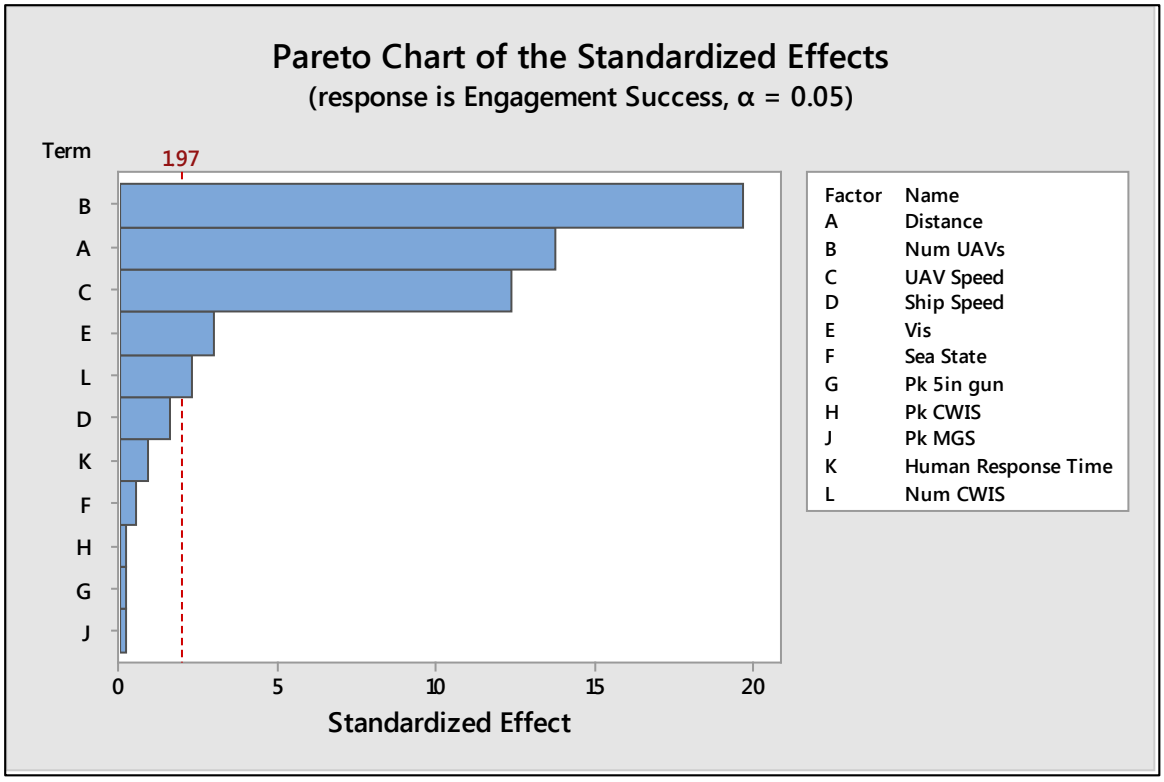

Figure 43. Pareto Chart of Standardized Effects Produced by Minitab 17

The analysis shows that the most significant factors are the number of UAVs in the swarm, the distance from the ship to the shore (where the UAVs are expected to be detected), and the speed of the UAVs. Besides those three main factors, there are two other factors with less significance but that still pass the 95 percent confidence reference line. These factors are the prevailing meteorological visibility (which affects the range that the Mk38 MGS can be employed) and the number of Mk15 CIWS systems on the ship. The other six factors did not significantly affect the simulation outcome and were therefore changed to fixed values in subsequent runs of the model. This factor screening reduced the original 11 factors down to just four controllable factors and one categorical factor (number of CIWS).

A second design of experiments was then created that focused on those four factors with significant impact on the simulation outcome. The purpose of this second set of runs was to provide a more filled out and statistically significant field of data with, which to fit a response surface. This is important because the regression fits are better

with many more design points. The authors conducted the factor analysis first in order to reduce the number of factors being studied, because if they used all 11 factors the total number of runs required would have been excessive (over 2 billion) and unmanageable.

Lessons learned by team members in SE3250 shows that a minimum of 30 replications are required in order to achieve statistical significance. Five levels for each factor have shown to provide adequate data for a reliable regression during early tests of the model. The authors chose to use five levels for each of the four factors with 40 replications because this combination of five levels, four factors and 40 replications yielded a round number of 25,000 runs. The predicted run time was approximately 13 minutes and actual run time was closer to 15 minutes.

E. OUTPUT ANALYSIS

With the final runs complete, the results were again pasted into Minitab 17 for analysis. This time the regression feature was used to fit a response surface to the output data. This generated the equations, shown in Figure 44 and Figure 45, which represent the likelihood of a successful engagement (blue ship victory) given the range of input factors. These equations can be considered a "meta-model," which is an equation, or set of equations, that represents the output of the simulation, and will be used to control the GUI.

```
Success = 0.0585 - 0.01707 NumUAVs + 0.06161 Dist + 0.001080 UAVSpeed
        + 0.5414 Vis - 0.000839 NumUAVs*Dist - 0.000053 NumUAVs*UAVSpeed
        - 0.01080 NumUAVs*Vis - 0.000149 Dist*UAVSpeed - 0.00312 Dist*Vis
        - 0.000958 UAVSpeed*Vis
```

Figure 44. Probability of Blue Force Victory (Success) with One CIWS

```
Success = 0.2561 - 0.01770 NumUAVs + 0.06419 Dist + 0.000431 UAVSpeed
        + 0.5675 Vis - 0.000839 NumUAVs*Dist - 0.000053 NumUAVs*UAV Speed
        - 0.01080 NumUAVs*Vis - 0.000149 Dist*UAVSpeed - 0.00312 Dist*Vis
        - 0.000958 UAVSpeed*Vis
```

Figure 45. Probability of Blue Force Victory (Success) with Two CIWS

Several contour plots were generated in order to visualize the data and study the behavior of the model. Each of these plots shows the relationship between three of the key factors. In each plot the response, Success, is represented on the z-axis by the change in colors as a function of the other two factors. For example, Figure 46 shows that with 20 UAVs, the probability of success is zero no matter how far away they start. On the other hand, with only five UAVs and at ranges as short as 6,000 feet, the probability of success is between 60 percent and 80 percent. All distances are in thousands of feet and speeds are in feet per second. The visibility metric is a percent degradation below ideal conditions. Figure 46 and Figure 47 show the expected behavior based on the simulation runs data analysed.

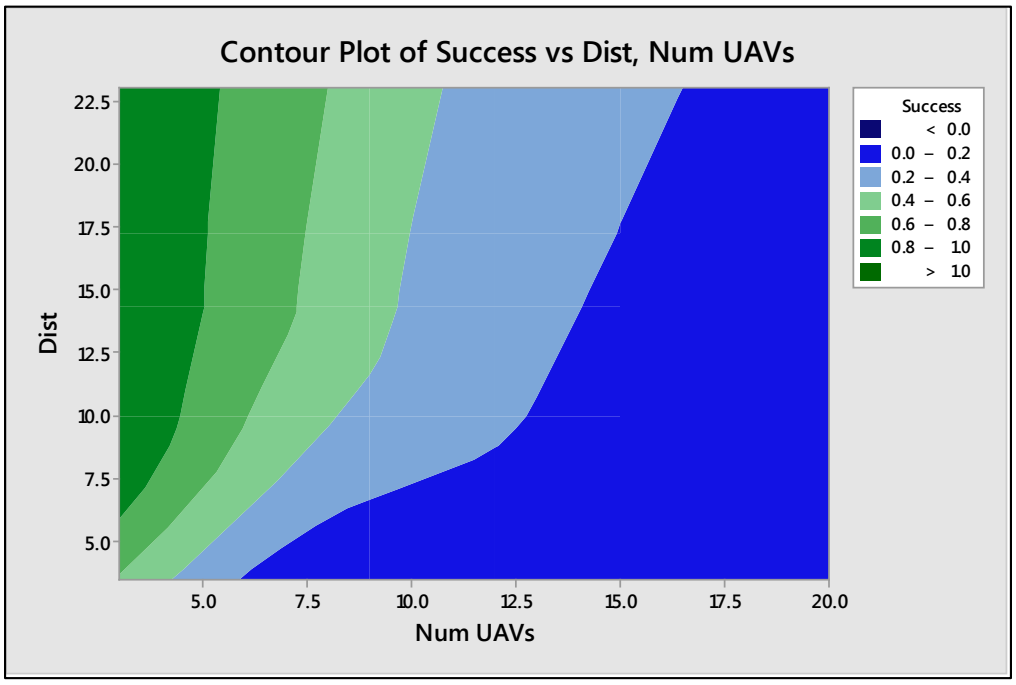

Figure 46. Contour Plot of Success Versus Distance and Number of UAVs.

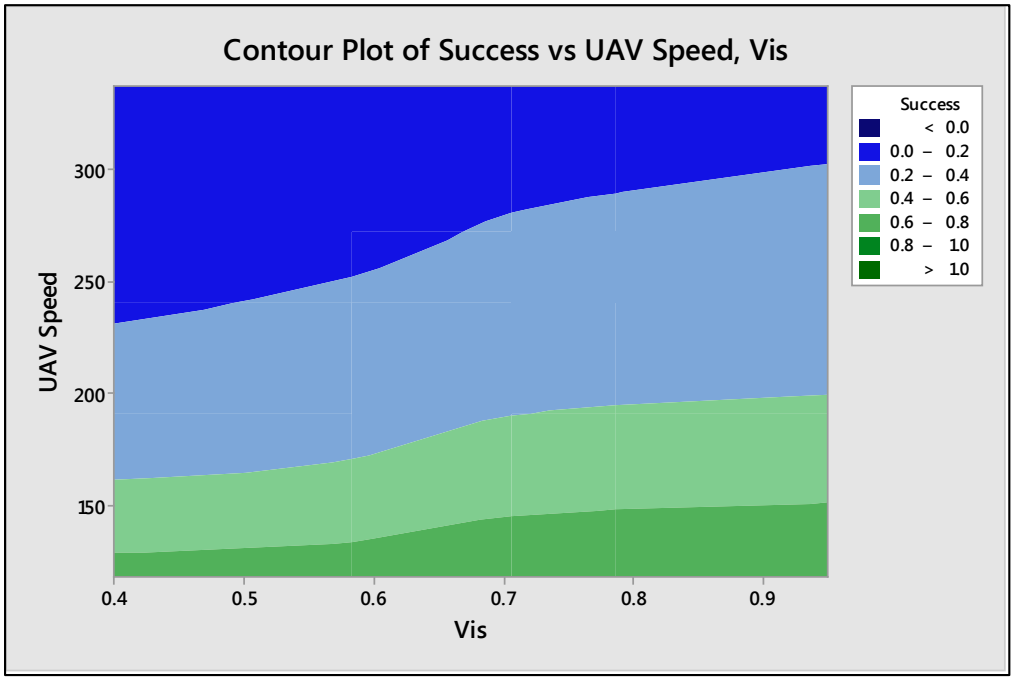

Figure 47. Contour Plot of Success Versus UAV Speed and Visibility

However, Figure 48 and Figure 49, indicate some surprising and complex interaction between factors. It is this complexity that proves the need for advanced modeling to predict the outcomes of tactical scenarios like the UAV swarm attack.

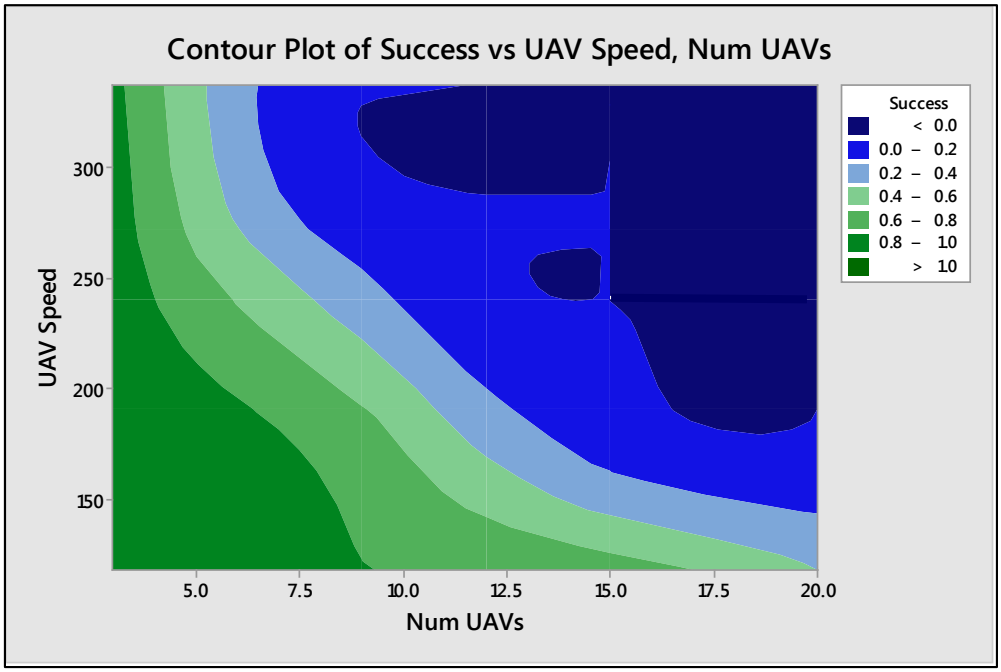

Figure 48. Contour Plot of Success Versus UAV Speed and Number of UAVs

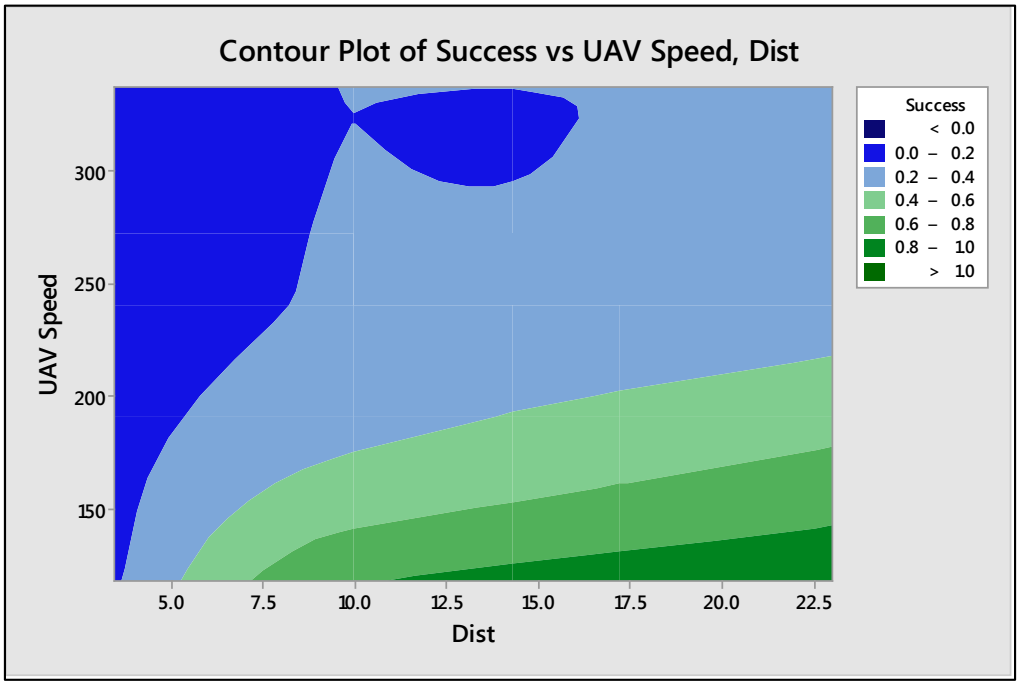

Figure 49. Contour Plot of Success Versus UAV Speed and Distance

F. INTEGRATION WITH THE GUI

The development of the ExtendSim 9 model proceeded in parallel with the design of the GUI and the demonstrations of the concept to the interviewees. As new requirements were generated and new revelations were discovered, the details of the simulation design had to evolve along the way. Most of these midpoint simulation design changes came in the form of minor features or characteristics of modeled weapon systems such as the Mk38 MGS's limitation of not being able to be used without a visual acquisition of a UAV. In general, the simulation design and development was in a supporting role to the GUI development in this project. In a few small ways, however, there was some reverse flow of design influence. For example, in some cases the simplifications desired by the authors developing the simulation feedback to the design of the GUI. Examples of this feedback are the preference to avoid the complexities of modeling the time of day and relative sun position effects and the desire to model only a limited number of ship classes based on the project scope, complexity and time available. As a result, the final GUI design has no time of day input and only allows the selection of two ship classes.

One of the most significant unexpected design changes and subsequent interactions between the two efforts came as a result of the second round of interviews. The interviewees stated that they would prefer to see a display output on the GUI that gave them an indication of the time available to react, and still have some confidence in victory, for a given scenario. This was a departure from the initial design assumption that the tool would be providing an indication of the risk for a given scenario. Not only did this drive a major change to the GUI controls and output display, it also caused a challenge for the simulation, which had been designed up to that point to produce an equation in terms of the probability of victory, not time.

After the authors returned from the second round of interviews, they researched how to produced equations that yielded response time as function of other controllable factors. The key challenge was that time was not one of the model factors, so the authors could not simply algebraically solve the equations for time. The authors realized, however, that time was a function of the UAV speed and the distance that they are

detected. They applied the alteration to the simulation data already in Minitab 17. The authors took the numerical value of the distance factor for each run and divided it by the value of the UAV speed for each run, and created a new column of data labeled "time." Then using Minitab's data analysis features, the authors fitted a regression equation to the data and time was selected as the response and included probability of success as one of the factors. This yielded the two equations in Figure 50 and Figure 51.

```
Time = -2.63 + 0.769 NumUAVs + 53.460 Dist - 0.0487 UAVSpeed - 24.39 Vis
       + 45.047 Success + 0.03779 NumUAVs*Dist + 0.002382 NumUAVs*UAVSpeed
       + 0.487 NumUAVs*Vis - 0.124281 Dist*UAVSpeed + 0.141 Dist*Vis
       + 0.0431 UAVSpeed*Vis
```

Figure 50. Required Response Time in Seconds with One CIWS

```
Time = -11.54 + 0.797 NumUAVs + 53.344 Dist - 0.0194 UAVSpeed - 25.56 Vis
       + 45.047 Success + 0.03779 NumUAVs*Dist + 0.002382 NumUAVs*UAVSpeed
       + 0.487 NumUAVs*Vis - 0.124281 Dist*UAVSpeed + 0.141 Dist*Vis
       + 0.0431 UAVSpeed*Vis
```

Figure 51. Required Response Time in Seconds with Two CIWS

These equations were used directly in the final MATLAB GUI code to provide the time required to respond as an output for a given set of user inputs, including the desired probability of success.

G. SIMULATION SUMMARY

Despite the significant limitations of schedule and resources, the authors were able to successfully create a discrete-event simulation that achieved two objectives. The authors were also very successful at manually integrating and using the three off-the-shelf software tools: ExtendSim 9, Minitab 17 and MATLAB GUI. The ExtendSim simulation paired with Minitab for data analysis, allowed to explore the trade-space and identify those input factors that had the greatest effect on the scenario outcome. Using Minitab, the authors were also able to generate a response equation to represent the simulation output. This response equation was used to drive the MATLAB GUI and enable immediate responses to user inputs, without the need to constantly re-run the simulation.

Ultimately this methodology enabled the authors to create a prototype that was closer to realism than the authors would otherwise have been able to create without these tools.

H. SIMULATION RECOMMENDATION FOR FURTHER STUDY

Extensive assumptions and simplifications were employed in this project in order to keep the effort unclassified and to remain within the constraints of schedule, computing power, and technical skills of the students. If this project were to be adopted as a program of record by the Navy, it should include the development of a higher fidelity simulation constructed by professional modelers, with the benefit of true classified intelligence data.

V. GRAPHICAL USER INTERFACE

Of equal importance to the discrete-event simulation developed by the authors to assess the threat of UAV swarm attacks under various conditions is the user interface developed to let users explore, interpret, and share those results. The authors developed a prototype user interface based in MATLAB GUI, emulating a web-based application. This prototype follows several generations whose development started with whiteboard drawings and guided through the iteration by stakeholder feedback.

A. INITIAL DESIGN DRIVERS

Through interviews with stakeholders, the authors identified four characteristics that were critical to the effectiveness and suitability of the interface design. The software solution must be lightweight, user friendly, exportable, and modifiable.

1. Lightweight

Since the original concept, the user interface has been imagined as a simple, lightweight, fast-running window into the simulation results, recorded externally, which does not include the simulation itself. The concept needed to be altered in the iterations to fit user needs but the main idea remained constant.

The simulation may take minutes to hours to produce results, depending on the variety of factors included and the level of statistical confidence desired. The authors do not expect it would be feasible to compensate with more powerful computers nor could they reasonably expect users to wait hours for their results. Instead, calling upon recorded simulation results could be done quickly enough for the user interface to respond instantly and constantly to user inputs.

The most robust means of recording the simulation results is a database. The user interface would query the database with the user inputs and provide the results. However, it would be faster and require less memory if the results could be refined into a polynomial expression. Analysis of the data in the statistical software Minitab 17

indicates that the results of the simulation, over the parameters investigated, are well represented by such an equation.

2. User Friendly

Though this application presents no unique requirements for usability, it, as all user software, stands to benefit in terms of acceptance and usefulness by addressing such concerns. To this end, the authors made an effort to develop a prototype that was familiar, intuitive, and aesthetic.

Familiarity with a tool can improve initial user acceptance and decrease the necessary training for users. The user interface includes common menu items such as *Save*, *Open*, *Quit*, and *Reset*. These are augmented with the usual keyboard shortcuts, for example, *Ctrl+S* for *Save*.

The desire for an intuitive design prompted the authors to place inputs along the left of the screen and outputs along the right – following the left-to-right flow common to English readers. Additionally, the colored indications of UAV threat were styled so that red indicates the highest threat and blue the lowest.

Though again limited by the experience of the authors—none being professional software developers—user feedback encouraged emphasis on aesthetics in the design. Even as early as the first digital prototypes, it was clear users focused on aspects of the design that had aesthetic appeal, or colloquially: coolness factor. This, in part, motivated the shift from a simple initial software prototype to the resulting more colorful, modernly styled final prototype.

3. Exportable

As confirmed by user feedback, a tool like the one the authors suggest here is likely to be operated by personnel who advise decision makers, but are not CO or other decision makers themselves. As such, it is important that the user be able to export the results. Users expressed interest in Microsoft Word, Microsoft PowerPoint, and Portable Document Format (PDF) document export formats. The development of these export features was not part of this prototype development effort.

4. Modifiable

Performing this work without the use of classified information has greatly limited the fidelity of the model the authors were able to produce and prove. Therefore, the authors consider essential that the architecture shall be flexible to future updates. The development of the user interface was done with good programming practices in mind, emphasizing readability of code, error handling, and general reusability.

B. DEVELOPMENT

The GUI development was based on a whiteboard drawing, static prototype and followed by a simple MATLAB GUI application that evolved as a web-app emulator.

1. Whiteboard Drawings

The initial concept for the user interface was fleshed out on a blackboard. At such an early stage, the concepts were not shown to stakeholders and instead were iterated based on internal feedback. One such blackboard diagram is shown in Figure 52.

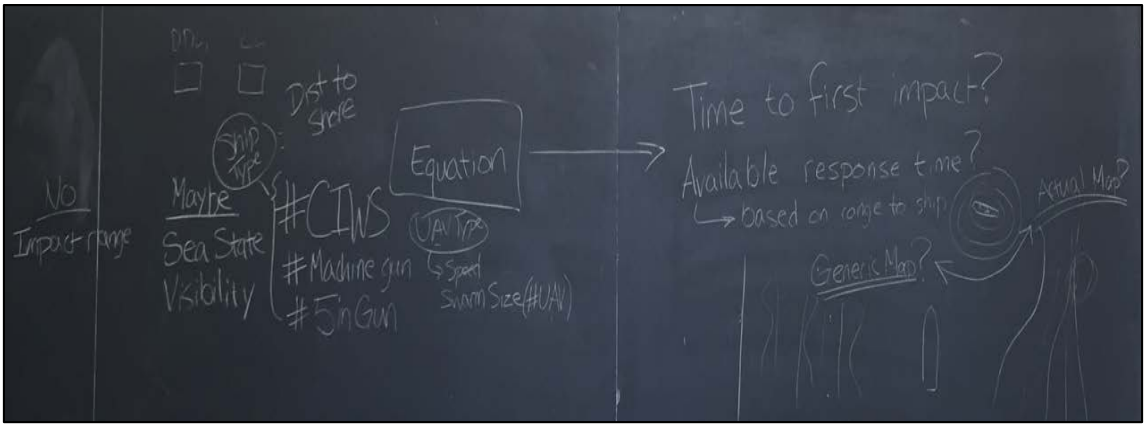

Figure 52. Blackboard Schematic: Early Model and User Interface Input and Output Refinement and a Few Basic Sketches of Output Visualization Concepts

Further refinement was done on whiteboards, both real and virtual, that the authors began to visualize the layout of the user interface. (The choice of whiteboard versus blackboard was merely one of availability.) Figure 53 represents the first prototypes to be shown to stakeholders.

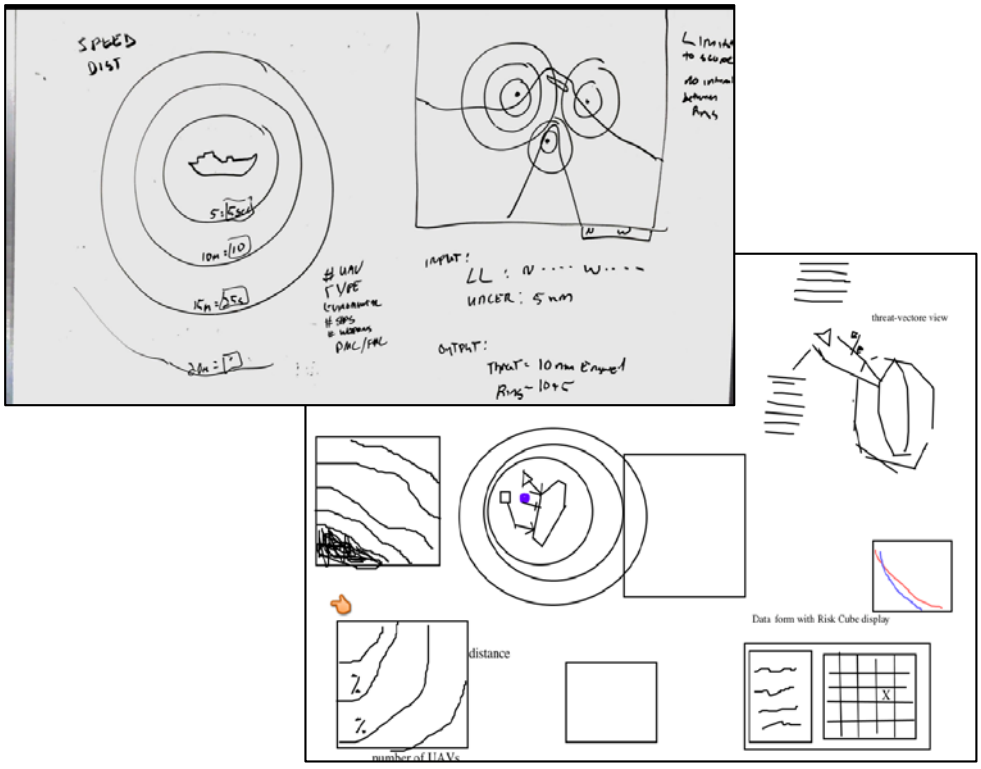

Figure 53. Whiteboard Drawings: A Whiteboard (Top Left) and Virtual Whiteboard (Bottom Right) Containing Drawings of Early Concepts of the User Interface Layout

2. Static Prototype

After the whiteboard drawings, the authors constructed a static prototype. This prototype looked like an operational Windows application. Though many of the fields were yet to be specified—see the overlaid text in Figure 54—this prototype gave a chance to get stakeholder feedback on the layout, appearance, and menu choices.

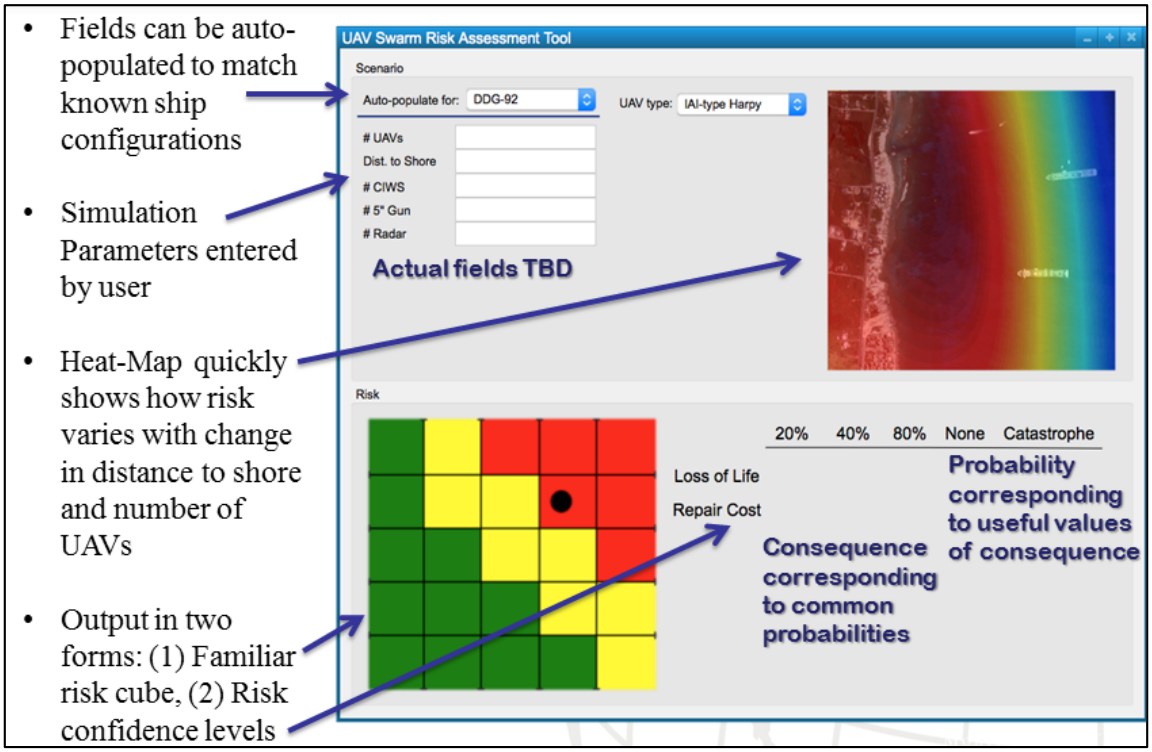

Figure 54. Static Prototype: A Nonfunctional but Realistic Looking Prototype

3. Simple MATLAB App

Motivated by available experience, the authors chose to develop the first functional prototype in MATLAB GUI. The initial task was to create a functional replica of the static prototype. The results of that effort are shown in Figure 55. This was completed partially in parallel with the development of the static app. The feedback received helped the authors shape the next functional prototype, which received the most user scrutiny of them all.

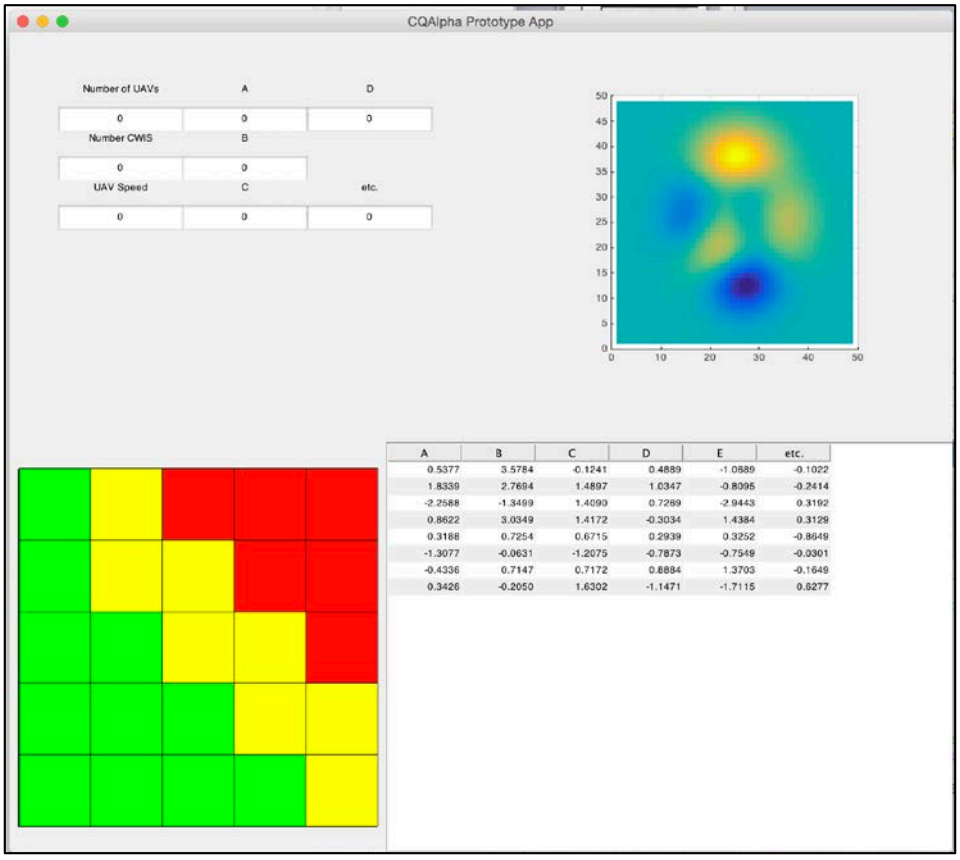

Figure 55. First Functional Prototype: This Prototype was Developed in MATLAB and was Designed to be a Functional Version of the Static Prototype.

The next MATLAB GUI prototype changed the layout, as shown in Figure 56, keeping some of the components and eliminating others. Interviews with stakeholders revealed that the characterization of UAV swarm threats as a risk was neither intuitive nor appealing. Instead, stakeholders wanted to emphasize the threat map. This prototype reincorporated the feature that was first introduced in the static prototype: ship configuration lookup. Rather than ask the user to manually enter every parameter, the authors added a dropdown menu (also called a popup menu) that the user can select the ship and the parameters will be adjusted to match.

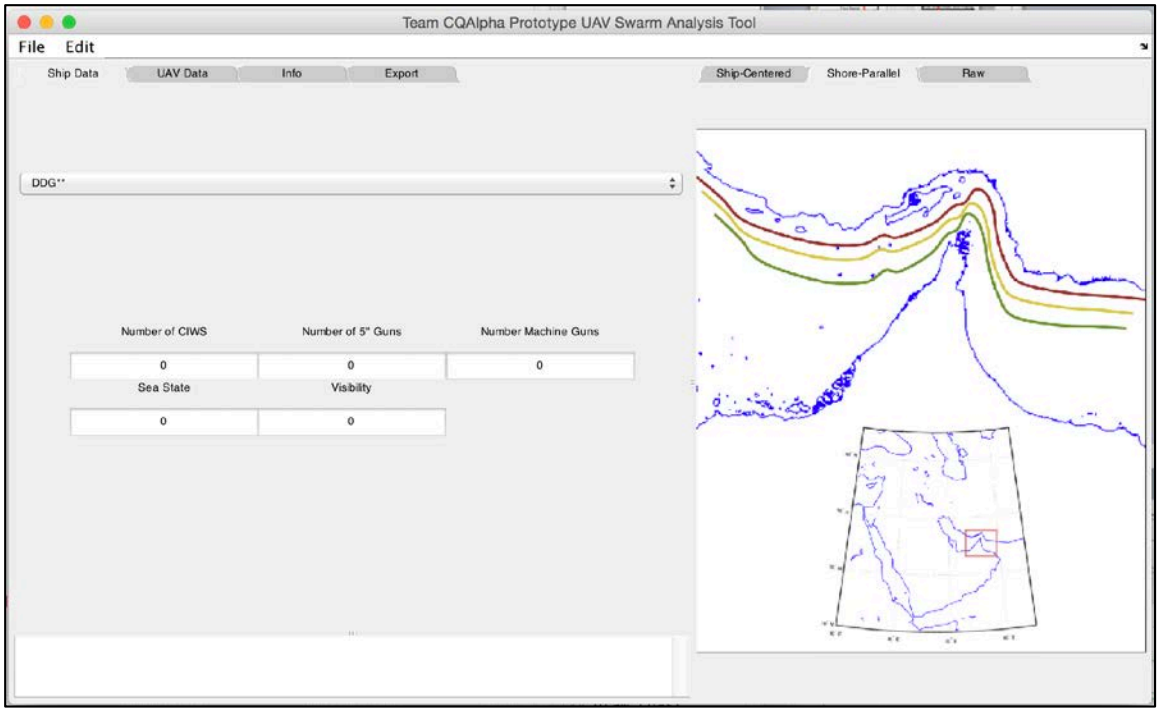

Figure 56. Second Functional Prototype: This Iteration Drastically Altered the Layout, Emphasized the Map, Got Rid of the Risk Cube, and Added the Ship Configuration Lookup Feature.

4. Web-App Emulator

The development path ends with the current design iteration. Following the feedback received primarily from a hands-on interview that was brought from the second functional prototype to a group of naval officers, the user interface layout was reimagined in Figure 57. In this iteration, the emphasis on the map is taken to the extreme. The characteristics of this final user interface prototype are discussed in the following section.

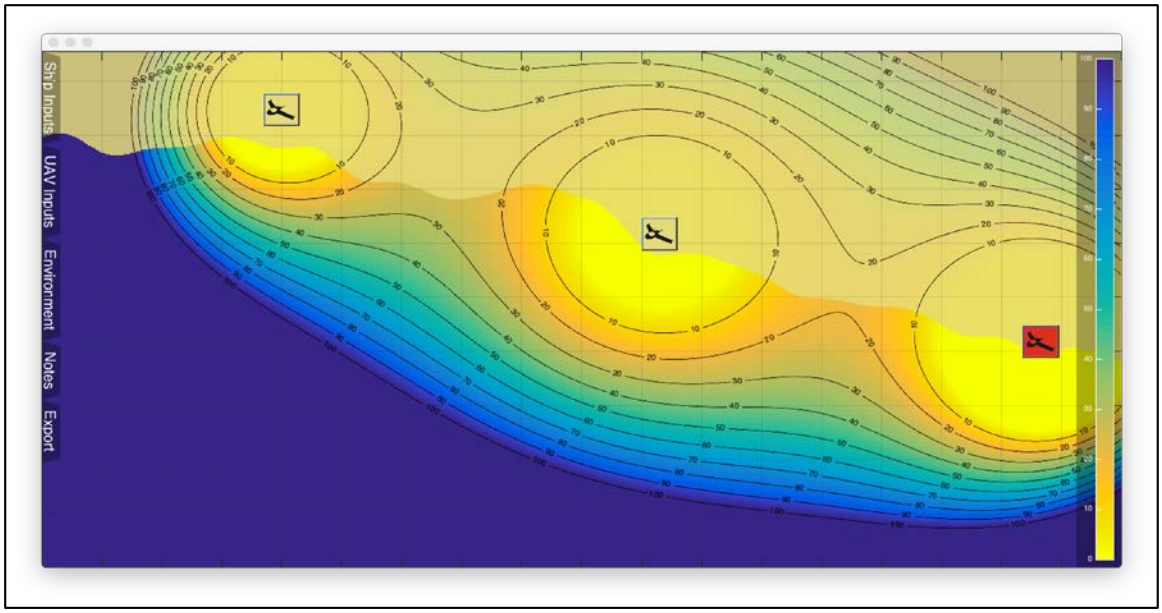

Figure 57. Web-App Prototype: The Final Prototype of This Effort.

C. USER GUIDE/DETAILED DESCRIPTION OF GUI

The final prototype developed as part of this effort was created to emulate a web-hosted application. In concept, users would access the application remotely across a SIPRNET connection. For practicality in development of this prototype, the interface was created using the MATLAB GUI language. By design, the use of MATLAB GUI is hidden from a user interacting with the interface. As such, the use of MATLAB GUI does not affect the way a user would interact with the application. MATLAB GUI enters into the discussion below only to describe the behind-the-scenes processing executed by the application. In addition, the MATLAB GUI complete script was included in Appendix B.

1. Entering Inputs

In Figure 58, the parameters that drive the application output can be specified within the tabbed sidebars along the left edge of the interface. The tabs categorize input fields by commonality. Inputs describing the defending ship are all contained on the Ship Inputs tab. The UAV Inputs tab allows the user to specify the characteristics of each UAV launch site. UAV threats are assumed to be homogeneous, but disparate sites are defined separately and need not share characteristics. Characteristics of the environment

during the engagement are specified in the Environment tab. In the Notes tab, users can add notes to the application to capture custom information in the exported documents.

For the purposes of this simulation, the defending ship is described only by the number of CIWS it carries. Though the simulation investigated other variables, a Pareto analysis whittled down the parameter space to only the significant factors.

Instead of specifying each of these values individually, users can select their platform from a list of known platforms and the parameters will be updated to match. Users can then adjust the parameters to customize the ship specification. See Figure 58 for an example of the ship input fields.

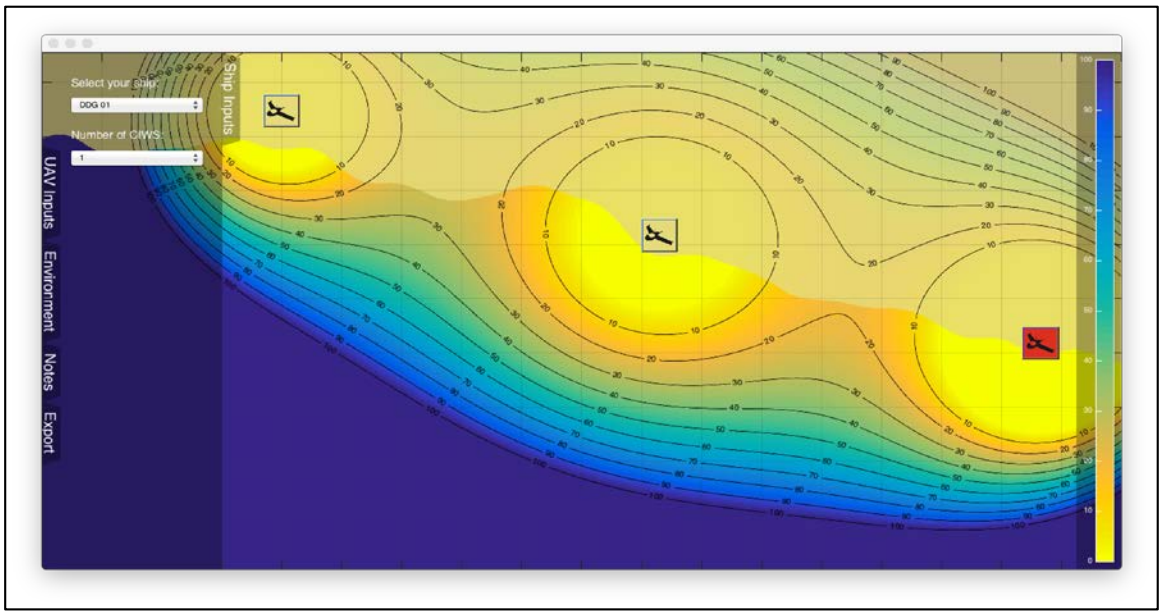

Figure 58. Ship Inputs

UAV launch sites can be created using the plus sign button at the bottom of the UAV tab. Existing UAV launch sites can be deleted using the nearby minus sign button. When a UAV launch site is created, the UAV characteristics are populated with default values. The user can change any of these values. The UAV parameters relevant to this simulation are quantity and speed.

As with the ship parameters, UAV characteristics can be specified quickly be selecting a known UAV from the available list. For a UAV type specified this way, the application will display other relevant information in the lower part of the tab. The UAV input fields are shown in Figure 59.

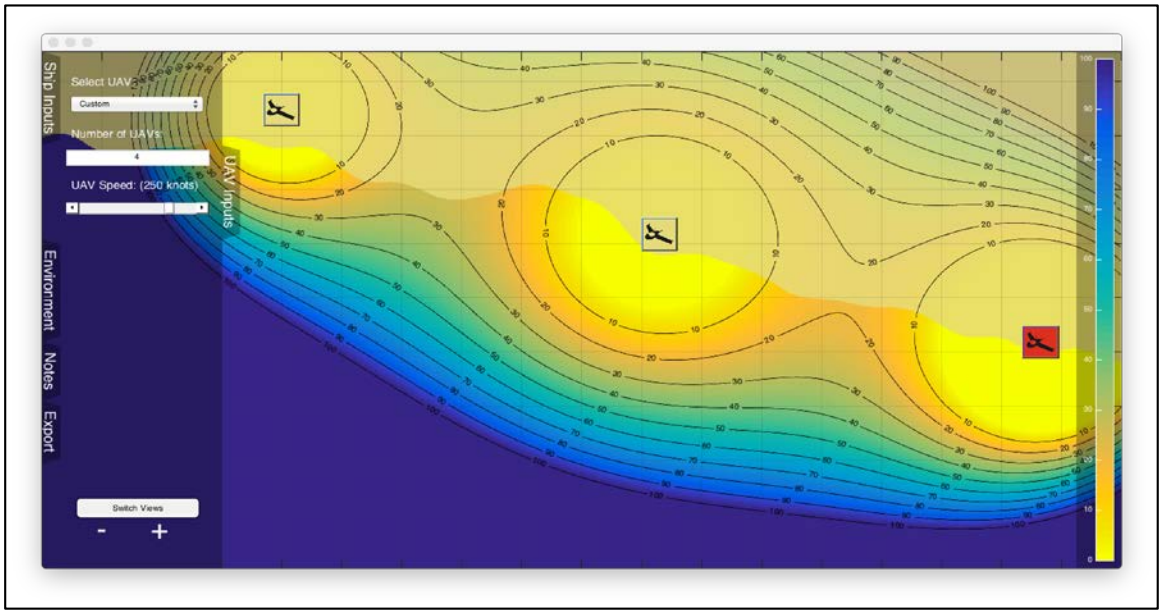

Figure 59. UAV Inputs

The simulation environment is specified in terms of visibility as shown in Figure 60. Visibility is specified as a multiplier on the probability of kill in the simulation. A visibility of 100 percent does not reduce the ship's defenses at all, whereas a visibility of 10 percent causes ten times more misses. Because visibility is naturally limited to a value between 0 and 100 percent, the visibility parameter is set by the user via a slider. Furthermore, because the polynomial approximation of simulation results is not reliable for visibilities close to zero, the lower limit of the slider is set to 20 percent.

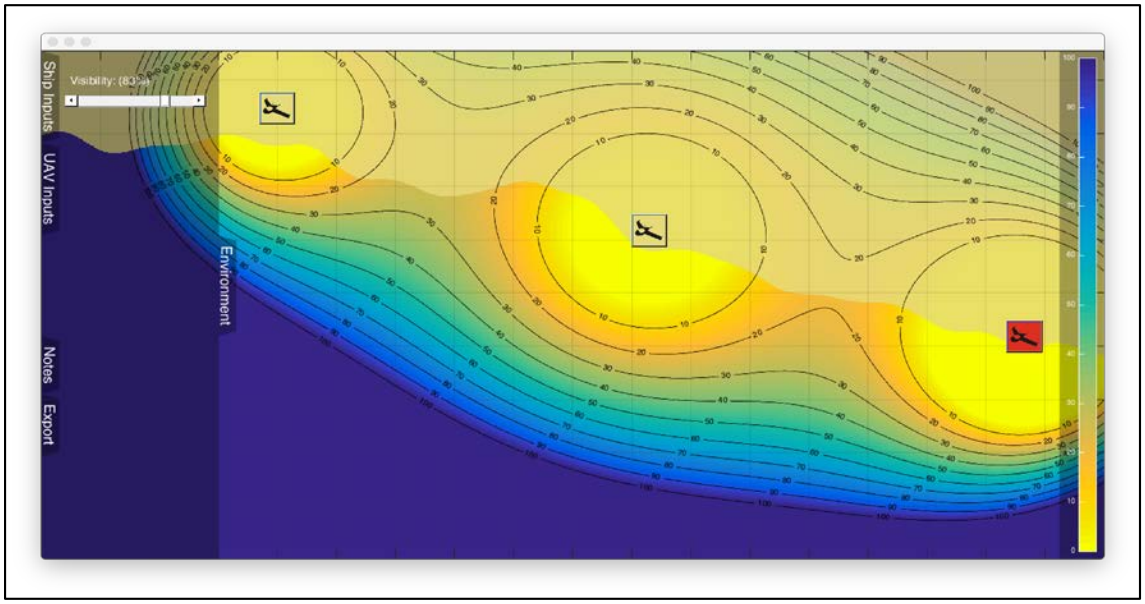

Figure 60. Environment Inputs

2. Interpreting the Map View

The primary output of this application is a map of the region of interest, colored to indicate threat level as shown in Figure 61. This tool characterizes threat level as a probability of successful engagement. Success here is the elimination of all UAVs with no damage to the ship. Therefore, this map view can be used to estimate risk when path planning. Overlaid on the colored map are contour lines every 10 percent. These make interpreting the colors easier and allow colorblind users to still interpret the results.

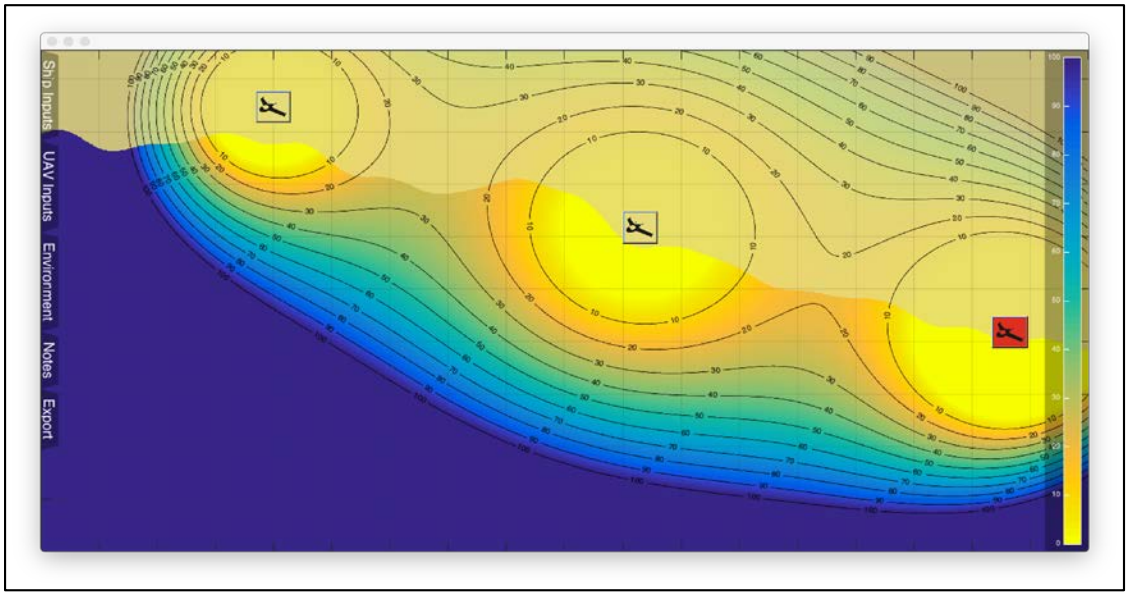

Figure 61. UAV Threat Map View

This map is intended to answer the question, “what are the chances that my ship can pass through this region and remain undamaged by this possible UAV threat?” In other words, it is the probability of a casualty-free encounter. The probability on the map corresponds to the UAV attack taking place while the ship is at that point. Because of this, the risk corresponding to a path is the maximum risk along that path, unless the user has reason to believe the attack will not occur while they pass through that maximally dangerous zone.

It is important to note that these values correspond to immediate engagement. The simulation models the ship as firing on the UAVs as soon as they cross over open water, where they are easier to detect. For other scenarios, see the next output view.

3. Interpreting the Response Time View

Once an engagement has begun the relative position of the UAVs and ship becomes more important than the absolute position of either. In this case, the application results can be displayed more simply as a threat level indicator versus distance from the ship as shown in Figure 62. Based on discussions with stakeholders, the authors chose to represent this threat level as the remaining decision time, that is, the amount of time in

seconds the ship can wait before firing on the UAVs and still maintain an acceptable probability of success, here defined to be 90 percent.

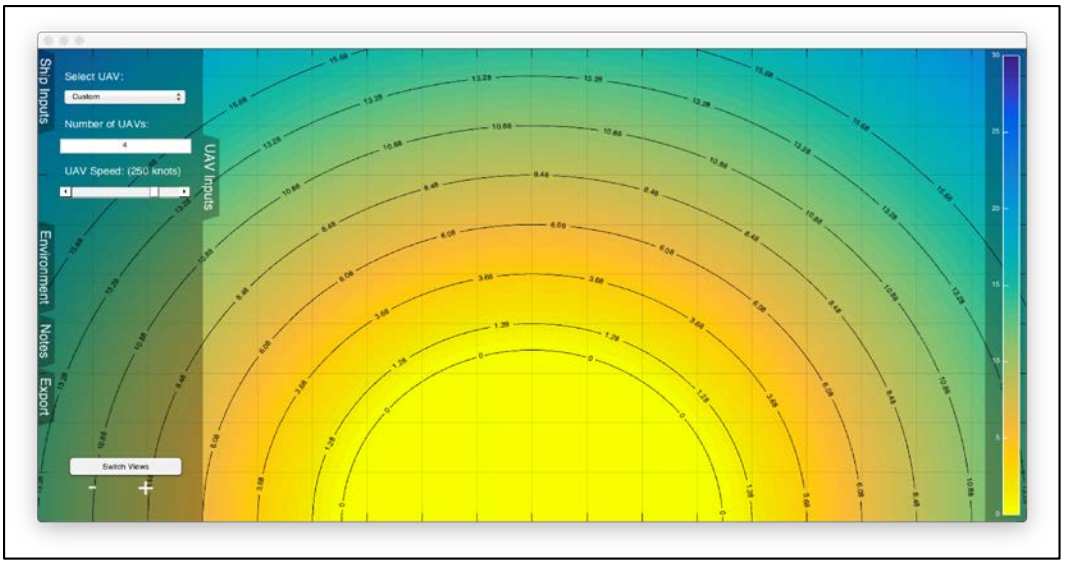

Figure 62. UAV Threat Response Time View

This chart is generated by calculating the distance that the probability of success equals the threshold value. Then the distance from this threshold range to the current UAV position is divided by the UAV speed to estimate the number of minutes available to the ship before an immediate response is necessary. This view can be shown for any one UAV launch site at a time. This view and the button to reveal it are shown in Figure 62.

4. Exporting the Results

Though the development of the export features was outside the scope of this project, the interface is shown in Figure 63. These buttons would link the user to an automatically generated file containing all of the inputs and outputs as well as the user's notes.

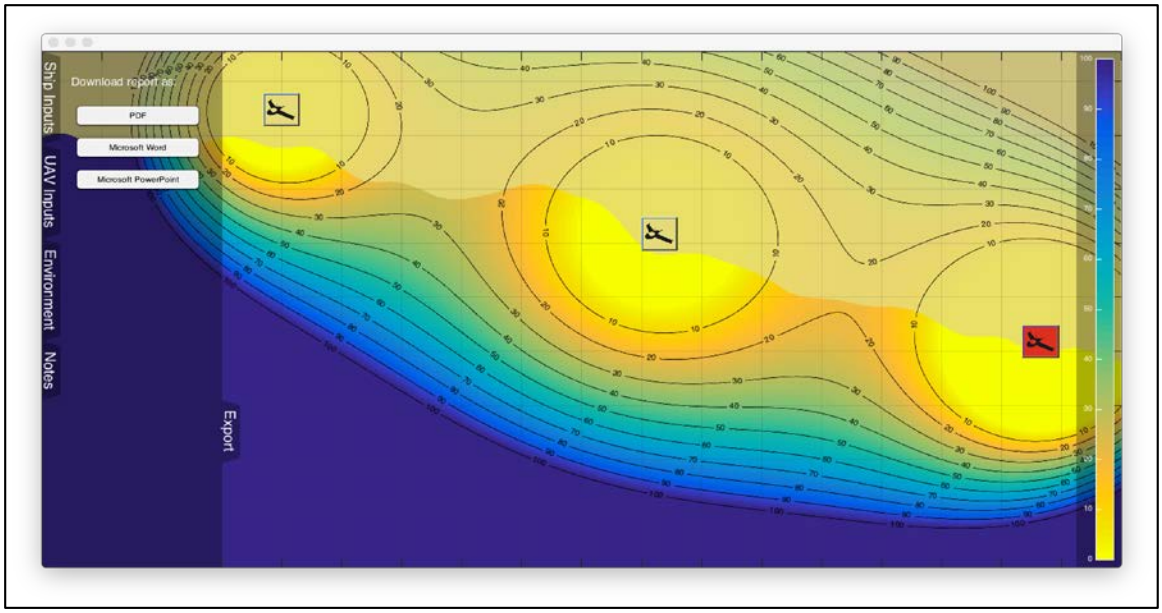

Figure 63. Export Menu

D. LESSONS LEARNED

The following are some lessons learned by the authors to improve some areas of the system in future studies to improve or expand on this work.

1. Up-to-Date Intelligence is Critical

The stakeholders interviewed made it clear that the information driving the results needed to be up to date at all times. They suggested that updates to the model should be made and pushed through to the user interface often. Specifically, they indicated that there was value even in daily updates.

2. Web Based is Possible

Keeping the user interface so up to date would be a monumental challenge if not for the second revelation revealed by the stakeholders. According to those interviewed, they are confident that SIPRNET connectivity is robust enough that a tool like the one authors envision could be hosted online and accessed remotely.

This greatly simplifies the process of keeping the user interface up to date. It was this realization that motivated the shift from a desktop app to an emulation of a web-based solution and the significant appearance changes that accompanied that shift.

3. Concrete versus Abstract

The trend from prototype to prototype shows clearly a shift from abstract displays of the results—the risk cube, generic contour plots, and chart output—to concrete ones – the map. Though charts, tables, diagrams, and other abstract visualizations of data have their place, the users responded most strongly to the concrete and intuitive nature of the map view. The concept of a map has become the core of the user interface and the primary means of communicating the results.

E. GUI COST

The efforts described in this report resulted in a working prototype. To take this working prototype and create a full production system requires funding, staffing and technical system requirements. This section details the estimated cost to take the prototype as it exists and transform it into a production system that could be used fleet-wide. The cost estimate was developed using the System Cost Model suite provided by NPS. The cost model suite was chosen as a tool for estimating cost of future system due to the authors familiarity with the cost model from the SE3011 Engineering Economics and Cost Estimating class at NPS. Raymond Madachy gave a lecture tittled “Constructive Cost Models” on January 30, 2014, at Naval Postgraduate School, Monterey. In his lecture, he introduced the Constructive Cost Model (COCOMO) suite and the Constructive Systems Engineering Cost Model (COSYSMO) model. In addition, Raymond Madachy gave a lecture tittled “Software Engineering Cost Model” on February 6, 2014, at Naval Postgraduate School, Monterey. In this lecture, he introduced COCOMO model for software engineering.

The output from the cost model contains relevant information that represents a decision factor for stakeholders. The cost model suite is an integrated tool that is used to perform cost estimates for various engineering disciplines including system engineering and software development. The COSYSMO is a tool within the System Cost Model Suite

that is used to estimate the system engineering costs of a project. Another tool in the suite is the COCOMO II, which estimates the costs of software development and sustainment. The authors have used these tools together to produce a comprehensive total ownership cost estimate for this project.

The COSYSMO model estimates the size of the project using four size factors: system requirements, interfaces, algorithms and operational scenarios. In addition, the model uses 14 effort multipliers and the labor rate. Both COSYSMO and COCOMO II models incorporate maintenance cost required to sustain the system through its lifecycle. The COSYSMO model input data is shown in Table 5. The authors added up all of the detailed software requirements, shown in Chapter III, Section C, and found that there were a total of 55 software system requirements. There were 13 system interfaces determined by the number of boundaries that exist between system functions (internal interfaces) and the external system functions. The system has one use case, which is termed as the operational scenario.

Table 5. COSYSMO Input (after Madachy 2014)

System Size	***Number***
System Requirements	55
System Interfaces	13
Operational Scenarios	1
System Cost Drivers (Effort Multiplier)	***Rating***
Requirement Understanding	Few undefined areas (high)
Architecture Understanding	Strong understanding of architecture and COTS, few unfamiliar areas (high)
Level of Service Requirements	High financial loss (high)
Migration Complexity	Everything is new; legacy system is completely replaced or nonexistent (nominal)
Technology Risk	Proven on pilot projects and ready to roll-out for production jobs (nominal)
Documentation	Extensive documentation and review requirements relative to life cycle needs, multiple revisions required (very high)
# and diversity of installations/platforms	Moderate environmental constraints; controlled environment (high)
# of recursive levels in the design	Extremely complex interdependencies coordination, and tradeoff analysis (very

System Size	***Number***
	high)
Stakeholder team cohesion	Shared project culture (nominal)
Personnel/team capability	Intellectual capability is System Engineers: 90th percentile (very high)
Personnel experience/continuity	3 years of continuous experience (nominal)
Process capability	Defined SE process, activities driven by benefit to project, SE focus is through operation, process approach driven by organizational processes tailored for the project (high)
Multisite coordination	Multi-city or multi-company, some time zone effects (nominal)
Tool support	Strong, mature SE tools, moderately integrated with other disciplines (high)
Maintenance	
Annual change percent	5
Maintenance duration	7 years
Software Labor Rates (dollars)	15,000

The effort multipliers for the cost model were rated based on the following assumptions:

1. Requirements understanding – Stakeholders' understands the users' needs from a system perceptive and the constraints that the system will be developed and operated.
2. Architecture understanding – The level of difficulty that will be encountered in developing and managing the system architecture with respect to "platforms, standards, components, connectors (protocols) and constraints" is minimal (Madachy 2014).
3. Level of service requirements – The level of service requirement, such as interoperability and security, will cause high financial loss if not met.
4. Migration complexity – The system will be integrated to an existing legacy system.
5. Technology risk – The working prototype has been proven by stakeholders and will go into full production for fleet use.

6. Documentation – Detail documentation is required for the system's lifecycle support.
7. Number and diversity of installations/platforms – The system will be used on different platforms and will operate in a secured environment.
8. Number of recursive levels in the design – System Engineering effort will be well coordinated and trade off analysis will be performed where necessary.
9. Stakeholder team cohesion – The program team shares a common objective and has strong team support.
10. Personnel/team capability – The program team has the skills and experience needed to start and complete the overall project.
11. Personnel experience/continuity – The program team has a minimum of three years continuous working experience.
12. Process capability – The program team has well documented and managed system engineering management plan.
13. Multisite coordination – Not all stakeholders are co-located. Some meetings are conducted via phone and visual interactive collaboration.
14. Tool support – Team has all necessary tools required to perform their job effectively and efficiently.

Also, the COCOMO II model has one size factor, which is either thousands of software lines of code (KSLOC) or function points, which it uses to estimate project size. The model uses five scale drivers and seventeen effort multipliers. Two estimates were developed for the software development portion of the cost estimate that is, software development cost for the GUI and ExtendSim. The COCOMO II input data is shown in Table 6. The software lines of code were obtained from the GUI. In addition, 300 function points was also obtained from the ExtendSim model.

Table 6. COCOMO II Input (after Madachy 2014)

Software Size	***Number***
New (SLOC)	1000
Reused (SLOC)	5000
Function Points	300
Language C	
Software Scale Drivers	***Rating***
Precedentedness	largely un-precedented (low)
Development Flexibility	general conformity (High)
Architecture / Risk Resolution	generally 75 percent (high)
Team Cohesion	highly cooperative (high)
Process Maturity	CMM Level 2 (nominal)
Software Cost Drivers	***Rating***
PRODUCT ATTRIBUTES	
Required Software Reliability	high financial loss (high)
Data Base Size (Database size/SLOC)	10 ≤D/P <100 (nominal)
Product Complexity	Use of simple graphic user interface (GUI) builders (low)
Developed for Reusability	across program (high)
Documentation Match to Life-Cycle Needs	very excessive for lifecycle needs (very high)
PLATFORM ATTRIBUTES	
Execution Time Constraint	≤ 70 percent use of available execution time (high)
Main Storage Constraint	≤ 50 percent use of available storage (nominal)
Platform Volatility	Major change: every 12 months. Minor: 1 month. (low)
PERSONNEL ATTRIBUTES	
Analyst Capability	75th percentile (high)
Programmer Capability	75th percentile (high)
Personnel Continuity	3 percent/year (very high)
Applications Experience	6 years (very high)
Platform Experience	3 years (high)
Language and Tool Experience	6 years (very high)
PROJECT ATTRIBUTES	
Use of Software Tools	strong, mature lifecycle tools, moderately integrated (high)
Multisite Development	fully collocated (extra high)
Required Development Schedule	100 percent (nominal)
Maintenance	
Annual change size (ESLOC)	500
Maintenance duration	7 years
Unfamiliarity	Mostly unfamiliar (0.8)
Software Labor Rates (dollars)	13,475

The software scale and cost drivers were rated based on the following assumptions:

1. Precedentedness – No similar software exists. It is an all new system; hence, precedentedness is rated low.
2. Development Flexibility – Software design and demonstration must conform to specified requirements.
3. Architecture/Risk Resolution – The program has well designed software architecture, risk reduction and mitigation plans.
4. Team Cohesion – The program team shares a common objective and has strong team support.
5. Process Maturity – Software processes are well design, managed and specific to the program.
6. Required Software Reliability – Delay in software functionality will cause high financial loss to the program.
7. Data Base Size – Data requirement will have minimal effect on software development.
8. Product Complexity – The software designers will use simple GUI builders to enhance user interface.
9. Developed for Reusability – The software is well documented and tested for possible other applications.
10. Documentation Match to Life-Cycle Needs – Detail documentation is required for the software's lifecycle support.
11. Execution Time Constraint – Software will be developed per requirement to meet specific execution time with minimal time constraint.
12. Main Storage Constraint – Minimal main storage constraint imposed on software system.
13. Platform Volatility – The software platform "includes any compilers or assemblers supporting the development of the software system" (Madachy 2014). The hardware will have a major change annually.

14. Personnel Capability/Continuity/Experience – The software team has the skills and experience needed to start and complete the overall project.
15. Platform Experience – The software team has a minimum of three years continuous working experience.
16. Language and Tool Experience – Team has six plus years of experience in software programming and tool.
17. Use of Software Tools – Team has all necessary tools required to perform their job effectively and efficiently.
18. Multisite Development – Not all stakeholders are co-located. Some meetings are conducted via phone and visual interactive collaboration.
19. Required Development Schedule – The software developers have no schedule constraint.

The cost estimate summary for the GUI is shown in Table 7. The total cost to build and maintain the GUI in the fleet is estimated at $1.74M. This estimate includes initial development and seven years of maintenance cost. Person-month is the measure of effort. System engineering effort requires 39 person-months; that is the amount of work done by 7.8 persons if they work full-time for five months. Similarly, software development effort requires 67 person-months; thus, 3.5 full-time persons are needed to work full time for 19 months. The maintenance duration for each effort is 84 months with two person-months for system engineering and one person-month for software engineering. This information was obtained from the System Cost Model Suite based the input data from Table 6 and 7.

Table 7. Cost Summary

	Effort (Person-month)	Schedule (month)	Cost ($)
System Engineering	39	5	591,017
Software Development	67	19	895,871
Maintenance	3	168	257,684
Total			**1,744,572**

F. SUGGESTIONS FOR FUTURE DEVELOPMENT

The following are suggestions for future development of this project. The suggestions were based on user needs and stakeholders interactions.

1. True Web App

The authors' suggest for future work the development of a truly web-based prototype. This would not only serve as a proof of possibility and an exploration of the available tools but as a means of reaching a greatly increased number of users for prototype testing. Also, by integrating this tool into a webpage, the COs would know that they are using the most effective and latest version of the tool. A desire for the version and date displayed was expressed; however, this need would be eliminated by putting this onto the Internet. The program office would be able to update this software frequently to support the fleet to the fullest extent. With frequent updates, the tool would be able to cover the constantly changing environment the U.S. Navy operates in. The webpage would be easily accessible since the majority of ships have SIPRNET capability via satellite.

2. SIPRNET Reliability

The unclassified nature of this effort prevents from extending the understanding of SIPRNET's reliability beyond the anecdotes the authors received. Because of the critical dependence of this concept on classified connectivity, future work stands to benefit from an improved understanding of the robustness and limitations of that network.

3. Nonkinetic Energy

Nonkinetic energy weapons threaten and inflict damage through a method other than physical application to the body of a human, ship or aircraft. These methods include sound, radiation, and chemical agents among others. Many interviewees expressed the desire to utilize this method of warfare. These nonkinetic methods of fighting eliminate the firing of projectiles from the ship; rather they involve the firing of a widespread gas, sound, or wave of some sort. This would aid to increase the P_k by being able to fire in the

direction of the UAV rather than directly at it. The laser weapon system that is fielded on a single DDG currently has been tested on UAVs before. However, the testing results are classified and therefore out of scope for this report. The fleet will be gaining several other nonkinetic weapons within the next several years, which could potentially be tested to be used to defeat UAVs. While these capabilities are not currently within the scope of this project, with further testing that will likely be classified, the weapon systems that will be added to the ships in the future could be used as a weapon against UAVs and assimilated within the tool in the classified realm.

4. Kill Radius

The interviewees also discussed the desire to have a kill radius surrounding the ship. Not all UAVs would have to actually come into physical contact with a ship in order for it to cause damage. It was expressed that a visual representation of this radius that the UAV would have to come within to be counted as a kill would be extremely useful. This would alter the simulation as currently the kill ring is based on the size of the ship. By accounting for the type of UAV (one that could cause damage without contact) a new kill ring would need to be determined. This could be developed in the future as the types of enemy UAVs evolve.

5. Identification of UAV Type

A useful piece of information that would be necessary prior to engaging with the UAVs would be the classification of UAV. In the future, by incorporating a tab with an output of various known UAVs, the tool could print out a file that would include characteristics of the hostile UAVs. This tab would identify the threats presented by the particular attacking UAV. It would also include a picture of the UAV for reconnaissance.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION AND AREAS OF FURTHER STUDIES

The following are the conclusions based on the research questions and stakeholders' needs. The recommendations are geared towards future development studies of this topic.

A. CONCLUSIONS

The goal of this project was to examine the need for a UAV Swarm Risk Assessment Tool and create a UAV Swarm Operational Risk Assessment System GUI. The authors made use of human-centered design principles in conjunction with more traditional system engineering tools to explore the problem space, identify requirements, analyze functions, and design a solution for the problem. The final outcome was a working prototype that is responsive to the listed requirements and stakeholder needs. The GUI can assist the Navy's decision makers in assessing risk of UAV swarm threats in littoral environments, near potentially hostile countries, based on the latest intelligence.

The system concept consisted of several parts: a discrete-event simulation of UAV swarm attacks using ExtendSim, a statistical analysis of the simulation data analyzed using Minitab, and a graphical user interface (GUI) that evolved as a web-app prototype using MATLAB. Data from the simulation were analyzed and used to generate equations that calculate the effect of critical factors such as physical environment, number of UAVs, distance from land, and the ship's defensive weapons. The GUI uses these equations to provide users with the capability to vary factors relevant to a UAV swarm attack and analyze different courses of action and risk.

This project created a physical GUI web-app that can be used, tailored or expanded for future studies and updated with the latest intelligence. This web-app prototype can assist decision makers in assessing risk of UAV swarm threats in littoral environments, near potentially hostile countries. The GUI gives information critical to assess the risk of swarm attacks in littoral environments to analyze different course of action and risk. As the project progressed, the answers to the research questions were discovered and documented.

1. Research Questions

This project was to provide commanding officers, carrier task force commanders, fleet commanders, operational planning teams or area of responsibility commanders with a way to better assess the risks presented to them by UAV swarm attacks. In order to design an appropriate system to assess risk, a variety of research questions were asked.

a. What are the types, capabilities, and flight envelopes for unmanned aerial "attack" vehicles of hostile foreign countries?

In order to answer the threat UAV types, capabilities, and flight envelopes, an analysis of UAV technologies available to hostile nations was conducted. As shown in Table 1 of the report, the flight envelopes of current day threat UAVs are drastically different from that of conventional air threats such as fighter jets and cruise missiles. Their relatively low speed (80 to 200 kts), operational ceilings (less than 15,000 ft MSL), operational ranges (27 to 270 nm), and low payloads (16 to 100 lbs) make them dramatically less capable than traditional air threats; however, these differences also give them a distinct advantage in terms of how they are tracked, engaged, and ultimately defeated.

The primary driver for performance of UAVs, as compared to manned aircraft, is their overall size. In general, they are much smaller than their manned counterparts, which is one of the primary drivers of their low costs. Aircrew life support systems and human machine interfaces are not required for UAVs. Specific design criterion such as survivability and reliability may also become less important without human aircrew in the airframe. This overall decrease in cost leads to the potential for more air threats attacking simultaneously.

Modern air defense radar systems were designed for traditional air threats that are larger and fly much faster. Since the likelihood of more UAVs attacking at one time is increased, the detection and identification of singular threat becomes much more difficult. As such, the timelines for battle commanders to decide "who" to engage, "how" to engage, and "when" to engage are also impacted. Ultimately, hostile countries can use the characteristics of "less" capable UAVs to their advantage. The drastic differences

between UAVs and traditional air threats are ultimately what make UAVs a new and formidable threat to surface ships.

b. What tangible information, systems, and interactions are needed by surface ship commanding officers, carrier task force commanders, fleet commanders, operational planning teams or area of responsibility commanders to make a risk decision for an operation?

Tangible information, systems, and interactions needed by surface ship commanding officers, carrier task force commanders, fleet commanders, operational planning teams or area of responsibility commanders to make a risk-based decision for an operation was determined using interviews of stakeholders for user needs. Interviewees covered the full kill chain from CAPTs to LTs to gain a wide variety perspective tactical and strategic. The interview results were dissected into user needs and placed into affinity diagrams. These affinities organized user needs for model input, model output, tool/application, user interface, blue force capabilities, and rules of engagement.

A major factor was UAV detection. Stakeholders addressed issues with how the littoral environment has impacted the capability to detect and track. The environmental factors included time of day, visibility, sea state, weather and clutter. These factors, along with the availability and readiness of the CIWS, five-inch gun, and MGS scope the engagement context to evaluate a risk-based decision for an operation.

The other aspect of concern to the stakeholders was the type, number and location of UAVs. This directly relates to the time for a CO to react to detection. Stakeholders are uncomfortable with attacking a UAV until they identify it. Hostile intent or act is even more difficult to identify. The CO wants to know how much time he has to engage before he is vulnerable.

c. What are the standard and nonstandard navy tactics for UAV swarm attack?

The standard and nonstandard navy tactics for UAV swarm attack were examined based on ships operating in a littoral environment. A constrained environment such as a straits transit presents several challenges. Challenges include restricted maneuvering, a potential for high air and surface traffic to include civilian, commercial and military all

within close proximity of each other. The Law of the Sea dictates that warships and military aircraft:

> …must avoid "any threat or use of force against the sovereignty, territorial integrity or political independence of the State bordering the strait,"… (Alexander 1991, 92)

Identifying a hostile act with a UAV proves much more difficult than the identification of a manned aircraft. A UAV may be observing or capturing data of the ship at a distance and not actually engaging. The tactics for a surface ship against a swarm of UAVs is typically to maneuver out of the swarm environment. Surface ships are hesitant to engage UAVs with weapons systems.

d. What are the gaps in technology and processes for UAV swarm response and what are the current work-arounds being used by the fleet?

Gaps in technology and processes for UAV swarm were also examined. It is easy to conclude that there is very little technology out there that addresses the problems described in this project. The available references covering the current capabilities relevant to the project fell into two categories: UAV swarm attack simulations and operational planning user tools. There are UAV swarm attack simulations but the results have not been implemented into any other technology. Operational planning tools were discovered but none were UAV centric or addressed the problem this project is trying to solve.

2. Requirements

User needs were used to determine the requirements. One of the major needs revolved around the visual representation of hostile threat geographically. Inputs such as type, number, and location of UAVs are needed along with types of ships/combinations of ships in order to produce a unique risk output to their situation. Time to react gave the ship the ability to maneuver instead of engage. If the ship were to engage, this tool would also inform the amount of time the ship had to counter the attack.

3. Simulation and GUI

The production of an ExtendSim based discrete-event simulation of the UAV swarm engagement allowed the authors to explore the trade-space and identify those input factors that have the greatest effect on the scenario outcome. The analysis shows that the most significant factors are: the number of UAVs in the swarm, the distance from the ship to the shore, and the speed of the UAVs. Besides those three factors, there are two other factors with less significance but that still pass the statistical test for significance. These factors are the prevailing meteorological visibility (which affects the range that the Mk15 MGS can be employed) and the number of Mk38 CIWS systems. The other six factors consider in this study, as part of the initial set of simulation runs discussed in Chapter IV Section D, did not significantly affect the simulation outcome and were changed to fixed values for the final runs of the simulation. This factor screening reduced the original eleven factors down to just four controllable factors and one categorical factor (Number of CIWS). This allowed keeping the total number of runs down to a manageable level.

Using Minitab, a regression was used to generate a response equation to represent the simulation output. This response equation drives the MATLAB GUI enabling immediate responses to user inputs, without the need to wait for a new simulation run. Ultimately this methodology enabled the authors to create a prototype that was closer to realism than the authors would otherwise have been able to create without these tools.

Of equal importance to the discrete-event simulation is the user interface developed to let users explore, interpret, and share the simulation results. A prototype GUI was developed in MATLAB, emulating a web-based application. This prototype followed several generations of iterative improvement starting with whiteboard drawings and guided through the iteration by stakeholder feedback. The final version is a refined and very capable web interface that was well received by the stakeholders.

The GUI prototype is the result of many iterations of human-centered design reflecting the feedback of many potential users or user representatives. Stakeholder interest in responsive design, low wait times, integration into existing computer systems,

and assurance of up-to-date intelligence guided the authors to a web-based solution. The final GUI from this effort, though developed in MATLAB, was built to emulate a web application.

The user interface is designed to be intuitive and exportable. The results are easy to produce, transport, and interpret because the user in front of the computer is often not the decision maker who needs the information based on feedback received.

Guided by the feedback of the stakeholders, the interface presents the UAV threat information in two ways; details of the GUI are discussed in Chapter V. First is a map, displaying user-specified launch sites and a colored contour map of likelihood of a zero-casualty encounter. (The simulation does not characterize the impact of a UAV hitting the ship and stakeholders indicated they would not trust a tool that did.) The second output is the inverse and is centered on the ship rather than the UAVs. It indicates the remaining reaction time available to the ship before they must open fire in order to have a reasonable probability of repelling all of the UAVs according to the simulation.

B. RECOMMENDATIONS FOR FURTHER STUDIES

Future groups who might carry this project to the next phase should plan to revisit the extensive set of assumptions and simplifications built into the ExtendSim simulation. These were necessary during this first attempt, in order to keep the effort unclassified and to remain within the constraints of schedule, computing power, and technical skills of the students; however, a higher fidelity modeling effort may yield insights into the nature of the UAV swarm engagement that could affect the GUI design or the general concept.

A follow on effort should focus on contacting the NAVSEA AEGIS Integrated Combat System Threat Analysis Assistant Program Manager (APM) Program Executive Office (PEO) Integrated Weapons Systems (IWS) to investigate the potential for making this an official software acquisition project. PEO IWS may choose to have professional modelers and web developers create a high fidelity simulation based on classified intelligence analysis to produce true decision quality inputs for the GUI. They may also have web developers produce a web page or have the Aegis weapon system prime contractor provide a quote for integrating such a capability into the weapon system. The

follow on strategy could also research what organization would be most appropriate to conduct the analyses needed to periodically update the model inputs.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. UAV SWARM SIMULATION DESIGN

An overview of the discrete-event simulation was provided in Chapter IV. This appendix provides a detailed step-by-step description of the model. Using ExtendSim9, the simulation was laid out following the basic process flow architecture shown in Figure 64. The first task upon initialization of the simulation is to read in the variables for each run from an input database.

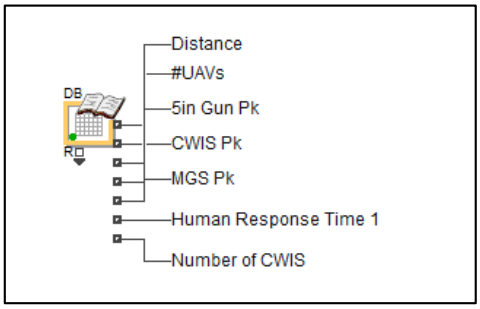

Figure 64. Model Read-In from Database

Next the simulation must calculate several pieces of information before and during the run. Given the distance and applying a uniform random distribution for the azimuth angle, it assigns the X and Y coordinates of the starting position of the UAV swarm as depicted in Figure 65.

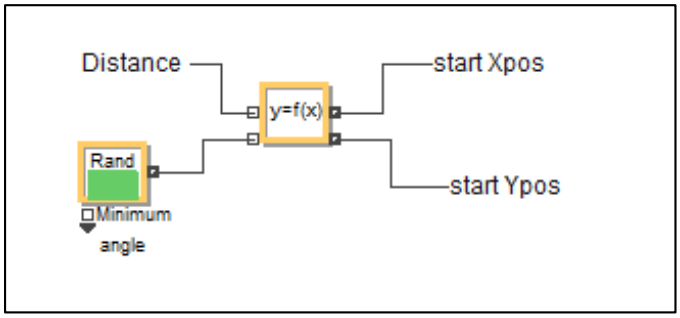

Figure 65. Setting the UAV Start Position

The sea state, Figure 66, is randomly set to integers 1, 2, 3, or 4 matching the standard maritime sea state definitions, and then converted to a degradation value used later in the Probability of Kill determination.

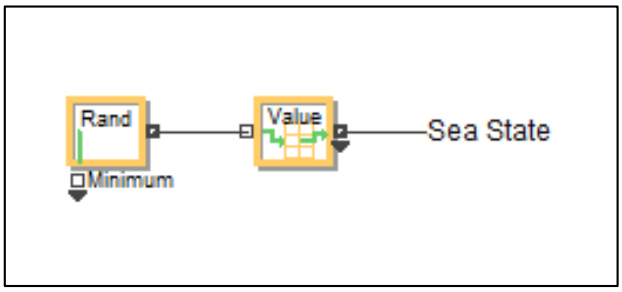

Figure 66. Calculating Sea State

The visibility distance, Figure 67, is set based on a uniform random distribution returning values between 20 miles and 0.2 miles.

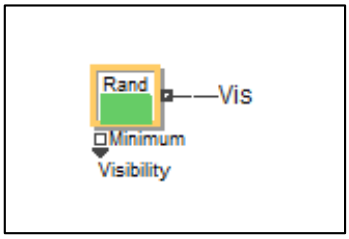

Figure 67. Setting Visibility

As the simulation runs, at each time step the new ship's position is based on the ship's speed, Figure 68, and starting location of (0,0). The ship moves along the X axis while the UAVs move in both X and Y in order to intercept the ship.

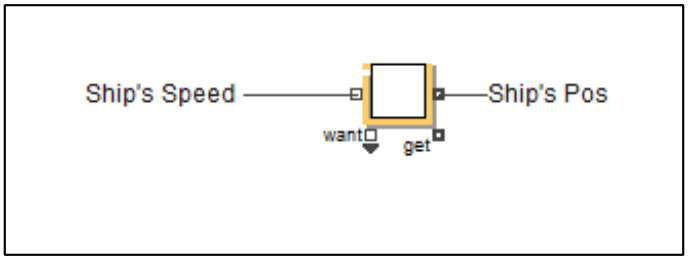

Figure 68. Calculating the Ships Position

The simulation then reads the number of UAVs, Figure 69, to be created each run and creates them with a slight random time delay between UAVs to simulate the fact that are probably not all launched at exactly the same time. The queue block that follows the create block is provided to ensure stability of the simulation and does not affect the outcome.

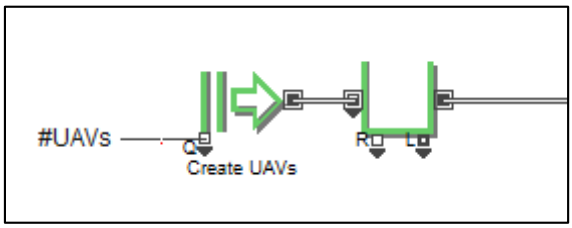

Figure 69. Creating UAVs

Next the attributes, Figure 70, of starting position and speed are assigned to the UAVs as they pass through the Set Attributes block.

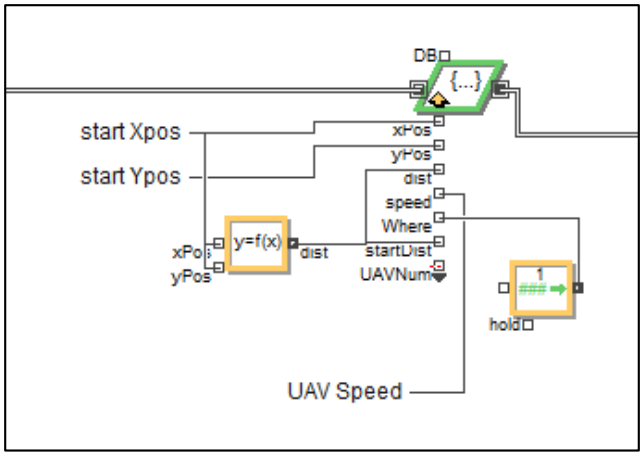

Figure 70. Setting UAV Attributes

Ship motion, Figure 71, is restricted to the X-axis, while the UAV can move in both the X- and Y- axes. The logic behind it is similar to that of a homing missile. Every time the ship moves the UAV must change direction in order to track toward the ship and intercept it. This makes the UAV's motion dependent on the ship's position. The math behind this "homing" behavior is performed in the motion generator equation block.

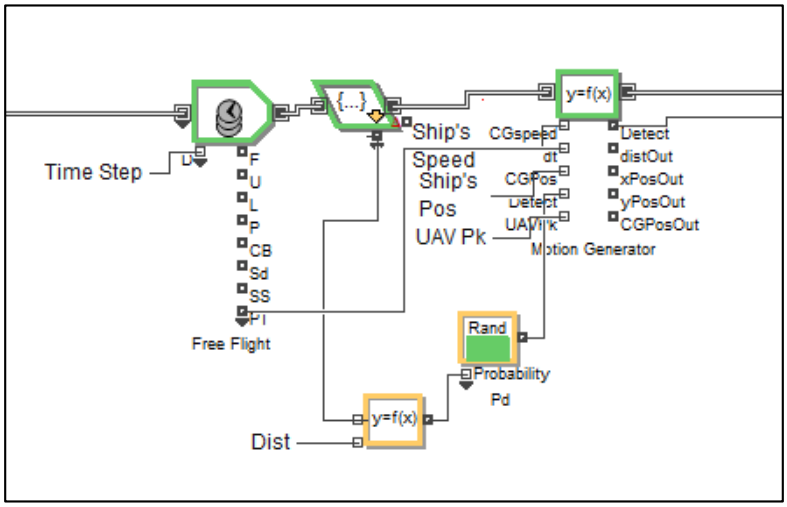

Figure 71. UAV Motion Generator

After the motion and distance have been determined, the simulation must decide when the UAVs are detected by the ship. Figure 72 shows the blocks that calculate the

range and use the Probability of Detection to determine at what point the UAVs have been detected.

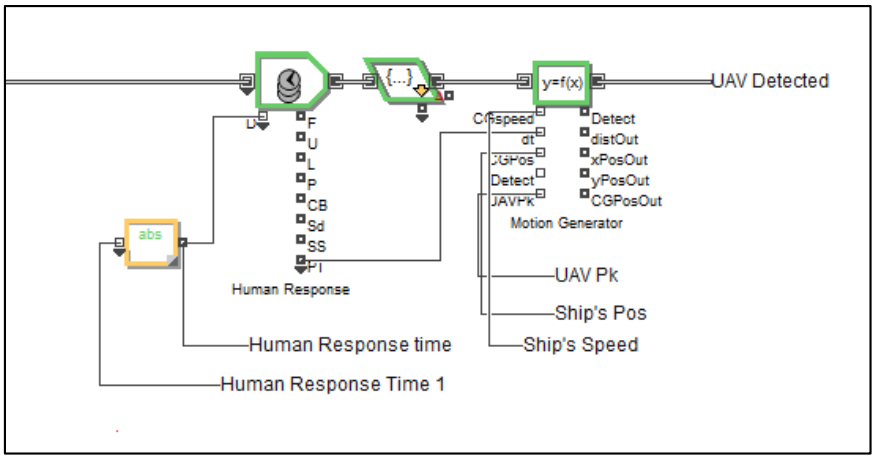

Figure 72. UAV Detection

Once the UAVs are detected, they can be engaged by the ship's weapon systems. This next set of blocks in Figure 73, handles the weapons engagement, determining if each shot attempt kills or misses the UAV. Each weapon system has its own unique P_K based on range, sea state and visibility. The closest UAV always has the priority for engagement. The appropriate weapon is selected is based on each UAVs distance to the ship. The equation block on the lower left of Figure 73 determines the maximum effective range of those weapons that are limited by visibility and also adjusts the Pk based on sea state. The three random number generators, one for each weapon system, randomly generate a number, between zero and one, for each shot attempt. If the randomly generated number is less than the Pk for that weapon system, then the UAV is hit and destroyed; otherwise, it is a miss. This comparison is done in the equation block on the upper right of Figure 73, and the appropriate signal is sent to the sorting block on the left side of Figure 74, to route the UAV through the model based on its status. While the engagement unfolds, the remaining UAVs continue to close with the ship, so range is constantly decreasing. Sea state and visibility remain constant within each run.

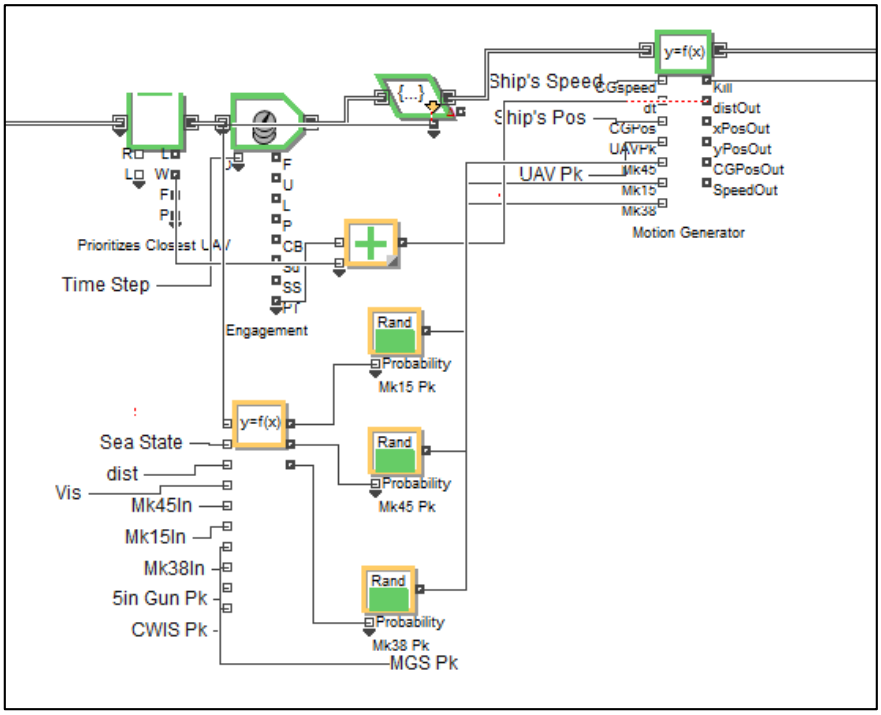

Figure 73. Weapon System Engagement

Based on the outcome of the equation block on the right side of Figure 73, each UAV is determined to have reached the ship, or to still be flying, or to have been destroyed by one of the three weapon systems. That equation block controls the sorting of through the "select out" box on the left side of the Figure 74. UAVs that are still flying and have not reached the ship yet are routed back through the engagement blocks in Figure 73. If a UAV reaches the ship, Figure 74, it is routed to the end block labeled "Ship Kill" and the simulation run ends. If the UAV is shot down, it is routed to the end block corresponding to the weapon system that shot it down.

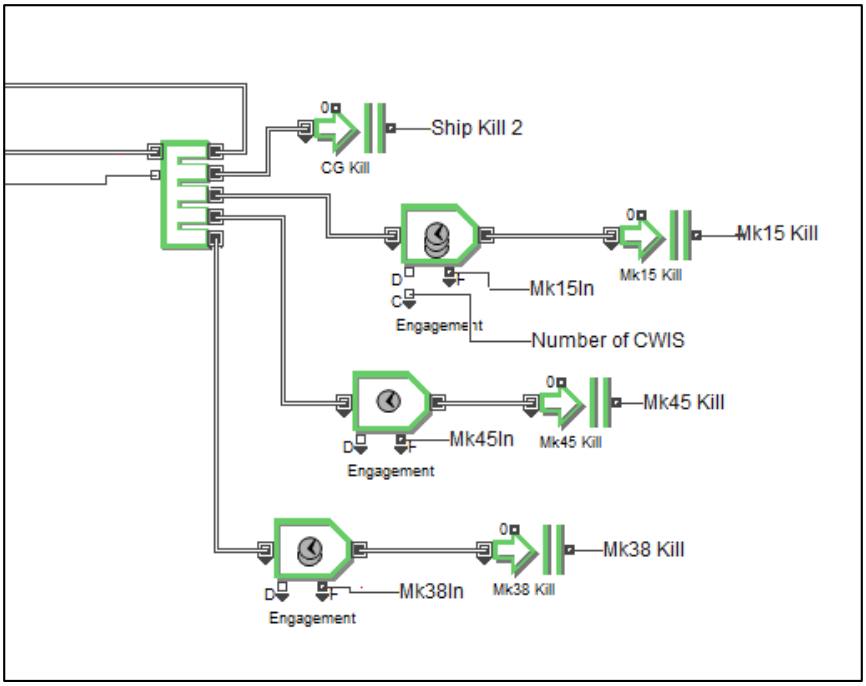

Figure 74. UAV Disposition

This equation block, shown in Figure 75, ends the current run when the number of UAVs created is less than or equal to the sum of the UAVs shot down by the three weapon systems. It will also stop the run when any UAV makes it to the ship.

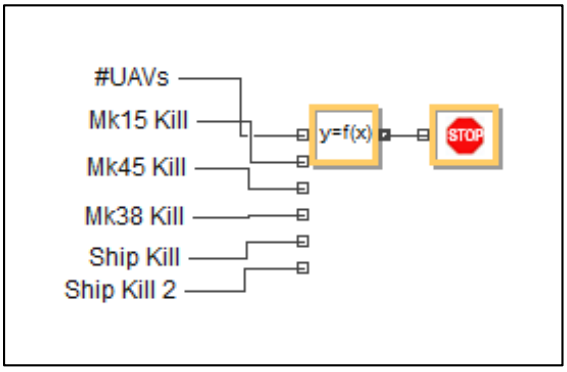

Figure 75. Simulation Stop Logic

The final step in each run is to write the engagement statistics to an ExtendSim database, Figure 76. This database was exported to Minitab for post-simulation analysis.

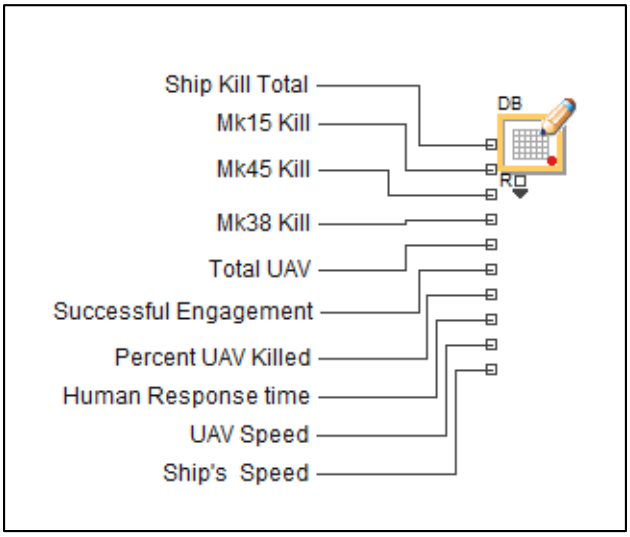

Figure 76. Writing the Output to a Database

APPENDIX B. MATLAB SCRIPT

The final prototype application is run from a MATLAB function called uav_threat_application.m. The contents of this function are copied below.

```
function uav_threat_application
% variable parameters
groundcolor = [0.9,0.85,0.5];

fig = gcf;
set(fig,'Visible','off')
clf
ax = gca;

set(gcf,'MenuBar','none')
set(gcf,'DockControls','off')
set(gcf,'ToolBar','none')
set(gcf,'NumberTitle','off')

% Set random seed for repeatability
R = RandStream.create('mrg32k3a','Seed',134);%'shuffle'); % was 134
F = designfilt('lowpassiir', 'FilterOrder', 4, ...
    'PassbandFrequency', 15e3, 'PassbandRipple', 0.2,...
    'SampleRate', 200e3);

% Define coastline
N = 1e3;
x = 0:N;
y = N - 150*x/N - 300*(x/N-1/2).^2;
y = y + filtfilt(F,cumsum(randn(R,size(x))));

% Hard code the X axis limits
xl = [410,590];
% Then calculate sensible y limits (used to be yl = [864,948];)
yl = y(x>xl(1)&x<=xl(2));
set(ax,'Units','pixels');
apos = get(ax,'Position');
yl = min(yl)+diff(xl)*apos(4)/apos(3)*[-2,3]/5;
% set(ax,'


% init with dummy then empty it
threat(1) = addThreat(0,0); threat(1) = [];
% add some real launch sites for the demo
threat(end+1) = addThreat(513,917);
threat(end+1) = addThreat(450,940);
threat(end+1) = addThreat(570,915);
uavCallback(threat(1).button)

% Define image vars for threat plotting
sqrtpixelcount = 1e3;
x0 = linspace(xl(1),xl(2),sqrtpixelcount);
y0 = linspace(yl(1),yl(2),sqrtpixelcount);
[X,Y] = meshgrid(x0,y0);
% Calculate color index from distance to threats using
% alpha mapping to indicate land vs. water
land = Y >= interp1(x,y,X);
threatmap = imagesc(x0([1,end]),y0([1,end]),inf(size(X)),'AlphaData',~land+0.15);
hold on
[~,threatlines] = contour(X,Y,X,0:10:100,'ShowText','on','LineColor','k');
hold off

% Tweak axes settings
```

```
set(ax,'Color',groundcolor,'Units','normalized','Position',[0,0,1,1],...
    'YDir','normal','CLim',[0,100])
grid(ax,'on')
set(ax,'XTick',xl(1):10:xl(2))
set(ax,'YTick',yl(1):10:yl(2))
cmap = flipud(colormap('parula'));
colormap(cmap)
% Setup colorbar with background for clarity
patch(xl(2)-diff(xl)/24*[0,1,1,0],yl(1)+diff(yl)*[0,0,1,1],[0,0,0],...
    'EdgeColor','none','FaceAlpha',0.3)
hc = colorbar(ax,'east');
set(hc,'Color',[1,1,1])

% Create a patch for the Ship inputs/info tab
tabs(1) = createTab('Ship Inputs',1);
% tabs(end).children(end+1) = uicontrol('Style','text','String','test',...
%     'Position',[30,600,170,20]);
tabs(end).children(end+1) = text(xl(1)+diff(xl)*0.027,yl(1)+diff(yl)*0.94,...
    'Select your ship:','FontSize',14,'Color',[1,1,1]);
ships = {'DDG 01','DDG 02'};
tabs(end).children(end+1) = uicontrol('Style','popupmenu','String',ships,...
    'Position',[30,570,170,20]);
tabs(end).children(end+1) = text(xl(1)+diff(xl)*0.027,yl(1)+diff(yl)*0.84,...
    'Number of CIWS:','FontSize',14,'Color',[1,1,1]);
tabs(end).children(end+1) = uicontrol('Style','popupmenu','String',{'1','2'},...
    'Position',[30,504,170,20],'Callback',@updateThreatMap);
numCiws = @() get(tabs(end).children(end),'Value');
% Create a patch for the UAV inputs/info tab
tabs(end+1) = createTab('UAV Inputs',tabs(end).bottom);
tabs(end).children(end+1) = text(xl(1)+diff(xl)*0.027,yl(1)+diff(yl)*0.94,...
    'Select UAV:','FontSize',14,'Color',[1,1,1]);
uavs = {'Harpy','Custom'};
tabs(end).children(end+1) = uicontrol('Style','popupmenu','String',uavs,...
    'Position',[30,570,170,20]);
tabs(end).children(end+1) = text(xl(1)+diff(xl)*0.027,yl(1)+diff(yl)*0.84,...
    'Number of UAVs:','FontSize',14,'Color',[1,1,1]);
tabs(end).children(end+1) = uicontrol('Style','edit','String',1,...
    'Position',[30,504,170,20],'Callback',@updateThreat,'UserData','num');
tabs(end).children(end+1) = text(xl(1)+diff(xl)*0.027,yl(1)+diff(yl)*0.74,...
    'UAV Speed: (200 knots)','FontSize',14,'Color',[1,1,1]);
tabs(end).children(end+1) = uicontrol('Style','slider',...
    'Position',[30,438,170,20],'Min',50,'Max',300,'Value',200,'UserData','speed',...
    'Callback',{@setLabel,tabs(end).children(end),'UAV Speed: (%.0f
knots)',@updateThreat});
tabs(end).children(end+1) = uicontrol('Style','pushbutton',...
    'Position',[40,60,150,30],'String','Switch Views',...
    'Callback',@switchViews);
tabs(end).children(end+1) = text(xl(1)+diff(xl)*0.10,yl(1)+diff(yl)*0.072,...
    '+','FontSize',36,'Color',[1,1,1],'ButtonDownFcn',@newThreat);
tabs(end).children(end+1) = text(xl(1)+diff(xl)*0.05,yl(1)+diff(yl)*0.072,...
    '-','FontSize',36,'Color',[1,1,1],'ButtonDownFcn',@rmvThreat);
% Create an Environment tab
tabs(end+1) = createTab('Environment',tabs(end).bottom);
tabs(end).children(end+1) = text(xl(1)+diff(xl)*0.027,yl(1)+diff(yl)*0.94,...
    'Visibility: (75%)','FontSize',14,'Color',[1,1,1]);
tabs(end).children(end+1) = uicontrol('Style','slider',...
    'Position',[30,570,170,20],'Min',20,'Max',100,'Value',75,...
    'Callback',{@setLabel,tabs(end).children(end),'Visibility: (%.0f%%)'});
visibility = @() get(tabs(end).children(end),'Value');
% Create the Notes tab
tabs(end+1) = createTab('Notes',tabs(end).bottom);
tabs(end).children(end+1) = uicontrol('Style','edit','String','Add your notes
here...',...
    'Position',[30,30,170,600],'Max',2,'HorizontalAlignment','left');
% Create the Export tab
tabs(end+1) = createTab('Export',tabs(end).bottom);
tabs(end).children(end+1) = text(xl(1)+diff(xl)*0.027,yl(1)+diff(yl)*0.94,...
    'Download report as:','FontSize',14,'Color',[1,1,1]);
tabs(end).children(end+1) = uicontrol('Style','pushbutton',...
```

```
    'Position',[40,550,150,30],'String','PDF');
tabs(end).children(end+1) = uicontrol('Style','pushbutton',...
    'Position',[40,510,150,30],'String','Microsoft Word');
tabs(end).children(end+1) = uicontrol('Style','pushbutton',...
    'Position',[40,470,150,30],'String','Microsoft PowerPoint');

% Minimize all but one tab
switchTabs(tabs(end).text)
% Minimize last tab
% switchTabs(tabs(end).text)

% Show the user!
drawnow % make sure everything is pretty first
set(fig,'Visible','on')


% the DeleteFcn of the uicontrol will take care of the rest
delete(threat(end).button)

    function [Z,levels] = genThreatMap
        Z = inf(sqrtpixelcount,sqrtpixelcount,max(1,length(threat)));
        vis  = 1 - visibility()/100; % read it once only to save time
        if strcmp(get(threat(1).button,'Visible'),'on') % map view
            for iii = 1:length(threat)
                dist = sqrt((X-threat(iii).x).^2+(Y-threat(iii).y).^2);
                if numCiws() == 1
                    Z(:,:,iii) = 0.0585 + 0.01707*threat(iii).num + 0.06161*dist...
                        - 0.00108*threat(iii).speed - 0.5414*vis -
0.000839*threat(iii).num*dist...
                        -0.000053*threat(iii).num*threat(iii).speed -
0.01080*threat(iii).num*vis...
                        - 0.000149*dist*threat(iii).speed - 0.00312*dist*vis -
0.000958*threat(iii).speed*vis;
                else % numCiws() == 2
                    Z(:,:,iii) = 0.2561 + 0.0177*threat(iii).num + 0.06419*dist...
                        - 0.000431*threat(iii).speed - 0.5675*vis -
0.000839*threat(iii).num*dist...
                        -0.000053*threat(iii).num*threat(iii).speed -
0.01080*threat(iii).num*vis...
                        - 0.000149*dist*threat(iii).speed - 0.00312*dist*vis -
0.000958*threat(iii).speed*vis;
                end
            end
            % Z = 100*min(Z,[],3); % treat each attack independently
            Z = 100*prod(Z,3); % treat all attacks jointly (but not overlapping)
        else % ship centered view
            iii = find([threat.active],1);
            dist = sqrt((X-xl(1)-diff(xl)/2).^2+(Y-yl(1)).^2);
            desiredSuccessRate = 0.90;
            levels = 10:10:100; % distances at which to plot time remaining (in minutes)
            if numCiws() == 1
                Z = (0.585 + 0.01707*threat(iii).num...
                    - 0.00108*threat(iii).speed - 0.5414*vis...
                    - 0.000053*threat(iii).num*threat(iii).speed...
                    - 0.01080*threat(iii).num*vis...
                    - 0.000958*threat(iii).speed*vis - desiredSuccessRate)...
                    ./...
                    (0.06161 - 0.000839*threat(iii).num...
                    - 0.000149*threat(iii).speed - 0.00312*vis);
                levels = (levels + Z)./(threat(iii).speed)*60;
                Z = (dist + Z)./(threat(iii).speed)*60;
            else % numCiws() == 2
                Z = (0.2561 + 0.0177*threat(iii).num...
                    - 0.000431*threat(iii).speed - 0.5675*vis...
                    - 0.000053*threat(iii).num*threat(iii).speed...
                    - 0.01080*threat(iii).num*vis...
                    - 0.000958*threat(iii).speed*vis - desiredSuccessRate)...
                    ./...
                    (0.06419 - 0.000839*threat(iii).num...
```

```
                    - 0.000149*threat(iii).speed - 0.00312*vis);
                levels = (levels + Z)./(threat(iii).speed)*60;
                Z = (dist + Z)./(threat(iii).speed)*60;
            end
        end
    end

    function threat = addThreat(x,y)
        threat.x = x;
        threat.y = y;
        wx = 1/30; % button width
        ux = (x-xl(1))/diff(xl) - wx/2;
        wy = diff(xl)/diff(yl)/30;
        uy = (y-yl(1))/diff(yl) - wy/2;
        png = which('uav_icon.png');
        str = ['<html><img src="file:',png,'" width="40" height="40"/></html>'];
        threat.button = uicontrol('Style','pushbutton','Parent',fig,...

'Units','normalized','Position',[ux,uy,wx,wy],'BackgroundColor',groundcolor,...
            'Callback',@uavCallback,'String',str,...
            'DeleteFcn',{@updateThreatMap});
        threat.num = 1;
        threat.speed = 200;
        threat.active = false;
    end

    function tab = createTab(name,top)
        tab.name = name;
        tab.pos = [xl(1) diff(xl)/6 yl(1) diff(yl)];
        tab.patch = patch(tab.pos(1)+tab.pos(2)*[0,1,1,1.1,1.1,1,1,0],...
            tab.pos(3)+tab.pos(4)*[0,0,0.1,0.11,0.89,0.9,1,1],[0,0,0],...
            'EdgeColor','none','FaceAlpha',0.3);
        tab.text = text(tab.pos(1)+1.035*tab.pos(2),...
            tab.pos(3)+(top-0.02)*tab.pos(4),name,'FontSize',18,...
            'Color',[1,1,1],'Rotation',270,...
            'HorizontalAlignment','left',...
            'VerticalAlignment','middle',...
            'ButtonDownFcn',{@switchTabs});
        % Reposition the tab patch to match the text
        ext = get(tab.text,'Extent');
        y1 = (ext(2)       -tab.pos(3))/tab.pos(4) - 0.01;
        y2 = (ext(2)+ext(4)-tab.pos(3))/tab.pos(4) + 0.01;
        ydata = tab.pos(3)+tab.pos(4)*[0,0,y1-0.01,y1,y2,y2+0.01,1,1];
        set(tab.patch,'YData',ydata)
        tab.top = top;
        tab.bottom = y1-0.01;
        tab.children = []; % to be filled later
    end

    function switchTabs(caller,~)
        this = [tabs.text]==caller;
        xpos = cellfun(@(c)c(1),get([tabs.text],'Position'));
        [~,ismax] = max(xpos);
        if any(diff(xpos)) % max was greater and not equal to the others
            this(ismax) = false; % always minimize active tab
        end
        n = 15;
        for iii = 1:n
            nudgeTab(tabs(this),  1/n)
            nudgeTab(tabs(~this),-1/n)
        end
    end

    function nudgeTab(tabs,step0)
        w = get(fig,'Position');
        w = w(3); % figure width for use below
        % a step of 1 is the full tab width
        for iii = 1:length(tabs)
            tab = tabs(iii);
```

```
            xdata = get(tab.patch,'XData');
            tpos = get(tab.text,'Position');
            step = step0*diff(xdata(1:2));
            if xdata(1)+step>xl(1) || xdata(2)+step<xl(1)
                continue % refuse to detach tab from left edge
            end
            set(tab.patch,'XData',xdata+step)
            tpos(1) = tpos(1)+step;
            set(tab.text,'Position',tpos)
            for iiii=1:length(tab.children)
                nudgeChild(tab.children(iiii),step,w)
            end
        end
        textOnTop
        drawnow
    end

    function nudgeChild(h,step,w)
        switch get(h,'Type')
            case 'uicontrol'
                pos = get(h,'Position');
                pos(1) = pos(1)+step*w/diff(xl);
                set(h,'Position',pos)
            case 'text'
                pos = get(h,'Position');
                pos(1) = pos(1) + step;
                set(h,'Position',pos)
            otherwise
                disp(get(h,'Type'))
        end
    end

    function textOnTop
        h = get(ax,'Children');
        these = arrayfun(@(h)isa(h,'matlab.graphics.primitive.Text'),h);
        h = [h(these);h(~these)];
        set(ax,'Children',h)
    end

    function updateThreatMap(caller,~)
        if strcmp(get(fig,'Visible'),'on') && exist('threatmap','var') &&
ishandle(threatmap)
            threat([threat.button]==caller) = [];
            if strcmp(get(threat(1).button,'Visible'),'on'); % map view
                Z = genThreatMap;
                set(threatlines,'ZData',Z,'LevelList',10:10:100)
            else
                [Z,levels] = genThreatMap;
                levels = [0,round(levels(levels>0)*100)/100];
                set(threatlines,'ZData',Z,'LevelList',levels)
            end
            set(threatmap,'CData',Z)
        end
    end

    function setLabel(caller,~,label,format,varargin)
        if ~isempty(varargin)
            feval(varargin{1},caller,[])
        end
        set(label,'String',sprintf(format,get(caller,'Value')))
        updateThreatMap(0)
    end

    function updateThreat(caller,~)
        ii = find([threat.active],1);
        if ~isempty(ii)
            switch get(caller,'UserData');
                case 'num'
                    threat(ii).num = nanmax(0,round(str2double(get(caller,'String'))));
```

```
                case 'speed'
                    threat(ii).speed = get(caller,'Value');
            end
        end
        updateThreatMap(0)
    end

    function uavCallback(caller,~)
        set([threat.button],'BackgroundColor',groundcolor)
        set(caller,'BackgroundColor',[0.85,0.2,0.15])
        ii = find([threat.button]==caller,1);
        [threat.active] = deal(false);
        threat(ii).active = true;
        h = findobj('UserData','num');
        if isempty(h) % GUI not initialized yet
            return
        end
        set(h,'String',num2str(threat(ii).num))
        recall(h)
        h = findobj('UserData','speed');
        set(h,'Value',threat(ii).speed)
        recall(h)
    end

    function recall(h)
        f = get(h,'Callback');
        if iscell(f)
            feval(f{1},h,[],f{2:end})
        else
            feval(f,h)
        end
    end

    function switchViews(~,~)
        wasmapview = strcmp(get(threat(1).button,'Visible'),'on');
        if wasmapview
            set([threat.button],'Visible','off')
            set(threatmap,'AlphaData',ones(size(land)))
            set(ax,'CLim',[0,30])
        else
            set([threat.button],'Visible','on')
            set(threatmap,'AlphaData',~land+0.15)
            set(ax,'CLim',[0,100])
        end
        updateThreatMap(0)
    end

    function rmvThreat(~,~)
        delete(threat([threat.active]).button)
        if ~isempty(threat)
            threat(end).active = true;
        end
    end

    function newThreat(~,~)
        [gx,gy] = ginput(1);
        threat(end+1) = addThreat(gx,gy);
        updateThreatMap(0);
    end
end % application
```

LIST OF REFERENCES

Alexander, Lewis M. 1991. “International Law Studies the Law of Naval Operations.” *Naval War Collage Press,* edited by Horace B. Robertson, Jr., 64: 92. http://purl.fdlp.gov/GPO/gpo3903.

Button, Robert W., David R. Frelinger, Brian A. Jackson, and Micheal J. Lostumbo. 2008. *Evaluating Novel Threats to the Homeland: Unmanned Aerial Vehicles and Cruise Missiles.* Alexandria, VA: RAND Corporation. http://site.ebrary.com/lib/nps/Doc?id=10240746.

Cenciotti, David. 2013. “Iran has Unveiled a New Drone Based on a Captured U.S. Boeing Scan Eagle.” *The Aviationist,* November 9. http://theaviationist.com/2013/09/29/yasir-drone/.

Cobb, Brandon J. 2011. “Adaptive Discrete Event Simulation for Analysis of Harpy Swarm Attack.” Naval Postgraduate School: Monterey, California. http://edocs.nps.edu/npspubs/scholarly/theses/2011/September/11Sep_Cobb.pdf.

Curry, Richard G. 2011. *Radar Essentials a Concise Handbook for Radar Design and Performance Analysis.* Raleigh, NC: SciTech Pub. http://libproxy.nps.edu /login?url=http://app.knovel.com/web/toc.v/cid:kpREACHRD3.

Davis, Lynn E., Michael J. McNerney, James Chow, Thomas Hamilton, Sarah J. Harting, and Daniel Byman. 2014. “Armed and Dangerous? UAVs and U.S. Security.” *International Security and Defense Policy Center and Rand National Security Research Division,* Santa Monica, CA. http://www.rand.org/pubs/research_reports/RR449.html.

Gortney, W.E., and C.D. Haney. 2013. “Introduction to the Readiness Kill Chain.” Accessed April 26, 2015. http://www.public.navy.mil/usff /documents/rkc_booklet_final-w-signatures_11apr13.pdf.

Grasso, Raffaele, Marco Cococcioni, Baptiste Mourre, Jacopo Chiggiato, and Michel Rixen. 2012. “A Maritime Decision Support System to Assess Risk in the Presence of Environmental Uncertainties: The REP10 Experiment.” *Ocean Dynamics* 62, 469-93. doi: 10.1007/s10236-011-0512-6

Jane’s Information Group. 2013a. “Mk 45 Single 5 in L/54 Gun Mount Mod 0-2/Mk 45 Single 5 in L/62 Gun Mount Mod 4.” *Jane’s Electro-Optic Systems,* Jane’s Defense Equipment Library.

Jane’s Information Group. 2013b. *Jane's Unmanned Aerial Vehicles and Targets: Harpy IAI and Cutlass.* Jane's Defense Equipment Library.

Jane’s Information Group. 2014a. “Jane's Unmanned Aerial Vehicles and Targets. HESA Ababil.” Jane's Defense Equipment Library.

Jane's Information Group. 2014b. “Mk 15 Close-in Weapons System, Phalanx.” *Jane's Electro-Optic Systems.* Jane’s Defense Equipment Library.

Jane’s Information Group. 2014c. “Mk 38/STARC 25/Mk 96/Mk88 Mounting/Type 98/Type 75S.” *Jane's Electro-Optic Systems.* Janes Defense Equipment Systems.

Jane’s Information Group. 2014d. “Under the Radar: A New Crop of Unmanned Aircraft.” *Jane's International Defense Review* 47 (2). http://search.proquest.com/docview/1477897571?accountid=12702.

Jane’s Information Group. 2015a. “Arliegh Burke (Flight I and II) Class.” *Jane's Fighting Ships* Jane's Defense Equipment Library: 26 April 2015.

Jane’s Information Group. 2015b. “Jane’s Unmanned Aerial Vehicles and Targets. Boeing-Insitu Scan Eagle.” Jane’s Defense Equipment Library.

Jane’s Information Group. 2015c. “Ticonderoga class. Jane's Fighting Ships.” Jane’s Defense Equipment Library.

Jennings, Gareth. 2014. “Iran Claims to Have Flown Reverse-Engineered U.S. Stealth UAV.” *IHS Jane's Defense Weekly.*

Mack, Patrick V. 2000. “THORN: a Study in Designing a Usable Interface for a Geo-Referenced Discrete Event Simulation.” Master’s thesis, Naval Postgraduate School. http://hdl.handle.net/10945/9410.

Madachy, Raymond. 2014. “Systems Engineering Cost Estimation Workbook” Monterey, CA: Naval Postgraduate School.

Martine, S. 1995. “Effective Visual Communication for Graphical User Interfaces.” Presentations: Worchester Polytechnic Institute. http://web.cs.wpi.edu/~matt/courses/cs563/talks/smartin/int_design.html.

Pham, Loc V; Dickerson, Brandon; Sanders, James; Casserly, Michael; Maldonado, Vicente; Balbuena, Demostenes; Graves, Stephen; and Pandya, Bhavisha. 2012. “UAV Swarm Attack: Protection System Alternatives for Destroyers.” Master’s thesis, Naval Postgraduate School. http://hdl.handle.net/10945/28669.

Piatko, Christine D., Carey Priebe, Lenore Cowen, I-Jeng Wang, and Paul McNamee. 2001. “Path Planning for Mine Countermeasures Command and Control.” *Proceedings SPIE* 4394: 836–43. doi:10.1117/12.445447.

Reagan, Jason. 2014. “DARPA Project Could Lead to Thinking Drones.” *DRONELIFE.com*, November 5. Accessed July 18, 2015. http://dronelife.com/2014/11/05/darpa-project-lead-thinking-drones/.

Ross, Sheldom M. 2013. *Simulation.* Amsterdam: Academic Press. http://libproxy.nps.edu/login?url=http://app.knovel.com/web/toc.v/cid:kpSE000016.

Salem, Anita. 2015 *Introduction to Design Thinking* [cited 15 June 2015]. Available from https://cle.nps.edu/access/content/group/69c0f2b9-3512-43c7-b986-7b396c23df2b/Modules%20Rev/Module_1_CourseIntro/Concept%20Introduction/Introduction%20to%20Design%20Thinking.ppt.

Shneiderman, Ben. 1998. *Designing the User Interface – Strategies for Effective Human-Computer Interactions.* Menlo Park, CA: Addison-Wesley Longman Inc.

Simonite, Tom. 2014. “A Brain-Inspired Chip Takes to the Sky.” *MIT Technology Review,* November 4. Accessed July 18, 2015. http://www.technologyreview.com/news/532176/a-brain-inspired-chip-takes-to-the-sky/.

Turner, Richard. 2007, April. “Toward Agile Systems Engineering Processes.” *Systems and Software Consortium*, 2.

U.S. Energy Information Administration. 2014. *World Oil Transit Chokepoints.* Washington, DC: U.S. Energy Information Administration.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California