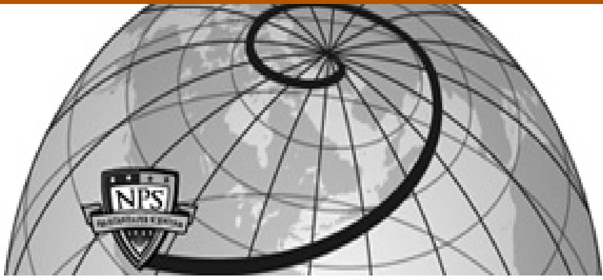


2008-03

Adapting the Dynamic Allocation of Fires and Sensors (DAFS) model for use in maritime combat analysis

Hattaway, Scott B.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/4228

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ADAPTING THE DYNAMIC ALLOCATION OF FIRES AND SENSORS (DAFS) MODEL FOR USE IN MARITIME COMBAT ANALYSIS

by

Scott B. Hattaway

March 2008

Thesis Advisor:	Arnold Buss
Second Reader:	Ronald D. Fricker, Jr.

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2008	**3. REPORT TYPE AND DATES COVERED** Master's Thesis
4. TITLE AND SUBTITLE Adapting the Dynamic Allocation of Fires and Sensors (DAFS) Model for Use in Maritime Combat Analysis		**5. FUNDING NUMBERS**
6. AUTHOR(S) Scott B. Hattaway		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
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.		
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE** **A**

13. ABSTRACT (maximum 200 words)

The U.S. Navy employs several models of maritime combat to provide analytical rigor to force structure and weapon system procurement policies. All of the models currently used are high resolution and deterministic, providing very detailed results but without any measurement of variance or any statistical manner of evaluating risk. This thesis provides the initial groundwork for a low resolution stochastic maritime combat model that may provide an initial evaluation and shape future detailed studies. The framework for the model is a Discrete Event Simulation (DES) Model fed by Extensible Mark-up Language (XML) input and output modules. The simulation loads scenario inputs from XML files forming the baseline values of entities, the rules employed for movement and combat, and the general concept of the scenario. During simulation run, the model makes intermittent calls to an optimization package to allocate weapons based on a multi-dimensional knapsack problem simulating a networked force. Upon completion of the simulation run, the model generates an XML output that can be later read for statistical analysis and data mining. Because of the stochastic nature of the model, it provides an increased level of analytical quality to its results as well as the ability to calculate the risk involved with the force structure and units employed.

14. SUBJECT TERMS Dynamic Allocation of Fires and Sensors, DAFS, stochastic, search and detect, XML, discrete event simulation, Java, Simkit, Maritime combat, Anti-Surface warfare, ASuW, Anti-Air Warfare, AAW, combat simulation, combat model			**15. NUMBER OF PAGES** 143
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

ADAPTING THE DYNAMIC ALLOCATION OF FIRES AND SENSORS (DAFS) MODEL FOR USE IN MARITIME COMBAT ANALYSIS

Scott B. Hattaway
Lieutenant Commander, United States Navy
B. S., United States Naval Academy, 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 2008

Author:	Scott B. Hattaway
Approved by:	Dr. Arnold Buss Thesis Advisor
	Dr. Ronald Fricker, Jr. Second Reader
	Dr. James Eagle Chairman, Department of Operations Analysis

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The U.S. Navy employs several models of maritime combat to provide analytical rigor to force structure and weapon system procurement policies. All of the models currently used are high resolution and deterministic, providing very detailed results but without any measurement of variance or any statistical manner of evaluating risk. This thesis provides the initial groundwork for a low resolution stochastic maritime combat model that may provide an initial evaluation and shape future detailed studies. The framework for the model is a Discrete Event Simulation (DES) Model fed by Extensible Mark-up Language (XML) input and output modules. The simulation loads scenario inputs from XML files forming the baseline values of entities, the rules employed for movement and combat, and the general concept of the scenario. During simulation run, the model makes intermittent calls to an optimization package to allocate weapons based on a multi-dimensional knapsack problem simulating a networked force. Upon completion of the simulation run, the model generates an XML output that can be later read for statistical analysis and data mining. Because of the stochastic nature of the model, it provides an increased level of analytical quality to its results as well as the ability to calculate the risk involved with the force structure and units employed.

THIS PAGE INTENTIONALLY LEFT BLANK

DISCLAIMER

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The U.S. Navy utilizes a large array of combat models to provide analytical rigor to its policy and programming decisions. These models are generally large and complex involving millions of variables and hundreds of hours to run a single iteration. However, many of the decisions that face battle group staffs and Pentagon action officers grant a limited time for analysis. Instead of the months of research and coding required for a single run of these complex simulation models, analysts must use tools that can provide accurate and credible results in a matter of days. In an effort to investigate a tool that might meet this requirement, this thesis utilizes Dr. Arnold Buss' Dynamic Allocation of Fires and Sensors (DAFS) model to analyze a modern naval combat scenario.

DAFS was originally used to analyze the predicted abilities of the Army's Future Combat System (FCS) and its role in net-centric warfare. Specific analysis was conducted to examine the fact that with the data of the various units linked, there were now more targets available to a greater mix of platforms. This proposed the use of optimization to determine the best mix of weapons from the various platforms to cause maximum damage to the enemy. This same optimized combat solution dove-tails with the Navy's net-centric warfare philosophy utilizing near real-time tactical data links (TADIL) and even Cooperative Engagement Capability (CEC) for fire control data. Because of this pre-existing Command and Control architecture, DAFS seemed a logical choice to be adapted for use in the modern naval environment.

This thesis is directed at adapting DAFS for use as a naval combat simulation model by providing three distinct products. First, DAFS required an implementation of a mathematical radar model for detection and engagement of contacts. Currently, DAFS uses a modified ACQUIRE algorithm, as defined by the Army Modeling and Simulation Office (AMSO), to determine detection of contacts via visual means. Since modern naval combat utilizes electronic sensors for prosecution of targets, the ACQUIRE algorithm is an unsatisfactory method of adjudicating detections. For the DAFS maritime version, the ACQUIRE algorithm was replaced with a more traditional electronic detection algorithm based off of detection and radar theory.

The second product of the thesis was the simulation of naval ordnance as applied to modern maritime combat. The vast majority of weapons employed in ground combat utilize flat or low apogee ballistic trajectories. However, many anti-surface, anti-air, and anti-ground weapons utilize more complex flight paths involving way-points and non-ballistic trajectories. The basis for implementing way-pointed missile flight paths is developed in the model as well as a quick analysis of their potential with regards to the radar model. These weapons were simulated in the model, providing greater fidelity to the warfare they modeled and a basis for understanding the effects of these weapons on the final results.

Lastly, after multiple runs of a set scenario, the thesis provides examples of a methodology to interpret the run results. The method identifies key indicators of victory conditions as well as possible vignette points in which experimentation can be made to further understand the factors involved. It provides probabilities of success as well as identifies critical units, weapons, or sensors in the scenario. This methodology also examines distributions of results to provide predictive value from the findings. Key to this methodology are the tools of data mining, statistical analysis, and campaign analysis.

Through this thesis, DAFS has been demonstrated to posses the potential to be an effective and flexible tool for quick analysis of maritime warfare. The component based design inherent in DAFS allowed for the easy implementation of a mathematical radar model. The stochastic nature of the model provided a range of variability for point estimates as well as a rich set of statistical data to base inferences from. Additionaly, the DAFS model can be utilized for rough-cut analysis of a problem to suggest further more detailed analysis by complex models. In this manner, the more time consuming models such as ITEM, NSS, or GCAMs may be better applied to the more intricate aspects of the problem.

The results obtained in this thesis are representative of the potential of DAFS as a combat model. Through the process of writing this thesis and utilizing the model, several improvements were indentified that would enable the model to be better utilized by analysts. The DAFS combat model also possesses the potential to be a tool for operational planners who require rapid proofing of their battle plans. Continued refinement of the DAFS model, in particular with a focus on naval warfare, would present the U.S. Navy with an invaluable tool for rapid analysis of combat related issues.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

During the course of this thesis, many individuals have provided insight, assistance, or motivation to complete the work. I thank each of those people who have enabled me to finish this thesis.

First and foremost, I must thank my wife, Takako, who listened to my mathematical ramblings with deaf ears. Without her reminders about what is important, and the occasional chai tea, cookies, and cheese cake, this thesis would have been much more difficult.

I must also thank Professor Arnold Buss for patiently listening to my rants about structured coding while at the same time guiding my work towards the right answer. The experience of working beside you has been an education in simulation modeling without peer. I only hope that through future effort I may repay the time you spent with me.

I would also like to thank Professor Ronald Fricker, who patiently waded through my prose and polished the rough spots.

I also must thank the NPS Operations Research faculty and the fellow students of my cohort. Though you may not have known it, you tacitly inspired me to work late at night and long hours on the weekend, if nothing more so that I did not feel so far behind.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The U.S. Navy utilizes a vast array of computer generated combat models to provide analytical rigor to long range strategic and policy decisions. These models provide an interface between the war fighters making policy and the mathematical analysts investigating the issues. The models encapsulate millions of calculations representing the interactions between the environment, weapons, units, and a host of other variables into a seamless performance providing an end result for the analysts to use as the basis of their recommendation.

The models the Navy currently uses have been developed over several decades as the Navy and the civilian sector work together to provide a more comprehensive product for the Navy's use. These models have grown in complexity as both the warfare they emulate and the computers that support them have grown more complex and powerful. However, because the models are continuations of their original forms, many of them suffer from design choices that were made two to three decades ago when the computers used for the model were not as powerful or sophisticated enough to run more complex algorithms. These design choices that seemed perfectly natural in the past are now binding restraints on the model.

One of these hindering design choices is discrete time based simulation. Although a natural progression from the way people think and act, discrete time based simulation requires the model to perform actions and calculations in defined blocks of time. The model performs all of its calculations and interactions only at these specific points in time

In very complicated simulations, the calculations for the next block of time may take hundreds or thousands of times longer than the actual time block it simulates. In the case of the Air Forces EADSIM model that modeled the early part of the air campaign in Operation Desert Storm, it required 8 days of real time to model three days of simulated time.[1] These simulations also suffer when there are very few interactions in

[1] Case (1993).

a given time block because the model must still perform all of its routine calculations at each time step. Because of this, campaign models built on a time stepped system tend to have large periods of wasted simulation time.

The second limiting design choice is making a simulation deterministic. Because of the complexity and immense number of calculations required of a stochastic model, most combat models are made deterministic to save development and run time. However, this excludes the model from exploring the range of possible outcomes as well as providing a standard deviation of measurements. These two factors make any extrapolation of risk mere guesswork by a subject matter expert.

A. RESEARCH OBJECTIVES

This thesis is directed at providing a better radar model of detections to the DAFS combat model for use in further naval studies. Because DAFS was originally created as a combined work between NPS and TRAC-Monterey to investigate concepts for Future Combat Systems, its model of radar detections is based on the Army's ACQUIRE algorithm and not a traditional radar model. Although more than adequate for electro-optical and thermal sensors, the ACQUIRE algorithm is a poor representation of detection for radar, ESM, and other long range electronic sensors. Results derived from the ACQUIRE algorithm's model of radar are widely divergent from those of NSS, ITEM, and other certified naval combat models because of this difference.

Secondly, this thesis will incorporate the new radar model with the networked fires architecture of DAFS to provide a scenario that emulates modern naval combat. Because DAFS uses an optimization routine to determine what units to apply weapons to, it in effect mimics a Force Warfare Commander controlling over a networked battle force. This total picture form of combat is used today in the Navy's Link architecture and more particularly in the CEC architecture.

Lastly, this thesis will provide an investigation as to what characteristics in DAFS are most important when examining the results of a combat scenario. Because DAFS is a stochastic combat simulation, it provides the opportunity for multiple runs to compute the variance of a finding. It also provides the opportunity through hundreds or even thousands of runs to better explore the domain of a problem using simulation analysis techniques. Through examining a single complex scenario, this thesis provides a first step towards evaluating further DAFS simulations and provide a reference for further studies.

1. Stochastic Emulation of Radar Probability

Radar detections are normally computed through a series of equations, curve charts, and other complicated means based on the physics of radar propagation. These detections are based almost exclusively on physical factors such as radar cross section, effective receiving aperture, and other design features of the radar or the target itself. Although these factors are required for high resolution physics based models, they are superfluous to a low resolution model. Because DAFS is a low-resolution combat simulation, it does not investigate sensors and detections in physics based manner but relies on mathematical models of the detection itself.

In this thesis, detections will be based on a handful of factors in a mathematical model to maintain its low resolution nature and to ensure calculations are both simple and expedient. All factors are based on the traditional radar equation but only the most necessary factors (max radar range, radar cross section of target, PRF, etc.) will be utilized. This produces a relatively accurate depiction of the radar's effective range compared to the target as well as a quick determination of the probability of detection based on the sensor characteristics.

Lastly, a variation of the glimpse based detection model will be used to determine when detection occurs. More complex versions of this model are commonly used in other combat simulations and its pedigree extends back to World War II. In this thesis, a simplified single probability of detection model will be coupled with the target geometry and the PRF of the sensor to determine a single probability of detection of the target. This simplified detection model remains true to the low-resolution nature of DAFS while maintaining a robust depiction of radar detection.

2. Emulation of Modern Naval Weapons

Modern naval combat is no longer solely about single ships attacking ships – it is a unified and controlled application of force between two or more groups of units. To facilitate this type of engagement, an over arching target selection process must be implemented that examines all sensor and weapons data collected from the units and decides how to distribute weapons among them. DAFS already has a linear programing and integer solver designed to do just that which enables the units to act in a networked manner similar to the US Navy's linked architecture.

Secondly, the majority of weapons employed in naval combat are not linear geometry ballistic weapons but cruise missiles and aircraft. These modern weapons require more complex flight geometries as the transit to their targets. However, because DAFS is a low-resolution model, this thesis will construct simplified versions of their flight paths to better emulate the more complex versions. This provides greater realism to the model but also increases the display time of incoming missiles and aircraft presenting a greater opportunity for them to be destroyed by anti-missile weapons.

3. Methodology to Analyze Scenario Results

Although DAFS is able to provide measurements of any factor in the simulation, some factors are more important than others. DAFS is able to report the results of a scenario to a data file for review as both a summary and as a step-by-step process. This enables an analyst to examine both the playback of the scenario as well as the end results as a source of information in a database. This dual perspective allows for both a macro and a micro view of the results.

In this thesis, an examination of the base scenario will be performed from multiple runs of a single scenario. These results will be examined via data mining and statistical analysis as well as the more time consuming manner of visual playback for those scenarios deemed anomalous or statistically significant.

II. METHODOLOGY

As stated in the introduction, the goal of this thesis is to provide a better radar model to the DAFS suite and to implement it in a maritime combat scenario for evaluation and analysis. This problem can be split into three elements and each tackled separately to provide a comprehensive solution. The first element is to design a radar model grounded in traditional radar theory but simple enough to be conducted quickly by the DAFS suite. The second element is to implement non-ballistic naval weapons into the DAFS suite and combine this with the radar model to provide a rich demonstration of maritime SUW and AAW combat. Lastly, after combining the two elements into a complex scenario, to test the scenario and provide an analysis of the results.

This project approaches these elements with a functional DAFS model that has been developed looking almost exclusively at ground combat and US Army requirements. Several of the functional components in DAFS are co-opted in this project providing for a more reliable development of the model. These elements and their integration are discussed in Chapter III.

A. THE RADAR MODEL

The first element to be developed is a simplified and DAFS friendly version of the radar model. In essence, any model implemented in DAFS is a mathematical representation of the actual system or event that it represents. However, there are two directions from which this model can be formed. The first is to observe the process and formulate an equivalent mathematical representation for the end result, or the "outside-in" approach. This method usually requires little understanding of the process itself but an immense amount of trial and data to develop robust enough models.

The second direction that may be taken is to start from the inner workings of the process and build a mathematically equivalent system that functions in a similar manner. This "inside-out" approach requires a much greater understanding of the process as the modeler is now tasked with developing equivalent systems in his model that produce a final result similar to the actual process. However, this approach requires much less trial and data as the modeler bases his entire model in the roots of the system and not the outcome. In this project, the second approach will be followed to develop the radar model. To maintain the validity of the model, it is critical that it be rooted in the well developed physics of radar theory.

1. The Radar Equation

The radar equation is the fundamental representation of radar's ability to detect a target based on the radar cross section (RCS) of the target and the physical characteristics of the radar itself. Developed from both the physics of radar waves and from empirical observations encapsulated in tables and charts, it provides a robust means of calculating a radars maximum effective range against a given size target. It also forms the basis of the radar model developed in this thesis.

$$R^4{}_{\max} = \frac{P_t G_t A_e \sigma}{(4\pi)^2 S_{\min}} \tag{2.1}$$

R_{max}	Maximum radar range
P_t	Antenna transmission power
G_t	Antenna gain
A_e	Effective aperture area
σ	Target radar cross section
S_{min}	Receiver minimum detectable signal

As can be seen from Equation 2.1, the radar's maximum range is heavily dependent on its physical characteristics. However, to simplify our model, we assume that all variables in the equation remain constant with the exception of the target's radar cross section and the Maximum radar range. In this manner, we are able to calculate an effective radar range for our sensor based on the target's RCS. Making this change, the equation becomes:

$$R^4{}_{\max} = \sigma K \tag{2.2}$$

R_{max}	Maximum radar range
σ	Target radar cross section
K	Grouped constant generated from $P_t\ G_t\ A_e/(4\pi)^2 S_{min}$

Because we assume that the radar's physical characteristics remain constant between detections of different targets, we are able to form an effective radar range relation using Equation 2.2. This new relation allows us to base all effective radar ranges off of a known value listed for the radar and therefore produce accurate effective ranges for the sensor.

$$\frac{R^4{}_{\max}}{\sigma_{ref}} = K = \frac{R^4{}_{eff}}{\sigma_{t\arg et}} \tag{2.3}$$

R_{max}	Maximum radar range
σ_{ref}	Reference radar cross section
K	Grouped constant generated from $P_t\ G_t\ A_e/(4\pi)^2 Smin$
R_{eff}	Maximum effective radar range
σ_{target}	Target radar cross section

For the purposes of this thesis, we assume that the reference radar cross section used to determine the maximum radar range is 25 m^2. This size reference target is quite common for testing of air search radars and thus is an industry standard for determining a production units maximum range. Applying this assumption, the equation for determining a sensors maximum effective range becomes:

$$R_{eff}=R_{max}\left(\frac{\sigma_{target}}{25m^2}\right)^{1/4} \tag{2.4}$$

With this equation we now have a simple manner of determining the maximum detectable distance of any target based off of its RCS and the sensors maximum radar range. In practice, this formula will accurately predict the maximum effective range of any target with an RCS equal to or greater than 0.1 m^2. However, for targets with an RCS less than 0.1 m^2 (so called "stealthy" targets), this relationship no longer holds true.[2] For the purposes of this thesis, a radar relationship equation for stealthy targets will not be investigated.

[2] Knott, Shaefer, and Tulley (2004).

2. The Glimpse Model

The second part of our mathematical model of radar involves determining an actual probability of detection. The most common manner of calculating a probability of detection for any periodic sensor is a glimpse model. A glimpse model assumes that an observer or sensor has a set number of opportunities to detect a target. For each opportunity of detection, an independent Bernoulli trial is conducted to determine if the observer detects the target. As the sensor is given more opportunities to detect the target, the overall probability of detection increases as the Cumulative Distribution Function (CDF) of a geometric random variable.

$$F(k) = 1 - \left(1 - P\right)^{k} \tag{2.5}$$

F(k)	Probability of at least one successful trial given k trials
P	Probability of success for each independent Bernoulli trial
k	Total number of trials conducted

In this manner, a single calculation for the probability of detection by the observer can be generated based on the number of glimpses conducted and the individual probability of detection at each glimpse. However, for the equation to hold true as the CDF of a geometric random variable, each Bernoulli trial must have the same probability of success. It is also noted that for any given positive probability of success, the probability of at least one successful trial is always 1 given an infinite number of glimpses.

Because radar can not radiate and receive at the same time, it effectively provides periodic glimpses over a given search area, and thus a glimpse. These individual glimpses of observation (or dwells) can be calculated based on the periodic rotation frequency (PRF) of the given radar. The PRF provides a measurement of the number of scans it can conduct in a given period of time, thus providing a dwell count. This dwell count represents the number of independent Bernoulli trials conducted by the radar as it attempts to detect a target.

Several factors can cause the probability of success at each individual glimpse to fluctuate. In the case of radar detecting an incoming target, the strength of the return from the target will increase as the target closes the range. This in effect increases the probability of detection by the sensor. The radar cross section of a target also changes depending on the aspect of the target towards the sensor. An aircraft approaching nose on provides only its nose and wing edges to reflect radar waves back to the transmitter while a banking aircraft may provide a sizeable portion of the aircraft body and both wings to reflect radar waves. This too would also change the overall probability of detection at each glimpse. However, to maintain the simplicity of the mathematics behind the model, a single probability of detection of 0.01 at each glimpse will be maintained for all targets. This allows us to continue to use the more fundamental equations of search theory without resorting to differential equations or abstract averages.

Combining the CDF of the geometric random variable with our assumptions and glimpse count, the model now provides a single probability of detection. This model does not require each individual Bernoulli trial to be conducted as in a time-stepped simulation because of the use of the CDF. It also provides a quick manner to determine the probability the target is detected by the sensor based solely on the number of glimpses that it will conduct.

$$P_{\det} = 1 - (1 - 0.01)^{k} \tag{2.6}$$

P_{det}	Overall probability of detection based on the number of dwells
k	Number of dwells conducted by the sensor

It is not enough to know whether a detection of the target occurs but also when. Previously we made the assumption that the probability of detection at each dwell was a uniform 0.01. Extending this assumption, it would be natural then to determine when a target is detected by a random draw from a geometric random variable. However, this idea fails inspection by experienced radar operators who know that the individual glimpses are not memory-less and that the kinematics of the target also plays an important role in the probability of detection. In practice, by using a geometric random variable to determine when the detection occurs, we imply that the sensor is just as likely to make its detection on its farthest dwell as it is its nearest, which as stated before is not reflected in reality. To compensate for this, this model will utilize a gamma random variable shaped so that the greatest likelihood of detection occurs at approximately 1/3 of the total number of dwells.

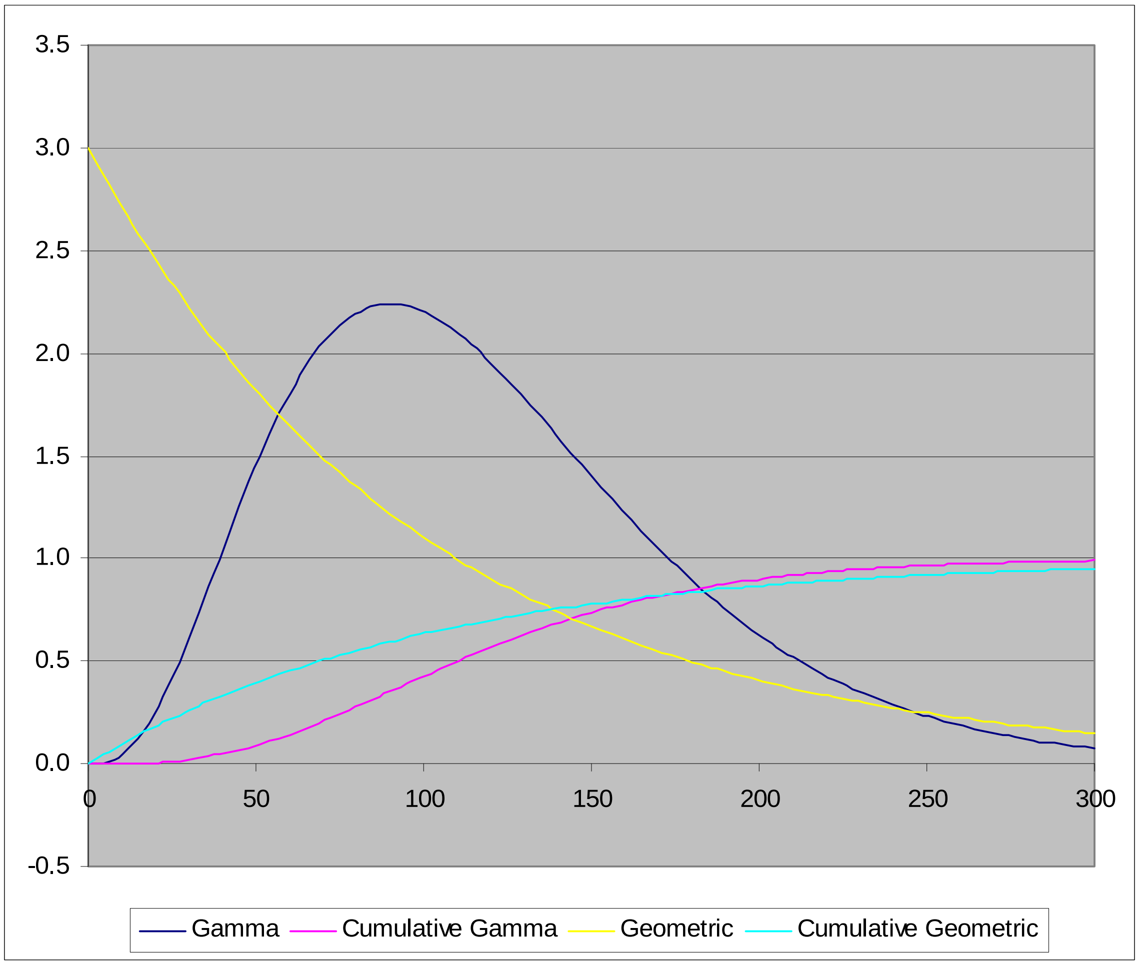


Figure 1. Comparison of Gamma and Geometric Distributions over 300 Dwells

In a side by side comparison, there is some difference in the overall outcome of the two techniques. As an example, the above sensor has 300 dwells to achieve detection. If a Gamma Distribution (α=4, β=0.1) is used, the peak likelihood of detection occurs at 90 dwells. For a Geometric Distribution (p=0.01), there is no peak likelihood because each dwell has the same chance of detection. Looking at both distributions' CDFs over the domain of 300 dwells, it is apparent that the cumulative Gamma Distribution covers the domain better than the cumulative Geometric Distribution (300 dwells reaches a total probability of 0.9897 for the Gamma compared to 0.9510 for the

Geometric). However, the Gamma Distribution only provides an answer for when the detection takes place given that detection occurred, and thus a conditional answer. By utilizing the Geometric Distribution, we may combine both the probability of detection with the time of detection as one single measurement. This will be further explained in the kinematic model.

3. Applying the Radar Equation to a Kinematic Model

Now that we have a manner of determining the radar sensor's maximum effective range and probability of detection, it is best to examine how this can be applied in a kinematic model. To facilitate this, a mathematical model of the process was created in Microsoft Excel to provide quick and accurate feedback of the results. This model combines the mathematics of the radar equation and glimpse model with a given kinematic target presentation to produce critical information in the detection process.

	X	**Y**							
ShipLocation	0.00	0.00	***TargetPath***						
					m		**c**		
	X	**Y**		**y=**	1.00	**x +**	50.00		
TargetInitialLoc	200.00	250.00		**a**		**b**		**c**	
TargetFinalLoc	-250.00	-200.00		1.00	**x +**	-1.00	**y +**	50.00	**= 0**
TargetRCS	15	m^2							
TargetSpeed	700	kts	***CPAPath***						
					m		**c**		
SensorMaxRange	250	nm		**y=**	-1.00	**x +**	0.00		
SensorMaxDetection	220	nm							
SensorPRF	14	spm			**X**	**Y**			
			CPALoc		-25.00	25.00			
InitialDetection	220	nm							
TargetCPA	35.36	nm			**X**	**Y**			
DistInitLocToCPA	318.20	nm	**InitDetLoc**		129.38	179.38			
DistInitDetToCPA	218.33	nm							
TimeToInitDet	8.56	min							
TimeToCPA	27.27	min							
TimeInitDetToCPA	18.71	min							
GlimpsesToCPA	262	dwells							
ProbDet	0.9281								

Table 1. Kinematic Model Inputs

The initial information required for the Excel model is the sensor's position (normally the origin), the maximum detection distance, and the PRF of the radar. For the target, the model requires a starting position, an ending position, speed in knots, and the target's RCS. Utilizing simple geometric relationships, the model computes the track of the target and computes an equation of motion in both slope-intercept and general equation of a line forms.

$$Y = \left(\frac{y_{fin} - y_{init}}{x_{fin} - x_{init}} \right) X + \left[y_{init} - \left(x_{init} * \frac{y_{fin} - y_{init}}{x_{fin} - x_{init}} \right) \right] \tag{2.7}$$

$$\left(\frac{y_{fin} - y_{init}}{x_{fin} - x_{init}} \right) X - Y + \left[y_{init} - \left(x_{init} * \frac{y_{fin} - y_{init}}{x_{fin} - x_{init}} \right) \right] = 0 \tag{2.8}$$

X_{init}	Target's initial x-coordinate position
Y_{init}	Target's initial y-coordinate position
X_{fin}	Target's final x-coordinate position
Y_{fin}	Target's final y-coordinate position

Utilizing the equation of motion for the target, we are able to generate a closest point of approach (CPA). This CPA provides the cornerstone for the remaining calculations as we assume that no detection may occur after this point as the radar has had its best opportunity for detection up to this point. The slope-intercept equation for the CPA line is generated by Equation 2.9.

$$y = -\left(\frac{x_{fin} - x_{init}}{y_{fin} - y_{init}} \right) X + y_{ship} - \left[x_{ship} * -\left(\frac{x_{fin} - x_{init}}{y_{fin} - y_{init}} \right) \right] \tag{2.9}$$

X_{init}	Target's initial x-coordinate position
Y_{init}	Target's initial y-coordinate position
X_{fin}	Target's final x-coordinate position
Y_{fin}	Target's final y-coordinate position
X_{ship}	Sensor's x-coordinate position
Y_{ship}	Sensor's y-coordinate position

Combining Equation 2.7 and 2.9 and solving for a single X-Y coordinate, the kinematic model determines the CPA point utilizing Equations 2.10 and 2.11.

$$X_{cpa} = \frac{y_{ship} - \left[x_{ship} * -\left(\frac{x_{fin} - x_{init}}{y_{fin} - y_{init}}\right)\right] - \left[y_{init} - \left(x_{init} * \frac{y_{fin} - y_{init}}{x_{fin} - x_{init}}\right)\right]}{\left(\frac{y_{fin} - y_{init}}{x_{fin} - x_{init}}\right) + \left(\frac{x_{fin} - x_{init}}{y_{fin} - y_{init}}\right)} \tag{2.10}$$

$$Y_{cpa} = -\left(\frac{x_{fin} - x_{init}}{y_{fin} - y_{init}}\right) * X_{cpa} + y_{ship} - \left[x_{ship} * -\left(\frac{x_{fin} - x_{init}}{y_{fin} - y_{init}}\right)\right] \tag{2.11}$$

X_{init}	Target's initial x-coordinate position
Y_{init}	Target's initial y-coordinate position
X_{fin}	Target's final x-coordinate position
Y_{fin}	Target's final y-coordinate position
X_{ship}	Sensor's x-coordinate position
Y_{ship}	Sensor's y-coordinate position

At this stage in the calculations, the model is now able to determine the maximum effective sensor range and the point in the target's path that this occurs at. Utilizing Equation 2.4 and the data inputted the model computes the effective range of the radar versus the target's RCS. The model then computes the CPA distance using Equation 2.12.

$$D_{cpa} = \left| \frac{\left(\frac{y_{fin} - y_{init}}{x_{fin} - x_{init}}\right) * X_{ship} - Y_{ship}}{\sqrt{\left(\frac{y_{fin} - y_{init}}{x_{fin} - x_{init}}\right)^2 + 1}} \right| \tag{2.12}$$

X_{init}	Target's initial x-coordinate position
Y_{init}	Target's initial y-coordinate position
X_{fin}	Target's final x-coordinate position
Y_{fin}	Target's final y-coordinate position
X_{ship}	Sensor's x-coordinate position
Y_{ship}	Sensor's y-coordinate position

At this point, the model now has information for various legs of two related triangles. These triangles allow for the computation of distances and times along the equation of motion for the target itself using the Pythagorean Theorem. Utilizing these relationships, the model computes $D_{cpa,det}$ and $D_{det,init.}$

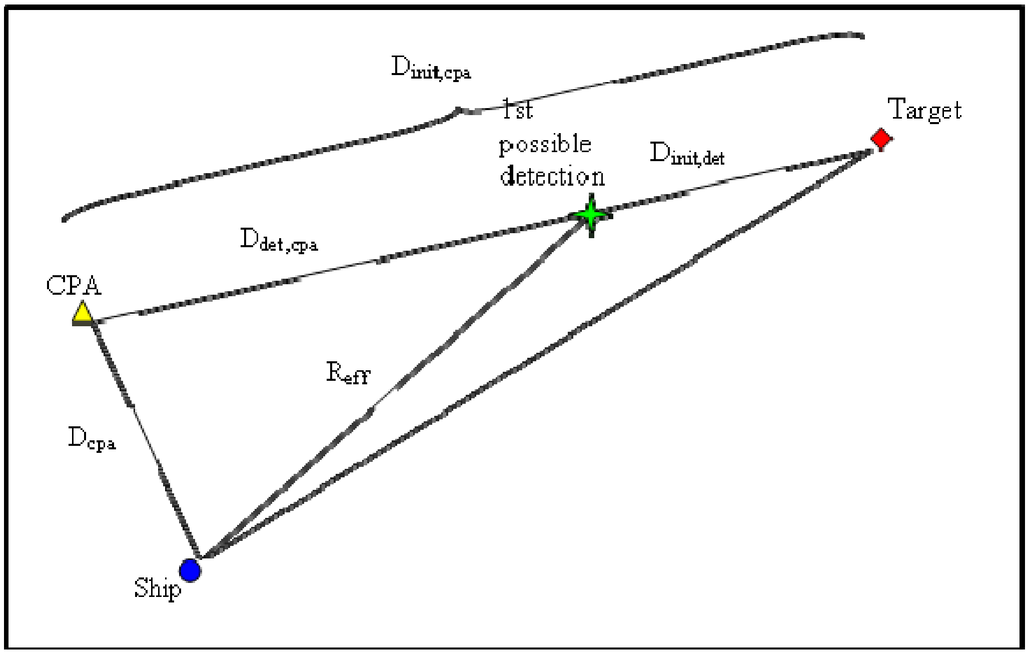


Figure 2. Related Triangles to Determine Distances

$$D_{init,cpa} = \sqrt{\left(X_{cpa} - X_{init}\right)^2 + \left(Y_{cpa} - Y_{init}\right)^2} \tag{2.13}$$

$$T_{cpa} = \left(\frac{D_{init,cpa}}{S_{tgt}}\right) * 60\,\text{min} \tag{2.14}$$

$$D_{\text{det},cpa} = T_{cpa} * \tan\left(\arccos\frac{T_{cpa}}{\text{R}_{eff}}\right) \tag{2.15}$$

$$T_{\text{det}} = \left(\frac{D_{init,cpa} - D_{\text{det},cpa}}{S_{tgt}}\right) * 60\,\text{min} \tag{2.16}$$

X_{init}	Target's initial x-coordinate position
Y_{init}	Target's initial y-coordinate position
X_{cpa}	Target's CPA x-coordinate position
Y_{cpa}	Target's CPA y-coordinate position
S_{tgt}	Target's speed in knots
R_{eff}	Effective range of the sensor against RCS of target

Utilizing similar triangles linking the origin, the CPA point, and both the maximum radar range and maximum effective radar range, the location of the first possible detection is determined using Equations 2.17 and 2.18.

$$X_{\det} = X_{cpa} + \left(X_{init} - X_{cpa} \right) * \frac{D_{\det,cpa}}{D_{init,cpa}} \tag{2.17}$$

$$Y_{\det} = Y_{cpa} + \left(Y_{init} - Y_{cpa} \right) * \frac{D_{\det,cpa}}{D_{init,cpa}} \tag{2.18}$$

X_{init}	Target's initial x-coordinate position
Y_{init}	Target's initial y-coordinate position
X_{cpa}	Target's CPA x-coordinate position
Y_{cpa}	Target's CPA y-coordinate position
$D_{det,cpa}$	Distance from first possible detection to CPA
$D_{init,cpa}$	Distance from initial location to CPA

Plotting the distances and points generated from the sample data, the model provides a display of the positions and measurements calculated on an X-Y graph. By altering the characteristics of the target or the radar, various kinematic solutions and initial detection point possibilities can be calculated.

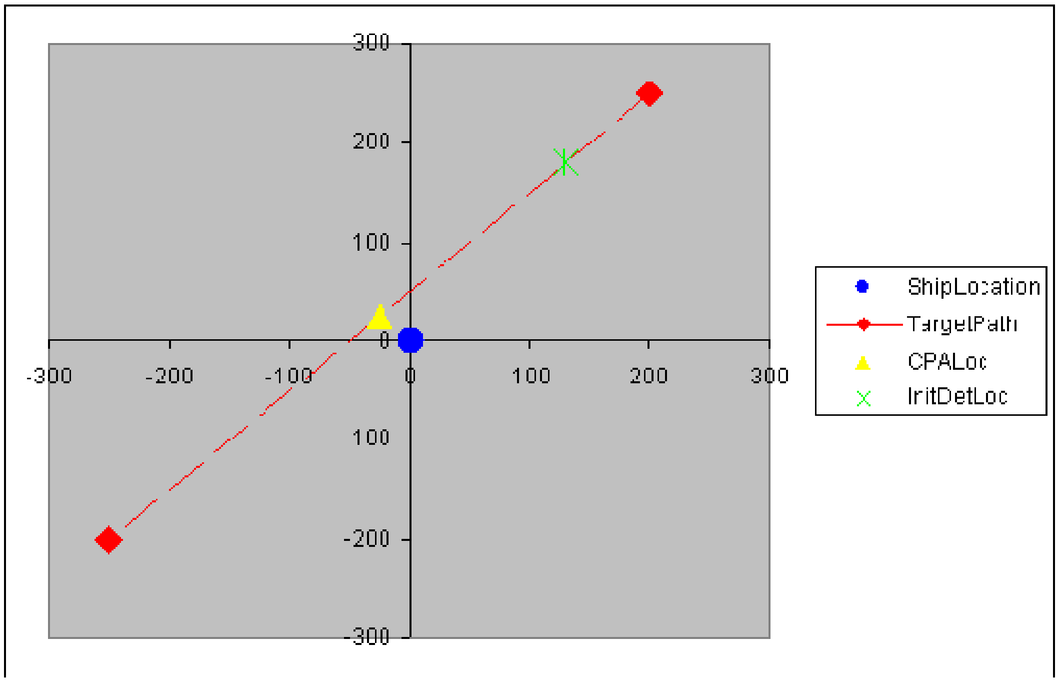


Figure 3. Graph of Kinematic Solution Calculated in Excel Model

At this point, the final calculations required deal solely with calculation of the probability of detection. Because the model has determined the distance from the initial possible detection point to the CPA, we can determine the amount of time that the sensor is permitted to detect the target based on the target's speed. This calculation can then be multiplied by the PRF of the radar and a floor function applied to determine the exact number of dwells conducted by the sensor.

$$T_{\det,cpa} = \frac{D_{\det,cpa}}{S_{tgt}} * 60\,\mathrm{min} \tag{2.19}$$

$$Dwells = \left\lfloor T_{\det,cpa} * PRF \right\rfloor \tag{2.20}$$

$D_{det,cpa}$	Distance from first possible detection to CPA
S_{tgt}	Target's speed in knots

Having the exact number of dwells conducted by the sensor, the model then calculates a probability of detection for the given kinematic scenario based on a glimpse probability of 0.01. In the example problem, the probability of detection is calculated to be 0.9281, providing a better than 90% chance that the target is detected before reaching CPA. By altering a single characteristic of the target, in this case RCS, it is possible to see the full spectrum of changes brought by the model. As the RCS decreases, so too does the maximum effective range of the radar providing fewer glimpse opportunities for detection. The effect is most apparent with the low RCS 0.1 m^2 target that has less than a 50% of detection before reaching CPA. This emulates small RCS sea skimming cruise missiles which may not be detected by the ship at all, or when detected, provide little opportunity for defensive actions. In the case of this same target, the radar has less than 5 minutes to detect the target before it reaches CPA.

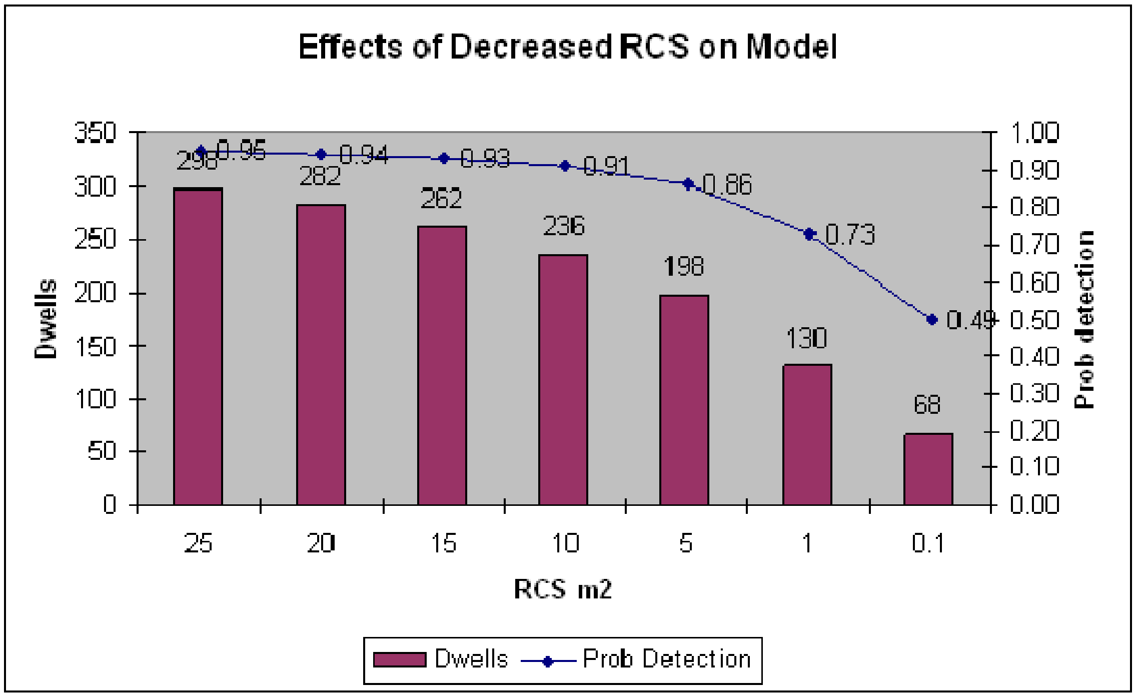


Figure 4. Effects of Decreased RCS on Model

4. Applying the Glimpse Model to Determine Time of Detection

Now that the model has determined the probability of detection, there are two methods for determining the time the target is detected. The first method conditions the time of detection on the fact that a detection exists. Knowing the probability of detection for the given kinematic problem, a draw is made from a standard Uniform Distribution to conduct a Bernoulli trial for detection. If the number drawn is less than or equal to the probability of detection, the sensor successfully detects the target. If the draw is greater than the probability of detection, then the trial fails and the target is never detected. Based on the condition that a successful detection was made, a second draw is conducted, this time from a tailored Gamma Distribution that produces a number between 0 and 1 that is multiplied by $T_{det,cpa}$ to determine when the detection occurs. As was stated earlier, the strength of this technique is that is allows the generated results to be tailored to empirical data or the observations of subject matter experts. The weakness in this method is that it relies on several random draws as well as deviates from the premise of the glimpse model that every glimpse is identical and memoryless.

A second method is to use the Geometric Distribution entirely to determine when and if detection occurs. In this method, a single draw from the Geometric Distribution with an argument of the glimpse probability of detection (in this case 0.01) produces the number of dwells required for detection to occur. In the event that the number drawn is less than the dwell count generated by the model, then the detection determination occurs in much the same manner as with the Gamma Distribution. If the draw is greater than the dwell count, then the detection occurs after CPA, and possible after the target has collided or over flown the sensor. It is also possible for the number drawn to be greater than twice the dwell count, and thus occurring outside the maximum effective range of the radar and in this case the detection must be discounted. The advantage of this method is that it does not require multiple draws from different distributions and it combines the probability of detection with the time of detection measurements. As was seen before, the CDFs of both the functions are similar enough that either may be used with little difference in their outcomes. However, for this thesis, the Gamma distribution will be used to determine when detection occurs.

B. EMULATING NAVAL WEAPONS

The second element to be developed for this thesis is an emulation of modern naval weapons. For surface and air warfare, all naval weapons can be divided in to two categories determined by their primary means of delivery. Gunnery or ballistic weapons have existed since the age of sail and continue in use to this day. They have evolved from the single shot weapons of the Napoleonic era to the rapid firing chain guns and large caliber weapons seen on modern naval warships. Despite this evolution in design, the original concept of launching an explosive shell down a straight path has changed little. The mathematics to model such a weapon also remains simple with the munitions' path being characterized by a straight path from a top down view or a ballistic parabola from a side view. Because these weapons remain simple to model, we will concentrate on the second form of naval weapon, the missile.

Missiles now form the backbone of most modern naval arsenals. They are launched from surface ships, aircraft, and even submarines. With the introduction of the SS-N-3 Shaddock and SS-N-2 Styx missiles by the U.S.S.R. in 1959 and 1960 respectively, a whole new era in naval combat was begun. These weapons pushed the lethal range of a ship past the horizon where ballistic weapons formed the lethal punch of a ship. Utilizing radar as the primary sensor, it allowed naval vessels to engage each other without ever being within visual range of each other. Despite this increase in range and revolution in naval combat, the fact remains that a missile is a munition following a path to its target. The major difference between a missile's flight path and a ballistic shell's flight path is that the missile is no longer required to fly straight to its target.

1. Straight Missile Flight Path

The simplest missile flight path to model is the straight missile flight path. This type of flight path is common among early anti-ship cruise missiles and all ship launched air defense missiles. The missile simply launches and then follows the shortest distance path to its intended target. To derive a mathematical model of this flight path, the thesis incorporated a second kinematic model created in Microsoft Excel. This model assumed that the launching platform was positioned at the origin of an X-Y grid measured in nautical miles. The target was provided an initial location and a current location (based on the time of weapon launch) to derive its course. Target speed was also given as an input. Lastly, a maximum range and missile speed were provided to complete the model. Utilizing this information, the model was able to compute the impact position of the missile and the distance required to make it there.

	X	**Y**	
ShipLocation	0.00	0.00	
	X	**Y**	
TargetInitialLoc	10.00	50.00	
TargetCurrentLoc	40.00	15.00	
TargetPredictLoc	45.21	8.93	
TargetSpeed	50		kts
MissileMaxRange	80		nm
MissileSpeed	500		kts
MissileFltPath0	**X**	**Y**	
-Initial	0.00	0.00	
-Terminal	45.21	8.93	
-Distance	46.08		nm

TargetPath				
		m		**c**
	y=	-1.17	**x +**	61.67
			theta=	-0.8622
MaxTargetTime			9.60	min
MaxTargetDist			8.00	nm
			X	**Y**
PredictedChange			5.21	-6.07
MissileStraightPath				
		m		**c**
	y=	0.20	**x +**	0.00
	r=	46.08	**theta=**	0.1949

Table 2. Missile Flight Path Kinematic Model Inputs

To determine the necessary information, the model first determines the target's path by transforming it into the equation of a line (Equation 2.7). Then it computes the angle theta in radians that describes the angular direction of the target.

$$\theta = \arctan\left(\frac{X_{fin} - X_{init}}{Y_{fin} - Y_{init}}\right) \tag{2.21}$$

X_{init}	Target's initial x-coordinate position
Y_{init}	Target's initial y-coordinate position
X_{fin}	Target's current x-coordinate position
Y_{fin}	Target's current y-coordinate position

As a check-sum to the process, the model also determines the maximum flight time of the missile as well as the maximum distance the target can travel in that given time.

$$T_{\max flight} = \frac{R_{\max}}{S_{missile}} 60 \min \tag{2.22}$$

$$D_{\max tgt} = \frac{S_{tgt} T_{\max flight}}{60 \min} \tag{2.23}$$

R_{max}	Maximum range of missile
$S_{missile}$	Speed of missile
S_{tgt}	Speed of target

Using the target's equation of motion and the maximum flight time of the missile, the model predicts the maximum change in position of the target. Adding this change to the known current location of the target establishes its predicted location at missile impact. This calculation is made with the implied assumption that the target will traverse its maximum distance given the flight time of the missile. This assumption is made to simply the later mathematics and because the Excel model is used to provide insight into the missile flight paths and not a granular physics based model. Suffice it to say, the seeker on this theoretical missile would be able to acquire the target within a specified range and therefore alleviate the requirement for a more exact impact position.

$$\Delta X_{tgt} = D_{\max tgt} \cos\theta \tag{2.24}$$

$$\Delta Y_{tgt} = D_{\max tgt} \sin\theta \tag{2.25}$$

D_{maxtgt}	Maximum distance traveled by target
θ	Angular direction of target

Connecting the launch platform's position to the target's predicted location, the model produces an equation of motion describing the straight line path the missile would make to intercept the target. The model also computes the straight line distance between the launch platform and the target to provide the flight distance of the missile.

$$Y = \left(\frac{Y_{pred} - Y_{ship}}{X_{pred} - X_{ship}}\right) X + Y_{ship} - X_{ship}\left(\frac{Y_{pred} - Y_{ship}}{X_{pred} - X_{ship}}\right) \tag{2.26}$$

$$D_{missile} = \sqrt{\left(X_{pred} - X_{ship}\right)^2 + \left(Y_{pred} - Y_{ship}\right)^2} \tag{2.27}$$

X_{pred}	Target's predicted x-coordinate position
Y_{pred}	Target's predicted y-coordinate position
X_{ship}	Launching ship's x-coordinate position
Y_{ship}	Launching ship's y-coordinate position

This simplified approach provides a reasonably accurate approximation of the missile's straight flight path for flight times of less than 30 minutes and for slow moving targets (<100 knots). For faster moving targets or longer missile flight times, the assumptions made for the target's position at impact weaken; however, this model is adequate enough to describe the motion of straight line missile shots for our purposes.

2. Missile Flight Path with Waypoints

The straight line missile flight path generated in the above model describes the motion of many missiles in the world's naval inventory. However, many of the Anti-Ship Cruise Missiles (ASCM) employed on modern naval warships travel to their targets through a series of waypoints. These waypoints allow the missile to attack its target from a bearing completely different than the one the launching ship may lie on. This tactic is also useful to avoid a ship's counter attack as the natural reaction of the target ship is to launch its own weapons down the bearing of the attacking missile. In either case, a model of the flight path of a way pointed missile has to be developed to apply these weapons to the DAFS model.

In looking at this problem, the previous Excel model was expanded upon. The mathematics used to determine the predicted intercept point of the missile was vital to determining the intercept point of a way pointed missile. However, the main data required from this model is not the equation of motion of the straight shot missile but the bearing and range of the intercept point. The way pointed missile is challenged with traveling through a series of points to eventually arrive at this same intercept point. The simplest way to build these waypoints is to determine a simple relationship to the intercept point and build the waypoints from that.

To do this, a geometric shape was imposed on the straight line flight path and the necessary waypoints determined from that shape. In missile flight path 1a, the missile travels the path completing an isosceles triangle over the straight line path. In missile flight path 1b, the missile again travels a triangular path, this time completing a scalene right triangle. Both of these simple flight paths involved one waypoint and could be simply calculated using known geometric relationships.

MissileFltPath0	**X**	**Y**
-Initial	0.00	0.00
-Terminal	45.21	8.93
-Distance	46.08nm	

MissileFltPath1a	**X**	**Y**
-Initial	0.00	0.00
-wp1	20.03	17.51
-Terminal	45.21	8.93
-Distance	53.21nm	

MissileFltPath1b	**X**	**Y**
-Initial	0.00	0.00
-wp1	31.67	18.00
-Terminal	45.21	8.93
-Distance	52.72nm	

Table 3 Single Waypoint Flight Coordinates and Distances

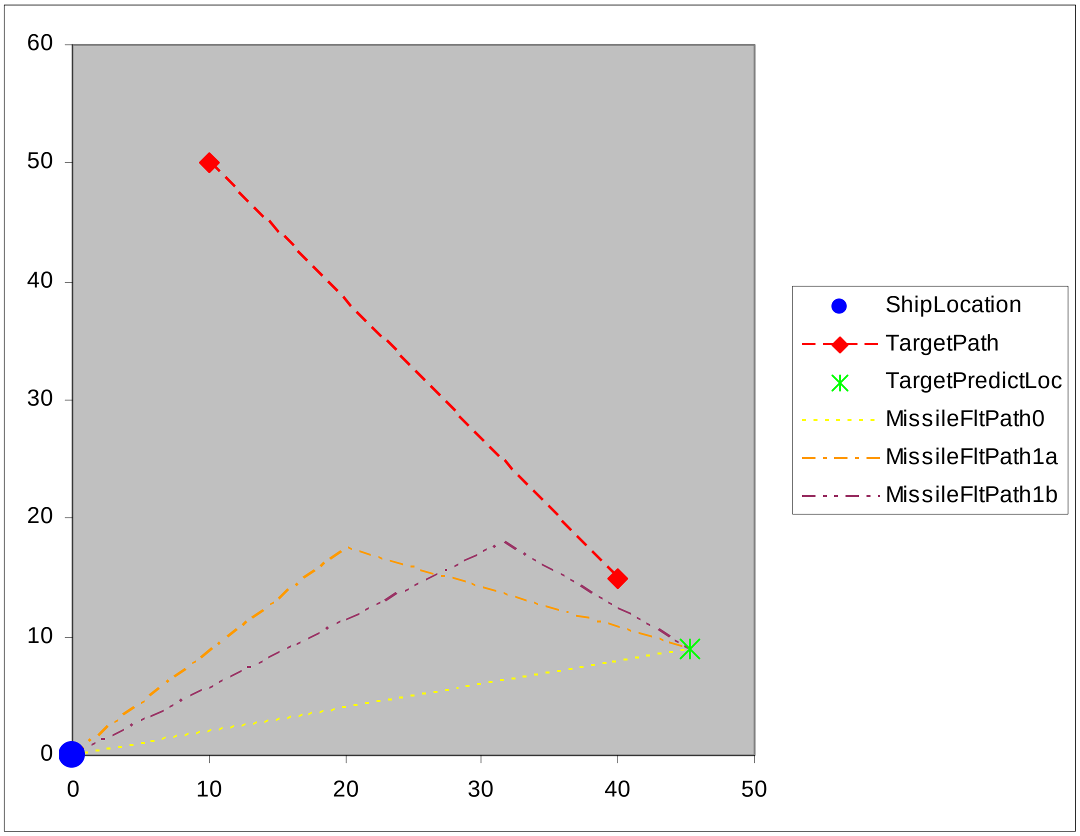


Figure 5. Single Waypoint Missile Flight Paths

Simple geometry is able to produce reasonable missile flight paths and waypoints for a single waypoint missile, but for two to three waypoints, a different approach was required. To produce reasonable waypoints for these types of flight paths, a few more assumptions were made to simplify the math. First, that the missile's first waypoint was always the same, in this case 30 degrees off axis from the straight line path and 0.75 of the straight line distance. This point corresponded to the exact same location as the waypoint for missile flight path 1b. The second assumption made was that all legs after that waypoint were of equal distance. These two assumptions allow the model to use chord mathematics to complete the distance from waypoint 1 to the intercept point.

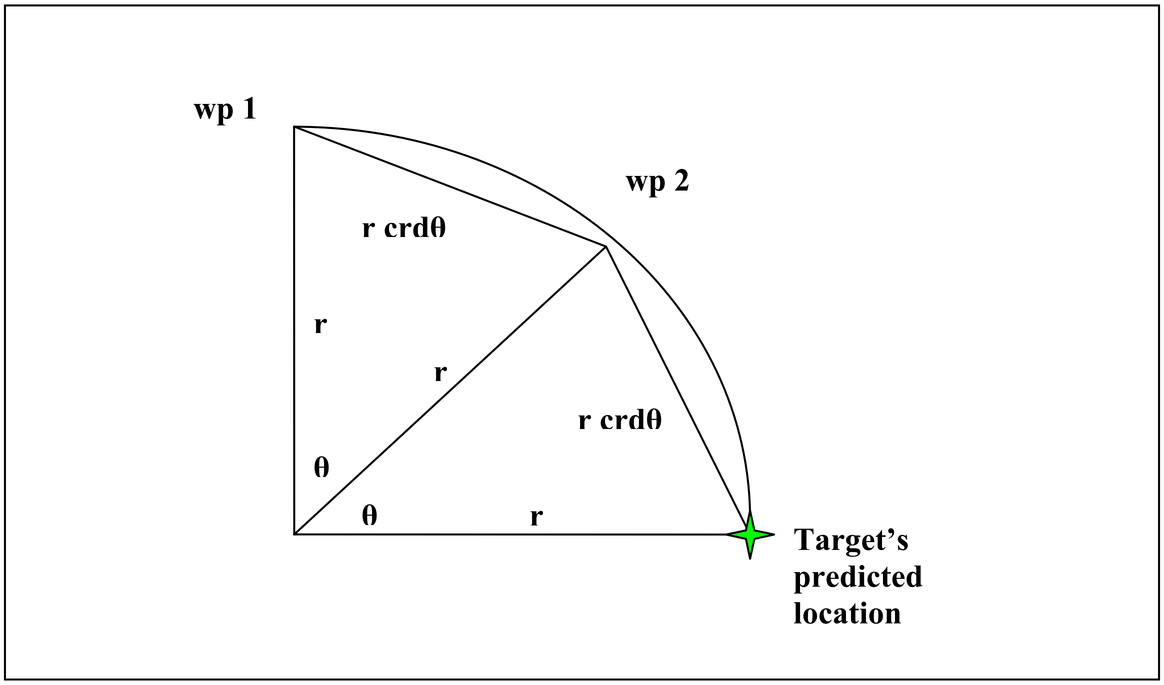


Figure 6. Chord Geometry to Determine Waypoints

Utilizing these assumptions, the model determines theta by dividing 90 degrees by the remaining number of waypoints and calculates the distance required between each waypoint. All of this is done using polar coordinates and the chord function of the derived angle.

$$crd\theta = \sqrt{(1-\cos\theta)^2 + \sin^2\theta} = \sqrt{2-2\cos\theta} = 2\sqrt{\frac{1-\cos\theta}{2}} = 2\sin\frac{\theta}{2} \qquad (2.28)$$

Table 5 shows the waypoints generated using this technique along with the required polar coordinates describing the change of the missile's flight path at each waypoint. It also demonstrates that this same technique may be used to determine a missile flight path to the left or right of the straight line path.

MissileFltPath1a	**X**	**Y**					
-Initial	0.00	0.00		**X**	**Y**	**r**	**theta**
-wp1	20.03	17.51	**wp1aChange**	20.03	17.51	26.60	0.7185
-Terminal	45.21	8.93					
-Distance	53.21nm						

MissileFltPath1b	**X**	**Y**					
-Initial	0.00	0.00		**X**	**Y**	**r**	**theta**
-wp1	31.67	18.00	**wp1bChange**	31.67	18.00	36.43	0.5167
-Terminal	45.21	8.93					
-Distance	52.72nm						

MissileFltPath2	**X**	**Y**					
-Initial	0.00	0.00		**X**	**Y**	**r**	**theta**
-wp1	31.67	18.00	**wp2aChange**	31.67	18.00	36.43	0.5167
-wp2	40.32	16.26	**wp2bChange**	8.65	-1.73	8.82	-0.1978
-Terminal	45.21	8.93					
-Distance	54.06nm						

MissileFltPath3	**X**	**Y**					
-Initial	0.00	0.00		**X**	**Y**	**r**	**theta**
-wp1	36.14	-4.61	**wp3aChange**	36.14	-4.61	36.43	-0.1268
-wp2	41.49	-1.98	**wp3bChange**	5.35	2.63	5.96	0.4567
-wp3	44.51	3.17	**wp3cChange**	3.02	5.14	5.96	1.0403
-Terminal	45.21	8.93					
-Distance	54.32nm						

Table 4. Multiple Waypoint Flight Coordinates and Distance

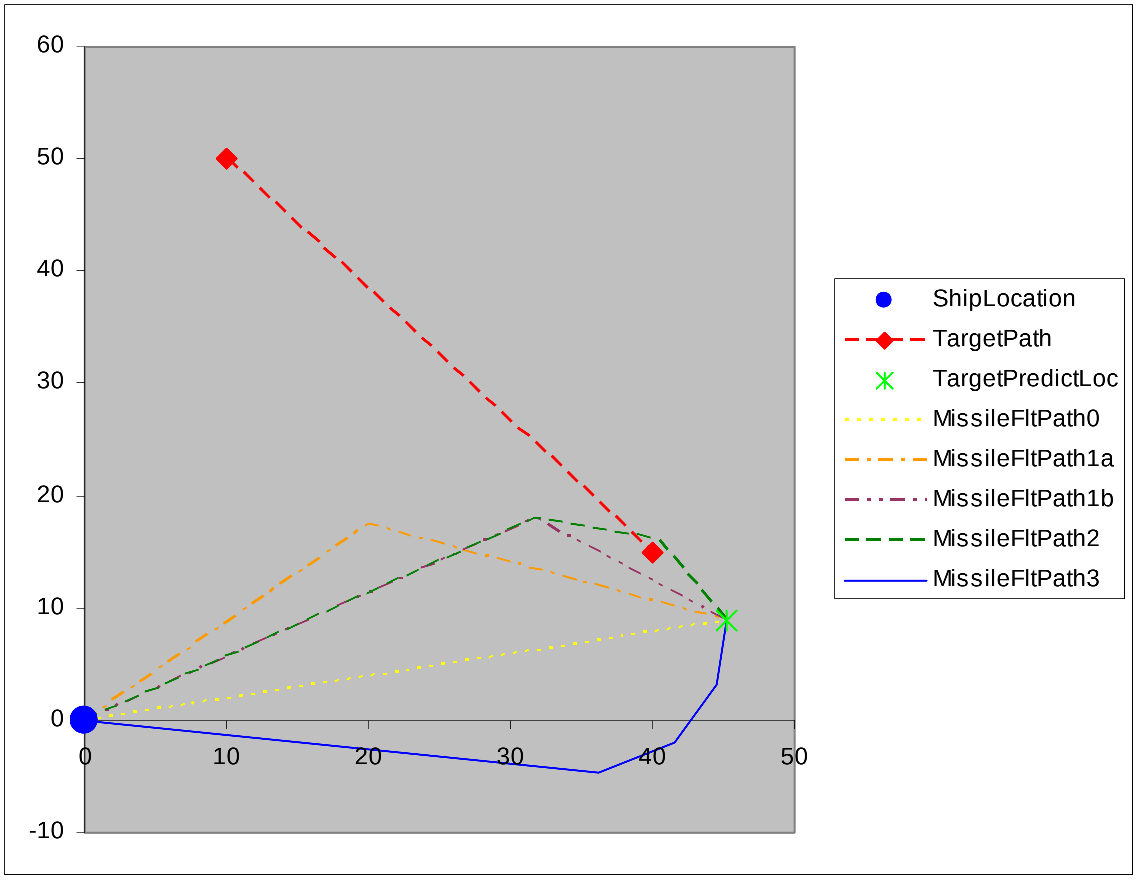


Figure 7. Multiple Waypoint Missile Flight Paths

3. Waypoints Do Not Provide More Insight

As was seen from the mathematics above, adding waypoints to a missile's flight path requires an additional overhead of calculations. Examining the end result of waypoints (additional distance traveled by the missile and therefore increased opportunity for detection), it was determined that waypoints provide little value added to our model. Using the example above, a missile fired at an intercept point of 46.08 nautical miles only increased its travel distance to 54.32 nautical miles by adding three additional waypoints. This additional distance of 8.24 nautical miles provides slightly less than one minute to the overall flight time of the missile. For a high PRF radar, this may provide upwards to 60 additional glimpse opportunities, but not enough to make a significant change in the overall probability of detection. In the example above, if the radar is able to detect the missile from launch to impact, it would have 392 dwells of opportunity for the flight path with waypoints and 332 dwells for the straight shot. The probability of detection is 98.05% for the way pointed missile and 96.44% for the straight shot, a difference of only 1.61%. This difference in detection probabilities is even less for lower PRF radars.

It can also be observed that for most low RCS missiles, the initial opportunity for detection happens at a much closer range. The driving factor for overall detection of such a weapon is no longer the flight time of the missile but the RCS which effectively mask the missile from any glimpses until it has reached the maximum effective range of the sensor. For these reasons, waypoints were not be incorporated into the overall DAFS model.

THIS PAGE INTENTIONALLY LEFT BLANK

III. IMPLEMENTATION

The Dynamic Allocation of Fires and Sensors (DAFS) simulation model provides a partially developed combat model to add the newly developed radar model. DAFS combines the kinematic mathematics inherent in Dr. Arnold Buss' Simkit with an XML library and Graphic User Interface (GUI) to drive the combat simulations. It incorporates a Constrained Value Optimizer (CVO) to provide a method of decision making for the individual units in the model thus allowing the user to pre-assign the desired logic and allow the simulation to run its course. This methodology improves on the conventional combat simulations by applying the best characteristics of both a decision engine (CVO) with event-based and stochastic simulation (Simkit).

The following sections provide a detailed description of DAFS and its individual components to provide the reader with a better understanding of the object-oriented model, how the individual units interact with one another, and the basic implementation steps required to add the radar model. The information contained in the sections describing DAFS components and their functions is liberally borrowed from LT Michael Havens' thesis Dynamic Allocation of Fires and Sensors.

A. MAJOR COMPONENTS

One of the most significant aspects of the DAFS model is its component-based architecture. Within the simulation model, there are two types of components that work together to provide DAFS its overall capability. Some of the elements represent physical items such as sensors, movers, and munitions; others represent functionality like the sensor's detection of a contact or the command to engage a target. Both of these types of elements are equally important and provide the benefit of a "plug-and-play" architecture. The individual elements may be switched out with different versions without altering any other portion of the model. The physical elements represented in DAFS are platforms, sensors, weapons and munitions. The functional elements are the command element, mover managers, kill probabilities, and the CVO. The physical elements will be discussed first followed by the functional elements and interaction fundamentals.

1. Physical Components

The physical components of DAFS are designed to represent actual physical components found in actual combat. Keeping the component interchangeability concept to the forefront, these components are designed to represent the most basic building blocks of the items they represent. They are also designed to allow seamless replacement without affecting any other component in the simulation. The platform forms the basic component for a unit to which zero or more components may be attached to further flesh out the simulated unit.

a. Platforms

Within the simulation, platforms represent the foundation structure of any combat unit involved in the scenario such as ships, aircraft, or even unmanned aerial vehicles (UAVs). Platforms may also represent non-mobile entities like radar stations and missile batteries, but the platform element is still used as the primary reference point for all other physical elements. Platforms are only responsible for knowing their current position and velocity and reporting the same to requesting sources. Additionally, when either the magnitude or the direction of a platform's velocity vector is changed in any way, a property change is fired that may be received by other entities listening for the change. The use of listeners is key to this particular style of modeling and occurs frequently throughout DAFS[3]. The listener feature refers to the fact that an element may be registered to listen for specific actions that occur within the simulation. These actions may be property changes or simulation events and may be triggered by other entities or by the simulation routine itself. These actions then may elicit a response on the part of the registered listener. Property change sources and listeners are resident in JAVA and the simulation event counterparts are in Simkit.

As the foundation structure for all physical entities in the simulation, the platform may have associated with it any number of the sensors, weapons and communications elements described below. One may look at this as a direct analogy to constructing an actual combat unit where the body is first manufactured and then all of the

[3] Buss (2000), Buss & Sanchez (2002).

weapons systems are installed. From an operational point of view, this means that wherever the foundation, or body, of the unit goes, so does the attached system. Therefore, the only entity that really needs to know its location is the foundation, or the platform. In the case of DAFS, once the platform entity is created, the associated sensors, weapons, munitions and communications are given a reference to the platform as they are created. This is conceptually depicted in Figure 7.

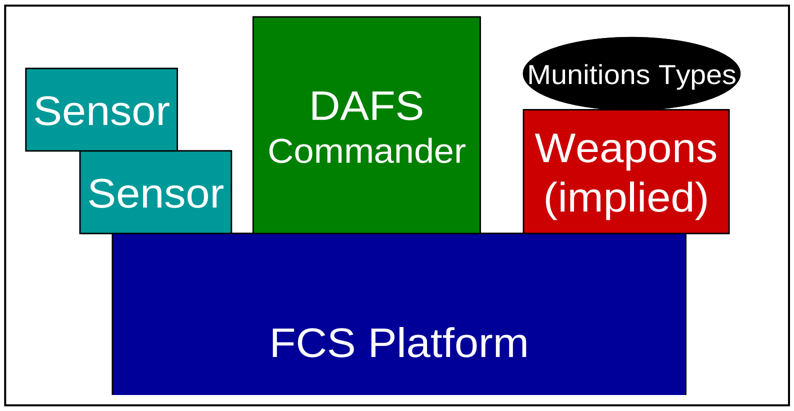


Figure 8. Entity Structure Example[4]

b. *Sensors*

Like the platform, the sensor component is very limited in its functionality. The sensor maintains only its basic capabilities in the form of its type, max range and footprint. Additionally, it maintains a container for its detections. Once created, a sensor object is given a reference to its associated platform in order to locate itself. The capability of a sensor to process detections is accomplished through the use of the functional objects called mediators and referees along with the listening process described earlier. The concept of referees and mediators, or adjudicators, is repeated between munitions and targets and a description of how all these items interact is contained in the primary interactions section below.

[4] Ahner (2005).

c. Weapons

From a functional point of view, the only task that a weapon accomplishes is launching munitions. Thus, the weapon itself does not play a critical role in the basic analysis of the effectiveness of the force. It is the munitions that interact with targets and therefore represent the real objects of focus when it comes to combat adjudication. The weapon object has been developed though to allow more analysis of unit configurations. Specifically, because a weapon object defines a platforms ability to deliver particular munitions types, the collection of weapons configured into a platform largely defines its potential employment.

d. Munitions

Munitions objects draw on the same process used to make the sensors function. Like the sensor, the munitions object only keeps track of its type and footprint. The adjudication of a weapon-target interaction is handled by the referee/adjudicator combination described below. At runtime, only the inventories of munitions by type are established on each platform. During the running of a simulation, if a munitions object is needed, it is instantiated on the fly provided the inventory level is greater than zero. This methodology minimizes the number of active objects in the simulation and improves performance.

2. Primary Component Interactions

There are two primary interaction templates that give DAFS the majority of its functional capability. The first is the referee-mediator and referee-adjudicator template, which apply to sensors and munitions respectively. The second is the source-listener template that allows two things. First, it allows the monitoring property changes throughout the model as a data gathering medium for analysis and second, as briefly described above, it allows elements within the simulation to act based on the actions or property changes of other elements. Table 6 captures the basic organization and function of these templates.

Template	Type	Function
Referee/Mediator	Referee/Sensor Mediator	Determines platform interactions
	Referee/Munitions Adjudicator	Determines munitions effects
Source/Listener	Property Change	Triggers actions based on the state change of another entity
	Simulation Event	Triggers actions based on the occurrence of a particular event

Table 5. Interaction Templates[5]

a. Sources and Listeners

The source-listener protocol is essential to the success of discrete event programming. As a tool, the protocol is one of the items that separates discrete event simulation from time step simulation. In time step simulation, all potential interactions must be checked at each time step to resolve whether or not an interaction is occurring, an evaluation load on the order of N^2 for each time step, where N represents the total number of entities in the scenario. This also means that interactions that would have begun in the mid-point of the time step are delayed and thus alter the level of "reality" attained. Discrete event simulation, on the other hand, by implementing the source-listener template, calculates the precise time of interactions and schedules the event at that time. At most this requires an evaluation load on the order of N for every event or property change. As the events are reached on the event list, the appropriate actions are taken, and the simulation continues.

The two main uses for the source-listener template are simulation control and data gathering. However, both function in exactly the same manner. The primary difference being that when a data gathering listener "hears" a change in the simulation, it only records the information and does not subsequently affect the remainder of the simulation run.

[5] Havens (2002).

The key elements of the source-listener template are the sources, the listeners and the registration process between the two. Sources, as the name implies, are the source of a trigger that may or may not require action on the part of another entity within the simulation. It doesn't matter if there is a registered listener or not, if it is something that could affect something else, the source is responsible to "fire" the information. The listener is the receiver of this information, and is responsible to process it however it has been programmed to do so.

The critical link is the registration process. The listener must be registered as such with the source in order to receive the information. This registration process provides the benefit of reducing the processing load to only those entities that have the need or capability to deal with the particular information fired. For example, a detections counter would be registered as a listener to a particular sensor, the source. Every time the sensor fires a detection event, the counter will hear it and tally that a detection event had occurred. This is an example of a data-gathering listener. If the parent platform of the sensor was also registered as a listener, it may alter its course as a result of the detection event. That would be a simulation control item.

Sources and listeners are used extensively throughout DAFS. One of the most impacting uses is in the evaluation of interactions between weapons or sensors and the platforms in the battle space. For this application, referees are registered as listeners and oversee the potential for interactions.

b. Referees and Mediators/Adjudicators

The referee-mediator/adjudicator template is used extensively in DAFS. The concept of mediators and adjudicators is exactly the same except that the mediator applies to sensor-target interactions and the adjudicator to munitions-target interactions. For the sake of brevity, only the referee-mediator template is described here with references to the adjudicators as necessary.

The referee may be viewed as a simulation monitor that listen for changes within the simulation that may lead to interactions between entities, or to changes in previously determined interactions. These events could be the appearance of a new entity, a change in an entity's velocity vector or the detectable properties that an entity may be emitting. In essence, a referee is a focused "eye-in-the-sky" that monitors whether changes in entities it is responsible for might result in subsequent interactions. Once this potential is determined, the referee passes entities that may have interactions to the appropriate mediator or adjudicator.

In the case of sensors and targets, the referee listens for changes in targets that would potentially create or change a detection event. If, for instance, a target maneuvers, the referee hears the change and executes its process. The referee takes the target's new course and speed, and with it, determines what sensors the target will come within range of. The referee only considers the sensors that have the ability to detect the target. For each of the sensors that will have the target enter its footprint, the referee passes the target and the sensor information to a mediator. The mediator then uses the detection algorithm associated with its footprint to determine whether or not a detection event will occur. If so, the detection will be scheduled on the event list and the simulation will go on. If not, nothing occurs. If the sensor already has a detection scheduled for a particular target and it will no longer occur, or will be different, the appropriate changes are made.

The referee-adjudicator template follows the same logic described for the referee-mediator and is applied when a munitions object fires an impact event. The referee then accomplishes the same task with the munitions footprint and the targets within it. Adjudicators determine the extent of damage occurring to targets based on the munitions type and distance from the impact.

3. Functional Components

Functional components within a simulation handle administrative matters and serve as decision or organization modules. Within DAFS, there are three significant functional components that will be discussed: mover managers, command elements and kill probability objects. Additionally, the inventory object will be described. The inventory object does not, at this point, play a critical role in the simulation. However, its concept and functionality will become increasingly beneficial as the research in this area grows more complex.

a. Mover Managers

Mover managers, as the name implies, manage the movement behaviors of the platforms. Each mover manager object type represents a specific movement pattern that a platform may engage in. Current forms of mover managers are patrolling, intercepting and basic path following. Each mover manager gets its unique form through different combinations of location control and behavior. Each uses JAVA Point2D objects for location management and simulation event protocol for its behavior. Table 4 summarizes these mover manager types.

b. Command Element

The command element is a functional element associated with each platform. This element organizes priorities, objectives and capabilities within each unit. The command element has two primary functions. First, it acts as a priority filter to keep the highest desired action at the top of the list. Second, it maintains track over requirements, such as reporting criteria or munitions inventory status, and ensures the entity complies with actions as necessary. The command element makes use of the listener protocol to accomplish its monitoring functions. It is the command control element that controls which of the mover managers is currently being used by the platform and whether or not it will engage targets within range.

Mover Manager	**Location Control**	**Behavior**
PathMoverManager	List of JAVA Point2D objects	Sequences through the list of points and stops at the end.
PatrolMoverManager	List of JAVA Point2D Objects	Sequences through the list of points and repeats a set number of times or unlimited until another mover manager takes control
InterceptMoverManager	Single JAVA Point2D object	Proceeds to the point and triggers the behavior contained

Table 6. Mover Manager Descriptions[6]

[6] Havens (2002).

c. Kill Probability Objects

Kill probability objects contain the ability to generate the expected probability of kill for a particular munitions type against a particular platform type as a function of range. The basic template for these objects does not presume the method that will be used to generate the value. Rather, the kill probability interface requires a contract set of methods that the user must employ so that any kill probability generator will work. Kill probability implementations currently in DAFS include linear, piecewise linear and exponential functions. A kill probability implementation that utilizes a lookup table was also developed. Other functional forms may be developed and used, as long as the kill probability interface is implemented.

d. Inventory Objects

Also stemming from an interface, inventory objects were developed to allow DAFS some level of benefit from logistic considerations. The interface for this object defines basic inventory methods including adding inventory, reducing inventory, returning the level for a specific item and many other standard inventory functions. Currently, the inventory object is used to track munitions inventory levels to assist in both the VPA calculation and eventual use of munitions. Again, because the objects stem from an interface, the user may design several other inventory objects for specific purposes and give them additional methods required to complete the functionality desired.

4. Weapon Assignment Components

A basic requirement of all combat simulations is the assignment of weapons to targets as an interaction between the represented units. In DAFS, a value optimized calculation is made from a variety of factors to assign the optimal weapon to the given known targets. In this manner, DAFS is able to emulate a networked battle force in which knowledge of the targets is shared throughout. This mathematical characterization of networked systems is carried out by the Value of Potential Assignments and the Constrained Value Optimizer components.

a. Value of Potential Assignments

VPA refers to the overall potential value of an assignment pairing between a friendly unit and a non-friendly unit. This assignment is not necessarily a firing or sensing assignment though it may potentially be used to that end. Rather, it is a general assignment based on a number of potential factors such as engagement potential, tracking benefit, the overall threat the unit may present and many others. Placing a value on such an assignment is a necessary element and provides a means for analysis within the battle space. These factors turn out to be critical for analysis in networked fires.

The VPA concept takes into account several factors that contribute to assigning a particular friendly unit the responsibility of an enemy unit. As with the term assignment, responsibility is used here in a general sense to indicate focus of attention for a friendly unit. The idea is to take several potential factors available in the battle space that may influence a unit's actions, and process them in such a way that a final value or set of values is generated. Once this is done for each potential pairing of friendly to non-friendly units, those values are applied to an objective function designed to maximize the total value of a particular assignment set, based on a mission goal and a user-defined set of constraints.

b. VPA Usage

For the purposes of this thesis, a set of factors was chosen to capture the range of considerations while avoiding excessive detail. The general categories of factors chosen are probability of kill (P_k), threat (expressed as reverse probability of kill), inherent value of friendly and non-friendly units and the type of action engaged in (e.g. defense, peace keeping). While this may, at first glance, appear to be a very brief list of factors that would provide a limited factor space for exploration, indeed it is not. In each of these general areas there are extensive considerations and assumptions that may be made.

Some of the sub-factors related to the primary factors listed above are explored explicitly and some are explored implicitly and are presented in Table 9. The implicit factors listed are influencing factors within the associated primary factor, which for the purposes of this research, are considered captured to a sufficient extent in the parent factor. Explicit modeling of these factors would cloud the process and provide an increased fidelity that is not necessary at this point in the development of DAFS.

Primary Factor	**Explicit Sub-Factor**	**Implicit Sub-Factor**
Probability of Kill (P_k)	Range Munitions Firing unit type Targeted unit type	Target Location Error (TLE) Munitions accuracy Munitions reliability
Threat	Range Firing Enemy unit type Targeted Friendly unit type	Munitions Munitions accuracy Munitions reliability
Unit Values	Unit type Scenario type	Strategic value Monetary value
Action Type	General category (attack, defend, etc.)	

Table 7. Explicit and Implicit Sub-Factors[7]

The use of the VPA and the associated formula used to arrive at it are the two main variables used to evaluate the potential value to networked fires. The VPA is generated as a result of a value formula that takes into account whatever factors have been designed into it. For example, one might propose that the factor involved in determining the VPA from a blue unit to a red unit is the expected value of eliminating the red unit. In this case, the VPA would be the red unit's pre-assigned value times the probability of killing it, which would be a function of the range between the two units. This versatility in the VPA allows for a variety of algorithms to be implemented to simulate a warfare or unit commander's thought processes when assigning weapons to targets.

c. Constrained Value Optimizer

Once a value function is chosen and the subsequent VPA values generated, the values are applied as the coefficients in an overall objective function designed to optimize the total benefit of all the potential assignments. Of course, the result must satisfy a given

[7] Havens (2002).

constraint set. To continue with the example above, a potential objective function may be to maximize the sum of all potential assignments from blue to red. If that were the extent of it, the solution would be easy; make all assignments that have a positive VPA. However, as is usually the case, there are limits. In our example, suppose that each blue unit may be assigned to at most one red unit and that a VPA greater than 25 is desired in each case. This results in each acceptable VPA solution having to have a potential value of greater than 25 with the number of possible pairings being constrained by the assignment limitation.

Once applied, an optimized value for the sum of all the possible combinations of assignments is generated producing an assignment set. This is a relatively standard optimization problem. However, once a blue or red unit moves, or any other factor used in either the VPA formula or the subsequent optimization is changed, the assignments may not still be optimal. Managing the subsequent re-evaluation of the optimization turns out to be a key factor in the attempt to synthesize simulation and optimization.

The portion of the simulation that evaluates the battle space information and provides a solution to the implemented objective function is the Constrained Value Optimizer (CVO). The term "constrained" in the name refers to the fact that the resulting optimal solution generated by the CVO is constrained by either the passing of time or by subsequent events that may or may not invalidate the standing solution. In this manner, the weapon and target pairings of the warfare commander can be maintained at their optimal value at each calculation of the VPA and CVO but may differ as the events unfold in the simulation.

B. DAFS EXECUTION

The execution of DAFS can be broken down into three distinct areas: input, runtime and output. DAFS input is a collection of XML files that predefine every aspect of the participants, the scenario, the nature of the runs and the desired output. Many of the input components are independent of one another and therefore may be altered or replaced without any affect on the remaining pieces. Others contain several related components and must therefore be altered as a whole. However, sub elements within these larger input files may be swapped out in the same manner as long as the integrity of the overall file remains. Runtime for DAFS is consists of a standard discrete event simulation run that contains entities that are intermittently controlled by the use of a local optimization routine. The output is available in a number of formats and again, is dictated by input XML files. The user has the choice of displaying output to the screen, writing to files, generating XML files or any combination. XML output files are particularly beneficial as they may be altered using XML stylesheets or queried in a number of ways to present the results.

1. Input

Figure 8 is a graphic representation of the input scheme used by DAFS. Each of the blocks on the left side of the diagram represents a self contained XML document and the significant contents. From this diagram, it can be seen that the simulation entities input file must contain a significant amount of information, which is due to the nature of constructing a unit for participation. Because the components used in the construction of a unique platform are closely tied to each other, with respect to references, they must be generated at the same time so that the proper associations can be made. This does not mean that every entity must contain each of the listed items; it simply means that if any of the listed items are going to be a part of the entity, it must be contained in the appropriate XML tag structure associated with construction of an entity. The remaining blocks on the left side of the figure also represent potentially discrete input files, each having a particular tag structure.

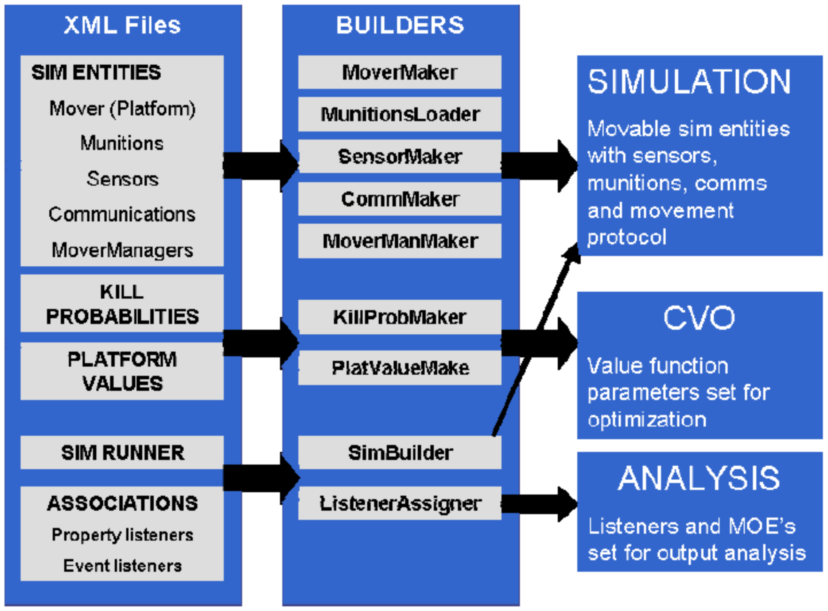


Figure 9. DAFS Input Design Structure[8]

[8] Havens (2002).

As the input files are discussed in the following paragraphs, the reader may find it useful to refer to Appendix A, which contains sample XML files. The experienced XML user will note that the structures of the input files may be defined and validated through the use of Document Type Definitions (DTD) or SCHEMA documents however.

The kill probabilities file contains the necessary information to generate kill probability object instances discussed in the previous section. Each kill probability instance covers all engagements between a particular munitions type and a particular platform type. Therefore, once the user has defined all of the possible interactions between munitions and platforms, this file does not need to be modified. When a new munitions type of platform type is desired in the scenario, the user may define new kill probabilities for the new entity and add them to the existing file. Kill probability instances should be generated to take into account friendly fire issues. Recall, the munitions-target referee will evaluate all targets within the munitions footprint regardless of the affiliation.

The platform values file is another file that once generated, can be used for all runs. Within the file, each platform type is assigned a value for a given scenario type. Once the user is happy with the choices, the file does not need to be altered unless new platforms are added or a different scenario type has been developed. Conversely, this file represents an excellent choice of a design point for analysis.

This simulation runner file contains the information necessary to implement the Schedule class in Simkit. This file contains the parameters that define the simulation stop criteria, non-data output options and the number of repetitions desired. The stopping criteria may either be set to an elapsed time or to the occurrence of a specific event, the tenth kill for example. The non-data output options refer to the simulation event output. The two categories are verbose and single step. If verbose is set to true, the simulation will generate an event list to the screen at each event change while the simulation is running. A selection of false will yield nothing. Single step will control the simulation by allowing it to progress one event at a time and, by definition, will invoke the verbose output method. This allows the user to view each discrete event as it occurs. The repetitions selection will reset all components to the original configuration and begin the simulation again. This is particularly beneficial for multiple run analysis of probabilistic scenarios as the simulation will begin the same but will not provide the same exact run due to the implementation of different random numbers.

The associations XML file is used to assign the listeners not already prescribed by DAFS. DAFS automatically registers the appropriate listeners necessary to accomplish successful running of the simulation. The associations contained in this file are for data gathering purposes.

The middle block of Figure 8 represents the collection of XML builders that take portions of the XML documents and use them to create the necessary JAVA objects. The DAFS main method is responsible for farming out the appropriate XML elements to the correct builder. In all cases, with the exception of the munitions loader and the listener assigner, the XML builder class instantiates JAVA objects corresponding with the name of the class. For instance, the mover maker instantiates a Simkit Mover object, the base component for each unit in the simulation. These were presented earlier as platforms.

The listener assigner and the munitions loader each have slightly different functions but do still convert XML file data into simulation information. The munitions loader by munitions type loads the inventory object on each platform with the corresponding number of rounds as initial inventory. Again referring to Figure 8, the munitions information comes from within the large XML file of simulation entities. Specifically, the munitions element is a sub element of the mover element. This structure is what allows the loader to associate the munitions with the correct platform. An example of a simulation entities file is contained in Appendix A.

The listener assigner is primarily to establish data gathering connections for simulation monitors. Typically simple statistics objects, these monitors listen to the objects to which they are assigned or to the simulation in general and tabulate events or property changes. The builder file in this case serves to register the appropriate objects as listeners. These monitor objects and their configuration is essential to retrieving usable output from a simulation run and the concepts are discussed more fully in the output section below.

2. Runtime

A DAFS simulation scenario currently involves two sides, red and blue, although there could be an arbitrary number of sides. Each side is given its objectives through the implementation of the mover managers and the level of aggressiveness protocol assigned to the command element. The mover managers dictate where and how the platforms will proceed as the simulation progresses and the aggressiveness factors dictate how the platform will behave upon interaction with other platforms.

Additionally, the blue side is provided a scenario posture, which affects the player values on both sides and has subsequent impact on the VPA values as they are calculated. The implementation of the CVO is only accomplished for the blue side and assumes the red side is using less sophisticated operational capabilities. Namely, the red side is assumed to operate as a conventional force with standing orders for objectives and rules of engagement (ROE) set from the beginning. The point of DAFS, at least initially, is to explore whether or not the networking of fires and sensors by a force has greater effectiveness than fighting with pre-designated routes, assignments, and ROE. Therefore, initial analysis with DAFS does not assume that the opponent is implementing the same technology so there is a visible difference in the results if indeed the networking effort has an affect.

The command control object associated with each platform provides it with a unique engagement behavior. When a platform of one side detects an opponent platform, as in the real world, it must do some analysis as to its course of action to follow. In the case of DAFS platforms, this is accomplished through its ROE in the command object to determine whether or not to engage. If the platform determines not to engage, the command element will dictate in what manner the platform will avoid engagement and implement the appropriate mover manager. This once again highlights the component nature of the DAFS simulation and its resident flexibility. Rather than employ a single mover manager with differing methods for the particular behaviors, each mover manager is a distinct object that can be removed, replaced or added. This allows the user to maintain behaviors that have proven successful and change only those that need further development.

If the platform elects to engage, the engagement protocol for the particular munitions will be called. This may implement a delay time designed to emulate set-up times associated with particular delivery systems. Currently this emulation is based purely on the munitions type and does not account for different delivery systems for the same munitions type.

The simulation will run in this manner until the designated stopping a criterion has been met. Upon completion, the output that was designated during the XML input process will be gathered and output according to the selected output methodology.

3. Output

Through extensive use of the listener functions described earlier, statistical objects are created and tasked with monitoring specific events within the simulation. As a result, the desired output is "programmed in" to any specific simulation run and subsequently provided to the user in a predetermined format. The tally and time varying statistics objects used for data gathering are both resident in Simkit. The tally version keeps track of simple values that only require counting, such as the number of red players killed or the number of missiles used. The time varying version keeps track of the level of a particular state and the corresponding times when the value changes. This state trajectory can then be used to retrieve quantitative values with respect to time, such as the number average number of contacts held or total time with a certain number of contacts held. Because the information is retained with time information, the time varying statistics object is capable of returning values as a function of time. In this case, the time-averaged mean of contacts would be computed by dividing the area under the curve by the current time.

Through the use of XML file writing functions in JDOM, the output values collected by the statistics objects can be selectively written to output XML files for future analysis. As an option, the output information may be written to the screen or to output text files. The various outputs are selected at runtime through the input process and the associations input document. XML output files are extremely beneficial to the user because they can be manipulated in a number of ways to present the output. Through the use of XML stylesheets, or XSL documents, the output values can be selectively extracted and displayed in several forms including web pages and as graphs.

C. RADAR MODEL IMPLEMENTATION

Before attempting to implement the new radar model in DAFS, a spiral development with designated benchmarks was implemented to ensure the stability of the model at each step and to minimize loss during coding. These benchmarks were decided upon prior to the models full implementation as a logical train of steps to reach the realization of the mature model in DAFS. These benchmarks also afforded an opportunity for "tagging" of the code in a digital repository to minimize any loss due to programming error or computer malfunction. The benchmarks for the model are outlined below.

Benchmark 0 (24 hour model)	A simple model of a single radar sensor mover against three identical platforms representing targets. The mover will move around in a semi-random patrol and interact with the platforms providing detection opportunities. The radar sensor is modeled as a simple cookie cutter sensor with an effective range equal to the maximum range of the sensor.
Benchmark 1	Still a simple model of a single radar sensor mover against three platforms representing targets. In this iteration, the platforms each have a different RCS. The cookie cutter radar sensor now has an effective range based off of the effective range against the target's RCS. A delay is added to the detection time based on the movers speed through the distance difference between maximum range and maximum effective range.
Benchmark 2	Still a simple model of a single radar sensor mover against three platforms representing targets. In this iteration, the platforms each have a different RCS. The radar sensor retains the detection delay based on effective range and adds a second delay based on a gamma distribution over the remaining distance from maximum effective range to the sensor. This gamma distribution provides a simple model of the delay based on number of dwells to determine detection.

Benchmark 3 (first stochastic)	Still a simple model with two radar sensors on the same mover against three platforms representing targets. In this iteration, the platforms each have a different RCS. The radar sensor retains the detection delay based on effective range. It calculates a probability of detection based on the number of dwells from max detection range to the sensor. If detection is determined to occur, then a second delay to detection is added based off of a gamma distribution over the remaining distance from maximum effective range to the sensor.
Benchmark 4	The model retains all the characteristics from Benchmark 3 and adds a property of altitude to the targets. Properties of minAltitude and maxAltitude are added to the two sensors on the mover effectively creating a surface search and an air search radar. This may be scrapped if it seems more feasible to create mediators for air search radars and surface search radars that prohibit certain types of targets.
Benchmark 5 (first DAFS Model)	Implement a simple 2 ship model utilizing the radar sensor and mediator in DAFS. Interceptor style missiles are not implemented at this stage.
Benchmark 6	Continue with the simple 2 ship DAFS model implementing helicopters and interceptor style missiles to examine a complete AAW picture.
Benchmark 7 (Final Model)	Create a more complex scenario involving 10+ warships with full weapons implementation in DAFS.

Table 8. Model Benchmarks

1. Simkit Implementation

Benchmarks 0 through 4 were conducted in Java utilizing Dr. Arnold Buss' Simkit library and a second library called smdx. This second library houses the sensor, referee, mediator, and other classes required for discrete event simulation of two dimensional movements. These same libraries are also used as the analytical engine for all movers and kinematic occurrences within DAFS. The two libraries were combined with graphical display applets to form the "Sandbox," a two-dimensional representation of the movement modeled in Simkit. This provided a test laboratory in which the modules required for the radar model could be tested on a smaller scale to insure their function prior to adapting them for DAFS. The actual code utilized for these modules is located in Appendix B.

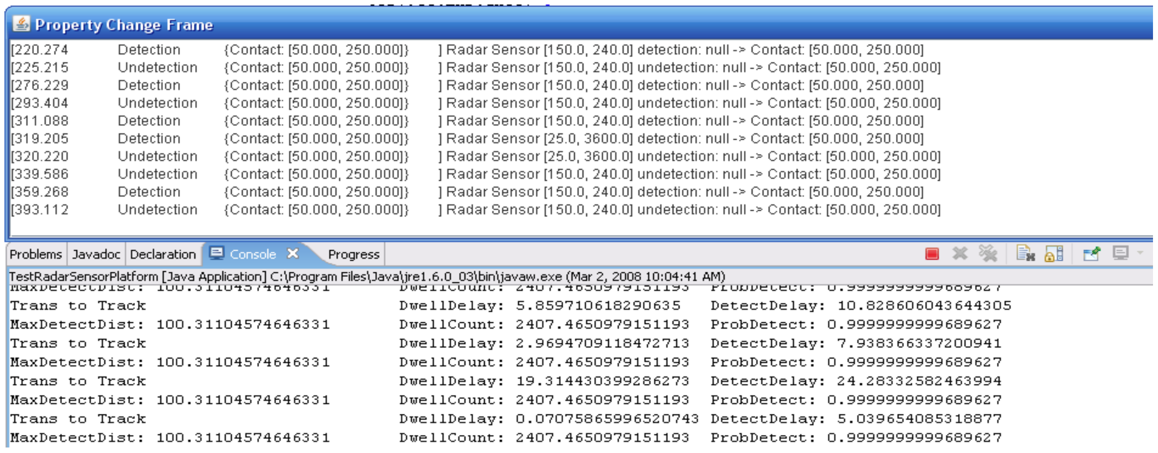
To implement the radar model in Simkit, two classes are required: the RadarSensor class and the RadarMediator class. The RadarSensor class contains the physical characteristics of the sensor such as Range, the mover it is assigned to, and the sensors Prf. Other than the constructor, this class only contains the methods required to set and retrieve the sensor information when an instance of the RadarSensor is created. The Radar Mediator class contains the actual kinematic mathematics required of the sensor. It includes a constructor that creates instances of a contact list and the two random variates (a Uniform[0,1] and a Gamma[1,1] in this case) required for the stochastic model. It also includes several methods required to set and retrieve information from the contact list and the random variates, but the main function of the class is encapsulated in the doEnterRange and doExitRange methods.

The doEnterRange method is activated when a contact enters the maximum range of the sensor. The method first verifies that the sensor type is of the RadarSensor class and then checks to see if the contact is already on the contact list. The situation when the contact is already on the contact list occurs when a doEnterRange event for that particular contact has already happened and the required time for detection has not elapsed. This situation is commonly seen when the kinematic solution of the sensor and target change before detection has occurred. If the contact is not on the contact list for the sensor, it will be added and the detection calculations will continue.

Utilizing the same mathematics outlined in Section II and the information manifest in the RadarSensor class, the maximum possible detection distance and the number of glimpses from that point to the target is calculated to determine the final probability of detection. A draw from the Uniform random variate is made and compared to the probability of detection to determine if a detection event occurred. If the detection was successful, a draw is made from the Gamma random variate to determine how many dwells were required for the detection and this time is added to the transit time to maximum detection distance to determine the delay until detection. From this information, a Detection event is then scheduled and the method is completed.

The doExitRange method does the opposite of the doEnterRange method. It is triggered when the contact leaves the maximum range of the sensor. If a Detection event was scheduled but had not occurred, the method removes that event from the event list to eliminate the possibility of detection. If the Detection event had already occurred, then the method schedules an Undetection event which will cause the sensor to stop reporting the target and prepare it for a new detection of the same contact. In this manner, the contact is able to leave the detection range of the sensor and effectively be lost from sight.

After building this implementation in Simkit, the model was tested utilizing the graphical display applet to verify its function and to provide feedback on the settings of the Gamma random variate. A third class, TestRadarSensorPlatform, was created to instantiate the necessary parts and to provide an input to the model. In Benchmark 4, a mover was assigned two radar sensors of differing ranges and tasked to move randomly among three different platforms. Each of the platforms had a different RCS to simulate different size of targets. The mover was placed in play and moved around interacting with all three targets. Data was collected from the run and displayed so that it could be examined more easily. This combination of exact positional and simulation time data and the easy to interpret graphical aide facilitated a quicker verification of the models function.



```
[220.274    Detection      {Contact: [50.000, 250.000]}    ] Radar Sensor [150.0, 240.0] detection: null -> Contact: [50.000, 250.000]
[225.215    Undetection    {Contact: [50.000, 250.000]}    ] Radar Sensor [150.0, 240.0] undetection: null -> Contact: [50.000, 250.000]
[276.229    Detection      {Contact: [50.000, 250.000]}    ] Radar Sensor [150.0, 240.0] detection: null -> Contact: [50.000, 250.000]
[293.404    Undetection    {Contact: [50.000, 250.000]}    ] Radar Sensor [150.0, 240.0] undetection: null -> Contact: [50.000, 250.000]
[311.088    Detection      {Contact: [50.000, 250.000]}    ] Radar Sensor [150.0, 240.0] detection: null -> Contact: [50.000, 250.000]
[319.205    Detection      {Contact: [50.000, 250.000]}    ] Radar Sensor [25.0, 3600.0] detection: null -> Contact: [50.000, 250.000]
[320.220    Undetection    {Contact: [50.000, 250.000]}    ] Radar Sensor [25.0, 3600.0] undetection: null -> Contact: [50.000, 250.000]
[339.586    Undetection    {Contact: [50.000, 250.000]}    ] Radar Sensor [150.0, 240.0] undetection: null -> Contact: [50.000, 250.000]
[359.268    Detection      {Contact: [50.000, 250.000]}    ] Radar Sensor [150.0, 240.0] detection: null -> Contact: [50.000, 250.000]
[393.112    Undetection    {Contact: [50.000, 250.000]}    ] Radar Sensor [150.0, 240.0] undetection: null -> Contact: [50.000, 250.000]
```

```
TestRadarSensorPlatform [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (Mar 2, 2008 10:04:41 AM)
MaxDetectDist: 100.31104574646331        DwellCount: 2407.4650979151193   ProbDetect: 0.9999999999689627
Trans to Track                           DwellDelay: 5.859710618290635    DetectDelay: 10.828606043644305
MaxDetectDist: 100.31104574646331        DwellCount: 2407.4650979151193   ProbDetect: 0.9999999999689627
Trans to Track                           DwellDelay: 2.9694709118472713   DetectDelay: 7.938366337200941
MaxDetectDist: 100.31104574646331        DwellCount: 2407.4650979151193   ProbDetect: 0.9999999999689627
Trans to Track                           DwellDelay: 19.314430399286273   DetectDelay: 24.28332582463994
MaxDetectDist: 100.31104574646331        DwellCount: 2407.4650979151193   ProbDetect: 0.9999999999689627
Trans to Track                           DwellDelay: 0.07075865996520743 DetectDelay: 5.039654085318877
MaxDetectDist: 100.31104574646331        DwellCount: 2407.4650979151193   ProbDetect: 0.9999999999689627
```

Figure 10. Data From Simkit Implementation of Radar Model

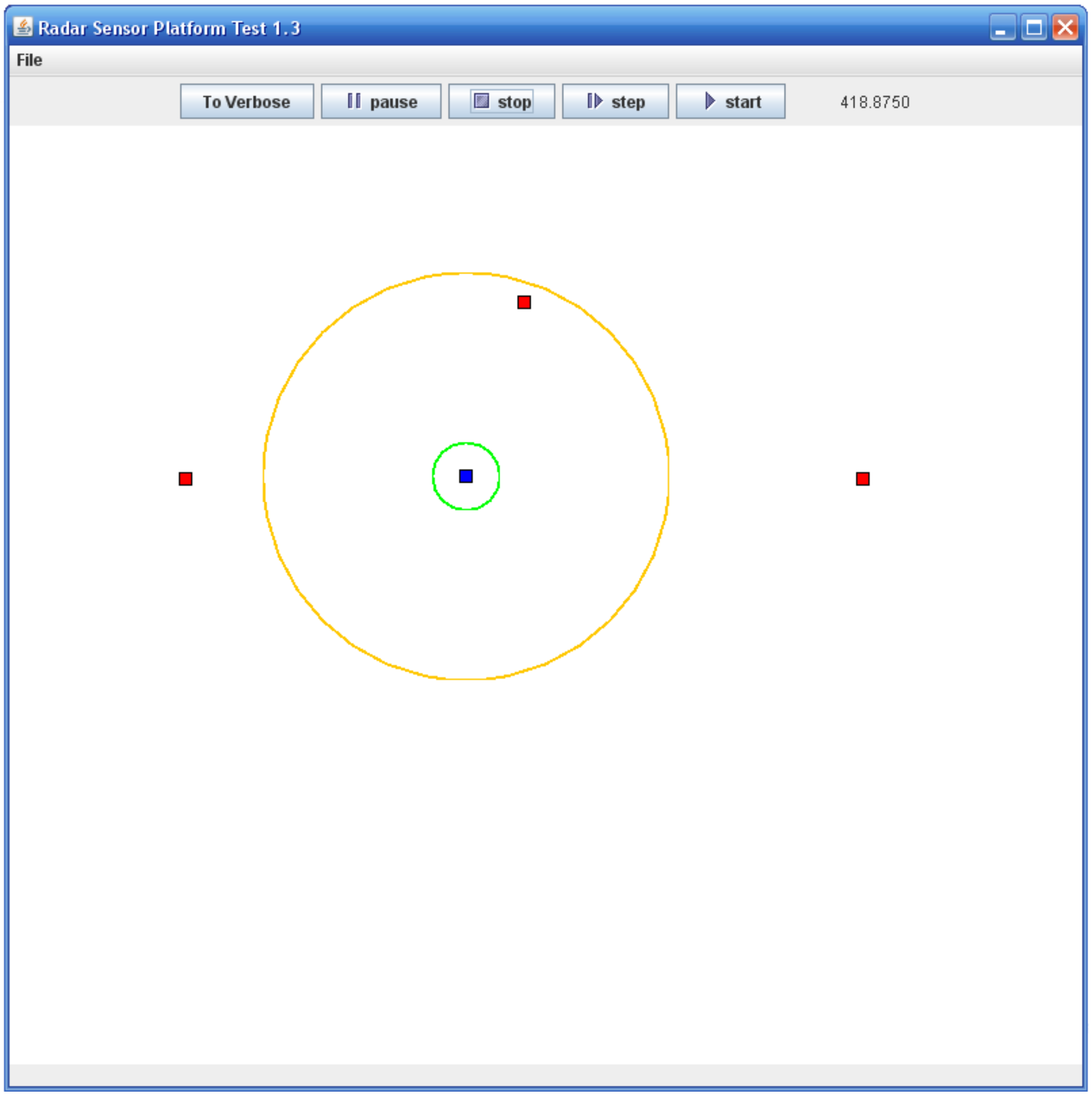


Figure 11. Visual Display of Simkit Implementation of Radar Model

2. DAFS Implementation

Because DAFS utilizes Simkit to drive its kinematic model, the classes developed for the Simkit implementation in the Sandbox can be plugged directly into the classes used in DAFS with minimal changes. The only modifications necessary for the implementation in DAFS was to change the location of where the classes pulled information for items such as radar cross-section and prf and a change of the name of the contact list within the mediator to the one that DAFS utilizes. The source code for the classes used to implement the radar model in DAFS can be found in Appendix C.

This modular style of implementation is a hallmark of the DAFS combat model where sensors, munitions, and other items within the simulation can be created and tested in an outside source and then plugged into the model with little effort. This allows a common user to develop simulation items derived from research requirements or new developments and implement them into the model quickly.

Despite this modular architecture, not all items in the model work in this manner. Code changes were made to several sections of the standard working code in DAFS to account for the new sensors and to address the added information of prf and RCS. These changes were made in separate modules designed to read the scenario database and produce the XML version of the scenario file. Because these requirements of a naval radar model were not incorporated into the original DAFS model, base code changes had to be made to add them.

A second complication that arose was the model's ability to detect munitions. Because the DAFS model was created with ground combat as the primary concept basis, the idea of detecting and then shooting down an inbound munition was never considered. In naval combat, the ability to destroy an enemies inbound ASCM is a key concept to the defense of the ship and must be accounted for. Changes to the DAFS base code were made to allow for detectable munitions to be seen by the sensors. This also enabled the platforms to react to these inbound treats and use their surface-to-air missiles accordingly.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SCENARIO AND ANALYSIS

With the implementation of the radar model in DAFS, it is now possible to create and analyze a more robust maritime combat scenario. Because radar is only utilized in Anti-Surface Warfare (ASuW) and Anti-Air Warfare (AAW), this scenario does not examine submarine combat at all. Additionally, due to the complexity of adding Defense Counter Air (DCA) air craft, the only aircraft included in this scenario are helicopters utilized in a search and targeting mode.

A. BASE SCENARIO DESCRIPTION

The United States Navy requires all ships scheduled to deploy overseas to complete a combined fleet exercise where they operate in a coordinated effort within a battle group. These battle groups consist of 7-8 ships including one aircraft carrier and a minimum of one cruiser. During the exercises, the aircraft carrier is tasked to conduct strike missions into a neighboring unfriendly nation requiring majority of the aircraft to be employed in this tasking. The accompanying surface combatants are required to provide protection to the aircraft carrier during this critical period of flight operations that can last for days if required.

Placed in opposition to the battle group are three other surface ships. They act as aggressor forces from the nation being attacked by the aircraft carrier and attempt to break the protective layer of the surface vessels in an attempt to sink the aircraft carrier. Although there are only three ships normally employed in this role, they are regenerated from time to time to represent from 6-9 enemy combatants. This enables the Navy to employ fewer vessels to carry out this role while still representing an equal number of enemy ships.

Despite this equality in numbers, the battle group is always able to win these simulated engagements. This can be attributed to a variety of reasons (superior technology, coordinated air coverage, coordinated attacks by the battle group, etc.) but the fact remains that the perceived belief is that a seven ship flotilla is able to adequately protect the aircraft carrier from a determined attack by up to nine aggressors. To test this belief, the scenario will incorporate these same number and qualities of ships in a repeated simulation to determine the veracity of this belief.

1. Scenario Units

The blue forces comprising the U.S. Navy assists represent a typical selection of vessels that would participate in a combined fleet exercise. These forces consist of 1 aircraft carrier, 2 TICONDEROGA class cruisers, 2 Flight I/II ARLEIGH BURKE class destroyers, 2 Flight IIA ARLEIGH BURKE class destroyers, and 4 SH-60R LAMPS helicopters launched from the cruisers and Flight IIA destroyers . The names of the vessels are drawn from the units of CARRIER STRIKE GROUP 5 based in Yokosuka, Japan and are used only as a convenient naming convention. The statistics and characteristics of the vessels are drawn from a generic profile for each class and do not contain specific information from the actual ship.

Blue Forces		**Red Forces**	
Name	Class	Name	Class
KITTY HAWK	America Class aircraft carrier		
SHILOH COWPENS	Ticonderoga class cruiser	LUYANG GUANGZHOU WUHAN	Type052B class destroyer
FITZGERALD STETHEM	Flight I/II Arleigh Burke destroyer	LANZHOU HAIKOU SUIHUA	TYPE052C class destroyer
LASSEN MUSTIN	Flight IIA Arleigh Burke destroyer	JIAXING PUTIAN YICHANG	Type53H3 class frigate
Warlord 05 Warlord 15 Warlord 25 Warlord 35	SH60-R LAMPS helicopters	Han 05 Han15 Han 25	Zhi-9 Multi-purpose helicopters

Table 9. List of Forces in Scenario

Arrayed against the blue forces are a selection of ships from the People's Republic Army-Navy (PLAN) consisting of 3 Type052B GUANZHOU I class destroyers, 3 Type052C LANZHOU II class destroyers, 3 Type53H3 JINAGWEI frigates, and 3 Zhi-9 multi-purpose helicopters. These ship classes were chosen as representing a selection of vessels of varying capability that are marketed as competitors against the United States Navy. Lesser vessels such as corvettes and ASW frigates were not chosen because they have little to no chance against an average U.S. force. Again, actual ship names are used in the scenario for ease of reference but only a generic version of the ship class is represented.

Each ship class was referenced in Janes' publications to determine their physical characteristics, sensors, and armaments. These items were further cross referenced in other sources of unclassified military information to determine actual values for entries such as maximum detection range, maximum missile range, size of warhead, and others to determine the input values for the scenario. The listings of each of these classes, their respective armaments and sensors, and the factors assigned to them in the model can be found in Appendix D.

2. Scenario Layout

The scenario is located off the coast of San Diego, California with a geographic center at latitude 32.5N longitude 119.0W. For the purposes of this scenario, the carrier air wing's strike targets are located in the vicinity of San Diego and thus require the positioning of the battle group off the coast. Blue forces are arrayed in a defensive perimeter around KITTY HAWK with MUSTIN operating as the shotgun destroyer. The remaining units in the battle group remain approximately 20 nm from the center of the formation to provide KITTY HAWK enough room to conduct flight operations on 045-225 degree legs. COWPENS occupies the down threat position between San Diego and the carrier. SHILOH is positioned as a protection for any ASW threats attempting to sneak up behind the battle group as well as for ASUW threats from the seaward direction.

The remaining destroyers are placed at the remaining corners to complete the protection perimeter. All 4 SH60-R will launch from problem start and commence a cyclic rotation around the battle group to provide an extended sensor range.

The red forces start from three boxes located approximately 150 NM to the south, west, and northwest directions. These boxes allow for a random placement of the four units assigned to them to provide a variety of dispositions as the simulation runs through multiple executions. This random positioning also allows the simulation to test both dispersed and massed formations of the enemy against the blue battle force. Each box is assigned one of each class of ship and helicopter to provide an effective mix of assets. The boxes are also located almost the same distance from the start location of the carrier to allow for a coordinated attack on the KITTY HAWK battle group.

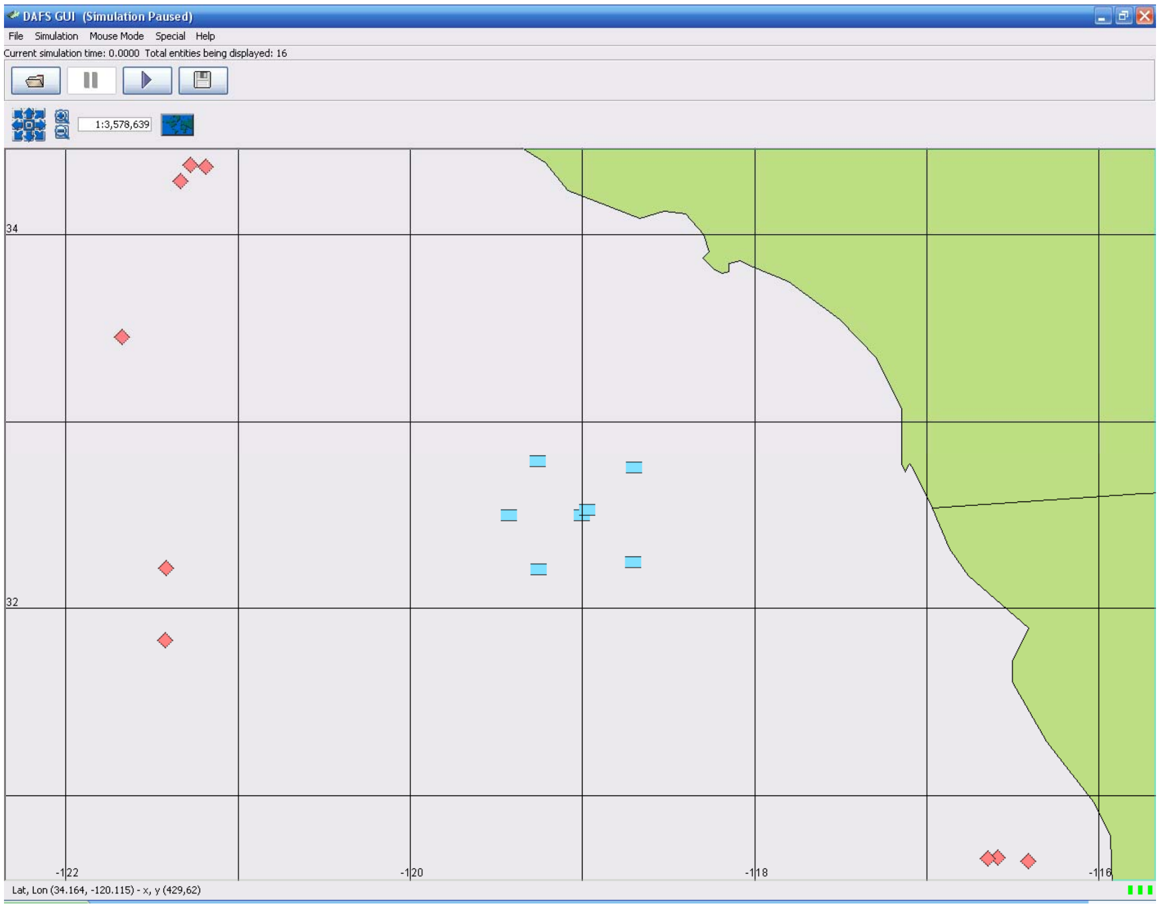

DAFS GUI (Simulation Paused)
File Simulation Mouse Mode Special Help
Current simulation time: 0.0000 Total entities being displayed: 16
1:3,578,639
34
32
-122
-120
-118
-116
Lat, Lon (34.164, -120.115) - x, y (429,62)

Figure 12. Scenario Start Positions in DAFS

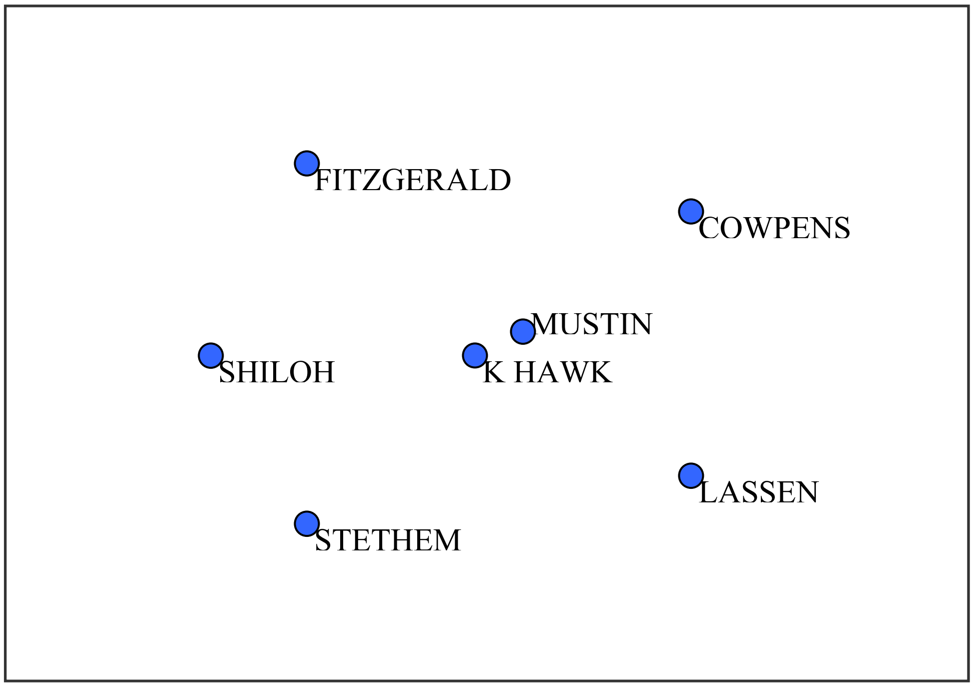


Figure 13. Blue Force Positions at Scenario Start

3. Scenario Victory Conditions

For this scenario, the victory conditions for either side are determined by the survivability of KITTY HAWK. If the blue side is able to destroy all red units and KITTY HAWK remains, then the blue forces will have won. If the red forces are able to destroy KITTY HAWK, no matter the extent of units lost, they will be considered the victor.

To enable the assets to arrive at a final result, the opposing red surface units are assigned a waypoint at KITTY HAWK's start position and instructed to procede to that location at best speed. The red helicopters are assigned the same waypoint but will proceed to it at a more cautious 45 knots so as not to sprint too far ahead of the red surface assets which provide the offensive power of the force. Both KITTY HAWK and MUSTIN are assigned two waypoints to simulate flight operations.

B. DESIGN OF EXPERIMENT

This experiment has been designed to look at an outcome of a possible exercise or combat action and not a specific design point or characteristic of the units. Because of this, the individual factors of the experiment remain the same with the exception of the random placement of the red forces and the stochastic results from the sensors and munitions.

1. Factors

As was stated above, no factors were changed in this experiment. However, it is possible to use this same model to investigate a specific characteristic of a platform, sensor, or formation by altering one or more factors over a given range. In this manner, the results could be statistically analyzed to see if these are a range of values that provide good results or even the minimum or maximum value to achieve a particular effect. The model lends itself to this type of analysis because of its stochastic nature and its short run time. A Nearly-Orthogonal Latin Hypercube (NOLH) may also be used to vary numerous factors and reduce the total number of simulation batches required to produce statistical results.

2. Scenario Replications

The scenario presented above was run for 300 replications of the first 360 minutes of the combat modeled. This represents over 1800 hours of real time naval combat operations. The actual elapsed time to run this many simulations was just over 6 and a half minutes. The results of all 300 runs were outputted to a Microsoft Access database for further query and analysis.

C. ANALYSIS

The analysis of the output data was conducted at several levels to attempt to gleam as much information from the runs as possible. Starting with the crudest instruments of statistical analysis, raw averages were used to identify trends and possible assertions. From there, the data was further analyzed using regression models to determine the strongest factors and to provide a deeper insight into the results.

1. Rough Analysis

The results of the data clearly indicate that the scenario tested is not an easy victory for the blue forces. Of the 300 simulation runs conducted, KITTY HAWK survived only 38% of them. This provides some evidence that when placed on an equal footing, the blue forces have a less than 50% chance of providing adequate protection to the force projection asset.

To examine this scenario further, the standard deviation of the probability of KITTY HAWK's survival was 48.62% providing a wide confidence interval ranging from 0% to 86.62% chance of survival using the first standard deviation alone. This wide confidence interval indicates that more iterations of the simulation must be run to provide a tighter estimate of the probability of mission success. Due to database limitations in MS Access and the limited scope of this thesis, more iterations of the scenario were not run to provide this better estimate of KITTY HAWK's survivability.

Continuing the examination of the results using the most basic of analysis tools, the data is further broken down between the iterations in which the blue victory condition was met and when it was not. Although the red casualty rate is always higher than that of blue, the difference between the two victory rates is less than a standard deviation for either side. This seems to indicate that the number of casualties suffered on either side has little to do with the overall victory and is more a factor of the risk involved in the combat scenario itself.

	Blue Victory	**Red Victory**	**Std Dev**
Average Blue Casualty Rate	47.13%	66.81%	22.10%
Average Red Casualty Rate	94.81%	88.80%	10.01%
Average Blue Attack Range (m)	47124.96	44682.81	9866.45
Average Red Attack Range (m)	45710.60	48816.62	9340.94

Table 10. Casualty Rates and Attack Ranges

Again, little insight can be found from the averages of the attack range for both sides. All four ranges are at approximately 75% of the longest sensor range providing adequate distance and time for the contact to be detected and then fired upon. The low standard deviation (+/- less than 5nm) indicates that this average distance holds true for the vast majority of the iterations.

Lastly, a histogram of the casualty rates was plotted to determine if any inferences can be made from their distribution. Blue casualty rates appear to follow a bi-modal distribution with maxima at 70% and 100%. Red casualties are similarly weighed towards the high end but with a distribution reminiscent of a reversed exponential distribution with the sharpest rise occurring between 90%-100%. This tends to indicate that the red forces will almost always suffer huge casualties in their effort to sink the KITTY HAWK. This may be do in part to the lethality of the weapons that both sides posses, especially considering the lack of anti-missile defense in this simulation. Contrary to this, the blue casualty level is much less predictable due to the wide dispersion of results and the bi-modal nature of its distribution. However, the casualties suffered by the blue forces will tend to be less than those of red based on probability.

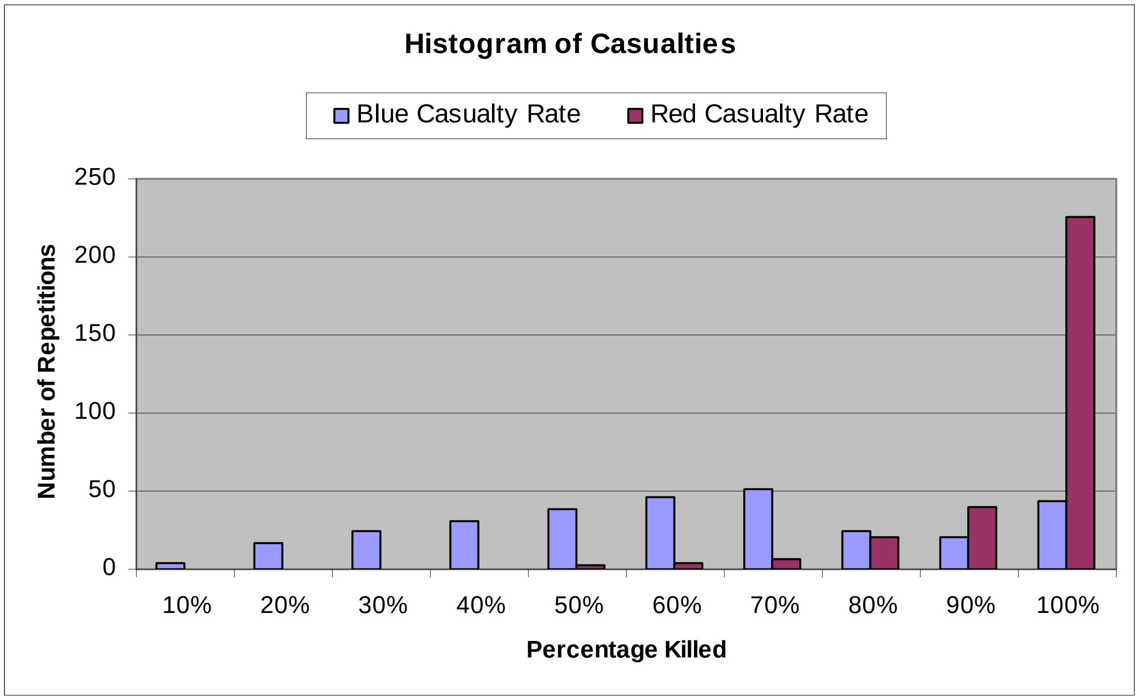


Figure 14. Histogram of Casualties

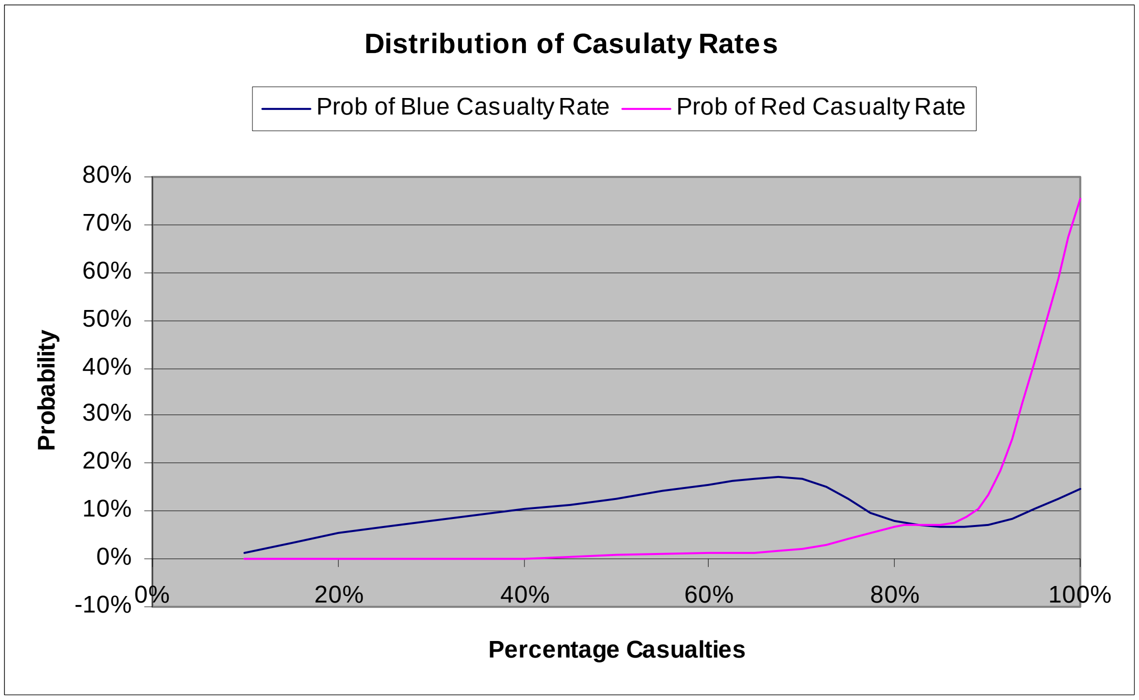


Figure 15. Distribution of Casualties

2. Logistic Regression Analysis

To perform a regression analysis of the victory conditions, the number of weapons employed and the number of detections by sensor were regressed against the blue victory outcome. An ordinal logistic regression was used to compare the strength and relevance of these factors to the actual outcome. Using a full factor model, the only factors that showed significance greater than 95% was the number of YJ62 missiles employed and the number of detections made by the KLC-1 and SPS-55 sensors. However, in examining the marginal change that could be made by increasing or decreasing other factors, it showed that a high number of SPS-64 or Type344 detections would have an even greater effect than the more linear change caused by the YJ62, KLC-1 and SPS-55 factors. Believing that the regression model was showing confounded results due to the inter-relation between sensors and weapons, the results were next modeled with each system regressed separately.

Term	Estimate	Std Error	ChiSquare	Prob>ChiSq
Intercept[0]	1.37219214	1.954061	0.49	0.4825
100mm	0.41366395	0.4550696	0.83	0.3633
HQ9 SAM	0.0039718	0.1166069	0.00	0.9728
RGM84	-0.0407674	0.0820113	0.25	0.6191
RIM-7	-0.0348611	0.0453896	0.59	0.4425
SA-N-12 SAM	-0.1108362	0.0960794	1.33	0.2487
SM2	0.01246085	0.0139856	0.79	0.3729
YJ62 SSM	**0.2932848**	**0.0829802**	**12.49**	**0.0004**
YJ81 SSM	-0.0016378	0.0402029	0.00	0.9675
YJ83 SSM	0.09578909	0.0575027	2.77	0.0957
KLC-1	**-0.0647346**	**0.0329938**	**3.85**	**0.0498**
LAMPS	-0.0280313	0.0326489	0.74	0.3906
SPQ9B	-0.0319247	0.0733181	0.19	0.6633
SPS55	**0.13489897**	**0.0679192**	**3.94**	**0.0470**
SPS64	-0.1125853	0.4681825	0.06	0.8100
SPS67	-0.0641925	0.0470773	1.86	0.1727
Type344	0.15456565	0.4693089	0.11	0.7419
Type360	-0.0412769	0.0603391	0.47	0.4939
Type364	0.01850652	0.0463543	0.16	0.6897

Table 11. Parameter Estimates for Full Model Logistic Regression

Prediction Profiler

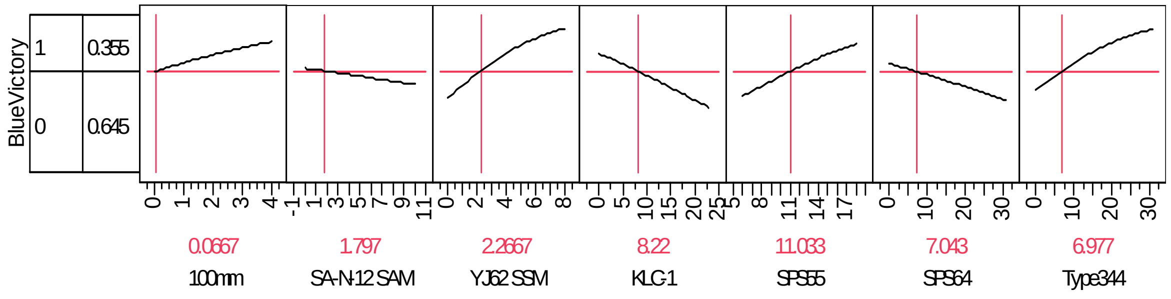


Figure 16. Prediction Profiler for Selected Factors of Full Model

When modeling the probability of the blue victory condition being met as a function of weapons employed, the dominant factor is much more discernable. Again utilizing logistic regression, the probability of a blue victory was regressed against the number of weapons employed. This time, the number of YJ62 and YJ83 employed were the strongest factors, with a larger number in both indicating a stronger probability of a blue loss. Surprisingly, the number of YJ81s employed seemed to have no effect on the outcome. A larger number of RGM-84 and RIM-7 missiles employed by the blue forces strengthened the chance of a blue victory but not strongly in comparison to the YJ62 and YJ83 factors.

These results tend to strengthen the idea that the main driver for red success is the employment of its two long range anti-ship cruise missiles. This inference is strengthened by the fact that the RGM-84 has a lesser power in predicting the outcome of the engagement. Examining the scenario, it is noted that the red forces must kill the KITTY HAWK to win, but do not have to kill any other vessel. However, the blue forces are required to kill all nine of the red surface vessels to properly safeguard the carrier. Because of this victory condition, if red simply sends a large number of missiles down range, they have a 1 in 7 chance per missile of hitting the KITTY HAWK in a blind shot. That gives the red forces a sizeable advantage when considering that the blue forces must destroy nine targets instead. In this manner, the results concur with the nature of the scenario.

Term	Estimate	Std Error	ChiSquare	Prob>ChiSq
Intercept[0]	-0.3810495	0.7347278	0.27	0.6040
100mm	0.36501304	0.3889876	0.88	0.3481
HQ9 SAM	0.09814595	0.0934978	1.10	0.2938
RGM84	-0.066368	0.0752937	0.78	0.3781
RIM-7	-0.0261908	0.0443098	0.35	0.5545
SA-N-12 SAM	-0.0367488	0.078451	0.22	0.6395
SM2	0.0074496	0.0130861	0.32	0.5692
YJ62 SSM	**0.3263633**	**0.0775864**	**17.69**	**<.0001**
YJ81 SSM	0.000424	0.0351885	0.00	0.9904
YJ83 SSM	**0.12309246**	**0.0519967**	**5.60**	**0.0179**

Table 12. Parameter Estimates for Logistic Regression (Weapons Only)

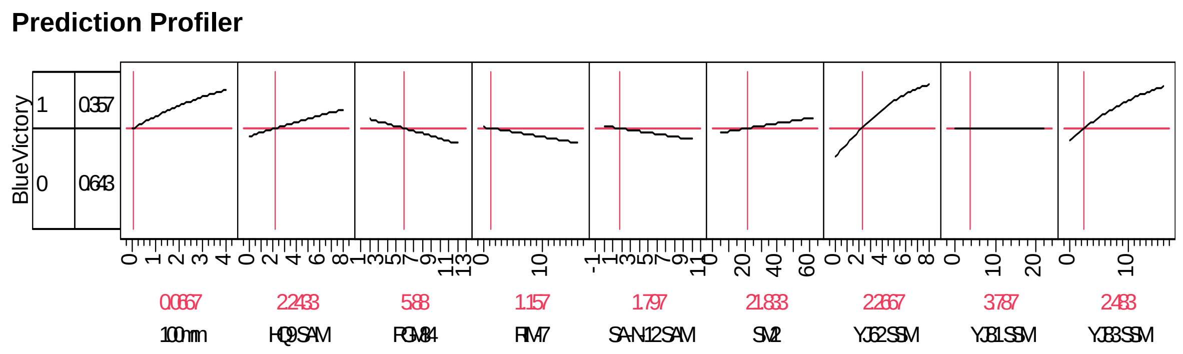


Figure 17. Prediction Profiler for Weapons Factors

The same analysis was conducted using sensors instead. This time, the KLC-1 was the only factor to show significance greater than 95%. However, a large number of KLC-1 detections strengthened the probability of a blue victory instead of weakening it, as would be expected from a red sensor. Also of interest was the Type 344 radar which showed a strong effect to decrease the probability of a blue victory, but only when the number of detections was greater than 10. The results from sensor detections do not appear to provide any further inference as to the results of each iteration. This may be due to the rational fact that sensor detections themselves cannot kill the units on either side and therefore do not show an affect with regard to the victory conditions.

Term	Estimate	Std Error	ChiSquare	Prob>ChiSq
Intercept[0]	0.62150508	1.4619724	0.18	0.6708
KLC-1	-0.0815833	0.0318417	6.56	0.0104
LAMPS	-0.0375581	0.0258144	2.12	0.1457
SPQ9B	-0.0125846	0.0675714	0.03	0.8523
SPS55	0.1187855	0.0637886	3.47	0.0626
SPS64	-0.0630176	0.4391008	0.02	0.8859
SPS67	-0.0323731	0.0431501	0.56	0.4531
Type344	0.13868825	0.4412429	0.10	0.7533
Type360	-0.0362921	0.0536663	0.46	0.4989
Type364	0.04388598	0.0421073	1.09	0.2973

Table 13. Parameter Estimates for Logistic Regression (Sensors Only)

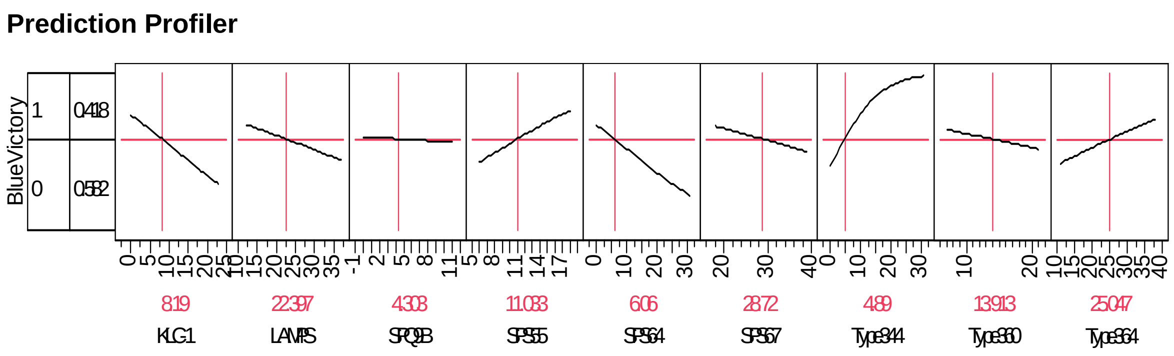


Figure 18. Prediction Profiler for Sensor Factors

3. Linear Regression Analysis

Because of the large variance demonstrated by the blue force casualty rates, a linear regression model was created to attempt to determine the main factors. Again, the regression models were split with one for sensors and one for weapons. Not surprisingly, the regression model for weapons showed all six of the red force weapons as the strongest factors, each with significance greater than 99%. Although this did not provide any greater inference into the source of blue casualties, it did demonstrate that the DAFS model was demonstrating proper causal relationships between the different units and the end results. The regression model of the weapons also demonstrated a low R-square of 0.545 indicating that there is still a great deal of variance that cannot be explained. A second regression model was created using the sensors as the independent factors but its R-square was only 0.483 and was thus discarded.

Summary of Fit

RSquare	0.545243
RSquare Adj	0.53593
Root Mean Square Error	0.157666
Mean of Response	0.575152
Observations (or Sum Wgts)	300

Analysis of Variance

Source	**DF**	**Sum of Squares**	**Mean Square**	**F Ratio**
Model	6	8.732834	1.45547	58.5499
Error	293	7.283584	0.02486	**Prob > F**
C. Total	299	16.016419		<.0001

Parameter Estimates

Term	**Estimate**	**Std Error**	**t Ratio**	**Prob>|t|**
Intercept	0.1925576	0.024638	7.82	<.0001
100mm	0.0723676	0.025687	2.82	0.0052
HQ9 SAM	0.0433007	0.006374	6.79	<.0001
SA-N-12	0.0233112	0.005577	4.18	<.0001
YJ62 SSM	0.0488867	0.004857	10.06	<.0001
YJ81 SSM	0.014318	0.002478	5.78	<.0001
YJ83 SSM	0.0296865	0.003454	8.60	<.0001

Table 14. Summary of Fit and Parameter Estimates for Linear Regression Model of Blue Casualty Rate (Weapons Only)

Similar models were created to examine the red force casualty rates but again both models displayed R-squares of less than 0.5.

4. Analysis Conclusions

The analysis conducted above provided examples of the type of statistical tools that could be utilized in conjunction with the DAFS model. Because of the unique capabilities of a stochastic simulation to show a variety of outcomes, a range of possibilities could be generated for the outcome of the scenario. It also demonstrated a rudimentary way to draw inferences from the casualty rates and provide some predictive capability to the results of the model.

Although little additional insight was gained by the regression models, they too demonstrated a manner to derive significant measurements from the results. In particular when testing a new system, it would be possible from these means to decide whether the new system showed any change in results as well as a measurement of how much change was produced. This is also a way to determine what further study is required from a cursory analysis where a more complex model may be necessary to determine the significance of a factor.

V. CONCLUSIONS

A. CONCLUSIONS

The Dynamic Allocation of Fires and Sensors model provides a suitable framework for analysis of modern naval combat. It is able to provide both a point estimate of the outcome of a given scenario as well as a means to investigate for cause and effect. The event driven simulation methodology housed in the DAFS model provides a timely turnaround of results with the additional enhancement of thousands of simulation runs in a matter of hours. The results reached in the previous chapter are merely examples of the most fundamental capabilities of the model.

The DAFS model framework provides several benefits over the current models and methodologies employed to examine modern naval warfare. The following advantages are specifically noted:

- Modular framework allowing for a plug-and-play architecture for quick analysis with little information requirements. Can build a representation of any system in question and then add it to the simulation with little modification of other components
- Event driven simulation architecture which wastes little time when no interactions occur while allocating adequate time for calculations and data requirements where they are.
- Rapid simulation with multiple stochastic runs to provide both a point estimate and an analysis of the risk and variance.
- Multiple sources of information derived from the results allowing for deeper analysis of the outcome.

B. RECOMMENDATIONS

This thesis represents a continuation of Dr. Arnold Buss and Lieutenant Michael Havens' work on the DAFS model. It demonstrates the possibility of using the DAFS model to examine aspects of maritime warfare and explores a new area of combat for the DAFS model. Continued efforts in DAFS are available in both operations research and the modeling with the conceptual framework largely in place. However, for the model's potential to be fully realized, several key enhancements must be implemented to answer the complexity of the applications needed to be explored.

Throughout the process of implementing the radar model in DAFS and modeling modern naval combat, several design choices made early on had drastic consequences when trying to utilize the model for a purpose other than ground combat. If these limitations are removed, then the model can be utilized for a greater diversity of projects and also be employed by more analytical agencies with in the military. The following recommendations are made with regards to modularity:

- A comprehensive scrubbing of the platform organization be conducted to allow for a more modular build. Wargaming experts could assist in this measure by providing insight into what elements are required and how they should be implemented
- A removal of Army specific code in the base source code (low resolution ACQUIRE algorithm) to be replaced with a modular installment fulfilling the same function.
- Allow for munition detection and engagement by weapons.

A second recommendation for the improvement of DAFS would be to simplify the scenario library and Graphic User Interface (GUI) utilization. As the model is currently constructed, there are several confusing connections in the scenario library that make it somewhat difficult to build a scenario because of the hidden connections between files. By incorporating a more modular design of the whole model, much of this can be alleviated. The GUI in the graphic simulation itself could also be improved. The following recommendations are made with regards to scenario library and GUI:

- Re-write the scenario library structure so that it is more modular allowing for a more plug-and-play approach to building platforms and missions.
- Improve the GUI to allow for importing of pre-designed maps and smaller scale areas.
- Improve the GUI to incorporate a variable distance measurement so that the scenario itself defines the scale of distance between points.
- Incorporate a symbol library to allow for easier recognition of the simulated units and better display to more senior audiences.
- Scale back information presented when selecting a simulation unit in information mode.

Lastly, for future usage by other military analysts, the DAFS model should incorporate a standing library of platforms and sensors to allow for quick turnaround analysis. These libraries could be created as both a classified version and an unclassified version based on the sources of the values held within. It would also be useful to have more control on what values are outputted by the model so that analysts may concentrate on those measurements they deem significant. The following recommendations are made with regard to future usage:

- Create standing libraries of platforms and scenarios to be used by analysts. They should demonstrate all forms of combat and various missions to allow for a baseline for less experienced analysts to build from.
- Provide more control as to the measurements produced in DAFS. It should be a separate section within a scenario file that delineates what measurements are to be taken from the simulation.

C. FOLLOW ON RESEARCH

Besides the continuing improvements to the DAFS model, there are several avenues for follow on research. Because this thesis merely skimmed the surface of examining naval combat, any further analysis in either weapons capabilities or tactical improvements would make an excellent continuation of the DAFS studies. A few more specific areas of follow on research are listed below:

- A more thorough analysis of a vignette or scenario examined by OPNAV N-81 would provide an excellent validation of the DAFS model as a naval simulation.
- Examination of the LCS weapons systems as applies to unit and group defense
- Implementation of Defense Counter Air (DCA) air craft and Air Tasking Order (ATO) flight scheduling
- Expansion of the factors in the CVO to account for greater complexity of combat fires decisions
- Expansion of a new CVO to determine command priorities so that platforms could shift from one role to another.

APPENDIX A: SAMPLE DAFS XML FILES

The following are portions of sample files used with DAFS and are representative of the full files used.

1. BASE SCENERIO FILE

```
<DAFSScenario version="2" type="Attack" bdaFactor="1" replications="1"
stopTime="400.0">
    <!--
This file was generated from jdbc:odbc:Driver={Microsoft Access Driver
(*.mdb)};DBQ=C:\Documents and Settings\Scott Hattaway\My
Documents\School Stuff\Thesis
Stuff\trunk.r1508\scenarios\NavalCombatScenerio.mdb
      at Tue Mar 04 20:44:05 PST 2008
      ProcessDBInput Version: $Id: ProcessDBInput.java 1508 2008-03-03
19:07:18Z ahbuss $
      DAFS Version: 1.0.0
-->
    <SimEntity>
        <Mover class="dafs.platform.Platform" qty="1" type="DDG51Flt1"
affiliation="Blue" assignment="combined" name="STETHEM">
            <Altitude>1</Altitude>
            <MaxSpeed>988.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Grid xLoc="-20000.0" yLoc="-35000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true" />
            <Sensor id="18" type="SPY1D"
class="dafs.sensor.DAFSRadarSensor" maxRange="272700.0" prf="3" />
            <Sensor id="20" type="SPS64" class="dafs.sensor.DAFSSensor"
maxRange="21946.0" />
            <Munitions>
                <Munition type="RGM84" qty="8" />
                <Munition type="SM2" qty="60" />
                <Munition type="5InchCVT" qty="60" />
                <Munition type="20mm CIWS" qty="8" />
                <Munition type="5InchCVT" qty="60" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="DDG51Flt2A"
affiliation="Blue" assignment="combined" name="LASSEN">
            <Altitude>1</Altitude>
            <MaxSpeed>957.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Grid xLoc="25000.0" yLoc="-25000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true" />
            <Sensor id="15" type="SPY1D"
class="dafs.sensor.DAFSRadarSensor" maxRange="272700.0" prf="3" />
            <Sensor id="21" type="SPS64" class="dafs.sensor.DAFSSensor"
maxRange="21946.0" />
            <Sensor id="48" type="SPS67"
class="dafs.sensor.DAFSRadarSensor" maxRange="93000.0" prf="60" />
            <Munitions>
                <Munition type="SM2" qty="60" />
                <Munition type="5InchCVT" qty="60" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="DDG51Flt2A"
affiliation="Blue" assignment="combined" name="MUSTIN">
            <Altitude>1</Altitude>
            <MaxSpeed>957.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Grid xLoc="3000.0" yLoc="0.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPatrolMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="24000" yLoc="21000" Speed="1000"
mover="MUSTIN" />
                <Waypoint xLoc="-18000" yLoc="-21000" Speed="500"
mover="MUSTIN" />
            </MoverManager>
            <Sensor id="16" type="SPY1D"
class="dafs.sensor.DAFSRadarSensor" maxRange="272700.0" prf="3" />
            <Sensor id="19" type="SPS64" class="dafs.sensor.DAFSSensor"
maxRange="21946.0" />
            <Sensor id="49" type="SPS67"
class="dafs.sensor.DAFSRadarSensor" maxRange="93000.0" prf="60" />
            <Sensor id="50" type="SPS67"
class="dafs.sensor.DAFSRadarSensor" maxRange="93000.0" prf="60" />
            <Munitions>
                <Munition type="SM2" qty="60" />
                <Munition type="5InchCVT" qty="60" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="CG47"
affiliation="Blue" assignment="combined" name="SHILOH">
            <Altitude>1</Altitude>
            <MaxSpeed>926.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Grid xLoc="-40000.0" yLoc="0.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPatrolMoverManager"
delay="0.0" startOnReset="true" />
            <Sensor id="17" type="SPY1B"
class="dafs.sensor.DAFSRadarSensor" maxRange="272700.0" prf="3" />
            <Sensor id="23" type="SPQ9B" class="dafs.sensor.DAFSSensor"
maxRange="30000.0" />
```

```
            <Sensor id="45" type="SPS55"
class="dafs.sensor.DAFSRadarSensor" maxRange="93000.0" prf="30" />
            <Sensor id="46" type="SPS64" class="dafs.sensor.DAFSSensor"
maxRange="21946.0" />
            <Munitions>
                <Munition type="RGM84" qty="8" />
                <Munition type="5InchCVT" qty="60" />
                <Munition type="SM2" qty="60" />
                <Munition type="20mm CIWS" qty="8" />
                <Munition type="20mm CIWS" qty="8" />
                <Munition type="5InchCVT" qty="60" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="CG47"
affiliation="Blue" assignment="combined" name="COWPENS">
            <Altitude>1</Altitude>
            <MaxSpeed>926.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Grid xLoc="30000.0" yLoc="30000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true" />
            <Sensor id="13" type="SPY1B"
class="dafs.sensor.DAFSRadarSensor" maxRange="272700.0" prf="3" />
            <Sensor id="22" type="SPQ9B" class="dafs.sensor.DAFSSensor"
maxRange="30000.0" />
            <Sensor id="43" type="SPS64" class="dafs.sensor.DAFSSensor"
maxRange="21946.0" />
            <Sensor id="44" type="SPS55"
class="dafs.sensor.DAFSRadarSensor" maxRange="93000.0" prf="30" />
            <Munitions>
                <Munition type="RGM84" qty="8" />
                <Munition type="SM2" qty="60" />
                <Munition type="5InchCVT" qty="60" />
                <Munition type="5InchCVT" qty="60" />
                <Munition type="20mm CIWS" qty="8" />
                <Munition type="20mm CIWS" qty="8" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="CV"
affiliation="Blue" assignment="sensor" name="KITTYHAWK">
            <Altitude>1</Altitude>
            <MaxSpeed>988.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Grid xLoc="0.0" yLoc="0.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPatrolMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="21000" yLoc="21000" Speed="1000"
mover="KITTYHAWK" />
                <Waypoint xLoc="-21000" yLoc="-21000" Speed="500"
mover="KITTYHAWK" />
            </MoverManager>
            <Sensor id="12" type="SPS67"
class="dafs.sensor.DAFSRadarSensor" maxRange="93000.0" prf="60" />
            <Sensor id="42" type="SPS48E"
class="dafs.sensor.DAFSRadarSensor" maxRange="230000.0" prf="8" />
            <Munitions>
                <Munition type="RIM-7" qty="8" />
                <Munition type="RIM-7" qty="8" />
                <Munition type="20mm CIWS" qty="8" />
                <Munition type="20mm CIWS" qty="8" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="Type052B"
affiliation="Red" assignment="combined" name="LUYANG">
            <Altitude>1</Altitude>
            <MaxSpeed>895.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Box ID="5" mover="LUYANG" minX="-200000.0"
minY="200000.0" maxX="-10000.0" maxY="205000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="0" yLoc="0" Speed="900" mover="LUYANG"
/>
            </MoverManager>
            <Sensor id="32" type="Type344"
class="dafs.sensor.DAFSSensor" maxRange="21946.0" />
            <Sensor id="33" type="Top Plate"
class="dafs.sensor.DAFSRadarSensor" maxRange="230000.0" prf="15" />
            <Sensor id="58" type="Type364"
class="dafs.sensor.DAFSRadarSensor" maxRange="100000.0" prf="1" />
            <Munitions>
                <Munition type="100mm" qty="50" />
                <Munition type="30mm CIWS" qty="8" />
                <Munition type="30mm CIWS" qty="8" />
                <Munition type="SA-N-12 SAM" qty="48" />
                <Munition type="YJ83 SSM" qty="8" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="Type052B"
affiliation="Red" assignment="combined" name="GUANGZHOU">
            <Altitude>1</Altitude>
            <MaxSpeed>895.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Box ID="8" mover="GUANGZHOU" minX="200000.0" minY="-
200000.0" maxX="-200000.0" maxY="-205000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="0" yLoc="0" Speed="900"
mover="GUANGZHOU" />
            </MoverManager>
            <Sensor id="28" type="Top Plate"
class="dafs.sensor.DAFSRadarSensor" maxRange="230000.0" prf="15" />
```

```
            <Sensor id="29" type="Type344"
class="dafs.sensor.DAFSSensor" maxRange="21946.0" />
            <Sensor id="57" type="Type364"
class="dafs.sensor.DAFSRadarSensor" maxRange="100000.0" prf="1" />
            <Munitions>
                <Munition type="100mm" qty="50" />
                <Munition type="30mm CIWS" qty="8" />
                <Munition type="30mm CIWS" qty="8" />
                <Munition type="SA-N-12 SAM" qty="48" />
                <Munition type="YJ83 SSM" qty="8" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="Type052B"
affiliation="Red" assignment="combined" name="WUHAN">
            <Altitude>1</Altitude>
            <MaxSpeed>895.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Box ID="10" mover="WUHAN" minX="-200000.0" minY="-
200000.0" maxX="-205000.0" maxY="200000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="0" yLoc="0" Speed="900" mover="WUHAN" />
            </MoverManager>
            <Sensor id="30" type="Type344"
class="dafs.sensor.DAFSSensor" maxRange="21946.0" />
            <Sensor id="31" type="Top Plate"
class="dafs.sensor.DAFSRadarSensor" maxRange="230000.0" prf="15" />
            <Sensor id="56" type="Type364"
class="dafs.sensor.DAFSRadarSensor" maxRange="100000.0" prf="1" />
            <Munitions>
                <Munition type="100mm" qty="50" />
                <Munition type="30mm CIWS" qty="8" />
                <Munition type="30mm CIWS" qty="8" />
                <Munition type="SA-N-12 SAM" qty="48" />
                <Munition type="YJ83 SSM" qty="8" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="Type052C"
affiliation="Red" assignment="combined" name="HAIKOU">
            <Altitude>1</Altitude>
            <MaxSpeed>895.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Box ID="7" mover="HAIKOU" minX="200000.0" minY="-
200000.0" maxX="-200000.0" maxY="-205000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="0" yLoc="0" Speed="900" mover="HAIKOU"
/>
            </MoverManager>
            <Sensor id="24" type="Type344"
class="dafs.sensor.DAFSSensor" maxRange="21946.0" />
```

```
            <Sensor id="25" type="Type348"
class="dafs.sensor.DAFSRadarSensor" maxRange="272700.0" prf="3" />
            <Sensor id="51" type="Type364"
class="dafs.sensor.DAFSRadarSensor" maxRange="100000.0" prf="1" />
            <Munitions>
                <Munition type="HQ9 SAM" qty="24" />
                <Munition type="100mm" qty="50" />
                <Munition type="YJ62 SSM" qty="8" />
                <Munition type="30mm CIWS" qty="8" />
                <Munition type="30mm CIWS" qty="8" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="Type052C"
affiliation="Red" assignment="combined" name="SUIHUA">
            <Altitude>1</Altitude>
            <MaxSpeed>895.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Box ID="9" mover="SUIHUA" minX="-200000.0" minY="-
200000.0" maxX="-205000.0" maxY="200000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="0" yLoc="0" Speed="900" mover="SUIHUA"
/>
            </MoverManager>
            <Sensor id="53" type="Type364"
class="dafs.sensor.DAFSRadarSensor" maxRange="100000.0" prf="1" />
            <Sensor id="54" type="Type348"
class="dafs.sensor.DAFSRadarSensor" maxRange="272700.0" prf="3" />
            <Sensor id="55" type="Type344"
class="dafs.sensor.DAFSSensor" maxRange="21946.0" />
            <Munitions>
                <Munition type="HQ9 SAM" qty="24" />
                <Munition type="100mm" qty="50" />
                <Munition type="YJ62 SSM" qty="8" />
                <Munition type="30mm CIWS" qty="8" />
                <Munition type="30mm CIWS" qty="8" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="Type053H3"
affiliation="Red" assignment="combined" name="JIAXING">
            <Altitude>1</Altitude>
            <MaxSpeed>988.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Box ID="11" mover="JIAXING" minX="-200000.0"
minY="200000.0" maxX="-10000.0" maxY="205000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="0" yLoc="0" Speed="900" mover="JIAXING"
/>
            </MoverManager>
            <Sensor id="34" type="Type344"
class="dafs.sensor.DAFSSensor" maxRange="21946.0" />
            <Sensor id="61" type="Type364"
class="dafs.sensor.DAFSRadarSensor" maxRange="100000.0" prf="1" />
            <Munitions>
                <Munition type="100mm" qty="50" />
                <Munition type="HQ7 SAM" qty="8" />
                <Munition type="YJ81 SSM" qty="16" />
                <Munition type="100mm" qty="50" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="Type053H3"
affiliation="Red" assignment="combined" name="PUTIAN">
            <Altitude>1</Altitude>
            <MaxSpeed>988.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Box ID="12" mover="PUTIAN" minX="200000.0" minY="-
200000.0" maxX="-200000.0" maxY="-205000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="0" yLoc="0" Speed="900" mover="PUTIAN"
/>
            </MoverManager>
            <Sensor id="35" type="Type344"
class="dafs.sensor.DAFSSensor" maxRange="21946.0" />
            <Sensor id="60" type="Type364"
class="dafs.sensor.DAFSRadarSensor" maxRange="100000.0" prf="1" />
            <Munitions>
                <Munition type="100mm" qty="50" />
                <Munition type="HQ7 SAM" qty="8" />
                <Munition type="YJ81 SSM" qty="16" />
                <Munition type="100mm" qty="50" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="Type053H3"
affiliation="Red" assignment="combined" name="YICHANG">
            <Altitude>1</Altitude>
            <MaxSpeed>988.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Box ID="13" mover="YICHANG" minX="-200000.0" minY="-
200000.0" maxX="-205000.0" maxY="200000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="0" yLoc="0" Speed="900" mover="YICHANG"
/>
            </MoverManager>
            <Sensor id="36" type="Type344"
class="dafs.sensor.DAFSSensor" maxRange="21946.0" />
            <Sensor id="59" type="Type364"
class="dafs.sensor.DAFSRadarSensor" maxRange="100000.0" prf="1" />
            <Munitions>
```

```
                <Munition type="100mm" qty="50" />
                <Munition type="HQ7 SAM" qty="8" />
                <Munition type="YJ81 SSM" qty="16" />
                <Munition type="100mm" qty="50" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="SH60R"
affiliation="Blue" assignment="sensor" name="Warlord05">
            <Altitude>1000</Altitude>
            <MaxSpeed>3000.0</MaxSpeed>
            <CrossSection>10</CrossSection>
            <OperationalEndurance>0.0</OperationalEndurance>
            <Position>
                <Grid xLoc="30000.0" yLoc="30000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPatrolMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="0" yLoc="70000" Speed="2000"
mover="Warlord05" />
                <Waypoint xLoc="-50000" yLoc="50000" Speed="2000"
mover="Warlord05" />
                <Waypoint xLoc="-70000" yLoc="0" Speed="2000"
mover="Warlord05" />
                <Waypoint xLoc="-50000" yLoc="-50000" Speed="2000"
mover="Warlord05" />
                <Waypoint xLoc="0" yLoc="-70000" Speed="2000"
mover="Warlord05" />
                <Waypoint xLoc="50000" yLoc="-50000" Speed="2000"
mover="Warlord05" />
                <Waypoint xLoc="70000" yLoc="0" Speed="2000"
mover="Warlord05" />
                <Waypoint xLoc="50000" yLoc="50000" Speed="2000"
mover="Warlord05" />
            </MoverManager>
            <Sensor id="37" type="LAMPS" class="dafs.sensor.DAFSSensor"
maxRange="60000.0" />
            <Munitions />
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="SH60R"
affiliation="Blue" assignment="sensor" name="Warlord15">
            <Altitude>1000</Altitude>
            <MaxSpeed>3000.0</MaxSpeed>
            <CrossSection>10</CrossSection>
            <OperationalEndurance>0.0</OperationalEndurance>
            <Position>
                <Grid xLoc="25000.0" yLoc="-25000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPatrolMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="70000" yLoc="0" Speed="2000"
mover="Warlord15" />
                <Waypoint xLoc="50000" yLoc="50000" Speed="2000"
mover="Warlord15" />
                <Waypoint xLoc="0" yLoc="70000" Speed="2000"
mover="Warlord15" />
```

```
                <Waypoint xLoc="-50000" yLoc="50000" Speed="2000"
mover="Warlord15" />
                <Waypoint xLoc="-70000" yLoc="0" Speed="2000"
mover="Warlord15" />
                <Waypoint xLoc="-50000" yLoc="-50000" Speed="2000"
mover="Warlord15" />
                <Waypoint xLoc="0" yLoc="-70000" Speed="2000"
mover="Warlord15" />
                <Waypoint xLoc="50000" yLoc="-50000" Speed="2000"
mover="Warlord15" />
            </MoverManager>
            <Sensor id="38" type="LAMPS" class="dafs.sensor.DAFSSensor"
maxRange="60000.0" />
            <Munitions />
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="SH60R"
affiliation="Blue" assignment="sensor" name="Warlord25">
            <Altitude>1000</Altitude>
            <MaxSpeed>3000.0</MaxSpeed>
            <CrossSection>10</CrossSection>
            <OperationalEndurance>0.0</OperationalEndurance>
            <Position>
                <Grid xLoc="3000.0" yLoc="0.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPatrolMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="-70000" yLoc="0" Speed="2300"
mover="Warlord25" />
                <Waypoint xLoc="-50000" yLoc="-50000" Speed="2000"
mover="Warlord25" />
                <Waypoint xLoc="0" yLoc="-70000" Speed="2000"
mover="Warlord25" />
                <Waypoint xLoc="50000" yLoc="-50000" Speed="2000"
mover="Warlord25" />
                <Waypoint xLoc="70000" yLoc="0" Speed="2000"
mover="Warlord25" />
                <Waypoint xLoc="50000" yLoc="50000" Speed="2000"
mover="Warlord25" />
                <Waypoint xLoc="0" yLoc="70000" Speed="2000"
mover="Warlord25" />
                <Waypoint xLoc="-50000" yLoc="50000" Speed="2000"
mover="Warlord25" />
            </MoverManager>
            <Sensor id="39" type="LAMPS" class="dafs.sensor.DAFSSensor"
maxRange="60000.0" />
            <Munitions />
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="SH60R"
affiliation="Blue" assignment="sensor" name="Warlord35">
            <Altitude>1000</Altitude>
            <MaxSpeed>3000.0</MaxSpeed>
            <CrossSection>10</CrossSection>
            <OperationalEndurance>0.0</OperationalEndurance>
            <Position>
                <Grid xLoc="-40000.0" yLoc="0.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPatrolMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="0" yLoc="-70000" Speed="2000"
mover="Warlord35" />
                <Waypoint xLoc="50000" yLoc="-50000" Speed="2000"
mover="Warlord35" />
                <Waypoint xLoc="70000" yLoc="0" Speed="2000"
mover="Warlord35" />
                <Waypoint xLoc="50000" yLoc="50000" Speed="2000"
mover="Warlord35" />
                <Waypoint xLoc="0" yLoc="70000" Speed="2000"
mover="Warlord35" />
                <Waypoint xLoc="-50000" yLoc="50000" Speed="2000"
mover="Warlord35" />
                <Waypoint xLoc="-70000" yLoc="0" Speed="2000"
mover="Warlord35" />
                <Waypoint xLoc="-50000" yLoc="-50000" Speed="2000"
mover="Warlord35" />
            </MoverManager>
            <Sensor id="40" type="LAMPS" class="dafs.sensor.DAFSSensor"
maxRange="60000.0" />
            <Munitions />
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="DDG51Flt1"
affiliation="Blue" assignment="combined" name="FITZGERALD">
            <Altitude>1</Altitude>
            <MaxSpeed>988.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
                <Grid xLoc="-20000.0" yLoc="35000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true" />
            <Sensor id="1" type="SPS64" class="dafs.sensor.DAFSSensor"
maxRange="21946.0" />
            <Sensor id="14" type="SPY1D"
class="dafs.sensor.DAFSRadarSensor" maxRange="272700.0" prf="3" />
            <Sensor id="47" type="SPS67"
class="dafs.sensor.DAFSRadarSensor" maxRange="93000.0" prf="60" />
            <Munitions>
                <Munition type="20mm CIWS" qty="8" />
                <Munition type="20mm CIWS" qty="8" />
                <Munition type="RGM84" qty="8" />
                <Munition type="SM2" qty="60" />
                <Munition type="5InchCVT" qty="60" />
            </Munitions>
        </Mover>
        <Mover class="dafs.platform.Platform" qty="1" type="Type052C"
affiliation="Red" assignment="combined" name="LANZHOU">
            <Altitude>1</Altitude>
            <MaxSpeed>895.0</MaxSpeed>
            <CrossSection>0</CrossSection>
            <Position>
```

```
                <Box ID="6" mover="LANZHOU" minX="-200000.0"
minY="200000.0" maxX="-10000.0" maxY="205000.0" />
            </Position>
            <MoverManager class="dafs.platform.DAFSPathMoverManager"
delay="0.0" startOnReset="true">
                <Waypoint xLoc="0" yLoc="0" Speed="900" mover="LANZHOU"
/>
            </MoverManager>
            <Sensor id="2" type="Type344" class="dafs.sensor.DAFSSensor"
maxRange="21946.0" />
            <Sensor id="26" type="Type348"
class="dafs.sensor.DAFSRadarSensor" maxRange="272700.0" prf="3" />
            <Sensor id="52" type="Type364"
class="dafs.sensor.DAFSRadarSensor" maxRange="100000.0" prf="1" />
            <Munitions>
                <Munition type="YJ62 SSM" qty="8" />
                <Munition type="30mm CIWS" qty="8" />
                <Munition type="30mm CIWS" qty="8" />
                <Munition type="HQ9 SAM" qty="24" />
                <Munition type="100mm" qty="50" />
            </Munitions>
        </Mover>
        <CVO class="dafs.command.CVO2" type="fires" name="Fires CVO">
            <Property name="maxAssign" value="10"
class="java.lang.Integer" />
            <Property name="maxCover" value="10"
class="java.lang.Integer" />
            <Property name="minCover" value="0"
class="java.lang.Integer" />
            <Property name="implementationInterval" value="1.1"
class="java.lang.Double" />
            <Property name="print" value="true"
class="java.lang.Boolean" />
            <VPA class="dafs.command.OutsideRangeVPA">
                <Property name="minPK" value="0"
class="java.lang.Double" />
                <Property name="maxThreatPK" value="0.8"
class="java.lang.Double" />
            </VPA>
        </CVO>
        <CVO class="dafs.command.CVOforBDA" type="sensor" name="BDA
CVO">
            <Property name="maxAssign" value="100"
class="java.lang.Integer" />
            <Property name="maxCover" value="4"
class="java.lang.Integer" />
            <Property name="minCover" value="0"
class="java.lang.Integer" />
            <Property name="implementationInterval" value="1.2"
class="java.lang.Double" />
            <Property name="print" value="false"
class="java.lang.Boolean" />
            <VPA class="dafs.command.VPAforBDA" />
        </CVO>
        <Listener source="Fires CVO" listener="BDA CVO" />
        <Mediator sensorClass="dafs.sensor.DAFSSensor"
targetClass="dafs.platform.Platform"
mediatorClass="dafs.sensor.DAFSCookieCutterMediator" />
        <Mediator sensorClass="dafs.sensor.DAFSRadarSensor"
targetClass="dafs.platform.Platform"
mediatorClass="dafs.sensor.DAFSRadarMediator" />
        <Mediator sensorClass="dafs.sensor.DAFSRadarSensor"
targetClass="dafs.weapon.DAFSCircularImpactMunition"
mediatorClass="dafs.sensor.DAFSCookieCutterMediator" />
    </SimEntity>
    <MunitionsTypes>
        <MunitionType>
            <MUNITION>RGM84</MUNITION>
            <WEIGHT>222.0</WEIGHT>
            <MER>1.0</MER>
            <MINRANGE>2000.0</MINRANGE>
            <MAXRANGE>146304.0</MAXRANGE>
            <LOAD>8.0</LOAD>
            <SPEED>16915.0</SPEED>
            <ALGORITHM>OLD_DAFS</ALGORITHM>
            <BURST_SIZE>1</BURST_SIZE>
            <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
            <Detectable>true</Detectable>
            <CrossSection>5.0</CrossSection>
        </MunitionType>
        <MunitionType>
            <MUNITION>SM2</MUNITION>
            <WEIGHT>90.0</WEIGHT>
            <MER>1.0</MER>
            <MINRANGE>500.0</MINRANGE>
            <MAXRANGE>180000.0</MAXRANGE>
            <LOAD>60.0</LOAD>
            <SPEED>49750.0</SPEED>
            <ALGORITHM>OLD_DAFS</ALGORITHM>
            <BURST_SIZE>1</BURST_SIZE>
            <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
            <Detectable>true</Detectable>
            <CrossSection>10.0</CrossSection>
        </MunitionType>
        <MunitionType>
            <MUNITION>20mm CIWS</MUNITION>
            <WEIGHT>50.0</WEIGHT>
            <MER>1.0</MER>
            <MINRANGE>5.0</MINRANGE>
            <MAXRANGE>2000.0</MAXRANGE>
            <LOAD>8.0</LOAD>
            <SPEED>13000.0</SPEED>
            <ALGORITHM>OLD_DAFS</ALGORITHM>
            <BURST_SIZE>1</BURST_SIZE>
            <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
            <Detectable>false</Detectable>
        </MunitionType>
        <MunitionType>
            <MUNITION>5InchCVT</MUNITION>
            <WEIGHT>75.0</WEIGHT>
        <MER>1.0</MER>
        <MINRANGE>500.0</MINRANGE>
        <MAXRANGE>18200.0</MAXRANGE>
        <LOAD>60.0</LOAD>
        <SPEED>13000.0</SPEED>
        <ALGORITHM>OLD_DAFS</ALGORITHM>
        <BURST_SIZE>1</BURST_SIZE>
        <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
        <Detectable>false</Detectable>
    </MunitionType>
    <MunitionType>
        <MUNITION>YJ62 SSM</MUNITION>
        <WEIGHT>300.0</WEIGHT>
        <MER>1.0</MER>
        <MINRANGE>2000.0</MINRANGE>
        <MAXRANGE>300000.0</MAXRANGE>
        <LOAD>8.0</LOAD>
        <SPEED>17910.0</SPEED>
        <ALGORITHM>OLD_DAFS</ALGORITHM>
        <BURST_SIZE>1</BURST_SIZE>
        <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
        <Detectable>true</Detectable>
        <CrossSection>10.0</CrossSection>
    </MunitionType>
    <MunitionType>
        <MUNITION>HQ9 SAM</MUNITION>
        <WEIGHT>200.0</WEIGHT>
        <MER>1.0</MER>
        <MINRANGE>500.0</MINRANGE>
        <MAXRANGE>90000.0</MAXRANGE>
        <LOAD>24.0</LOAD>
        <SPEED>79600.0</SPEED>
        <ALGORITHM>OLD_DAFS</ALGORITHM>
        <BURST_SIZE>1</BURST_SIZE>
        <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
        <Detectable>false</Detectable>
        <CrossSection>0.0</CrossSection>
    </MunitionType>
    <MunitionType>
        <MUNITION>30mm CIWS</MUNITION>
        <WEIGHT>50.0</WEIGHT>
        <MER>1.0</MER>
        <MINRANGE>5.0</MINRANGE>
        <MAXRANGE>2000.0</MAXRANGE>
        <LOAD>8.0</LOAD>
        <SPEED>13000.0</SPEED>
        <ALGORITHM>OLD_DAFS</ALGORITHM>
        <BURST_SIZE>1</BURST_SIZE>
        <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
        <Detectable>false</Detectable>
    </MunitionType>
    <MunitionType>
        <MUNITION>100mm</MUNITION>
        <WEIGHT>75.0</WEIGHT>
        <MER>1.0</MER>
        <MINRANGE>500.0</MINRANGE>
        <MAXRANGE>18200.0</MAXRANGE>
        <LOAD>50.0</LOAD>
        <SPEED>13000.0</SPEED>
        <ALGORITHM>OLD_DAFS</ALGORITHM>
        <BURST_SIZE>1</BURST_SIZE>
        <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
        <Detectable>false</Detectable>
    </MunitionType>
    <MunitionType>
        <MUNITION>RIM-7</MUNITION>
        <WEIGHT>39.0</WEIGHT>
        <MER>1.0</MER>
        <MINRANGE>500.0</MINRANGE>
        <MAXRANGE>16000.0</MAXRANGE>
        <LOAD>8.0</LOAD>
        <SPEED>49750.0</SPEED>
        <ALGORITHM>OLD_DAFS</ALGORITHM>
        <BURST_SIZE>1</BURST_SIZE>
        <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
        <Detectable>false</Detectable>
        <CrossSection>0.0</CrossSection>
    </MunitionType>
    <MunitionType>
        <MUNITION>SA-N-12 SAM</MUNITION>
        <WEIGHT>200.0</WEIGHT>
        <MER>1.0</MER>
        <MINRANGE>500.0</MINRANGE>
        <MAXRANGE>50000.0</MAXRANGE>
        <LOAD>48.0</LOAD>
        <SPEED>59700.0</SPEED>
        <ALGORITHM>OLD_DAFS</ALGORITHM>
        <BURST_SIZE>1</BURST_SIZE>
        <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
        <Detectable>false</Detectable>
        <CrossSection>0.0</CrossSection>
    </MunitionType>
    <MunitionType>
        <MUNITION>YJ81 SSM</MUNITION>
        <WEIGHT>165.0</WEIGHT>
        <MER>1.0</MER>
        <MINRANGE>2000.0</MINRANGE>
        <MAXRANGE>80000.0</MAXRANGE>
        <LOAD>16.0</LOAD>
        <SPEED>17910.0</SPEED>
        <ALGORITHM>OLD_DAFS</ALGORITHM>
        <BURST_SIZE>1</BURST_SIZE>
        <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
        <Detectable>true</Detectable>
        <CrossSection>10.0</CrossSection>
    </MunitionType>
    <MunitionType>
        <MUNITION>YJ83 SSM</MUNITION>
        <WEIGHT>165.0</WEIGHT>
        <MER>1.0</MER>
```

```
            <MINRANGE>2000.0</MINRANGE>
            <MAXRANGE>250000.0</MAXRANGE>
            <LOAD>8.0</LOAD>
            <SPEED>25870.0</SPEED>
            <ALGORITHM>OLD_DAFS</ALGORITHM>
            <BURST_SIZE>1</BURST_SIZE>
            <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
            <Detectable>true</Detectable>
            <CrossSection>10.0</CrossSection>
        </MunitionType>
        <MunitionType>
            <MUNITION>HQ7 SAM</MUNITION>
            <WEIGHT>200.0</WEIGHT>
            <MER>1.0</MER>
            <MINRANGE>5.0</MINRANGE>
            <MAXRANGE>10000.0</MAXRANGE>
            <LOAD>8.0</LOAD>
            <SPEED>45770.0</SPEED>
            <ALGORITHM>OLD_DAFS</ALGORITHM>
            <BURST_SIZE>1</BURST_SIZE>
            <SUBMUNITION_COUNT>1</SUBMUNITION_COUNT>
            <Detectable>false</Detectable>
            <CrossSection>0.0</CrossSection>
        </MunitionType>
    </MunitionsTypes>
    <SensorTypes>
        <SensorType name="SPY1D" maxRange="272700.0" prf="3"
class="dafs.sensor.DAFSRadarSensor" />
        <SensorType name="Type348" maxRange="272700.0" prf="3"
class="dafs.sensor.DAFSRadarSensor" />
        <SensorType name="SPS64" maxRange="21946.0" prf="30"
class="dafs.sensor.DAFSSensor" />
        <SensorType name="Type344" maxRange="21946.0" prf="30"
class="dafs.sensor.DAFSSensor" />
        <SensorType name="LAMPS" maxRange="60000.0" prf="60"
class="dafs.sensor.DAFSSensor" />
        <SensorType name="SPQ9B" maxRange="30000.0" prf="60"
class="dafs.sensor.DAFSSensor" />
        <SensorType name="Top Plate" maxRange="230000.0" prf="15"
class="dafs.sensor.DAFSRadarSensor" />
        <SensorType name="SPS48E" maxRange="230000.0" prf="8"
class="dafs.sensor.DAFSRadarSensor" />
        <SensorType name="SPS67" maxRange="93000.0" prf="60"
class="dafs.sensor.DAFSRadarSensor" />
        <SensorType name="SPY1B" maxRange="272700.0" prf="3"
class="dafs.sensor.DAFSRadarSensor" />
        <SensorType name="Type364" maxRange="100000.0" prf="1"
class="dafs.sensor.DAFSRadarSensor" />
        <SensorType name="SPS55" maxRange="93000.0" prf="30"
class="dafs.sensor.DAFSRadarSensor" />
    </SensorTypes>
    <DamageDataHolder />
</DAFSScenario>
```

2. PLATFORM VALUES

```
<PlatformValues>
        <ScenarioValues scenarioType="Attack">
            <Value platformType="YJ82 SSM">3000.0</Value>
            <Value platformType="YJ81 SSM">3000.0</Value>
            <Value platformType="DDG51Flt1">8950.0</Value>
            <Value platformType="Type052C">7000.0</Value>
            <Value platformType="RGM84">3000.0</Value>
            <Value platformType="YJ62 SSM">3000.0</Value>
            <Value platformType="SM2">2000.0</Value>
            <Value platformType="HQ9SAM">2000.0</Value>
            <Value platformType="SH60R">1000.0</Value>
            <Value platformType="Type052B">7000.0</Value>
            <Value platformType="Type053H3">3250.0</Value>
            <Value platformType="CG47">9957.0</Value>
            <Value platformType="CV">83960.0</Value>
            <Value platformType="DDG51Flt2A">9188.0</Value>
            <Value platformType="Zhi-9 Helo">1000.0</Value>
        </ScenarioValues>
    </PlatformValues>
```

3. KILL PROBABILITIES

```
    <KillProbabilities>
        <KillProbability munitionType="100mm" platformType="DDG51Flt1"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.01" maxRangePK="0.0050"
minRange="500.0" maxRange="18000.0" />
        </KillProbability>
        <KillProbability munitionType="20mm CIWS" platformType="YJ62
SSM" class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.4" maxRangePK="0.3" minRange="5.0"
maxRange="2000.0" />
        </KillProbability>
        <KillProbability munitionType="30mm CIWS" platformType="RGM84"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.4" maxRangePK="0.3" minRange="5.0"
maxRange="2000.0" />
        </KillProbability>
        <KillProbability munitionType="5InchCVT" platformType="Type052C"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.01" maxRangePK="0.0050"
minRange="500.0" maxRange="18000.0" />
        </KillProbability>
        <KillProbability munitionType="HQ9 SAM" platformType="RGM84"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.6" maxRangePK="0.6" minRange="500.0"
maxRange="90000.0" />
        </KillProbability>
        <KillProbability munitionType="RGM84" platformType="Type052C"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.8" maxRangePK="0.6" minRange="2000.0"
maxRange="146304.0" />
        </KillProbability>
        <KillProbability munitionType="SM2" platformType="YJ62 SSM"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.7" maxRangePK="0.6" minRange="500.0"
maxRange="180000.0" />
        </KillProbability>
        <KillProbability munitionType="YJ62 SSM"
platformType="DDG51Flt1" class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.8" maxRangePK="0.6" minRange="2000.0"
maxRange="300000.0" />
        </KillProbability>
        <KillProbability munitionType="20mm CIWS" platformType="YJ81
SSM" class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.4" maxRangePK="0.3" minRange="5.0"
maxRange="2000.0" />
        </KillProbability>
        <KillProbability munitionType="20mm CIWS" platformType="YJ82
SSM" class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.4" maxRangePK="0.3" minRange="5.0"
maxRange="2000.0" />
        </KillProbability>
        <KillProbability munitionType="RGM84" platformType="Type052B"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.8" maxRangePK="0.6" minRange="2000.0"
maxRange="146304.0" />
        </KillProbability>
        <KillProbability munitionType="RGM84" platformType="Type053H3"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.9" maxRangePK="0.7" minRange="2000.0"
maxRange="146304.0" />
        </KillProbability>
        <KillProbability munitionType="RIM-7" platformType="YJ81 SSM"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.5" maxRangePK="0.3" minRange="500.0"
maxRange="16000.0" />
        </KillProbability>
        <KillProbability munitionType="RIM-7" platformType="YJ82 SSM"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.5" maxRangePK="0.3" minRange="500.0"
maxRange="16000.0" />
        </KillProbability>
        <KillProbability munitionType="SM2" platformType="YJ81 SSM"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.7" maxRangePK="0.6" minRange="500.0"
maxRange="180000.0" />
        </KillProbability>
        <KillProbability munitionType="SM2" platformType="YJ82 SSM"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.7" maxRangePK="0.6" minRange="500.0"
maxRange="180000.0" />
        </KillProbability>
        <KillProbability munitionType="YJ81 SSM" platformType="CG47"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.6" maxRangePK="0.4" minRange="2000.0"
maxRange="80000.0" />
        </KillProbability>
        <KillProbability munitionType="YJ81 SSM" platformType="CV"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.3" maxRangePK="0.1" minRange="2000.0"
maxRange="80000.0" />
        </KillProbability>
        <KillProbability munitionType="YJ81 SSM"
platformType="DDG51Flt1" class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.6" maxRangePK="0.4" minRange="2000.0"
maxRange="80000.0" />
        </KillProbability>
        <KillProbability munitionType="YJ81 SSM"
platformType="DDG51Flt2A" class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.6" maxRangePK="0.4" minRange="2000.0"
maxRange="80000.0" />
        </KillProbability>
        <KillProbability munitionType="SA-N-12 SAM" platformType="SH60R"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.7" maxRangePK="0.6" minRange="500.0"
maxRange="50000.0" />
        </KillProbability>
        <KillProbability munitionType="100mm" platformType="DDG51Flt2A"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.01" maxRangePK="0.0050"
minRange="500.0" maxRange="18000.0" />
        </KillProbability>
        <KillProbability munitionType="100mm" platformType="CG47"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.01" maxRangePK="0.0050"
minRange="500.0" maxRange="18000.0" />
        </KillProbability>
        <KillProbability munitionType="100mm" platformType="CV"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.0010" maxRangePK="5.0E-4"
minRange="500.0" maxRange="18000.0" />
        </KillProbability>
        <KillProbability munitionType="100mm" platformType="SH60R"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.9" maxRangePK="0.5" minRange="0.0"
maxRange="10000.0" />
        </KillProbability>
        <KillProbability munitionType="5InchCVT" platformType="Type052B"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.01" maxRangePK="0.0050"
minRange="500.0" maxRange="18000.0" />
        </KillProbability>
        <KillProbability munitionType="5InchCVT"
platformType="Type053H3" class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.01" maxRangePK="0.0050"
minRange="500.0" maxRange="18000.0" />
        </KillProbability>
        <KillProbability munitionType="5InchCVT" platformType="Zhi-9
Helo" class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.9" maxRangePK="0.5" minRange="500.0"
maxRange="10000.0" />
        </KillProbability>
        <KillProbability munitionType="RIM-7" platformType="YJ62 SSM"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.5" maxRangePK="0.3" minRange="500.0"
maxRange="16000.0" />
        </KillProbability>
        <KillProbability munitionType="SM2" platformType="Type052B"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.2" maxRangePK="0.2" minRange="500.0"
maxRange="30000.0" />
        </KillProbability>
        <KillProbability munitionType="RIM-7" platformType="Type052B"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.2" maxRangePK="0.2" minRange="500.0"
maxRange="15000.0" />
        </KillProbability>
        <KillProbability munitionType="SM2" platformType="Type052C"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.2" maxRangePK="0.2" minRange="500.0"
maxRange="30000.0" />
        </KillProbability>
        <KillProbability munitionType="RIM-7" platformType="Type052C"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.2" maxRangePK="0.2" minRange="500.0"
maxRange="15000.0" />
        </KillProbability>
        <KillProbability munitionType="SM2" platformType="Type053H3"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.3" maxRangePK="0.3" minRange="500.0"
maxRange="30000.0" />
        </KillProbability>
        <KillProbability munitionType="RIM-7" platformType="Type053H3"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.3" maxRangePK="0.3" minRange="500.0"
maxRange="15000.0" />
        </KillProbability>
        <KillProbability munitionType="YJ62 SSM"
platformType="DDG51Flt2A" class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.8" maxRangePK="0.6" minRange="2000.0"
maxRange="300000.0" />
        </KillProbability>
        <KillProbability munitionType="YJ62 SSM" platformType="CG47"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.8" maxRangePK="0.6" minRange="2000.0"
maxRange="300000.0" />
        </KillProbability>
        <KillProbability munitionType="YJ62 SSM" platformType="CV"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.5" maxRangePK="0.3" minRange="2000.0"
maxRange="300000.0" />
        </KillProbability>
        <KillProbability munitionType="SA-N-12 SAM" platformType="RGM84"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.5" maxRangePK="0.5" minRange="500.0"
maxRange="50000.0" />
        </KillProbability>
        <KillProbability munitionType="YJ83 SSM" platformType="CG47"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.6" maxRangePK="0.4" minRange="2000.0"
maxRange="250000.0" />
        </KillProbability>
        <KillProbability munitionType="YJ83 SSM" platformType="CV"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.3" maxRangePK="0.1" minRange="2000.0"
maxRange="250000.0" />
        </KillProbability>
        <KillProbability munitionType="YJ83 SSM"
platformType="DDG51Flt1" class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.6" maxRangePK="0.4" minRange="2000.0"
maxRange="250000.0" />
        </KillProbability>
        <KillProbability munitionType="YJ83 SSM"
platformType="DDG51Flt2A" class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.6" maxRangePK="0.4" minRange="2000.0"
maxRange="250000.0" />
        </KillProbability>
        <KillProbability munitionType="HQ9 SAM" platformType="SH60R"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.8" maxRangePK="0.7" minRange="500.0"
maxRange="90000.0" />
        </KillProbability>
        <KillProbability munitionType="HQ7 SAM" platformType="RGM84"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.4" maxRangePK="0.4" minRange="500.0"
maxRange="10000.0" />
        </KillProbability>
        <KillProbability munitionType="HQ7 SAM" platformType="SH60R"
class="dafs.weapon.LinearKillProbability">
            <Params minRangePK="0.8" maxRangePK="0.7" minRange="500.0"
maxRange="10000.0" />
        </KillProbability>
    </KillProbabilities>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: SIMKIT IMPLEMENTATION OF RADAR MODEL

The following are the three classes written to implement the radar model in Simkit. These classes were used in conjunction with the Simkit and Actions java libraries written by Dr. Arnold Buss.

1. NEWRADARSENSOR CLASS

```java
package finalProject;

import Simkit.smdx.Mover;
import Simkit.smdx.CookieCutterSensor;

/**
 * An extension of the cookieCutterSensor. Adds a prf (Periodic Rotation
 * Frequency) to the sensor for later detection of dwell times in
 * calculating time to detection.
 *
 * @author Scott Hattaway
 * @version $Id$
 */
public class NewRadarSensor extends CookieCutterSensor {

   private double prf;

   /**
    * @param range
    *           Maximum range
    * @param mover
    *           Platform sensor is mounted on
    * @param prf
    *           Radar sensors Periodic Rotation Frequency (prf) in
scans/min
    */
   public NewRadarSensor(double range, Mover mover, double prf) {
      super(range, mover);
      setPrf(prf);
   }

   /**
    * @return the prf
    */
   public double getPrf() {
      return prf;
   }

   /**
    * @param prf
    *           the prf to set
    */
   public void setPrf(double prf) {
      if (prf > 0.0) {
         this.prf = prf;
      } else {
         throw new IllegalArgumentException("PRF must be > 0.0: " +
prf);
      }
   }

   /**
    * @return String version of this object
    */
   public String toString() {
      return "Radar Sensor [" + getMaxRange() + ", " + getPrf() + "]";
   }

}
```

2. NEWRADARMEDIATOR CLASS

```
package finalProject;

import java.beans.PropertyChangeEvent;
import java.util.Map;
import java.util.WeakHashMap;
import Simkit.SimEntityBase;
import Simkit.random.RandomVariate;
import Simkit.random.RandomVariateFactory;
import Simkit.smdx.Contact;
import Simkit.smdx.Mover;
import Simkit.smdx.Sensor;
import Simkit.smdx.SensorTargetMediator;
import Simkit.smdx.SensorTargetMediatorFactory;

/**
 * Mediator for NewRadarSensor. At this iteration the mediator evaluates
 * the reduction in range of the sensor due to the RCS of the target and
 * schedules the detection event appropriately. A probability of
 * detection is calculated based on the number of dwells from the target
 * to the sensor. If a detection occurs, a second delay is added
 * generated by a Gamma random variate to account for the number of
 * dwells required before the contact transitions to a track.
 *
 * @author Scott Hattaway
 * @version $Id$
 */
public class NewRadarMediator extends SimEntityBase implements
      SensorTargetMediator {

   private RandomVariate gammaVariate;

   private RandomVariate uniformVariate;

   /** list of contacts keyed by their originating target */
   private Map<Mover, Contact> contacts;

   public NewRadarMediator() {
      contacts = new WeakHashMap<Mover, Contact>();
      setGammaVariate(RandomVariateFactory.getInstance("Gamma", new
Object[] {
            new Double(1.0), new Double(1.0) }));
      setUniformVariate(RandomVariateFactory.getInstance("Uniform",
            new Object[] { new Double(0.0), new Double(1.0) }));
   }

   /** Clear contacts list */
   public void reset() {
      contacts.clear();
   }

   /**
    * @param sensor
```

```
     *            Sensor whose range is entered
     * @param target
     *            Mover that just entered range
     */
    public void doEnterRange(Sensor sensor, Mover target) {
       if (sensor instanceof NewRadarSensor
             && SensorTargetMediatorFactory.getInstance().getMediatorFor(
                   sensor.getClass(), target.getClass()) == this) {
          Contact contact = (Contact) contacts.get(target);
          if (contact == null) {
             contact = new Contact(target);
             contacts.put(target, contact);
          }
          double maxDetectDist = sensor.getMaxRange()
                * (Math.pow((Double) target.getProperty("radarFactor") /
25,
                      0.25));
          double dwellCount = (Double) sensor.getProperty("prf") *
maxDetectDist
                / sensor.getMover().getMaxSpeed();
          double probDetect = 1 - Math.pow(0.99, dwellCount);

          System.out.println("MaxDetectDist: " + maxDetectDist
                + "\tDwellCount: " + dwellCount + "\tProbDetect: " +
probDetect);

          if (uniformVariate.generate() <= probDetect) {
             double dwellDelay = dwellCount * gammaVariate.generate()
                   / (Double) sensor.getProperty("prf");
             double detectDelay = ((sensor.getMaxRange() - maxDetectDist)
/ sensor
                   .getMover().getMaxSpeed())
                   + dwellDelay;

             System.out.println("Trans to Track \t\t\t\tDwellDelay: "
                   + dwellDelay + "\tDetectDelay: " + detectDelay);

             sensor
                   .waitDelay("Detection", detectDelay, new Object[] {
contact });
          }
       }
    }

    /**
     * @param sensor
     *            Sensor whose range was just exited
     * @param target
     *            Mover that just exited range
     */
    public void doExitRange(Sensor sensor, Mover target) {
       if (sensor instanceof NewRadarSensor
             && SensorTargetMediatorFactory.getInstance().getMediatorFor(
                   sensor.getClass(), target.getClass()) == this) {
          Object[] contact = new Object[] { contacts.get(target) };
```

```
            if (contact[0] != null) {
               sensor.interruptAll("Detection", contact);
               sensor.waitDelay("Undetection", 0.0, contact);
            }
         }
      }

      /** Not used in this class */
      public void propertyChange(PropertyChangeEvent propertyChangeEvent) {
      }

      public RandomVariate getGammaVariate() {
         return gammaVariate;
      }

      public RandomVariate getUniformVariate() {
         return uniformVariate;
      }

      public void setGammaVariate(RandomVariate rv) {
         gammaVariate = rv;
      }

      public void setUniformVariate(RandomVariate rv) {
         this.uniformVariate = rv;
      }
   }
```

3. TESTRADARSENSORPLATFOM CLASS

```
package finalProject;

import animate.PingThread;
import animate.SandboxFrame;
import java.awt.Color;
import java.awt.geom.Point2D;
import Simkit.Schedule;
import Simkit.random.RandomVariate;
import Simkit.random.RandomVariateFactory;
import Simkit.smdx.Mover;
import Simkit.smdx.RandomLocationMoverManager;
import Simkit.smdx.Sensor;
import Simkit.smdx.SensorTargetMediatorFactory;
import Simkit.smdx.SensorTargetReferee;
import Simkit.smdx.UniformLinearMover;
import Simkit.util.PropertyChangeFrame;

/**
 * Unit test of the NewRadarSensor. At this iteration, the mover has two
 * NewRadarSensors with a maximum detect range. One radarSensor
 * simulates an air search radar while the second shorter range sensor
 * simulates a surface search radar. Platform A and C can be detected by
 * the air search radar while all of the three platforms can be detected
 * by the surface search radar. Each platform has a different RCS (radar
 * cross section) effectively creating three different detection ranges.
 * The radar sensor mediator also adds a second delay of detection for
 * the number of dwells required before the target is acquired.
 *
 * @author Scott Hattaway
 * @version $Id$
 *
 */
public class TestRadarSensorPlatform {

   static {
      SensorTargetMediatorFactory.addMediator(NewRadarSensor.class,
            PlatformC.class, NewRadarMediator.class);
   }

   /**
    * @param args
    */
   public static void main(String[] args) {
      Mover sensorPlatform = new UniformLinearMover("Sensor Platform",
            new Point2D.Double(0.0, 0.0), 10.0);
      Sensor[] sensor = new Sensor[] {
            new NewRadarSensor(150.0, sensorPlatform, 240.0),
            new NewRadarSensor(25.0, sensorPlatform, 3600.0) };

      Mover[] target = new Mover[] {
            new PlatformC("Platform A", new Point2D.Double(50.0, 250.0),
0.0,
                  5.0, 1.0),
            new PlatformC("Platform B", new Point2D.Double(300.0,
380.0), 0.0,
                  1.0, 1.0),
            new PlatformC("Platform C", new Point2D.Double(550.0,
250.0), 0.0,
                  0.1, 1.0) };

      SensorTargetReferee airSensorRef = new SensorTargetReferee();
      airSensorRef.register(sensor[0]);
      airSensorRef.register(target[0]);
      airSensorRef.register(target[2]);

      SensorTargetReferee surfSensorRef = new SensorTargetReferee();
      surfSensorRef.register(sensor[1]);
      surfSensorRef.register(target[0]);
      surfSensorRef.register(target[1]);
      surfSensorRef.register(target[2]);

      System.out.println(airSensorRef);
      System.out.println();
      System.out.println(surfSensorRef);

      Point2D[] path = new Point2D[] { new Point2D.Double(0.0, 400.0),
            new Point2D.Double(400.0, 400.0), new Point2D.Double(400.0,
0.0),
            new Point2D.Double(0.0, 0.0) };

      if (args.length > 0 && args[0].equals("patrol")) {
         PatrolMoverManager pmm = new PatrolMoverManager(sensorPlatform,
path);
         pmm.setStartOnReset(true);
         System.out.println(pmm);
      } else {
         RandomVariate[] rv = new RandomVariate[] {
               RandomVariateFactory.getInstance("Uniform", new Object[]
{
                     new Double(path[3].getX()), new
Double(path[1].getX()) }),
               RandomVariateFactory.getInstance("Uniform", new Object[]
{
                     new Double(path[3].getY()), new
Double(path[1].getY()) }) };
         RandomLocationMoverManager rlmm = new
RandomLocationMoverManager(
               sensorPlatform, rv);
         rlmm.setStartOnReset(true);
      }

      SandboxFrame frame = new SandboxFrame("Radar Sensor Platform Test
1.3");
      frame.setSize(800, 800);
      ((PingThread) frame.getControlPanel().getController())
            .setMillisPerSimtime(50);
      frame.getSandbox().setOrigin(new Point2D.Double(80, 510));

      for (int i = 0; i < target.length; ++i) {
         frame.addMover(target[i], Color.red);
      }
      frame.addMover(sensorPlatform, Color.blue);

      frame.addSensor(sensor[1], Color.green);
      frame.addSensor(sensor[0], Color.orange);

      Schedule.reset();

      PropertyChangeFrame pcf = new PropertyChangeFrame();
      for (int i = 0; i < sensor.length; ++i) {
         sensor[i].addPropertyChangeListener("detection", pcf);
         sensor[i].addPropertyChangeListener("undetection", pcf);
      }

      frame.setLocation(10, 10);
      frame.setVisible(true);
      pcf.setLocation(frame.getLocationOnScreen().x + frame.getWidth(),
frame
            .getLocationOnScreen().y);
      pcf.setVisible(true);
   }

}
```

APPENDIX C: DAFS IMPLEMENTATION OF RADAR MODEL

The following are the two classes written to implement the radar model in DAFS.

1. DAFSRADARSENSOR CLASS

```java
package dafs.sensor;

import Simkit.SimEventListener;
import Simkit.smdx.CookieCutterSensor;
import Simkit.smdx.Mover;

/**
 * An extension of the DAFSSensor. Adds a prf (Periodic Rotation
 * Frequency) to the sensor for later detection of dwell times in
 * calculating time to detection.
 *
 * @author Scott Hattaway
 * @version $Id$
 */
public class DAFSRadarSensor extends DAFSSensor {

   protected static final double DEFAULT_ALTITUDE = 1.0;

   private SensorType type;

   private SimEventListener[] listeners;

   private double prf;

   /**
    * @param range
    *            Maximum range
    * @param mover
    *            Platform sensor is mounted on
    */
   public DAFSRadarSensor(double range, Mover mover) {
      super(range, mover);
   }

   /**
    * @param range
    *            Maximum range
    * @param mover
    *            Platform sensor is mounted on
    * @param prf
    *            Radar sensors Periodic Rotation Frequency (prf) in
scans/min
    */
   public DAFSRadarSensor(double range, Mover mover, double prf) {
      super(range, mover);
      setPrf(prf);
    }

    public void setSensorType(SensorType type) {
        this.type = type;
    }

    public SensorType getSensorType() {
        return type;
    }

    public double getAltitude() {
        double altitude = DEFAULT_ALTITUDE;
        Object alt = mover.getProperty("altitude");
        if (alt instanceof Number) {
            altitude = ((Number) alt).doubleValue();
        }
        return altitude;
    }

    /**
     * @return the prf
     */
    public double getPrf() {
        return prf;
    }

    /**
     * @param prf
     *            the prf to set
     */
    public void setPrf(double prf) {
        if (prf > 0.0) {
            this.prf = prf;
        } else {
            throw new IllegalArgumentException("PRF must be > 0.0: " +
prf);
        }
    }

    /**
     * Removes all of the SimEventListeners.
     */
    public void shutDown() {
        listeners = getSimEventListeners();
        for (int i = 0; i < listeners.length; i++) {
            this.removeSimEventListener(listeners[i]);
        }
    }

    /**
     * Clears the Contact list.
     */
    public void clearContacts() {
        contacts.clear();
    }
```

```
   /**
    * Restores all of the SimEventListeners to this sensor.
    */
   public void restart() {
      if (listeners != null) {
         for (int i = 0; i < listeners.length; i++) {
            this.addSimEventListener(listeners[i]);
         }
      }
   }

   /**
    * @return String version of this object
    */
   public String toString() {
      return "Radar Sensor [" + getMaxRange() + ", " + getPrf() + "]";
   }

}
```

2. DAFSRADARMEDIATOR CLASS

```
package dafs.sensor;

import java.beans.PropertyChangeEvent;
import java.util.LinkedHashMap;
import java.util.logging.Logger;
import Simkit.random.RandomVariate;
import Simkit.random.RandomVariateFactory;
import Simkit.smdx.CookieCutterMediator;
import Simkit.smdx.Mover;
import Simkit.smdx.Sensor;
import Simkit.smdx.SensorTargetMediatorFactory;
import dafs.platform.Platform;

/**
 * Mediator for DAFSRadarSensor. At this iteration the mediator
 * evaluates the reduction in range of the sensor due to the RCS of the
 * target and schedules the detection event appropriately. A probability
 * of detection is calculated based on the number of dwells from the
 * target to the sensor. If a detection occurs, a second delay is added
 * generated by a Gamma random variate to account for the number of
 * dwells required before the contact transitions to a track.
 *
 * @author Scott Hattaway
 * @version $Id$
 */
public class DAFSRadarMediator extends CookieCutterMediator {

   private RandomVariate gammaVariate;

   private RandomVariate uniformVariate;

   public static Logger log = Logger.getLogger("dafs.sensor");

   /** list of contacts keyed by their originating target */
   protected LinkedHashMap contacts;

   public DAFSRadarMediator() {
      setGammaVariate(RandomVariateFactory.getInstance("Gamma", new
Object[] {
            new Double(1.0), new Double(1.0) }));
      setUniformVariate(RandomVariateFactory.getInstance("Uniform",
            new Object[] { new Double(0.0), new Double(1.0) }));
      contacts = new LinkedHashMap();
   }

   /** Clear contacts list */
   public void reset() {
      super.reset();
      contacts.clear();
   }

   /**
```

```
     * @param sensor
     *           Sensor whose range is entered
     * @param target
     *           Mover that just entered range
     */
   public void doEnterRange(Sensor sensor, Mover target) {
        if ((sensor instanceof DAFSRadarSensor) &&
                this ==
SensorTargetMediatorFactory.getInstance().getMediatorFor(sensor.getClass
(),
                target.getClass())) {
            Object contact = contacts.get(target);
            if (contact == null) {
                contact = new DAFSContact((Platform) target);
                contacts.put(target, contact);
            }

            Double rcs = (Double) target.getProperty("crossSection");
            if (rcs != null) {

                double maxDetectDist = sensor.getMaxRange() *
(Math.pow(rcs.doubleValue() / 25, 0.25));
                double dwellCount = (Double) sensor.getProperty("prf") *
maxDetectDist / sensor.getMover().getMaxSpeed();
                double probDetect = 1 - Math.pow(0.99, dwellCount);

                if (uniformVariate.generate() <= probDetect) {
                    double dwellDelay = dwellCount *
gammaVariate.generate() / (Double) sensor.getProperty("prf");
                    double detectDelay = ((sensor.getMaxRange() -
maxDetectDist) / sensor.getMover().getMaxSpeed()) + dwellDelay;

                    sensor.waitDelay("Detection", detectDelay, new
Object[]{contact});
                }
            }
        }
    }

   /**
     * @param sensor
     *           Sensor whose range was just exited
     * @param target
     *           Mover that just exited range
     */
   public void doExitRange(Sensor sensor, Mover target) {
      if ((sensor instanceof DAFSRadarSensor)
            && this == SensorTargetMediatorFactory.getInstance()
                  .getMediatorFor(sensor.getClass(), target.getClass()))
{
         Object contact = contacts.get(target);
         if (contact != null) {
            sensor.interrupt("Detection", new Object[] { contact });
            sensor.waitDelay("Undetection", 0.0, new Object[] { contacts
                  .get(target) });
         }
      }
   }

   /** Not used in this class */
   public void propertyChange(PropertyChangeEvent propertyChangeEvent) {
   }

   public RandomVariate getGammaVariate() {
      return gammaVariate;
   }

   public RandomVariate getUniformVariate() {
      return uniformVariate;
   }

   public void setGammaVariate(RandomVariate rv) {
      gammaVariate = rv;
   }

   public void setUniformVariate(RandomVariate rv) {
      this.uniformVariate = rv;
   }
}
```

APPENDIX D: SIMULATION UNIT CHARACTERISTICS

The information for each class of ship or weapon used in the simulation were researched in the applicable Jane's reference and then defined in DAFS terminology for the scenario. Some weapons systems were not implemented in the scenario due to their redundancy or lack of interaction within the scenario. All values for the weapons were converted to metric units to conform with the common measurement system used in the DAFS model.

1. AMERICA CLASS AIRCRAFT CARRIER (CV)

Characteristic	Jane's Listing	DAFS Definition
Displacement	83,960 tons (full load)	83960 assigned value for CVA
Speed	32 knots	Converted to 988m/min for speed
Missiles	2 Mk26 NSSM Launchers 2 Mk49 RAM Launchers	RIM-7 munition implemented with 4 launchers
Guns	2 20mm CIWS	20mm CIWS implemented
Radars	SPS-48E SPS-49(V)5 Mk 23/7 TAS SPS-67	Implemented as DAFSRadarSensor Not implemented (2d Radar) Not implemented (FC radar) Implemented as DAFSSensor
Helicopters	4 SH-60F (logistics helo) 3 HH-60H (SAR helo)	Not implemented

2. TICONDEROGA CLASS CRUISER (CG)

Characteristic	Jane's Listing	DAFS Definition
Displacement	9957 tons (full load)	9957 assigned value for CVA
Speed	30+ knots	Converted to 926m/min for speed
Missiles	2 Mk141 Harpoon CLS 2 Mk41 VLS	RGM-84 munition implemented SM-1 munition implemented
Guns	2 5"/54 Mk 45 Guns 2 20mm CIWS	5"CVT munition implemented 20mm CIWS implemented
Radars	SPY-1B SPS-49(V)7 SPS-55 SPS-64 SPQ-9B	Implemented as DAFSRadarSensor Not implemented (redundant radar) Implemented as DAFSRadarSensor Implemented as DAFSSensor Implemented as DAFSSensor
Helicopters	2 SH-60B LAMPS	Implemented as SH-60R LAMPS

3. ARLEIGH BURKE CLASS (FLIGHT I/II) DESTROYER (DDG)

Characteristic	Jane's Listing	DAFS Definition
Displacement	8950 tons (full load)	8950 assigned value for CVA
Speed	32 knots	Converted to 988m/min for speed
Missiles	2 Mk141 Harpoon CLS 2 Mk41 VLS	RGM-84 munition implemented SM-1 munition implemented
Guns	1 5"/54 Mk 45 Guns 2 20mm CIWS	5"CVT munition implemented 20mm CIWS implemented
Radars	SPY-1D SPS-67 SPS-64	Implemented as DAFSRadarSensor Implemented as DAFSRadarSensor Implemented as DAFSSensor
Helicopters	None	

4. ARLEIGH BURKE CLASS (FLIGHT IIA) DESTROYER (DDG)

Characteristic	**Jane's Listing**	**DAFS Definition**
Displacement	9188 tons (full load)	9188 assigned value for CVA
Speed	31 knots	Converted to 957m/min for speed
Missiles	2 Mk41 VLS	SM-1 munition implemented
Guns	1 5"/54 Mk 45 Guns	5"CVT munition implemented
Radars	SPY-1D SPS-67 SPS-64	Implemented as DAFSRadarSensor Implemented as DAFSRadarSensor Implemented as DAFSRadarSensor
Helicopters	2 SH-60B LAMPS	Implemented as SH-60R LAMPS

5. LUYANG II (TYPE 052C) CLASS DESTROYER (DDGHM)

Characteristic	Jane's Listing	DAFS Definition
Displacement	7000 tons (full load)	7000 assigned value for CVA
Speed	29 knots	Converted to 895m/min for speed
Missiles	YJ-62 HHQ-9	YJ-62 munition implemented HHQ-9 munition implemented
Guns	1 100mm/56 gun	100mm munition implemented
Radars	Type 517 Knife Rest Type 348 Type 364 Type 344	Not implemented (redundant) Implemented as DAFSRadarSensor Implemented as DAFSRadarSensor Implemented as DAFSSensor
Helicopters	2 Zhi-9A Haitun	Implemented as

6. LUYANG I (TYPE 052B) CLASS DESTROYER (DDGHM)

Characteristic	Jane's Listing	DAFS Definition
Displacement	7000 tons (full load)	7000 assigned value for CVA
Speed	29 knots	Converted to 895m/min for speed
Missiles	YJ-83 SA-N-12 Grizzly	YJ-83 munition implemented SA-N-12 munition implemented
Guns	1 100mm/56 gun	100mm munition implemented
Radars	Top Plate Type 364 Type 344	Implemented as DAFSRadarSensor Implemented as DAFSRadarSensor Implemented as DAFSSensor
Helicopters	1 Zhi-9A Haitun	Implemented as

7. JIANGWEI II (TYPE 053H3) CLASS FRIGATE (FFGHM)

Characteristic	Jane's Listing	DAFS Definition
Displacement	3250 tons (full load)	3250 assigned value for CVA
Speed	32 knots	Converted to 988m/min for speed
Missiles	YJ-81 HQ-7	YJ-81 munition implemented HQ-7 munition implemented
Guns	2 130mm	130mm munition implemented
Radars	Type 517 Knife Rest Type 364 Type 344	Not implemented (redundant) Implemented as DAFSRadarSensor Implemented as DAFSSensor
Helicopters	None	

8. SURFACE-TO-SURFACE MISSILES

Missile	Characteristics	Jane's Listing	DAFS Implementation
RGM-84 (HARPOON)	Speed Range Warhead	0.85 Mach 80 nm 221.6 kg	Converted to 16915 m/min for speed Converted to 146304m for range Assigned weight of 222
RIM-7 (Surface Mode)	Speed Range Warhead	2.5 Mach 15 km 38.6 kg	Converted to 49750 m/min for speed Converted to 15000m for range Assigned weight of 39
SM-2 (Surface Mode)	Speed Range Warhead	2.5 Mach 15 nm 90 kg	Converted to 49750 m/min for speed Converted to 30000m for range Assigned weight of 90
YJ-62 (C-602)	Speed Range Warhead	0.9+ Mach 300 km 300 kg	Converted to 17910 m/min for speed Converted to 300000m for range Assigned weight of 300
YJ-81 (CSS-N-4)	Speed Range Warhead	0.9 Mach 80 km 165 kg	Converted to 17910 m/min for speed Converted to 80000m for range Assigned weight of 165
YJ-83 (C-803)	Speed Range Warhead	1.3+ Mach 250 km 165 kg	Converted to 25870 m/min for speed Converted to 250000m for range Assigned weight of 165

9. SURFACE-TO-AIR MISSILES

Missile	Characteristics	Jane's Listing	DAFS Implementation
HQ-7	Speed Range Warhead	2.3 Mach 10 km 10 kg	Converted to 45770 m/min for speed Converted to 10000m for range Assigned weight of 10
HQ-9	Speed Range Warhead	4 Mach 90km 180 kg	Converted to 79600 m/min for speed Converted to 90000m for range Assigned weight of 180
RIM-7 NSSM	Speed Range Warhead	2.5 Mach 16 km 38.6 kg	Converted to 49750 m/min for speed Converted to 16000m for range Assigned weight of 39
SA-N-12 Grizzly	Speed Range Warhead	3 Mach 50 km 70 kg	Converted to 59700 m/min for speed Converted to 50000m for range Assigned weight of 70
SM-2 (Air Mode)	Speed Range Warhead	2.5 Mach 180 km 90 kg	Converted to 49750 m/min for speed Converted to 180000m for range Assigned weight of 90

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Ahner, Daryll, Leroy Jackson and Donovan Phillips. 2005. DAFS: A Low Resolution Modeling Approach: Architecture and Implementation. *Proceedings of The 10th Annual International Conference on Industrial Engineering Theory, Applications & Practice*; December 2005.

Buss, Arnold. 2000. Component Based Simulation Modeling. *Proceedings of the 2000 Winter Simulation Conference*, edited by P.A. Fishwick, K. Kang, J. A. Joines, and R. R. Barton.

Buss, Arnold and Daryll Ahner. 2006. Dynamic Allocation of Fires and Sensors (DAFS): A Low-Resolution Simulation for Rapid Modeling. *Proceedings of the 2006 Winter Simulations Conference*, edited by L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto.

Buss, Arnold and Paul Sanchez. 2002. Building Complex Models with LEGOS (Listener Event Graph Objects). *Proceedings of the 2002 Winter Simulation Conference*, edited by E. Yucesan, C. H. Chen, J. L. Snowden, J. M. Diarmes.

Case, Frederick T., Christopher W. Hines and Steven N. Satchwell. 1993. Analysis of air operations during DESERT SHIELD and DESERT STORM. *Naval Research Logistics*, volume 42, issue 4, Wiley Periodicals, Inc.

Chinese Defense Today. 2006. 9M317 / Shtil (SA-N-12) Ship-to-Air Missile. http://www.sinodefence.com/navy/navalmissile/9m317.asp. Last updated 29 April 2006. Accessed 28 February 2008.

Chinese Defense Today. 2006. Naval HQ-7 Ship-to-Air Missile. http://www.sinodefence.com/navy/navalmissile/hq7naval.asp. Last updated 30 April 2006. Accessed 28 February 2008.

Chinese Defense Today. 2006. Naval HQ-9 Ship-to-Air Missile. http://www.sinodefence.com/navy/navalmissile/hq9naval.asp. Last updated 29 April 2006. Accessed 28 February 2008.

Chinese Defense Today. 2007. Type 052B Luyang Class Missile Destroyer. http://www.sinodefence.com/navy/surface/type052b_luyang.asp. Last updated 22 June 2007. Accessed 28 February 2008.

Chinese Defense Today. 2007. Type 052C Luyang-II Class Missile Destroyer. http://www.sinodefence.com/navy/surface/type052c_luyang2.asp. Last updated 22 June 2007. Accessed 28 February 2008.

Chinese Defense Today. 2007. Type 053H2G/H3 Jiangwei Class Missile Frigate. http://www.sinodefence.com/navy/surface/type053h3_jiangwei.asp. Last updated 22 June 2007. Accessed 28 February 2008.

Chinese Defense Today. 2006. YJ-62 Anti-Ship Cruise Missile. http://www.sinodefence.com/navy/navalmissile/yj62.asp. Last updated 7 October 2006. Accessed 28 February 2008.

Chinese Defense Today. 2007. YJ-8 (C-801) Anti-Ship Missile. http://www.sinodefence.com/navy/navalmissile/yj8.asp. Last updated 9 July 2007. Accessed 28 February 2008.

Chinese Defense Today. 2007. YJ-83 Anti-Ship Missile. http://www.sinodefence.com/navy/navalmissile/yj83.asp. Last updated 9 July 2007. Accessed 28 February 2008.

Chinese Defense Today. 2007. Z-9C (AS 565 Panther) Naval Helicopter. http://www.sinodefence.com/airforce/helicopter/z9c.asp. Last updated 28 July 2007. Accessed 28 February 2008.

Havens, Michael E., editor. 2002. Dynamic Allocation of Fires and Sensors, Masters Thesis, Operations Research Department, Naval Postgraduate School, Monterey CA.

Hooten, E. R. 2006. *Jane's Naval Weapon Systems*, Issue 44, Jane's Information Group, Inc., Alexandria, VA.

Jackson, Leroy and Donovan Phillips. 2005. Using A Low Resolution Entity Level Modeling Approach. The Bulletin of Military Operations Research: Phalanx, 38-2: 15-26.

Koxinga. 2004. A Brief History of Chinese Naval Radar and EW Developments. China-Defense.com. http://www.china-defense.com/naval/plan_radar_ew/PLA-N%20Radar%20and%20EW.pdf.

Knott, Eugene F., John F. Shaeffer and Micheal T. Tuley. 2004. *Radar Cross Section.* Second Edition, Scitech Publishing, Inc., Raleigh.

Law, Averil M. and David W. Kelton. 2000. *Simulation Modeling and Analysis*, 3rd edition, McGraw Hill.

Mazumdar, Mrityunjoy and Glenn Levick. The People's Liberation Army Navy (PLAN) The Future Fleet of 2050. China-Defense.com. http://www.china-defense.com/naval/plan2050/plan2050-1.html.

The Johns Hopkins University Applied Physics Laboratory. 2006. *OPNAV Analytical Models Catalog*. Prepared by National Security Analysis Department.

Ragsdale, Cliff T. 2001. *Spreadsheet Modeling and Decision Analysis*, 3rd edition, South Western Publishing, Cincinnati, OH.

Saunders, Stephen, editor. 2006. *Jane's Fighting Ships 2006-2007*, Jane's Information Group, Inc., Alexandria, VA.

Skolnik, Merrill I., et alia. 1990. *Radar Handbook*, 2nd Edition, McGraw-Hill.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Arnold H. Buss
   Naval Postgraduate School
   Monterey, California

4. Professor Ronald D. Fricker, Jr.
   Naval Postgraduate School
   Monterey, California

5. CAPT Robert Adrion, USN
   Sea Strike (N81T) Branch Head
   OPNAV N81 Assessment Division
   Pentagon, Arlington Virginia

6. Christina K. Jurgens
   Sea Strike (N81T) Senior Analyst
   OPNAV N81 Assessment Division
   Pentagon, Arlington Virginia

7. LTC Darryl K. Ahner
   TRAC-Monterey
   Monterey, California

8. LTC Jeffrey B. Schamburg
   TRAC-Monterey
   Monterey, California

9. MAJ Manuel Ugarte
   TRAC-Monterey
   Monterey, California