Exploring Fog of War Concepts in Wargame Scenarios

Dillon N. Tryhorn

EXPLORING FOG OF WAR CONCEPTS IN WARGAME SCENARIOS

THESIS

Dillon N. Tryhorn, Second Lieutenant, USAF

AFIT-ENG-MS-21-M-086

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-21-M-086

EXPLORING FOG OF WAR CONCEPTS IN WARGAME SCENARIOS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Dillon N. Tryhorn, B.S.C.E.

Second Lieutenant, USAF

March 2021

AFIT-ENG-MS-21-M-086

EXPLORING FOG OF WAR CONCEPTS IN WARGAME SCENARIOS

THESIS

Dillon N. Tryhorn, B.S.C.E.
Second Lieutenant, USAF

AFIT-ENG-MS-21-M-086

Abstract

This thesis explores fog of war concepts through three submitted journal articles. The Department of Defense and U.S. Air Force are attempting to analyze war scenarios to aid the decision-making process; fog modeling improves realism in these wargame scenarios. The first article "Navigating an Enemy Contested Area with a Parallel Search Algorithm" [1] investigates a parallel algorithm's speedup, compared to the sequential implementation, with varying map configurations in a tile-based wargame. The parallel speedup tends to exceed 50 but in certain situations. The sequential algorithm outperforms it depending on the configuration of enemy location and amount on the map. The second article "Modeling Fog of War Effects in AFSIM" [2] introduces the Fog Analysis Tool (FAT) for the Advanced Framework for Simulation, Integration, and Modeling (AFSIM) to introduce and manipulate fog in wargame scenarios. FAT integrates into AFSIM version 2.7.0 and scenario results verify the tool's fog effects for positioning error, hits, and probability affect the success rate. The third article "Applying Fog Analysis Tool to AFSIM Multi-Domain CLASS scenarios" [3] furthers the verification of FAT to introduce fog across all warfighting domains using a set of Cyber Land Air Sea Space (CLASS) scenarios. The success rate trends with fog impact for each domain scenario support FAT's effectiveness in disrupting the decision-making process for multi-domain operations. The three articles demonstrate fog can affect search, tasking, and decision-making processes for various types of wargame scenarios. The capabilities introduced in this thesis support wargame analysts to improve decision-making in AFSIM military scenarios.

AFIT-ENG-MS-21-M-086

For my wife.

Acknowledgements

I would like to thank my entire committee for guiding me through the research process. They provided me with the tools for success, the inspiration for this thesis, and they challenged my ideas to help me focus on the research accomplishments. I would also like to acknowledge the Air Force Research Laboratory and the 711th Human Performance Wing for sponsoring my research. They provided the resources and support to conduct these experiments.

Dillon N. Tryhorn

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AFSIM Advanced Framework for Simulation, Integration, and Modeling. iv, ix, x, 1, 4, 6, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 83, 84, 1

C2 command and control. x, 4, 5, 6, 7, 8, 9, 10, 11, 15, 16, 17, 18, 32

CLASS Cyber Land Air Sea Space. iv, 1

COA course of action. 3

DoD Department of Defense. 31

EW electronic warfare. 7

FAT Fog Analysis Tool. iv, 1, 2, 3, 4, 31, 32, 83, 84, 1

FIMM Fog Identification and Manipulation Methodology. 1, 3, 15, 32, 83, 84

GUI graphical user interface. 18, 22, 30

HTML Hypertext Markup Language. 30

IA information assurance. 7

IDE integrated development environment. 19

IRCs Information-related capabilities. 7, 8

JADC2 Joint all-domain command and control. 31, 32

JFC Joint Force Commander. x, 6, 7, 8

JPDC Journal of Parallel and Distributed Computing. 14

MDO	Multi-domain operations. 30, 31, 32
OITL	Operator-in-the-Loop. 19
OODA	Observe-Orient-Decide-Act. 10, 15, 16, 28
OPSEC	operations security. 7
OSI	Open Systems Interconnection. 27
RADAR	Radio Detection and Ranging. 23
RIPR	Reactive Integrated Planning aRchitecture. 28, 29
ROE	rules of engagement. 5
TDOA	Time Difference of Arrival. 24
USAF	United States Air Force. 15
WKF	Warlock Framework. 21, 22

EXPLORING FOG OF WAR CONCEPTS IN WARGAME SCENARIOS

I. Introduction

Military leaders use battlespace intelligence and environmental information to make timely strategic and tactical decisions that advance their operational objectives while attempting to deny their opponent's actions. Simulation programs of computer-based wargames model battlespace events to aid decision-making so leaders may choose a more optimal option, from a set of many, to efficiently accomplish objectives. This research explores areas of uncertainty in war scenarios to find sources of fog, that could hinder decision-making processes [4] documented in three submitted journal articles. The article "Navigating an Enemy Contested Area with a Parallel Search Algorithm" [1] explores fog by finding the configurations of enemy unit locations on the map that lead to the greatest speedup for a parallel search algorithm. Solutions exist where the sequential algorithm outperforms the parallel algorithm and these results demonstrate fog in enemy locations can be used to disrupt the search and tasking processes. The article "Modeling Fog of War Effects in AFSIM" [2] creates the Fog Identification and Manipulation Methodology (FIMM) to introduce fog into sensors and communications and develops the Fog Analysis Tool (FAT) to implement FIMM into the Advanced Framework for Simulation, Integration, and Modeling (AFSIM) for verification. The sensors and communications contribute to a commander's perception of enemy forces in the battlespace. Fog distorts a commander's perception of a situation, which can skew the decision-making process and cause mission failure. The article "Applying Fog Analysis Tool to AFSIM Multi-Domain CLASS scenarios" [3] utilizes FAT to investigate the impact of fog effects in multiple warfighting domains and uses the trends to support the idea FAT is effective in multi-domain operations and useful for future war analysis. FAT's effectiveness in multiple domains solidifies the idea that the identification and manipulation of fog in sensors and communications are effective for providing options for military simulation analysts to improve decision-making in wargames

1.1 Problem and Motivation

Military leaders must consider large amounts of information across multiple warfighting domains to make timely decisions and further the mission. Simulation models attempt to realistically emulate war scenarios to analyze courses of action and choose the optimal path. A method to provide options for viewing levels of uncertainty, or fog, in a war simulation can further the realism for war simulations. The method needs a general definition of fog sources and how the sources can be manipulated for simulation analysis. An implementation of the method for multi-domain analysis verifies the method's effectiveness for viewing the impact on future scenarios.

The problem stems from a need to create a decision aid for military leaders for multi-domain operations. The U.S. Air Force is researching an intuitive sensing grid concept that uses fused sensor data from multiple platforms to provide decision makers information about contested environments [5]. The sensing grid provides leaders at the strategic and tactical levels to gain and maintain a decision advantage against adversaries. The grid assumes sensors collect the information and transmit it through communication links to reach a centralized processing entity. Leaders want the most accurate depiction of the environment possible, so the identification of fog sources and how fog affects the decision-making process allow military analysts to account for fog when processing the information and sending the data to decision makers.

A decision-making agent parallels to the sensing grid concept, but the agent provides decision-making capabilities instead of being a decision aid for human decision-makers. When collecting, processing, and forming a decision based on environmental information, a decision-making agent could incorporate fog into a course of action (COA) analysis. Accounting for fog in COAs may alter which option is considered optimal. The identification and manipulation of fog for multiple domains allows for a more realistic approach to forming decisions and decision aids.

1.2 Research Roadmap

The following roadmap provides the overall research objectives and goals.

1. Identify sources of uncertainty in wargame scenarios that affect the decision-making process for commanders.
2. Create a method to identify and manipulate uncertainty in wargames with sensors for data collection and communications for data transmission.
3. Develop a tool for an existing wargame simulation to implement the method and verify the impact.
4. Analyze multi-domain scenarios to verify the tool's effectiveness in disrupting the decision-making process across all warfighting domains.

The roadmap focuses on the identification and manipulation of fog effects. The parallel search article focuses on finding sources of uncertainty in wargame scenarios. The paper that introduces FIMM and FAT target the second and third items. The last article provides results to support the fourth item. Each article presented in this paper is shown in the journal format.

II. Background

This chapter provides a background on wargames and military simulations relating to command and control (C2), fog of war, decision-making models and processes, the Advanced Framework for Simulation, Integration, and Modeling (AFSIM), and multi-domain operations. The Fog Analysis Tool (FAT) targets C2 processes and decision-making models in AFSIM to introduce and manipulate various fog effects at different levels. The topics discussed in this section directly apply to the methodology of each article. The articles consider introducing fog effects for wargame scenarios in AFSIM for disrupting decision making in joint operations.

2.1 Military Simulations and Wargaming

A wargame is an armed conflict simulation game with friendly, enemy, and control teams. The friendly, or blue team, clashes with the enemy forces, or red team, to accomplish opposing objectives. The control, or white team, provides adjudication, analysis, and after-action reporting [6]. A wargame has three major components: a map, a set of units, and rules. The map represents the battlespace and can provide a grid or geographic view. Units represent various military formations. Rules provide the framework for playing the game, as well as various tables for the resolution of combat and such. A wargame may aim to provide a large and realistic experience or a smaller and simpler experience depending on the needs of the analyst [7]. The following sections provide a background of wargames and the processes military decision-makers use in war.

2.1.1 Wargame Background

A wargame consists of the following components to provide a structure for simulating war. The battlespace is the area of conflict where both teams operate. Grid maps and geographic-based maps represent the battlespace with boundaries. The limitations of the wargame, such as operating within map boundaries and attacking certain units, act as the rules of engagement (ROE). ROE are directives military authorities issue to control combat engagement in war across the United States armed forces [8]. A commander makes decisions with gathered information concerning the adversary. The commander selects courses of action that leverage centers of gravity to affect the enemy. The centers of gravity are the sources of power that provides moral or physical strength, freedom of action, or will to act [9]. A course of action is a decision path that accomplishes a commander objective in support of the overall mission. The commander may weigh multiple courses of action against a defined scale to choose the best one. When there are no clear winners, the commander may spend more time deciding than acting [9]. The courses of action identify an enemy center of gravity the commander intends to attack and the commander uses the effect when weighing the decisions. Wargames follow a chain of command structure where a commander issues orders to subordinates and they execute the orders. The next section discusses C2 and how commanders structure the command chain.

2.1.2 Command and Control

Military C2 is the exercise of authority and direction by a properly designated commander over assigned and attached forces in the pursuit of the mission [10]. Wargames utilize C2 for structuring the chains of command, logistics and decision-making processes. C2 enhances the ability of the commander to make sound and timely decisions and successfully execute them. C2 uses decentralized execution of centralized, overarching plans to provide unity of effort over complex operations. The C2 tenets strengthen unity of effort: clearly defined authorities, roles, and relationships; mission command; information management and knowledge sharing; communication; timely decision making; coordination mechanisms; battle rhythm discipline; responsive, dependable, and interoperable support systems; situational awareness; and mutual trust [11]. Today, joint commanders must make rapid decisions based on overwhelming information available to them while incorporating the tenets of C2. The next section provides a background on fog and areas where commanders may experience uncertainty in decision-making.

2.2 Fog of War

War is a realm of uncertainty and a fog wraps around decision-making factors in war in a varied level of uncertainty [4]. Therefore, fog of war is the uncertainty in situational awareness experienced by participants in military operations. The goal is to define where fog exists and allow the analyst to manipulate fog effects in virtual wargame frameworks such as AFSIM. This section provides definitions for fog within different domains of combat and processes commanders use to combat the fog of war and create their desired effects.

2.2.1 Information-Related Capabilities

Joint Pub 3-13 introduces the information environment [12]. The Joint Force Commander (JFC) controls forces from different warfighting domains: air, land, maritime, and space domains, and the information environment including cyberspace. The information environment provides a prime opportunity to introduce fog in simulations because it includes physical, informational, and cognitive dimensions that continuously interact with individuals, organizations and systems. The physical dimension consists of C2 systems, decision makers and infrastructure that enable individual and organizations to create effects. Physical platforms and their communication networks reside in this dimension. The informational dimension collects, processes, stores, disseminates, and protects information. The actions in this dimension affect the content and flow of information. The cognitive dimension encompasses the minds of those who transmit, receive, and respond to or act on information. This dimension focuses on individual perception and decision making and factors in cognitive influencers such as beliefs, emotions and motivations. This dimension is the most important component of the information environment [12].

Information-related capabilities (IRCs) are the tools, techniques or activities that affect any of the three dimensions of the information environment. The JFC employs IRCs to influence the information flowing to and from a target audience in the physical and information dimensions to affect the decision-making process. The response of the target audience contributes to the desired end state. IRCs consist of capabilities such as operations security (OPSEC), information assurance (IA), counter-deception, physical security, electronic warfare (EW) support, and electronic protection. These capabilities enable and protect the JFC's C2 of force [12]. Table 1 provides the components for this process.

The dimensions of the information environment provide access points for influencing target audiences. Joint Pub 3-13 states the purpose of IRCs are to influence a target audience [12]. Rules, norms, and beliefs govern the behavior of individuals and groups. Rules are explicit regulative processes such as policies and laws. Norms are regulative mechanisms accepted by the society. Beliefs are the collective perception of fundamental truths governing behavior. The first step to cause an effect using IRCs is to identify the target audience. The joint force studies the rules, norms, and beliefs of the target audience. The joint force then produces effects to modify the behavior

Table 1: Process of Enabling and Protecting JFC's C2 of Forces: The components of this process aid the JFC to affect a target audience in the information environment [12].

Component	**Description**
Information	Data in context to inform or provide meaning for action.
Data	Interpreted signals that reduce uncertainty.
Knowledge	Information in context to enable direct action. Explicit knowledge articulates through words, diagrams, formulas, etc. Tacit knowledge cannot articulate through those means.
Influence	The act of power to produce a desired output or end of a target audience.
Means	The resources available to a national government, non-nation actor or adversary in pursuit of its ends including public and private sector assets.
Ways	How means apply to achieve the desired state. Ways are persuasive or coercive.
Information-Related Capabilities	Tools and techniques using information to create effects and operationally desirable conditions within the dimensions of the information environment.
Target Audience	An individual or group selected for influence.
Ends	A consequence of the way of applying IRCs

of the target audience based on the information. The JFC then decides which IRCs it can apply to individuals, organizations or systems to produce the desired affect. Figure 1 illustrates the full concept of using IRCs in the information environment. The joint force identifies the target audience and gathers information. Then the joint force develops ways and means to influence the target audience and specific IRCs to accomplish the objectives. Over time, the target end state changes.

Section 2.3.2 provides more information on C2 in a cognitive modeling context. The next section identifies fog elements in command chains through established definitions of C2 functions.

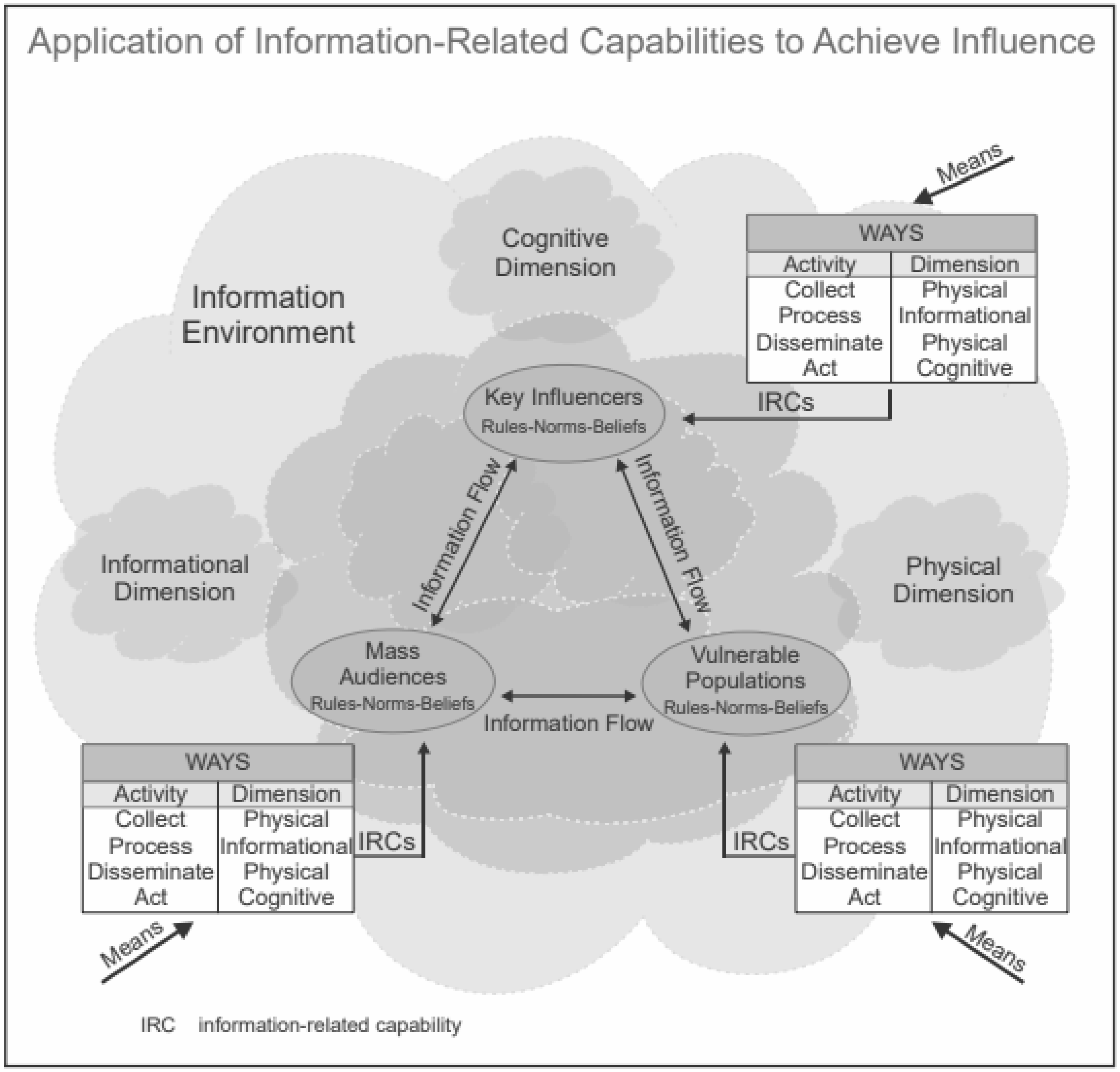

Application of Information-Related Capabilities to Achieve Influence
Information Environment
Cognitive Dimension
Informational Dimension
Physical Dimension
Means
WAYS
Activity
Dimension
Collect
Process
Disseminate
Act
Physical
Informational
Physical
Cognitive
IRCs
Key Influencers Rules-Norms-Beliefs
Mass Audiences Rules-Norms-Beliefs
Vulnerable Populations Rules-Norms-Beliefs
Information Flow
IRC information-related capability

Figure 1: Information Environment: This figure shows the application of information-related capabilities to achieve influence [12].

2.2.2 Identifying Fog Elements in Command Chains

This section identifies areas where information flowing to/from the commander and the weapon system and sensor platform subordinates may become denied, tampered, or inaccurate. This requires a definition of what functions make up the characteristics of C2. Martin van Creveld lists the defining characteristics of C2 as the following eight functions [13]. These functions provide similar processes to Lawson's

C2 model [14] and the Observe-Orient-Decide-Act (OODA) loop [15] in terms of collecting information, analyzing the data, making a decision, and executing a plan.

The first function from Table 2 provides multiple factors of uncertainty. Information collection is based on different types of measurement tools, sensors, and human intelligence. Sensors may not always provide accurate readings and it is possible for them to experience cutouts and degradation. Many measurement tools uses various types of sensors to collect information such as weather, temperature, and air pressure. Therefore, an accurate analysis of the sensors requires the analyst to vary the noise on the data collection. The second function for handling the data provides areas for fog as well. Communication networks for this research are a linked collection of nodes where data enters a node and exits another node linked to the first one. If data is already corrupted when entering a node, it remains so when exiting the network. However, if data is accurate and precise when entering the network, it may not remain so. The network may drop packets of information or physically lose a link to a node. An adversary may sit at a node the data is flowing through and tamper with

Table 2: Van Creveld Functions of C2: The eight functions list the defining characteristics of command and control [14].

Function	**Description**
1	Collecting information on own forces, the enemy, the weather and the terrain
2	Finding means to store, retrieve, filter, classify, distribute and display the information
3	Assessing the situation
4	Analyzing objectives and finding alternative means for achieving them
5	Making a decision
6	Planning based on the decision
7	Writing and transmitting orders as well as verifying their arrival and proper understanding by the recipients
8	Monitoring the execution by means of feedback, at which the process repeats itself

the information before sending it back into the network. If a subordinate is sending a commander coordinates to an enemy location, the commander may not receive the correct information from either an inaccurate sensor reading or an uncertainty issue with the communication network they use. For function seven, when the commander believes they are receiving accurate information, they may send their kinetic units to strike the area and end up hitting nothing or the wrong target. Functions three through six describe the process for the commander to analyzing the available data, form a decision, and create a plan for a tasking. Function eight also ties into the first function because the commander needs the collected information to see if the task is successful. This section identifies the critical areas in C2 decision-making to encompass communication and sensor assets. The sensor components provide the environment information and the communication components provide a channel for the information to flow to and from the commander. The next section discusses established methods for introducing fog into wargames and war scenarios.

2.2.3 Fog Elements in Wargames

Wargaming use different methods to introduce and define fog. John Setear provides a general method for defining and simulating fog in wargames [16]. Setear provides the sources of fog of war as uncertainty about the enemy, enemy intentions, and enemy forces. The natural environment facing the commander and the behavior of the friendly forces are also sources of fog. One last important source is the uncertainty about the underlying laws of war that govern the clash of arms on the battlefield. Setear stresses the sources of fog affect commanders and environments at all levels. This applies to strategic and tactical commanders and environments such as deserts, forests, and the sea. Setear's following methods aid the simulation of fog of war in paper-based and computer-based wargames.

1. **Referees Real and Simulated**. A referee holds all the information in the scenario and may delay or interfere with orders. Location-masking and strength-masking mechanisms allow the player to search for enemy strength and location of forces. This implements the fog source of the commander's need to know where the strength of the enemy resides.

2. **Counters that Conceal**. A counter-based method uses a single piece with a counter value to represent an unknown number of units. For example, a single fighter piece with a value of four represents four fighters. This method shows players the fog of war that results from uncertainty about the enemy's units. The untried units method keeps the counter value of all units hidden and reveals the counter values of two opposing units at the moment of combat to determine who wins.

3. **Counters Potentially in Play**. A die roll introduces a total-force-composition uncertainty by determining whether or when certain reinforcements enter a map based on the value of the roll. Wargames should use this method when initial order of battle is insignificant or when historical accuracy is unimportant.

4. **The Rulebook**. Rulebook tools fall into two areas of methods: modifications to sequence of play (or the method of determining the end of a game turn) and specific rule cases. The former methods introduce fog of war by limiting a player's ability to predict what the capabilities of both friendly and enemy units are in the next turn. The latter methods introduce fog in a relatively limited area for a relatively brief period of time. One example of Rulebook fog is introducing weather into a wargame. The Rulebook needs specific criteria for the type and duration of weather and the conditions for triggering it.

5. **Modifying the Map**. Another source of fog is the uncertainty of the terrain

in which players fight. The outcomes of fighting on a certain terrain are based on the rules of the scenario. However, this creates an area of importance for both players to learn the terrain values and create strategies based on those values. This excludes air and sea units.

6. **The Laws of War**. Two neglected areas of fog are the goals of the enemy and the laws of war. Since both players know the victory conditions, the wargamer knows the overall intentions of his opponent in all their details. History shows commanders and politicians do not exactly know the precise idea of the aims of their opponent. By introducing a roll or a choice of a victory condition from a menu of goals and recording it only to that player, the intentions of each opponent remains uncertain to an extent. The laws of war are relationships among physical variables that determine the outcome of a given engagement between units. A player's first few rounds of a wargame closely simulates the fog from the laws of war since the player is still learning how the game works.

The intent of most wargamers is not to closely simulate fog but to either best their opponent in a competitive setting or to learn historically how a battlefield exactly played out. Setear concludes rules simulating uncertainty help the efforts of a wargamer to learn why events happen as they do and discover in the long run what characteristics a good commander must possess [16].

Hagelback and Johansson scope the elements of fog in a real-time strategy game into the location of the enemy bases, exploration of unknown terrain, and the unpredictability of explored terrain [17]. Mason describes fog of war in most real-time strategy games is a term used to describe the mechanics of making only limited portions of a game map viewable for a combination of the areas immediately surrounding the player's character and all allied units. Unit movements shift these viewable zones and causes previously-visited areas to fade out of sight. This mechanic dynamically constrains the player's information, as areas outside their current viewing zones may contain active entities of interest. Progression requires eventual confrontation with whatever lies in the surrounding fog, forcing players to think strategically about how to prepare for the unknown [18]. This fog method focuses on uncertainty about terrain as well as the uncertain about the strength and location of enemy forces.

This section defines what fog elements are for both paper and computer wargames. The main areas of fog are the uncertainty of the strength and location of enemy forces, the natural environment facing the commander, behavior of the friendly forces, and the uncertainty about the underlying laws of war. This section also discusses how these areas of fog implement into wargames and real-time strategy games. The next section provides the mission routing problem to illustrate an instance of uncertainty in a scenario.

2.2.4 Mission Routing Problem

The journal article for Journal of Parallel and Distributed Computing (JPDC) uses a variant of the mission routing problem for a pilot to navigate a contested enemy battlespace. The mission routing problem is a type of combinatorial problem. The problem assumes a fleet of aircraft and multiple aircraft. Each aircraft is assigned a group of targets to visit. The problem is to determine the order in which to visit the targets. This maps to the traveling salesman problem where the starting location is the same as the ending city and the targets represent the cities to visit. The distance between cities is redefined as distance between starting location and cities [19]. The problem in the JPDC article represents the map as a grid. The tiles in the grid represent the starting location, enemy tiles, and the targets. The aircraft must avoid enemy tiles and reach the target. This variation of the mission routing problem assumes none of the tiles except the starting location are known.

2.3 Decision-Making Models and Processes

This section discusses several decision-making models and processes. These models and processes provide background for military decision-making and the C2 model used to create the Fog Identification and Manipulation Methodology (FIMM).

2.3.1 OODA Loop

The OODA loop is a four-step approach to decision-making that focuses on filtering available information, putting it in context and quickly making the most appropriate decision while also understanding that changes can be made as more data becomes available [20]. United States Air Force (USAF) Colonel John Boyd theorized the OODA loop to originally express an approach to tactical engagement, but later he expanded the idea to incorporate broad strategic action [15].

Maneuver warfare is a military strategy that emphasizes disrupting the opponent's decision-making process in order to defeat the enemy. The OODA loop uses this strategy to gain an advantage over the adversary. Mental models provide a representation of human behavior and the orientation phase uses these models. The decision-making phase relies on situational awareness. This concept is the comprehension of environmental stimuli and it involves perceiving and understanding all components of a situation. The OODA loop aims to minimize reaction time, the time that elapses between a stimulus and the response given to that stimulus [20].

The OODA loop consists of four main phases [21] [20] [22]. The first phase, Observe, gathers information pertinent to the decision at hand. Information appears internally through feedback loops and externally through sensors or other information sources. The second phase, Orient, consists of destruction and creation and involves the greatest amount of cognitive effort. The decision maker destructs the main problem into smaller sub-problems that he or she understands. With the understood knowledge of the sub-problems, the decision maker creates a plan of action using solutions for the sub-problems and combining the actions into a unified task. If the decision maker fails to create a plan, he/she may concede defeat. The third phase, Decide, contemplates the plans available to the decision maker. If the decision maker can only construct one feasible plan, the decision is simply whether or not to execute. If there is more than one overall plan, the decision maker chooses one as a course of action. The decision often involves weighing the risk or cost of a plan to its potential benefit. A single superior choice results in a confident and rapid decision. However, a few or many same-level decisions result in a longer decision-making time. The fourth phase, Act, represents the execution phase where the decision maker executes the chosen decision. The OODA loop phases execute simultaneously, not sequentially [22]. Figure 2 illustrates the OODA loop and its four phases.

The OODA loop provides a basic representation of the decision-making process. The next section presents a C2 model to illustrate the recursive nature of decision making.

2.3.2 Lawson's Command Control Model

Joel Lawson showcased his model in 1981 to create a military C2 system deriving from a higher national or political desire to maintain, or to change, the status quo in a contested battlespace [14]. The C2 system incorporates the ability to perceive or sense the state of its environment, compares the perception to a specified desired state, and takes action to force the environment into the desired state. This approach includes information gathering and processing as well as decision-making, similar to the OODA loop. The model functions at different levels of the chain of command and it requires the commander to receive some sort of visual representation of the environment such as charts, maps or electronic displays.

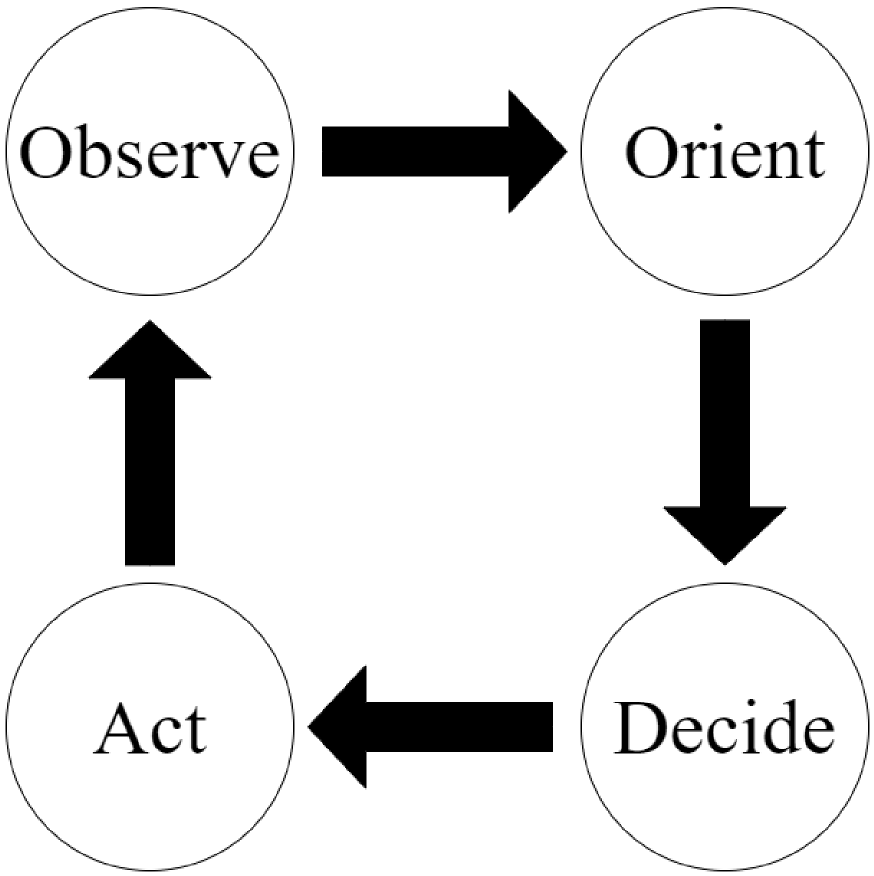


Figure 2: OODA Loop: The process consists of the four stages where the decision maker observes the environment, forms possible decisions, makes a decision and then executes the decision [20].

Figure 3 illustrates the Lawson C2 model process for a single unit. The individual unit senses the environment, processes the information and then takes action. At the same time, the commander up the chain manages multiple units and holds an array of sensors. This allows the process to occur recursively, however, this is above the scope of this research. Since decisions in war may save or lose lives, the cost of a wrong decision is higher in this model than in a management model. Management is the efficient use of resources internally to deal with a benign or neutral environment. C2 is the effective use of resources to deal with an external hostile environment. Therefore, time is an important parameter in this model so it introduces a time line analysis component. The time line analysis presents the timeline from the start to finish of

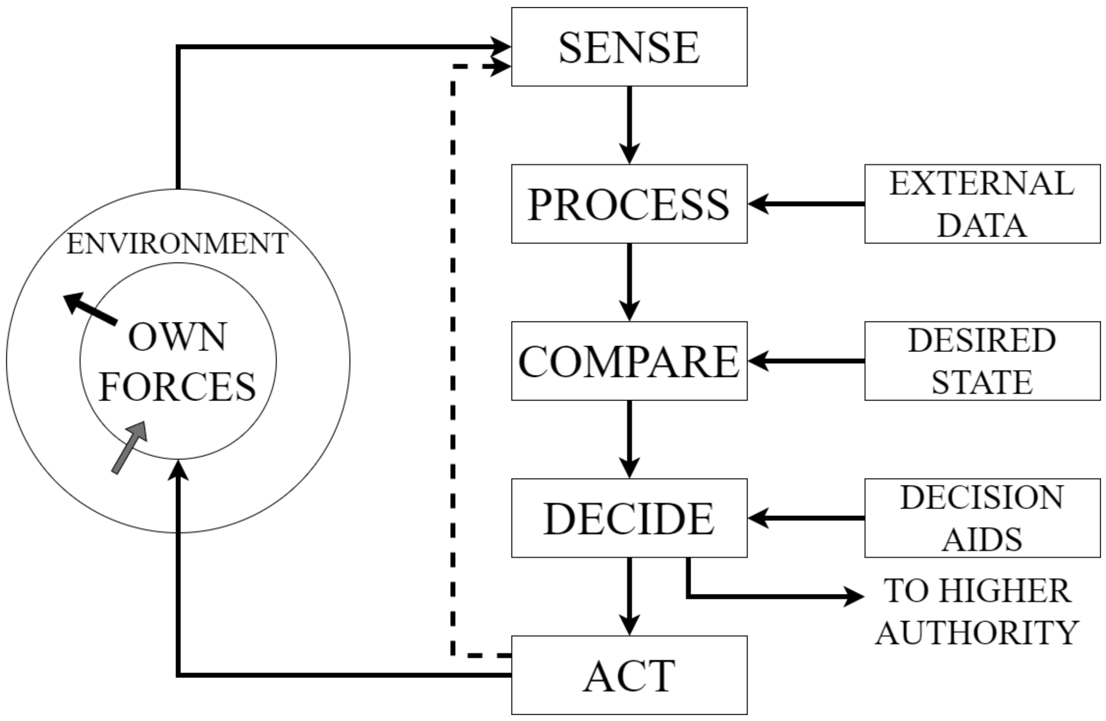


Figure 3: Lawson C2 Model: The Lawson C2 model process. The system perceives the environment, compares it to the desired state, and takes action to force the environment into the desired state [14].

each scenario. Each major event appears on the timeline from sensing to execution of command [14].

2.4 Advanced Framework for Simulation, Integration and Modeling

AFSIM is an objected-oriented C++ library that is used to create simulations that can model platform interactions in a geographic context. Platforms are top-level objects in the simulation and represent physical entities such as vehicles, buildings or living beings. Systems and attributes attach to platforms to provide functionality and specialization. Platforms interact internally or externally through processes such as sensor detections, collisions, and communications. AFSIM provides several core applications to simulate scenario input and generate post-analysis output. The AF-SIM ecosystem includes a suite of applications such as Wizard to provide a graphical user interface (GUI) for manipulation of the system [23]. This section explores the AFSIM applications, architecture, and source code development.

2.4.1 Main Suite Applications

This section describes the function of the core AFSIM suite applications. The general flow for a scenario consists of scripting the scenario in the Wizard integrated development environment (IDE), executing the scripts through the constructive Mission application or real-time Warlock application then viewing the completed simulation results in Mystic. The next three sections illustrate these applications and their slight differences.

2.4.1.1 Wizard

Wizard is an IDE that provides interfaces to aid with scripting and simulation as well as rapid platform integration. Wizard edits scenario files, executes AFSIM scenarios and visualizes the output [24]. The application also highlights file syntax, flags unknown commands, and provides context-sensitive documentation. It uses an auto-completion feature and a script debugger to minimize the time required to develop and debug models and scenarios [25].

2.4.1.2 Warlock

Warlock is the Operator-in-the-Loop (OITL) tool designed to interact with the simulation engine for real-time analytical capabilities. This application provides graphical views of AFSIM scenarios in real-time and allows for mid-simulation manipulation [24]. Warlock also facilitates the creation of cells of operators where each only have access to virtual information collected by that cell's platform. For example, there is a Blue cell of friendly platform operators and a Red cell of adversary platform operators. Both cells have imperfect information about the other. AFSIM can add further realism to the wargame by degrading the flow of information between members of the same cell. Warlock also supports the creation of a White cell that has perfect information about all platforms, where it leverages to control the flow of the overall wargame [25].

2.4.1.3 Mystic

Mystic (formerly Results Visualization) displays AFSIM components and platforms in a geographical context. The application allows the user to playback the scenario results in real-time, or faster, while viewing the position of each platform as well as platform interactions and sensor volumes [24].

2.4.2 Framework Architecture

AFSIM is an object-oriented, C++ simulation environment that emulates customized engagement and mission level warfare simulations. The framework includes a class hierarchy of simulation objects, including data driven platforms, movers, sensors, communications networks, processors, weapons, and simulation observers. Simulation and Event classes exist to control time and event processing for AFSIM-based models, and the logging of entity data. The framework supports a common geospatial environment and terrain representation. A general-purpose scripting language provides access to framework objects using text input files and the ability to run any AFSIM application in both constructive (batch processing) and virtual (real-time) modes [26]. Figure 4 illustrates the framework's high-level architecture while Table 3 provides descriptions for AFSIM services. The framework services make up the bulk of the simulation operations while platforms primarily use components.

Platforms are container data structures that hold components and information.

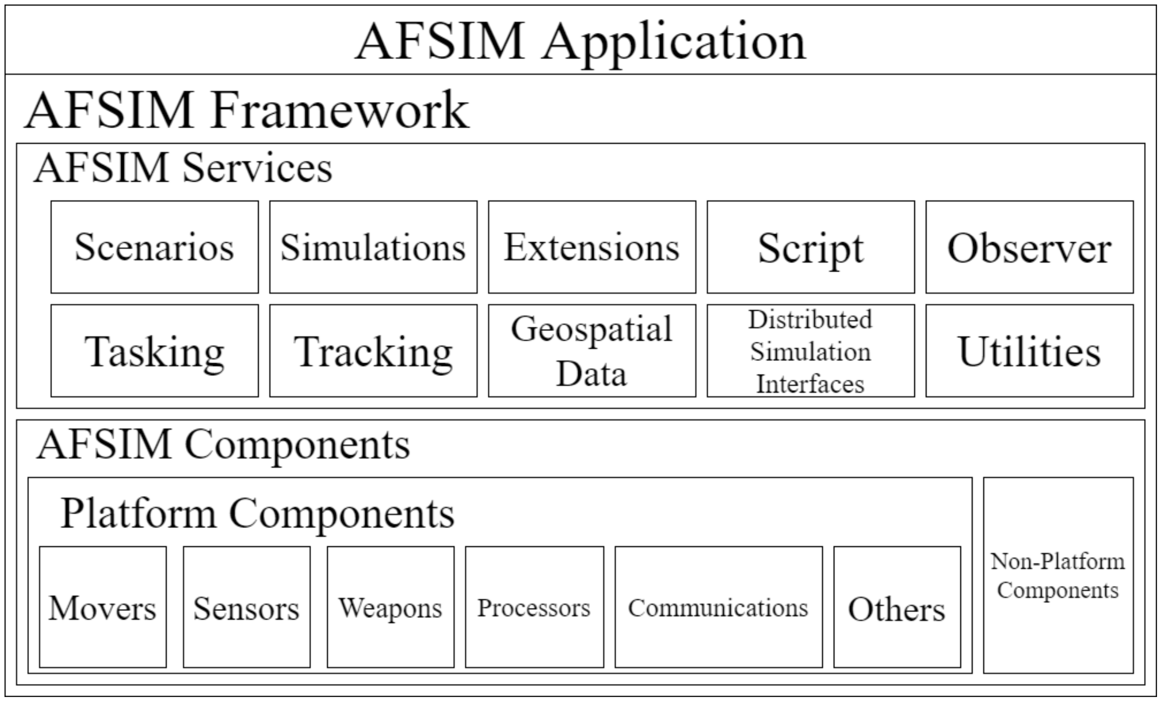


Figure 4: AFSIM High-Level Architecture: The framework consists of services and components that extend through extensions and plugins [27].

Platforms consist of physical components, mental components, information, attributes, and links. Figure 5 displays the architecture of a platform and the relationships between its internal components.

Movers maintain the kinematic state (position, orientation, speed, acceleration, etc.) of the platform. A communication device transmits and receives messages between platforms using external links. The framework allows for wired or wireless devices, using transmitters, receivers, and antennas to capture the full physical aspects of the communications systems. A sensor creates measurements and transmits them over links in track messages. Sensors often utilize transmitters, receivers, and antennas. A weapon prevents the operation of some other object either permanently or temporarily. A processor defines behaviors or computational algorithms much like a human brain or computer [23].

The Warlock Framework (WKF) is the common framework written to support all

Table 3: AFSIM Services and Descriptions: The services provide processes for creating simulations [23] [27].

Service	**Description**
Scenarios	Provide scenario input processing, type lists, and scripts.
Simulations	Provide time-based event processing and maintain platform lists.
Thread Management	Provides threading and multi-threading management capabilities.
Extensions and Plug-Ins	Supply a generic method of adding new services and components.
Script	Provides the infrastructure to implement and extend the AFSIM scripting language.
Observer	Supplies a generic publish-subscribe service for extracting data from simulations.
Tasking	Allows for inter-platform tasking and behavior modeling.
Tracking	Allows for track formation from sensor measurements and track correlation and fusion.
Geospatial	Data supplies terrain and line-of-sight data.
Distributed Simulation Interfaces	Simulation Interfaces applies interface standards for simulation interoperability (IEEE 1278 & 1516).
Utilities	Supply earth models, coordinate frames, math routines, artificial intelligence constructs, etc.

AFSIM GUI applications from section 2.4.1. Section 2.4.3 describes how plugins use the WKF and the plugin development process [28].

2.4.2.1 Tracks

Tracks are perceptions of other object relative to a single platform. Tracks only hold attributes for objects perceived as known and they contain information such as a position, coordinate, speed, etc. They relay the information back to the platform that perceives the object and owns the track. If the track does not perceive the object, it does not show data for that object [23].

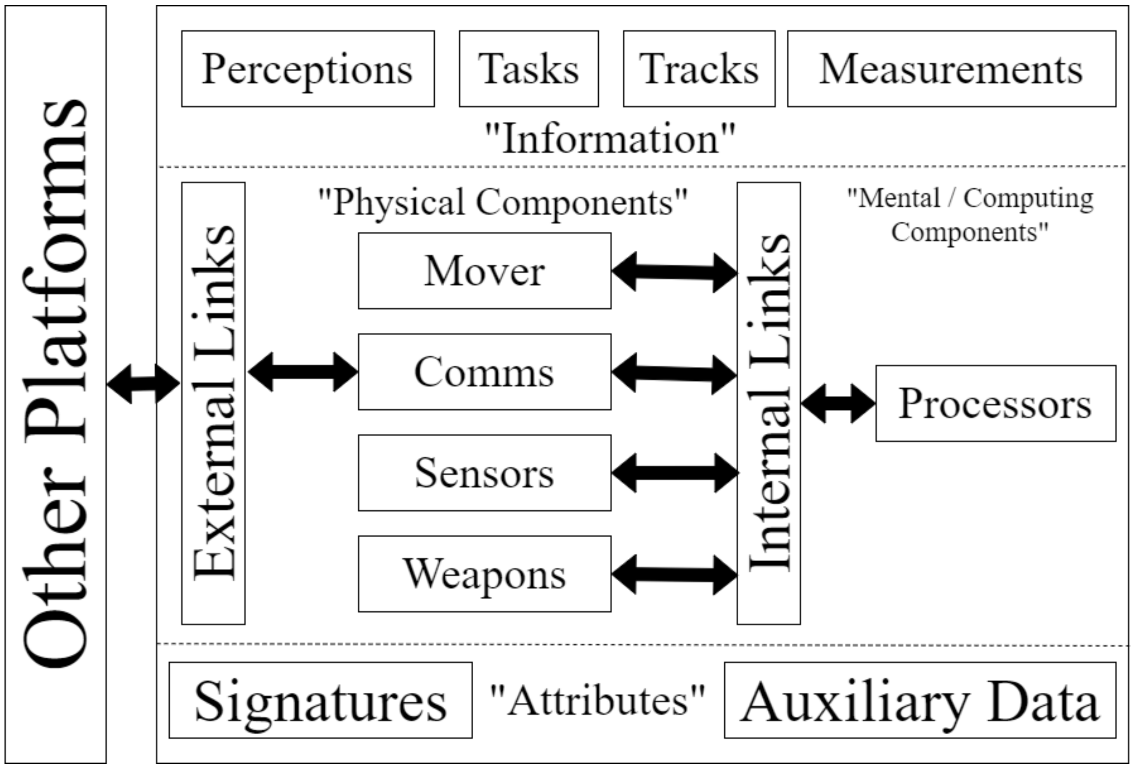


Figure 5: AFSIM Platform Architecture: A platform holds information about its environment and components to interact internally and externally. Each platform also has attributes to distinguish itself apart from other platforms. [23].

2.4.2.2 Sensors

A sensor provides the ability for a platform to detect other platforms and their components. AFSIM contains several predefined sensors including but not limited to Radio Detection and Ranging (RADAR), infrared, and acoustics [27]. The various types are in Table 5. Sensors use azimuth and elevation limits to define the area of influence and hold the abilities to detect jamming and to form tracks. The user may define error sigmas for sensors that form tracks and the sigmas provide error margins on the position measurements from the sensor [23].

Sensors use commands in the framework to define characteristics. Track reporting commands define the criteria for establishing a track and the quality of information reported in the tracks produced. Sensors take advantage of error models to provide a method for measuring error in the sensor object. Table 4 displays the sensor functions related to track reporting and error models.

The azimuth, elevation, range, and range rate error sigma commands allow the analyst to specify a standard deviation of the data based on the truth information the sensor reports. For example, a sensor reports a truth value of 800 meters but the error sigma for range is 20 meters. The perceived values could be any number in the range 780 to 820 meters. The hits to establish and maintain the track is M of N hits. This depends on the signal strength of the transmitter and if the perceived object remains in view. The establish and maintain probability provides an additional layer of whether the hits create or maintain the track of the perceived object.

The none error model is a dummy model equivalent to no error. The standard sensor error model only uses the azimuth, elevation, range and range rate error sigmas. The radar sensor error model uses the beam errors specified by the receiver or transmitter data.

The trimsim error model defines the various sources of error used in the Time Difference of Arrival (TDOA) algorithm. This pairs with the WSF_TRIMSIM_PROCESSOR. This processor models the effect of reference system errors on data fusion for air-to-ground targeting. The TDOA algorithm generates measurement errors of a target point in three dimensions based on errors from various sources. These errors are applied to the detection information of the master sensor. The sensors must be passive type sensors.

The bistatic error model uses the sensor mode azimuth error sigma and elevation error sigma. The instantaneous measurement calculates the dynamic range sigma. This command is mutually exclusive with range sigma, transmit only, and compute measurement errors. This model requires a direct line of sight to both the transmitter (direct signal) and to the target (reflected signal) to get a successful detection. AFSIM

Table 4: AFSIM Sensor Functions: These functions provide the capabilities for manipulating tracks [23].

Function	**Parameters**
Track Reporting Commands	
azimuth_error_sigma	[angle-value] OR [real-value] percent_of_true_range
elevation_error_sigma	[angle-value] OR [real-value] percent_of_true_range
range_error_sigma	[angle-value] OR [real-value] percent_of_true_range
range_rate_error_sigma	[speed-value]
hits_to_establish_track	M hits of N hits
hits_to_maintain_track	M hits of N hits
establish_track_probability	0.0 ... 1.0
maintain_track_probability	0.0 ... 1.0
Error Models	
error_model	none OR standard_sensor_error OR radar_sensor_error OR trimsim_error OR bistatic_error
Radar Sensor Error	
azimuth_beamwidth	[angle-value]
elevation_beamwidth	[angle-value]
pulse_width	[time-value]
receiver_bandwidth	[frequency-value]
doppler_resolution	[speed-value]
Trimsim Error	
north_position_error_sigma	[length-value]
east_position_error_sigma	[length-value]
down_position_error_sigma	[length-value]
reference_time_error	[time-value]
inter_system_time_delay	[time-value]
sensor_timing_error	[time-value]
atmospheric_refraction_residual	[unitless]
ground_target_altitude_error	[length-value]
Bistatic Error	
realistic_blurring	
time_reflected_sigma	[time-value]
time_direct_sigma	[time-value]
transmitter_position_sigmas	[length-value]

uses the sensors listed in Table 5. The table lists each sensor and a short description of the entity.

2.4.2.3 Weapons

A weapon prevents the operation of some other object either permanently or temporarily. Most weapons are explicit where the user creates a platform for the weapon. Implicit weapons do not represent as platforms such as the Jammer weapon. The user defines the type of weapon, the quantity of ammunition, if applicable, firing and reload information, and other weapon effects [23]. Weapon effects provide effects on targets [27].

2.4.2.4 Communications

A communication object provides the mechanism for platforms to communicate with each other as well as their own internal parts. Communication may occur in a wired or wireless setting with transmitters receivers and antennas. *WsfComm* is the

Table 5: AFSIM Sensor Types: The sensors listed collect data from different sources [23].

Type	**Description**
Acoustic	Simple passive acoustic sensor representing human hearing.
Composite	A sensor composed of other sensors.
EOIR	Baseline electro-optical or infrared sensor.
ESM	Baseline passive RADAR frequency detection sensor.
Geometric	Baseline sensor based on geometry.
IRST	Baseline infrared search-and-track sensor.
Optical	Simple electro-optical sensor.
OTH	Baseline over-the-horizon backscatter skywave RADAR sensor.
RADAR	Baseline RADAR sensor.
SAR	Baseline synthetic aperture RADAR.
Surface Wave RADAR	Surface-wave RADAR sensor.

base class for all communication implementations. A message is a unit of communication, there are many derived types. Messages pass over internal links within a platform that shows communications. Messages pass over external links when platforms wish to communicate with each other [27]. Communication objects in AFSIM use a 7-layer Open Systems Interconnection (OSI) model for implementation of communication types. Each object contains an object called a protocol stack that contains multiple layer objects that process messages sent from and received by a communication object. When an object sends a message, the message passes through every layer in the stack and each layer may pass the message or abort it [23].

The AFSIM comm object is a device that handles the transportation of messages between platforms. The comm object itself does not hold capabilities to modify the data in transit. However, the comm object uses multiple functions to deny, restrict the flow, and provide multiple paths of flow for information. Table 6 displays framework functions for the different layers of communication related to the flow of information.

Propagation speed is the speed of the message passing through a medium. This is by default the speed of light. Transfer rate is the amount of data able to trans-

Table 6: AFSIM Communication Functions: These functions affect the transmission of data [23].

Function	**Parameters**
Physical Layer	
propagation_speed	[random-speed-reference]
transfer_rate	[random-speed-reference]
packet_loss_time	[random-speed-reference]
Datalink Layer	
channels	[integer-value]
queue_type	fifo OR lifo OR priority
queue_limit	[queue-limit]
purge_interval	[time-value]
retransmit_attempts	[integer-value]
retransmit_delay	[time-value]

mit over a given time; for example, 100 bits per second means 100 bits transmit each second. Packet loss time introduces a delay in the information transmitted. These commands can interrupt the decision-making process by restricting the flow of information through the commander.

The datalink commands aid the datalink protocol to handle the scheduling and delivery of messages. The channels command supports simultaneous channels of transmission and therefore multiple information paths. The rest of the commands provide a structure for limiting and ordering the transmission and retransmission of data. These commands interact with the higher-level network membership the comm may use. Network objects use addressing and links to connect platforms internally or externally. The comm object may attach to a router and gateway to add another network.

2.4.2.5 Basic Agent Modeling

Agents are artificially intelligent entities that understand the world around it and its internal state. Agents operate in a virtual environment while making and carrying out decisions in a feedback loop. Agents follow a loop similar to OODA where they sense the world around them, collect knowledge about the environment, make a decision, and then take action [29]. AFSIM uses an artificial intelligence framework named Reactive Integrated Planning aRchitecture (RIPR) to create flexible agents with sophisticated behaviors. Users create RIPR agent scripts using behavior tree technology, a standard part of all script processors in the architecture [23].

RIPR agents contain a Perception processor and Quantum Tasker Processor. The agent senses the virtual world by querying the platform and subsystems for information. The agent builds knowledge, makes decisions, and takes action by controlling the platform accordingly [29]. The Quantum Tasker, shown in Figure 6, is used for commander subordinate interaction and task deconfliction. The Quantum Tasker operates by acquiring a perception of assets and a perception of threads from a cognitive model, generating and handling tasks, then assigning tasks over communications with a handshake [26].

A RIPR agent maintains its own perception of threats, assets, and peers. Therefore, the agents' brain holds limited and error-prone information. Users tune cognitive models to represent players of varying skill. A RIPR agent uses a RIPR behavior tree to define their behavior. A behavior is a compact modular piece of script that performs some unique action. A behavior tree connects the behaviors in interesting ways so they perform in certain orders or subsets [26]. Some agents use a Cluster Manger to perform clustering on threat or asset perception to think of the larger sets as smaller

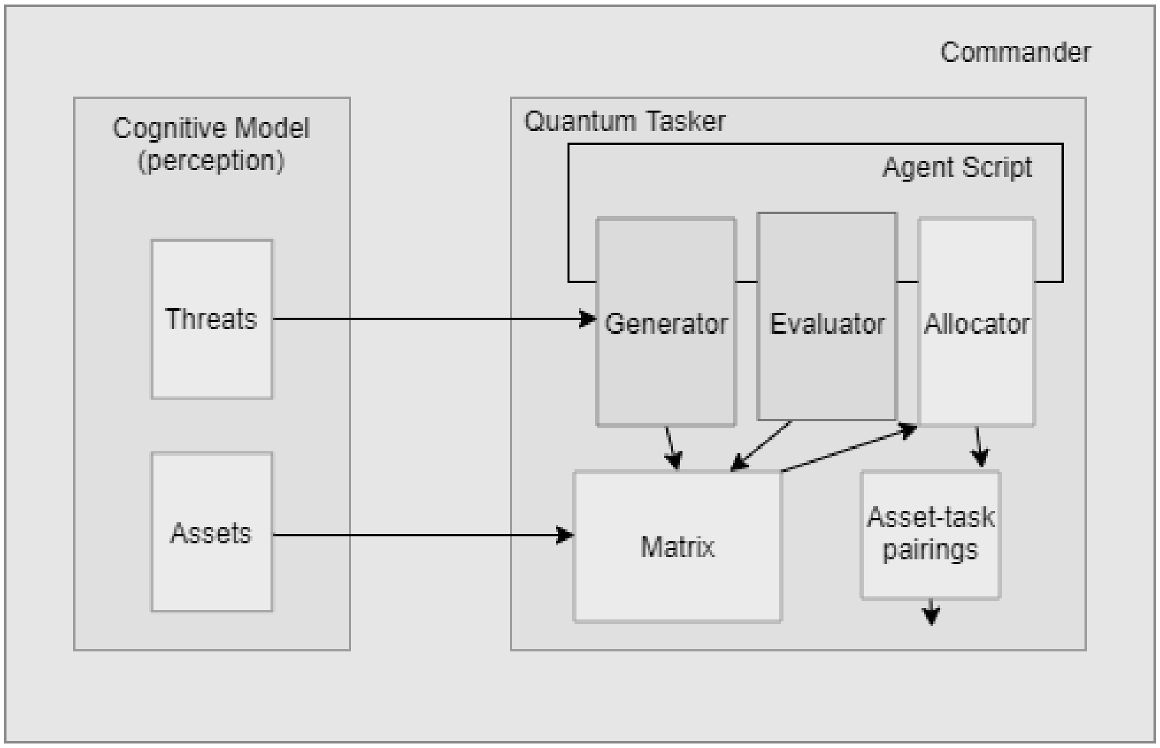


Figure 6: AFSIM Quantum Tasker: Quantum tasker mode of operation from commander. The tasker retrieves information from the cognitive model and uses it to calculate decisions [26].

groups. For example, a commander groups two incoming threats into two clusters so it sends two squadrons after separate groups [26].

2.4.3 Source Code Development

The AFSIM installation includes all source code in addition to a build of the current version [25]. The framework directory structure provides a full Hypertext Markup Language (HTML) documentation for the framework as well as training PowerPoint slides are specialized for either users or developers of the framework [23]. The developer training slides hold different examples for creating plugins to modify different pieces of the architecture. This section highlights the requirements to create and implement a plugin. AFSIM plugin creation requires CMake and Visual Studio for Windows. The CMake GUI allows the developer to select parts of the framework to install. For example, a developer holds the ability to create a custom build with only the Mission application. The custom configuration creates a solution file and Visual Studio builds it. The build named INSTALL creates the completed AFSIM build directory [30]. To create a plugin, the developer must navigate the plugin directory of the intended application and create a new folder with the name of the plugin. A standard plugin consists of a source folder, a user interface folder, a documentation folder, and a cmake file. The cmake file specifies the plugin name and source file path. The user interface folder provides files for the visual aspects of the plugin and how it displays in the chosen application. The source folder contains all source files and functionality of the plugin.

2.5 MDO and JADC2

Multi-domain operations (MDO) is a U.S. Army concept that defines the joint force as the Army, Navy, Air Force, Space Force, and Marines and the warfighting domains as air, land, maritime, space, and cyberspace. MDO describes how the joint force can counter and defeat a near-peer adversary capable of contesting the U.S. in all domains in both competition and armed conflict. The central idea is to compete successfully in all domains to deter a potential enemy. When deterrence fails, the joint force penetrates enemy anti-access and area denial systems. Then the joint force disintegrates, disrupts, degrades, or destroys enemy anti-access and area denial systems. Then it exploits the resulting freedom of maneuver by defeating enemy forces in all domains and attempts to re-compete to force the environment to a favorable state to the U.S. and allies [31]. MDO focuses on the U.S. Army but it emphasizes unifying the capabilities of the different branches of U.S. armed forces. It also acknowledges the need for all-domain dominance to maintain deterrence of enemies.

Joint all-domain command and control (JADC2) is a U.S. Department of Defense (DoD) concept to support sensor fusion of all U.S. military branches. JADC2 envisions providing a cloud-like environment for the joint force to share intelligence, surveillance, and reconnaissance data, transmitting across many communications networks, to enable faster decision-making. JADC2 intends to enable commanders to make better decisions by collecting data from numerous sensors, processing the data using artificial intelligence algorithms to identify targets, then recommending the optimal weapon to engage the target. Each U.S. military branch is attempting its own implementation of JADC2, such as the Air Force's Advanced Battle Management System (ABMS), the Army's Project Convergence, and the Navy's Project Overmatch [32].

MDO and JADC2 support the concepts of multi-domain warfighting and sensor fusion. A joint force commander uses assets and sensor data from multiple domains to enforce adversarial deterrence. FAT attempts to aid the simulation of war scenarios by introducing battlespace fog into data collection and transmission through sensors and communications. Incorporating fog effects into the decision-making process may allow the commander to make more accurate choices to accomplish the objectives.

2.6 Summary

This chapter provides information about wargame simulations, fog of war, the AFSIM framework, and multi-domain operations. This background chapter provides the information used for the journal articles in the next few sections. Chapter III explores the mission routing problem in a C++ based wargame environment. Chapter IV discusses the development of FIMM and FAT and its verification with C2 topics and decision-making processes. Chapter V analyzes FAT use in multi-domain scenarios using AFSIM, MDO, and JADC2.

III. Journal of Parallel and Distributed Computing

Navigating an Enemy Contested Area with a Parallel Search Algorithm

Dillon Tryhorn, Laurence D. Merkle, Richard Dill

Abstract

An enemy contested area consists of a tile map where each tile identifies as neutral, enemy or goal. The objective is to reach the goal tile without crossing into an enemy tile. A sequential modified breadth-first search algorithm with the Manhattan heuristic function provides an option to find a direct route to the goal while avoiding enemy tiles. The heuristic function provides a significant speedup to the search function. A parallel modification to the algorithm uses OpenMP on the Air Force Research Laboratory Mustang high performance computer to prove that the system achieves similar performance to the sequential algorithm. However, the system also achieves a substantial speedup with certain configurations of the tile map with respect to enemy tile placement.

Keywords: Wargame, Parallel Search, Heuristic, DOP, DFS

1. Introduction

Computer simulations allow for strategically planning war scenarios without expending manpower and unnecessary resources. The problem starts when a command unit tasks a bomber crew with the destruction of a critical enemy building. The bomber crew must avoid any enemy contact, mainly surface to air missile (SAM) sites. One or more paths exist that do not travel through enemy territory and they reach the objective. The area of engagement converts into a simple game board composed of tiles. Each tile has a x-y coordinate and an identifier of whether it is neutral or enemy territory or the goal. The bomber crew navigates through the neutral territory to the goal without entering enemy territory to avoid destruction (see Figure 1).

In this particular discrete-optimization (DOP) problem, the number of available search paths corresponds to the feasible solution space S. The

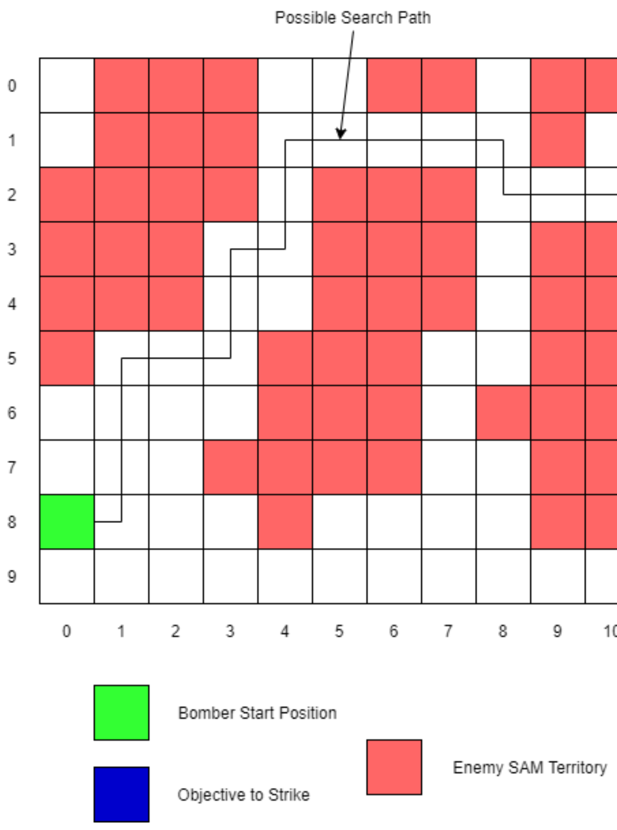


Figure 1: A problem example with a 10 x 18 tile grid.

length of the path corresponds to the cost function f. The objective of this problem is to find a feasible solution x_{opt}, such that $f(x_{opt}) \leq f(x)$ ∀x ∈ S or simply the shortest path to the objective without entering enemy territory (1). When the problem becomes very large, it would take a long time to find a route from the initial node to the goal node in a sequential algorithm. If the search space splits into smaller processes with tasks, each search space contributes to locating the goal node. There are different types of search algorithms that tackle similar problems. The next section analyzes these different types and identifies the search algorithm that the experiments modify and evaluate.

2. Background

DOPs are non-deterministic polynomial-time (NP)-hard problems and one may argue that there is no point in applying parallel processing to them (1). This is because the worst-case run time can never reduce to a polynomial unless there is an exponential number of processors. However, the average time complexity of heuristic search algorithms for many problems is polynomial time. For problems with large search spaces and real-time calculations, parallel search algorithms may be the only way to provide acceptable performance (1).

An example 8-puzzle problem contains a 3 x 3 grid containing eight tiles, numbered one through eight (see Figure 2). The ninth spot is the blank spot where the other adjacent tiles can move into. The tiles may only move up, down, left and right. The objective is to determine a shortest sequence of moves that transforms the initial configuration to the final configuration (2).

The 8-puzzle problem introduces the Manhattan distance (2). The Manhattan distance represents the distance between positions (i, j) and (k, l) as the expression $|i - k| + |j - l|$. In this problem, the sum of the distances between the initial and final positions of all tiles is an estimate of the number of moves required to transform the current configuration into the final configuration. If $h(x)$ is the Manhattan distance between configuration x and the final configuration then $h(x)$ is also a lower bound on the number of moves from configuration x to the final configuration This is also known as an admissible heuristic (2).

Depth-first branch-and-bound (DFBB) exhaustively searches the state space even after finding a solution path. Whenever it finds a new solution path, it updates the current best solution path. DFBB discards inferior

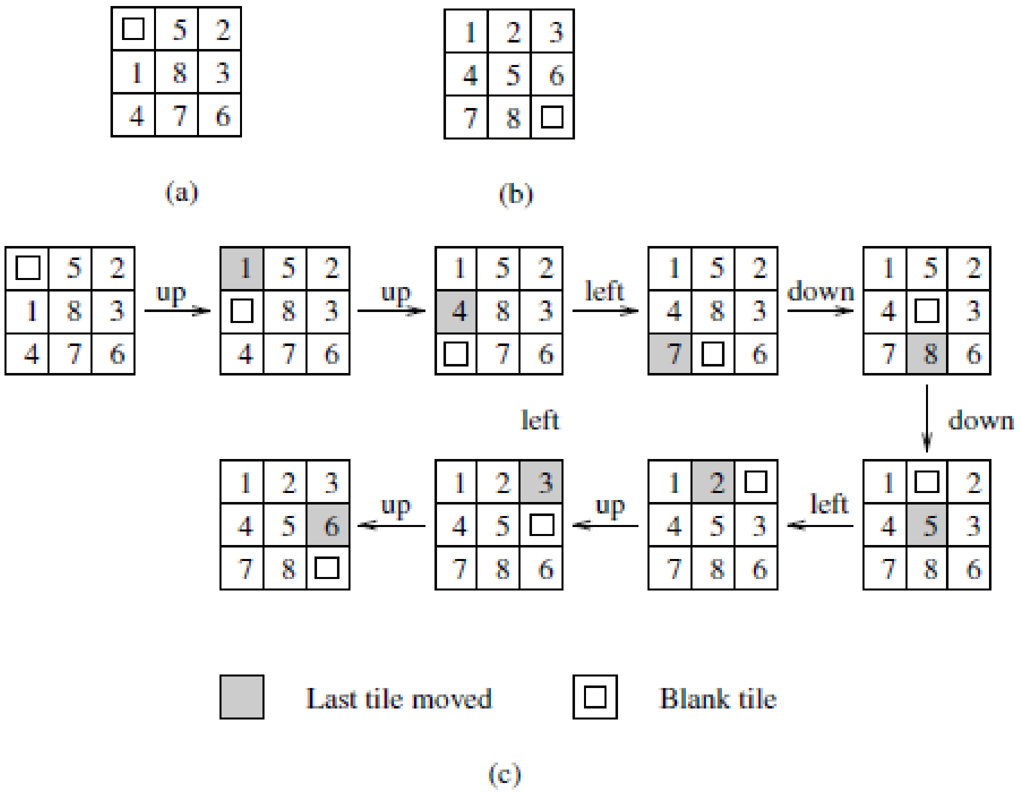


Figure 2: 8-puzzle problem instance: (a) Initial configuration; (b) Final configuration; and (c) A sequence of moves leading from the initial to the final configuration (2).

partial solution paths whose extensions are worse than the current best solution path. Upon termination, the current best solution is a globally optimal solution (2).

Pearl expands on the 8-puzzle problem by creating an NxN tile grid. Pearl claims that no effective upper cost-bounds are known for this problem, and hence DFBB cannot solve it efficiently. Even a relatively small 15-puzzle (N = 4) has a search space of $\frac{16!}{2}$ = 1013 states. Generally, it takes a few hundred million node expansions to solve a random problem instance of the 15-puzzle using the Manhattan heuristic (3).

The A* algorithm uses the lower bound function l as a heuristic evaluation function (2). For each node x, $l(x) = h(x) + g(x)$. The l function orders the nodes in the open list by their value. At each step, the algorithm removes the node with the smallest l value and expands it. The A* algorithm places the successors into the open list based on the l value and it places the node itself into the closed list (2). However because the memory requirement is linear in the size of the explored search space, memory is a limiting factor in the A* algorithm. The size of the search space is exponential in the depth of the expanded tree (6).

Iterative deepening combines the space efficiency of DFS with the optimality of breadth-first methods (4). A DFS algorithm may get stuck searching a deep part of the search space when a solution exists higher up on another branch. For such trees, an iterative deepening DFS (ID-DFS) imposes a bound on the depth to which the DFS algorithm searches. If the node about to expand is beyond the depth bound, then the node does not expand and the algorithm backtracks. If the search does not find a solution, then the algorithms searches the entire search space again using a larger depth bound (2).

Iterative deepening can also apply to an A* search. The resulting algorithm Iterative-deepening A* (IDA*) uses the l function from A* to repeatedly perform cost-bounded DFS over the search space. In each iteration, IDA* expands the nodes depth-first. If the l value of the adjacent node is greater than the cost bound, then the process backtracks. If the iterative fails to find a solution, the cost bound increases and another iteration occurs. The algorithm terminates when the iteration expands a goal node. IDA* simulates a best-first search by a series of depth-first searches with successively increased cost-bounds (5). IDA* has a low space overhead of O(d) that makes it feasible in applications where A* is unrealistic due to memory limitations (6). With an admissible heuristic estimate function, IDA* finds an optimal (shortest) solution. IDA* halts after finding a first solution and guarantees optimality by the iterative approach with the minimal cost-bound increments (3).

Depth-first search can execute in parallel by partitioning the search space into many small, disjunct parts (subtrees) that can expand concurrently. The parallel multithreaded iterative deepening A* (PMIDA*) algorithm (7) extends the sequential IDA* algorithm with the extension of multithreading and multiple processors. Figure 3 represents the PMIDA* algorithm graphically.

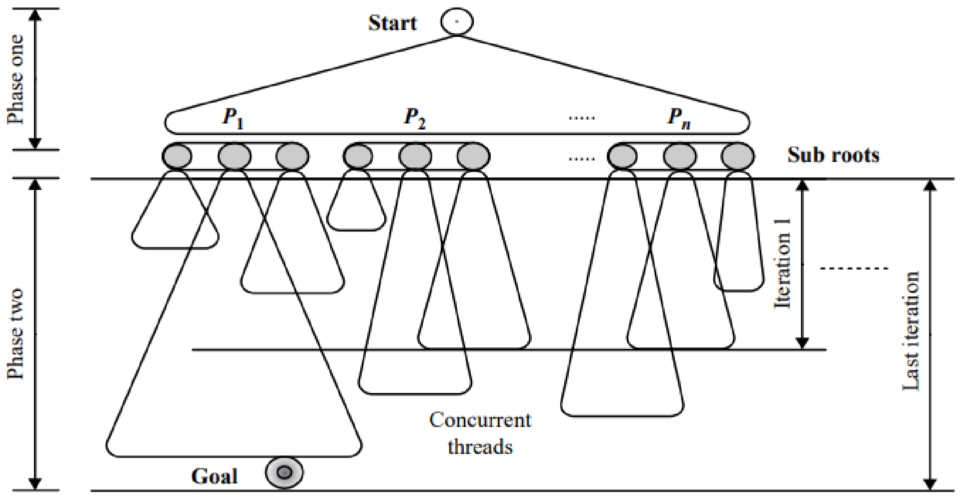


Figure 3: Graphical representation of PMIDA* Algorithm (7)

The PMIDA* algorithm consists of two major phases. In the first phase, one processor expands the tree, through a limited-depth BFS algorithm, to reach a specific level whose depth is determined at run-time depending on the nature and size of the problem and also on the number of processors. The generated nodes at this depth separate to multiple processors, where each processor receives the same number of nodes. In the second phase, each processor creates multiple equivalent threads. The constructed tree divides into equal partitions. All the created threads, concurrently, execute the sequential IDA* algorithm, each on its tree partition. Once any thread achieves the goal, it informs other threads in all processors to terminate. On the other hand, the search process stops if the tree exhausts the search space without any thread achieving the goal. Thus, until the algorithm finds the solution or the tree exhausts the search space, this process repeats multiple times with a distinct increased threshold value each time (7).

This problem is similar to the mission routing problem where a pilot selects ingress and egress routes to navigate enemy territory to accomplish a mission. The selection of a route is optimally the shortest path between the starting point and the target. The problem reduces to performing a search within the domain of the threat environment. The environment is represented as a tile map where the number of tiles determines the resolution of the problem. A higher number of tiles in the map allows the search to find a more routes but costs more time to search (10). This problem relates to the mission routing problem because it uses a tile map to navigate enemy territory to reach a target. The implementation of this problem is a C++ object-oriented approach that focuses on analyzing the effects of a sequential heuristic algorithm versus the parallel implementation.

3. Methodology

This section introduces a formal sequential algorithm, its implementation in Mustang, and Mustang system information and architecture structure. The section also introduces the parallel design of the new algorithm and its anticipated overheads and speedup with respect to the sequential algorithm. Then the implementation of the parallel algorithm takes an objected-oriented approach in C++ that compiles and executes on Mustang.

3.1. Sequential Algorithm

The problem solution derives from an informed modified depth-first search algorithm. The search must check the neighbors and verify they are not enemy territory before crossing into it. Once the search locates the target node, it terminates and prints the search path. Using the Manhattan distance function, the distance from the initial node to the goal node provides an admissible heuristic. The heuristic function guides the search path to the goal tile without iterating through branches of the search tree that move away from the goal tile. Assume a goal tile exists and there exists at least one path from the initial tile to the goal tile that avoids enemy tiles. Assume the system calculates all heuristic values *a priori*. Algorithm 1 demonstrates the solution.

Algorithm 1: Modified Depth First Search with Manhattan Heuristic Function

```
Execute
Data: tileMap, startTile
Result: boolean success
let searchPath be a list;
label startTile as discovered;
place startTile in searchPath;
if startTile is the goal then
    return startTile;
end
let neighbors be adjacent tiles of startTile;
order neighbors from lowest to highest heuristic value;
for neighbor in neighbors do
    if neighbor ID is not enemy ID then
        if Worker(neighbor) returns goal ID then
            return success;
        end
    end
end
return not success
Worker
Data: tile
Result: int ID
if tile is not discovered then
    label tile as discovered;
    place tile in searchPath;
end
if tile ID is goal ID then
    return tile ID;
end
let neighbors be adjacent tiles of startTile;
order neighbors from lowest to highest heuristic value;
for neighbor in neighbors do
    if neighbor ID is not enemy ID AND neighbor is not discovered
     then
        if Worker(neighbor) returns goal ID then
            return success;
        end
    end
end
return neutral ID;
```

3.2. Mustang Platform

The experiments utilize the Air Force Research Laboratory high performance computer Mustang (8). Mustang is an Hewlett Packard Enterprise (HPE) Silicon Graphics International (SGI) 8600 system and it supports two parallel programming models: Message Passing Interface (MPI) and Open Multi-Processing (OpenMP) 3.1 as well as a Hybrid MPI/OpenMP programming model. MPI is an example of the message or data passing models, while OpenMP uses only shared memory on a node by spawning threads. The hybrid model combines MPI and OpenMP models. Mustang has MPI libraries from HPE SGI and Intel. SGI's Message Passing Toolkit (MPT) and Intel's MPI support the MPI 3.0 standard (8).

The Mustang login and compute nodes use the Intel Xeon Platinum 8168 (Skylake) processors clocked at 2.7 GHz. Intel Skylake processors use directory-based coherency, the Opportunistic Snoop Broadcast (OSB), HitME cache and IO directory cache. For each node, there is a Caching and Home Agent (CHA) and the input/output (I/O) directory cache implements as an eight-entry directory cache per CHA. Skylake also uses sub-NUMA clustering which allows for the creation of two localized domains with each memory controller belonging to each domain. The processor also has a three-level cache, and each level uses a write-back protocol (9). Mustang uses the Intel Omni-Path interconnect in a Non-Blocking Fat Tree as its high-speed network for MPI messages and I/O traffic. Mustang has 1176 compute nodes that share memory only on the node; nodes do not shared memory between them. Each standard compute node has two 24-core processors (48 cores) sharing 192 GigaBytes (10^9) of Double Data Rate 4 (DDR4) memory, with no user-accessible swap space. Each large-memory compute node has two 24-core processors (48 cores) sharing 768 GigaBytes of DDR4 memory, with no user-accessible swap space. Each login node contains 384 GigaBytes of main memory. Each standard compute node contains 180 GigaBytes of user-accessible shared memory. Each large-memory compute node contains 744 GigaBytes of user-accessible shared memory. Mustang rates at 4.87 peak Peta (10^{15}) Floating Operations Per Seconds (PFLOPS) and has 8.4 PBytes (formatted) of parallel disk storage (8).

Considering only the standard nodes (1128 nodes) and assuming the tree is balanced, a Fat Tree network has a bisection width of $\frac{1128}{2} = 564$ links. The total number of links is $1128log_2(1128)$ which is approximately 11438 links. The diameter is $2log_2(1128)$ nodes which is approximately 20.28 nodes and the arc connectivity is 1 arc (2).

3.3. Parallel Algorithm Model

The parallel solution of this problem employs a work pool model (see Figure 4). This model dynamically maps tasks onto processes for load balancing in which any task may potentially perform on any process (2). Since the problem holds a search space of solutions, the parallel solution employs exploratory decomposition. This decomposition technique partitions the search space into smaller parts and searches those parts concurrently until it finds the solution (see Figure 5). Therefore, each task corresponds to a node to expand.

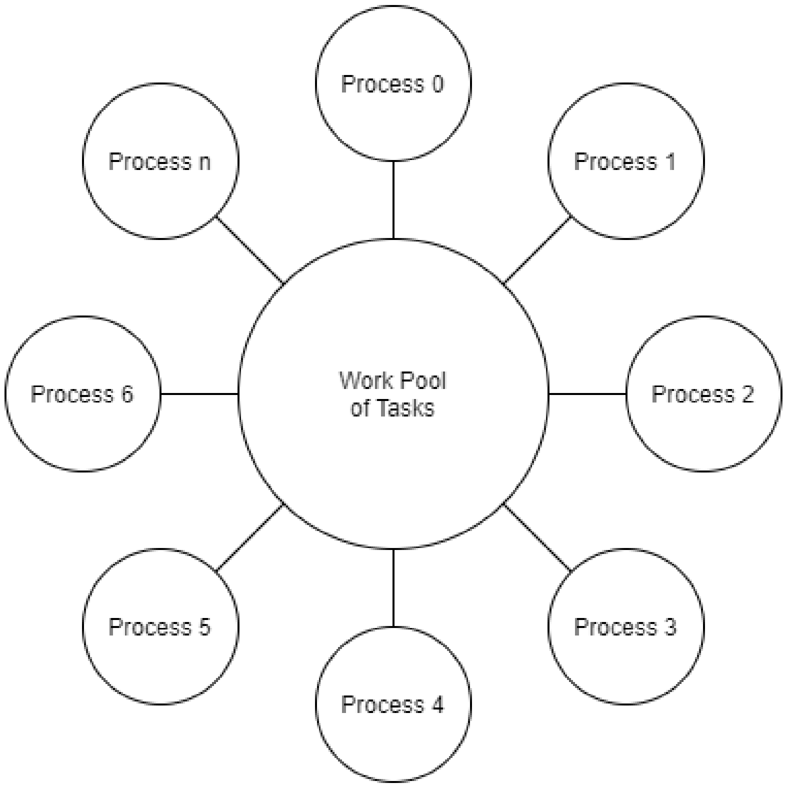


Figure 4: Work pool model with n processors.

Each node pointer resides in a shared list. The dynamic task generation occurs when each node expands and generates additional nodes. The mapping is a centralized scheme since the parallel tree search is central at the initial node. .

The tasks perform almost the same amount of work depending on the number of adjacent nodes to them. As the graph becomes larger, the granularity becomes more fine-grained and the maximum degree of concurrency becomes equal to the number of processes available to accept a task (see Figure 6).

Each resulting task is a tile to expand. Each tile holds the same information: (x,y) coordinate, tile identifier, heuristic value and discovered flag. The algorithm determines these values *a priori* but holds the ability to change the flag during execution. Since each task may not produce the same number of nodes to expand, the tasks are non-uniform and dynamically generated. Each task only generates the sub-tasks and places itself in the search path.

As a result of using the work pool model, the work pool centralizes the mapping of tasks. Each process requests a task from the work pool, performs the work, and repeats. The resulting search path tree that finds the goal node performs an all-to-one reduction and stores the resulting path into the global solution path. Once the process finds the solution, it performs a one-to-all broadcast to let other processes know they must terminate.

Some overhead occurs when each sub-task needs a copy of the local search path from the parent node. This produces an overhead of data replication. Nodes need to perform a check to verify if the neighboring node is an enemy node or if the node is already discovered. This check allows less data exchange and frequency of interaction between nodes. There is also the search overhead factor, the ratio of the work done by the parallel formulation to that done by the sequential formulation, or $\frac{Wp}{W}$ (2). Thus, the upper bound on speedup for the parallel system is $p * \frac{Wp}{W}$. The actual speedup, however, may be less due to other parallel processing overhead. In most parallel search algorithms, the search overhead factor is greater than one. However, in some cases, it may be less than one, leading to superlinear speedup. If the search overhead factor is less than one on the average, then it indicates that the serial search algorithm is not the fastest algorithm for solving the problem (2).

3.4. Implementation in Mustang

The implementation of the sequential and parallel modified depth-first search algorithms use the C++ coding language for an objected-oriented structured approach and the OpenMP library for parallel thread pooling. The tile graph class consists of tiles that fit in a X by Y grid. Each tile holds its coordinate, identifier of whether it is neutral, enemy or goal, the Manhattan distance for a heuristic and a Boolean flag to mark whether the

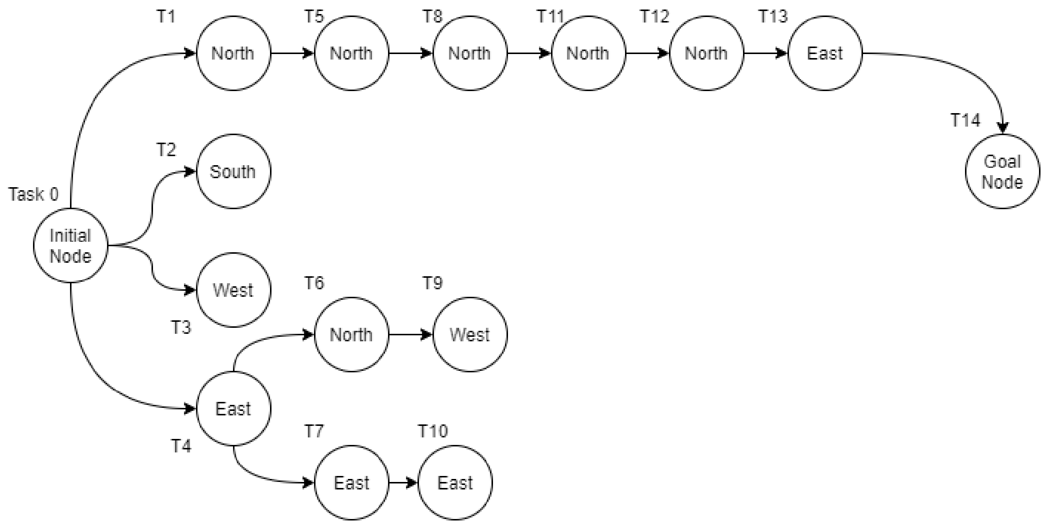


Figure 5: Task interaction graph with 15 tasks.

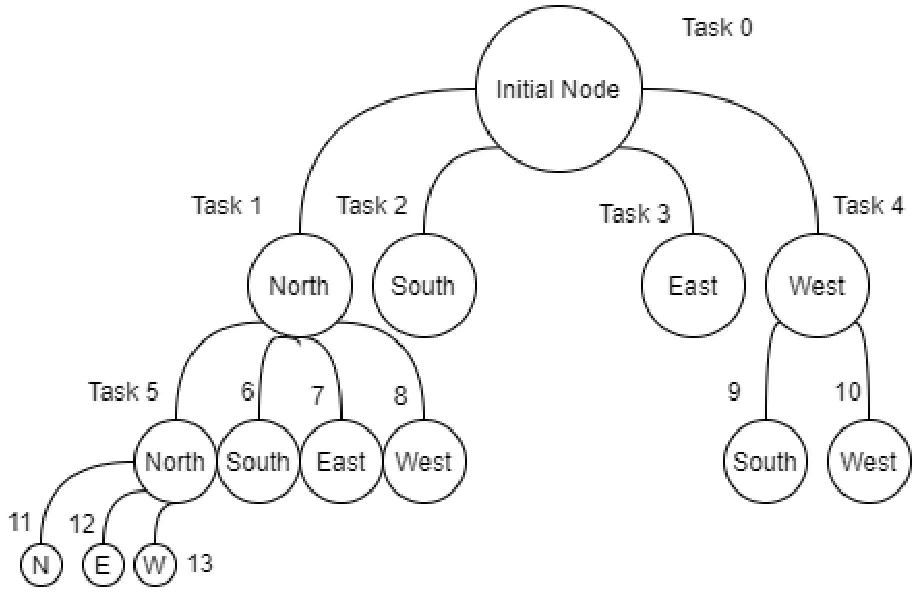


Figure 6: Example task dependency graphic with 14 tasks.

algorithm already traversed the tile. The program implements tiles as shared pointers so all processes can access them and the memory deallocates when the program terminates. The tile graph holds the logic for locating adjacent tiles and returning a list. The modified depth-first search (MDFS) class is a parent class for the sequential and parallel algorithm implementations. The MDFS class manipulates the tile information on the tile graph such as identifiers, discovered flags and heuristics. The MDFS class keeps track of the global search path used for the solution and the pointer to the goal tile. The iterative algorithm loops through the possible neighbor combinations using the heuristic value. Since the heuristic value provides an accurate representation of the distance to the goal, the path has a better chance of reaching the goal faster than without the heuristic. The parallel algorithm expands upon the sequential algorithm and uses OpenMP to parallelize the for loops that expand the neighbors out and place the tasks in a work pool.

The experiments analyze the program performance from the start of the search until the end of the search. This does not include the tile map creation and goal and enemy node selections. In the parallel program, due to the OpenMP library, once it finds the solution, the program loops through the remaining iterations while not performing any work. This causes an overhead of wasted computation and idling. However, since the processes pick a task from the work pool, there is little inter-process interaction overhead.

4. Results and Discussion

This section focuses on three simulation experiments: one where the sequential algorithm compares two version with and without the heuristic function, one where there are no enemy tiles and one where the enemy tile placement favors the parallel algorithm. Each experiment executes on the Mustang system with OpenMP handling the parallel work pool using a thread pool.

4.1. Sequential Algorithm Heuristic Improvement

This section focuses on three simulation experiments: one where the sequential algorithm compares two version with and without the heuristic function, one where there are no enemy tiles and one where the enemy tile placement favors the parallel algorithm. Each experiment executes on the Mustang system with OpenMP handling the parallel work pool using a thread pool.

4.2. Sequential Algorithm Heuristic Improvement

The sequential algorithm utilizes the Manhattan heuristic function to calculate the distance to the goal node from any given tile on the tile map. This experiment uses a 100 x 100 tile map with a goal tile at the coordinate (86, 67) and no enemy tiles. The experiment does not compare any parallel algorithm improvement. Table 1 provides the execution times and speedup for the heuristic improvement with the mean taken over 10 simulations. The mean execution times are in milliseconds.

Table 1: Table for Execution Times and Speedup for Heuristic Improvement

Configuration	Average Execution Time (**ms**)	Speedup
Non-Heuristic (baseline)	99.59	1
Heuristic	3.86	25.80

The heuristic implementation provides a substantial speedup to the sequential algorithm. However, under certain circumstances, the heuristic may not produce the best result. A later experiment illustrates this idea. The rest of the experiments compare speedups to the heuristic implementation of the sequential algorithm and the parallel implementation of the heuristic algorithm.

4.3. Search Without Enemy Tiles

The first experiment consists of an empty 2000 x 2000 tile grid. The initial node is located at the coordinate (0, 0). The goal node is located at the coordinate (954, 384). There are no enemy tiles on the map. Table 2 provides each execution time for the different number of threads with the mean taken over 10 simulations. The mean execution times are in seconds.

The multi-threaded approach to an empty map produces little speedup. This occurs because the sequential algorithm uses the Manhattan heuristic to locate the goal node in the shortest path possible. The parallel algorithm also utilizes the heuristic to guide the search towards the goal node but it uses the multiple threads to speed up the process to a maximum of 1.43 with 32 threads. The speedup may occur from the nodes on Mustang placing and executing the queued tasks faster than the sequential algorithm. Both the sequential and parallel algorithm produce the same number of elements in the search path. In the next section, the program modifies the tile map so

Table 2: Table for Execution Times and Speedup for No Enemy Tiles

Configuration	Average Execution Time (**seconds**)	Speedup
Sequential (baseline)	25.61	1
1 thread (Parallel)	25.87	0.99
2 threads	19.92	1.29
4 threads	19.92	1.29
8 threads	19.31	1.33
16 threads	19.71	1.30
32 threads	17.92	1.43
64 threads	20.83	1.23
128 threads	20.40	1.26
256 threads	19.65	1.30
512 threads	20.61	1.24
1024 threads	22.32	1.15

that it tricks the sequential algorithm to a longer solution and allows the parallel algorithm to attain a higher speedup.

4.4. Search That Favors Parallel Algorithm

To favor the parallel algorithm, the program must set up the tile map to trick the sequential algorithm into thinking the solution is closer than it actually is. This occurs when the sequential algorithm follows the heuristics to the goal node but the goal node becomes blocked by enemy tiles. For this experiment, the tile map is a 200 x 200 grid. The initial node is placed at (0, 100) and a line of enemy tiles goes from (1, 100) to (198, 100) and then from (198, 100) to (198, 0) as to create an L shape. The goal node is located at (199, 1). This configuration manipulates the sequential algorithm into thinking the goal node is north east, but it hits an enemy wall before it reaches the node and it backtracks to move around the wall. The parallel algorithm uses multi-threading to explore both paths and reach the goal in less time than the sequential algorithm. Table 3 provides the execution times and speedup for the different threaded configurations with the mean taken over 10 simulations. The mean execution times are in milliseconds.

Table 3: Table for Execution Times and Speedup for Parallel Favored Tile Map

Configuration	Average Execution Time (**ms**)	Speedup
Sequential (baseline)	4399.6	1
1 thread (Parallel)	13683.2	0.32
2 threads	90.77	48.47
4 threads	84.75	51.92
8 threads	86.60	50.81
16 threads	88.09	49.94
32 threads	87.64	52.20
64 threads	97.75	45.01
128 threads	90.11	48.83
256 threads	87.19	50.46
512 threads	96.46	45.61
1024 threads	111.53	39.45

Table 3 shows a significant speedup at two and above threads. When the parallel algorithm executes at one thread, it runs slower than the baseline sequential algorithm due to the overhead associated with the parallel implementation. The speedup peaks at 32 threads with a speedup of 52.20. The table also shows that a higher amount of threads starts to decrease the speedup due to the overhead of creating many threads. However, only using two threads uses fewer resources and provides a significant speedup to a single-threaded approach. This is due to the configuration of the tile map. Different configurations of the tile map with respect to the enemy tile placement determines whether the sequential algorithm or parallel algorithm is the better option.

5. Conclusion

A heuristic algorithm provides a benefit to the sequential portion of this problem. The heuristic provides an idea of how far away a certain tile is from the goal tile. This idea guides the search towards the solution. However, with this particular problem, certain configurations of the tile map with respect to the enemy tile placement severely affect the search execution time. With an empty tile grid, the parallel implementation provides a small benefit over the sequential algorithm but the benefit fails to have a significant increase as the threads increase. This occurs because the sequential algorithm uses the heuristic to guide the search without running into any enemy tiles. However, when the configuration of the tile map tricks the sequential algorithm into running into a wall of enemy tiles, the parallel algorithm provides a significant benefit by exploring multiple paths at a given time. The sequential algorithm is the optimal algorithm when the heuristic directly guides the search to the goal node without hitting a wall of enemy tiles. Otherwise, the parallel algorithm provides significant speedup to this problem.

6. Future Work

The algorithm currently has no bounded depth just as A* and IDA* do. The search tree continues to its end until there are no tiles left to explore in a given portion of the tile map. For example, with the simulation where the configuration favors the parallel algorithm, a bounded depth terminates the search tree after a certain number of iterations. When the sequential algorithm hits the wall of enemies, instead of continuing to search the area until all nodes exhaust, it reaches the bounded depth and picks the next neighbor from the initial node and traverses the path to find the goal tile. The overall concept requires additional map configurations to gauge the effectiveness of the parallel algorithm. A map generator would generate a tile map with a specified length and width and populate it with random enemy tiles while ensuring a path exists from the initial node to the goal node. Then each configuration would compare the speedup of the parallel algorithm to the sequential algorithm using different numbers of threads.

The OpenMP library provides limited access to the message passing interface commands but makes the process easier. An implementation of the parallel algorithm using the Message Passing Interface could provide additional benefits to the system and less overhead with more direct function calls. Then the OpenMP and MPI implementations compare the speedup and execution time data between them.

A different parallel model will provide different results to this experiment. This experiment uses a work pool model to distribute the work between the processes. However, another model such as the task graph model could implement the parallel algorithm. This model is typically employed to solve problems in which the amount of data associated with the tasks is large relative to the amount of computation associated with them (2). Since the nodes replicate data as the search path continues, the interaction between tasks is low and this model holds the ability to produce different results.

7. Disclaimer

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Air Force, the Department of Defense, or the U.S. Government.

References

[1] V. Kumar and Ananth Y. Grama. "Parallel Search Algorithms for Discrete Optimization Problems" *System Modelling and Optimization.* Proceedings of the 16th IFIP-TC7 Conference, Compiegne, France, 1993. pp. 55-69.

[2] Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar. *Introduction to Parallel Computing, Second Edition.* Addison Wesley, 2003. pp. 547-594.

[3] J. Pearl. *Heuristics. Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, Reading, MA, 1984.

[4] David Poole and Alan Mackworth. *Artificial Intelligence.* Poole Mackworth, Vancouver, Canada, 2010.

[5] R.E. Korf. *Depth-first iterative-deepening: An optimal admissible tree search.* Art. Intell. 1985. pp. 97-109.

[6] N.J. Nilsson. *Principles of Artificial Intelligence.* Tioga Publ., Palo Alto, CA, 1980.

[7] Basel A. Mahafzah. "Parallel multithreaded IDA* heuristic search: algorithm design and performance evaluation." *International Journal of Parallel, Emergent and Distributed Systems.* Vol. 26, No. 1, 2011. pp. 61-82.

[8] Air Force Research Laboratory. *HPE SGI 8600 (Mustang) User Guide* `https://www.afrl.hpc.mil/docs/mustangUserGuide.html` Accessed [June 25, 2020]

[9] WikiChip. *Skylake (server) - Microarchitectures - Intel.* `en.wikichip.org/wiki/intel/microarchitectures` Accessed [June 25, 2020]

[10] Olsan, J.B. *Genetic algorithms applied to a mission routing problem (No. AFIT/GCE/ENG/93-12).* Air Force Institute of Technology, Wright-Patterson AFB, OH. 1993.

IV. Journal of Defense Modeling and Simulation

Modeling Fog of War Effects in AFSIM

Journal of Defense Modeling and Simulation: Applications, Methodology, Technology
XX(X):1–12
©The Author(s) 2021
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
journals.sagepub.com/home/dms/
SAGE

Dillon Tryhorn[1], Richard Dill[1], Douglas Hodson[1], Michael R. Grimaila[1], and Christopher W. Myers[2]

Abstract

This research identifies specific communication sensor features vulnerable to fog and provides a method to introduce them into an Advanced Framework for Simulation, Integration, and Modeling (AFSIM) wargame scenario.
Military leaders use multiple information sources about the battlespace to make timely decisions that advance their operational objectives while attempting to deny their opponent's actions. Unfortunately, the complexities of battle combined with uncertainty in situational awareness of the battlespace, too much or too little intelligence, and the opponent's intentional interference with friendly command and control actions yield an abstract layer of battlespace fog. Decision-makers must understand, characterize and overcome this "battlespace fog" to accomplish operational objectives. This research proposes a novel tool, the Fog Analysis Tool (FAT), to automatically compile a list of communication and sensor objects within a scenario and list options that may impact decision-making processes. FAT improves wargame realism by introducing and standardizing fog levels across communication links and sensor feeds in an Advanced Framework for Simulation, Integration, and Modeling (AFSIM) scenario. Research results confirm that FAT provides significant benefits and enables the measurement of fog impacts to tactical command and control decisions within AFSIM scenarios.

Keywords

Fog of war, command and control, decision-making, AFSIM

1. Introduction

Military leaders must make timely strategic and tactical decisions from battlespace intelligence and available information. To improve real-world decision-making, commanders model their operation decisions in a virtual wargame to identify the optimal path to accomplish their objectives. The term fog describes a military decision-makers lack of information.[1] AFSIM is a modeling and simulation software that uses a C++ object-oriented approach to simulate geographically-based war scenarios.[2]

After reviewing several cognitive decision models, the Fog Identification and Manipulation Methodology (FIMM) was used to identify and adjust fog levels in a command and control (C2) scenario where sensor platforms collect and transmit information to the commander via communication links. The commander processes the information and transmits it back to effect-generating platforms for execution. Our tool, the Fog Analysis Tool (FAT), introduces fog effects to manipulate sensor platforms and communications links, universally. FAT is implemented as an AFSIM plugin. Simulation results obtained using FAT serve to aid the strategic and tactical decision-making process by introducing uncertainty across sensor platforms and communication objects that correspond to fog and directly support C2 functions. FAT and FIMM, in conjunction with AFSIM, produce a behavior analysis platform to improve operational decision-making in a contested environment. This research expands on Setear's ideas for introducing fog methods into wargames and on decision-making processes such as Lawson's C2 model and van Creveld's functions of C2.[3 4 5]

2. Literature Review

We first provide background and related research on topics needed to scope the decision-making process into a Command and Control (C2) context where information transmits over sensors and communication links. The following topics include a review of military simulations, battlefield C2, the battlefield fog, cognitive-behavioral-decision models, and AFSIM capabilities.

2.1. Military Simulations and Wargaming

A wargame represents a military scenario where two or more sides play against each other to accomplish their objectives. The battlespace is the conflict area where both teams operate, typically consisting of a grid or geographic-based map. The limitations of the wargame include operating within map boundaries and attacking specific units. A commander gathers information from the environment, forms decision

[1]Air Force Institute of Technology, USA
[2]Air Force Research Laboratory, USA

Corresponding author:
Dillon Tryhorn, Air Force Institute of Technology, Department of Electrical and Computer Engineering, Wright-Patterson AFB, Ohio, 45433, USA.
Email: dillon.tryhorn@afit.edu

paths, and selects a path that yields the highest probability to accomplish the planned objectives while minimizing loss and operational risk. Typically, the team identifies a series of adversary centers of gravity, power sources for moral or physical strength, freedom of action, or will to act. The commander may weigh multiple decision paths against a defined scale to choose the best one. When there are no clear winners, the commander may spend more time deciding than acting.[6] A command structure chain in a wargame is where a commander issues orders to subordinates to execute.

A few applications similar to AFSIM exist today. Next Generation Threat System (NGTS) is a military simulation environment produced by the Naval Air Warfare Center Aircraft Division (NAWCAD) that provides real-time military scenario simulations. NGTS models threat and friendly aircraft, ground, surface, subsurface platforms, corresponding weapons and subsystems, and interactions in a theater environment.[7] One Semi-Automated Forces (OneSAF) is another constructive and real-time simulation software developed by the U.S. Army for cross-domain analysis. OneSAF holds capabilities for simulating soldiers, logistical supplies, and communication systems.[8] These military simulation frameworks are similar to AFSIM, but each provides its distinct capabilities. The next section discusses military command and control (C2) and the functions of the command chain.

2.2. Command and Control

Military C2 exercises authority and direction by a properly designated commander over assigned and attached forces to accomplish a mission.[9] Wargames use C2 for structuring the chain of command, logistics, and decision-making processes. C2 enhances the commander's ability to make sound and timely decisions through decentralized execution of centralized, overarching plans. Joint Publication 1 lists the tenets of C2: clearly defined authorities, roles, and relationships; mission command; information management and knowledge sharing; communication; timely decision making; coordination mechanisms; battle rhythm discipline; responsive, dependable, and interoperable support systems; situational awareness; and mutual trust.[10] These tenets directly support the chain of command's goals: clear, quick, and accurate communication between the commander and subordinates. In contrast, these tenets also provide opportunities to introduce fog elements, potentially disrupting the adversary C2 capabilities. The next section focuses on the definition of fog and how analysts introduce fog elements in a wargame.

2.3. Fog of War

Clausewitz defines war as a realm of uncertainty. "Three quarters of the factors on which action in war is based are wrapped in a fog of greater or lesser uncertainty."[1] In modern warfare, battlefield fog is the uncertainty in situational awareness experienced by participants in military operations across multiple warfighting domains. The Joint Force Commander (JFC) controls forces from different warfighting domains: air, land, maritime, space domains, and the information environment, including cyberspace.[11] The information environment provides a prime opportunity to introduce fog in simulations because it includes physical, informational, and cognitive dimensions that continuously interact with individuals, organizations, and systems. The physical dimension consists of C2 systems, decision-makers, and infrastructure that enable individuals and organizations to create effects. Physical platforms and their communication networks reside in this dimension. The informational dimension collects, processes, stores, disseminates, and protects information. The actions in this dimension affect the content and flow of information. The cognitive dimension encompasses the minds of those who transmit, receive, and respond or act on information. This dimension focuses on individual perception and decision-making and cognitive influencers, such as beliefs, emotions, and motivations. Introducing uncertainty in the cognitive dimension could directly alter a decision maker's perception or indirectly alter how the perception develops in a decision-maker. This research focuses on extending the battlefield fog definition to include uncertainty in sensor data and the communication links that connect the commander to subordinates. The battlefield fog alters the perception of actors in the cognitive dimension to influence decision making.

Introducing fog effects at critical areas to deny, delay, or disrupt data to the adversary's decision-making processes while protecting one's C2 integrity is central to winning battles. Van Creveld lists defining characteristics of C2 as eight functions in Table 1.[5] These functions provide a baseline for identifying areas where fog can affect the collection and flow of information in a command chain. The first and second functions in Table 1 provide an uncertainty where information may become inaccurate during collection and transmission from a sensor platform to the commander. Functions three through six depend on the decision-making process implementation of the commander. These functions may provide areas for uncertainty but are not in the scope of this research. The seventh function is the communication of the commander to the effect-generating platforms. The eighth function is the commander's perception of the order execution. These last two functions may become delayed, inaccurate, or denied from orders move to and from the commander and subordinate. The cognitive models in the later sections discuss these functions of C2 in established decision-making processes.

Table 1. Creveld's eight functions list the defining characteristics of command and control.[5]

Function	Description
1	Collecting information on own forces, the enemy, the weather and the terrain
2	Finding means to store, retrieve, filter, classify, distribute, and display the information
3	Assessing the situation
4	Analyzing objectives and finding alternative means for achieving them
5	Making a decision
6	Planning based on the decision
7	Writing and transmitting orders as well as verifying their arrival and proper understanding by the recipients
8	Monitoring the execution by means of feedback, at which the process repeats itself

Wargames use different methods to introduce and define fog. John Setear provides a general method for defining and simulating fog in wargames.[3] The sources of battlefield fog are uncertainty about the enemy, enemy intentions, and enemy forces. The natural environment facing the commander and the behavior of the friendly forces are also sources of fog. The last source is the uncertainty about the underlying laws of war that govern the clash of arms on the battlefield. Setear stresses that the sources of fog affect commanders and environments at all levels. This result applies to strategic and tactical commanders and environments such as land, air, and the sea. Rules simulating uncertainty help the efforts of a wargamer to learn why events happen as they do and discover in the long run what characteristics a good commander must possess.

Hagelback and Johansson state the elements of fog in a real-time strategy game are the enemy bases' location, exploration of unknown terrain, and the unpredictability of explored terrain.[12] In most real-time strategy games, Mason states fog of war is a term used to describe the mechanic of making only limited portions of a game map viewable for a combination of the areas immediately surrounding the player's character and all allied units. Unit movement shifts these viewable zones and causes previously-visited areas to fade out of sight. This mechanic dynamically constrains the player's information, as areas outside their current viewing zones may contain active entities of interest. Progression requires an eventual confrontation with whatever lies in the surrounding fog, forcing players to think strategically about preparing for these unknowns.[13] These fog definitions focus on uncertainty about the terrain and the strength or location of enemy forces.

This section discussed fog from its definitions to its implementations in wargames. In this research, we define fog effects as critical areas that affect the information collection and dissemination to and from the commander and subordinates. These fog effects alter the development of perception for a decision-making entity and contribute to the enemy's uncertainty, friendly units, and the geographic environment. The next section discusses decision-making models and provides areas for scoping the decision-making process into a C2 context.

2.4. Decision-Making Cognitive Models

Decision-making cognitive models provide a process for perceiving the environment and executing decisions to shape the desired state. This section discusses Boyd's Observe Orient Decide Act (OODA) loop model and the Lawson Command and Control (C2) model and their decision-making processes.

The Observe Orient Decide Act (OODA) loop is a four-step approach to decision-making that models how a commander receives and filters information, formulates a decision, and then refines the decision as new information arrives.[14] United States Air Force Colonel John Boyd theorized the OODA loop to initially express an approach to tactical engagement, but later, he expanded the idea to incorporate broad strategic action.[15]

The OODA loop consists of four main phases.[14] The first phase, Observe, gathers information pertinent to the decision at hand. Information appears internally through feedback loops and externally through sensors or other information sources. The second phase, Orient, consists of destruction and creation and involves the most significant cognitive effort. The decision-maker destructs the main problem into smaller sub-problems that he or she understands. With the understood knowledge of the sub-problems, the decision-maker creates a plan of action using solutions for the sub-problems and combining them into a unified task. If the decision-maker fails to create a plan, he/she may concede defeat. The third phase, Decide, contemplates the plans available to the decision-maker. If the decision-maker can only construct one feasible plan, the decision is whether or not to execute. If there is more than one overall plan, the decision-maker chooses one as a course of action. The decision often involves weighing the risk or cost of a plan for its potential benefit. A single superior choice results in a confident and rapid decision. However, a few or many same-level decisions result in a longer decision-making time. The fourth phase, Act, represents the execution phase, where the decision-maker executes the chosen decision. The OODA loop phases execute simultaneously and not sequentially.[16] Figure 1 illustrates the model and its four phases.

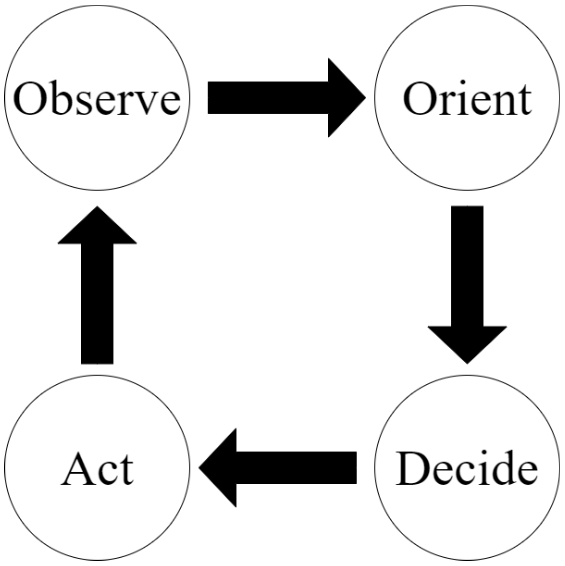


Figure 1. OODA Loop: The process consists of the four stages where the decision maker observes the environment, forms possible decisions, decides, and then executes the decision.[16]

In 1981, Joel Lawson showcased his cognitive model to create a military C2 system deriving from a higher national or political desire to maintain or change the status quo in a contested battlespace.[4] The C2 system incorporates the ability to perceive or sense the state of its environment, compares the perception to a specified desired state, and acts to force the environment into the desired state. This approach includes information gathering and processing as well as decision-making, similar to the OODA loop. The model functions at different levels of the chain of command, and it requires the commander to receive a visual representation of the environment, such as charts, maps, or electronic displays.

Figure 2 illustrates the Lawson C2 model process for a single unit. The individual unit senses the environment, processes the information, and then acts. Simultaneously, the commander up the chain manages multiple units and holds an array of sensors. The Lawson model may recursively use multiple commanders to form a command chain. However, the current model suffices. The next section discusses the framework AFSIM and its simulation capabilities.

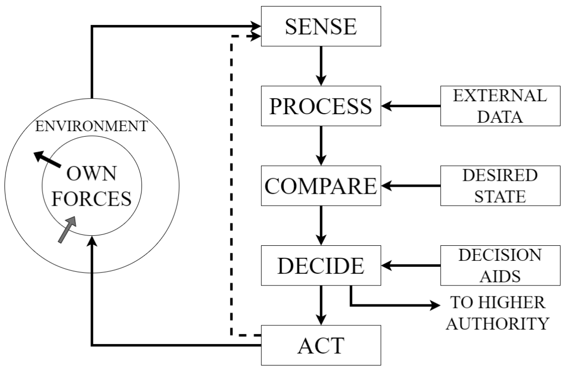


Figure 2. Lawson C2 Model: The Lawson C2 model process. The system perceives the environment, compares it to the desired state, and acts to force the environment into the desired state.[4]

The decision-making model in Figure 2 derives from the processes in the OODA loop and the Lawson C2 model. The OODA loop and the Lawson C2 model provide a basis for perceiving the environment, collecting the environmental data, forming a decision based on objectives, and executing the decision to produce an effect. The model in Figure 2 uses environment perception and information collection through sensors and information transmission through communication links. FIMM uses the derived model in Figure 2 to produce a process for introducing and manipulating fog in a C2 scenario. The next section discusses AFSIM and its capabilities to implement fog effects.

2.5. Advanced Framework for Simulation, Integration, and Modeling

AFSIM is an object-oriented, C++ simulation environment that prototypes customized engagement and mission level warfare simulations. The framework includes a class hierarchy of simulation objects, data-driven platforms, movers, sensors, communications networks, processors, weapons, and simulation observers. Simulation and Event classes exist to control time and event processing for AFSIM-based models and the logging of entity data. The framework supports a standard geospatial environment and terrain representation, a general-purpose scripting language to provide access to framework objects using text files, and the ability to execute any AFSIM application in both constructive (batch processing) and virtual (real-time) modes.[17] Figure 3 illustrates the high-level architecture. The services of the framework provide simulation functionality, while the components represent physical objects in the simulations.

A platform is a container data structure that holds physical and mental components and information, attributes, and

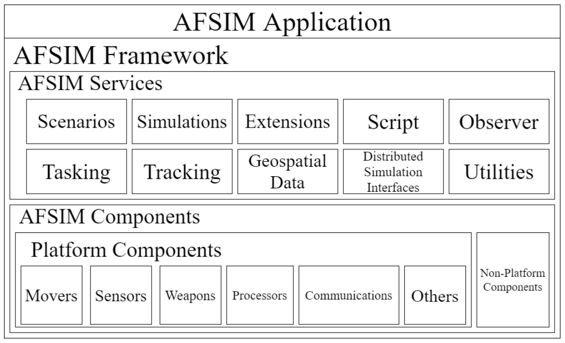


Figure 3. AFSIM High-Level Architecture: The framework consists of services and components that extend through extensions and plugins.[18]

links. Figure 4 displays the architecture of a platform and the relationships between its internal components.

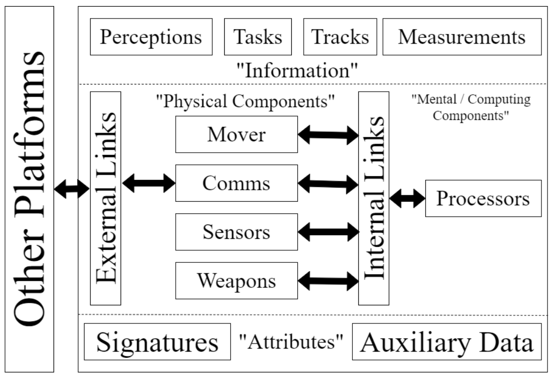


Figure 4. AFSIM Platform Architecture: A platform holds information about its environment and components to interact internally and externally. Each platform also has attributes to distinguish itself from other platforms.[18]

A sensor provides the ability for a platform to detect other platforms and their components. AFSIM contains several predefined sensors listed in Table 2. Sensors use azimuth and elevation limits to define their area of influence and hold the ability to form tracks.

The user may define error sigmas for sensors that form perceptions of objects, and the sigmas provide error margins on the position measurements from the sensor.[18] Sensors use commands in the framework to define their characteristics. Track reporting commands define the criteria for establishing a perception of another platform and the quality of information produced. Table 3 displays the sensor functions related to establishing and maintaining a perception of another platform.

The azimuth, elevation, range, and range rate error sigma commands allow the analyst to specify the standard deviation based on the true information the sensor reports. The error sigma is either a percent error of true range or an error value in units shown in the next two formulas.

$$R_p = V_t \pm V_t * P_e$$
$$R_p = V_t \pm V_e$$

Table 2. AFSIM Sensor Types.[18]

Type	Description
Acoustic	Simple passive acoustic sensor representing human hearing.
Composite	A sensor composed of other sensors.
EOIR	Baseline electro-optical or infrared sensor.
ESM	Baseline passive RADAR frequency detection sensor.
Geometric	Baseline sensor based on geometry.
IRST	Baseline infrared search-and-track sensor.
Optical	Simple electro-optical sensor.
OTH	Baseline over-the-horizon backscatter skywave RADAR sensor.
RADAR	Baseline RADAR sensor.
SAR	Baseline synthetic aperture RADAR.
Surface Wave RADAR	Surface-wave RADAR sensor.

R_p is the range of the perceived value, V_t is the truth value, V_e is the error sigma value with units, and P_e is the error sigma value in percent. The following is an example of the range error sigma function. A sensor reports a truth value of 800 meters, but the error sigma for the range is 20 meters. Therefore, the perceived values range from 780 to 820 meters. Next, the hits to establish and maintain the track are M of N hits where $N \leq 32$ due to framework constraints and $M \geq 1$, while $M \leq N$. To establish and maintain probability, provides an additional layer of whether the hits create or maintain the perceived perception object. The off/on commands provide options for introducing cutouts at scripted periods.

Table 3. AFSIM Sensor Functions.[18]

Function	Parameters
azimuth_error_sigma	[angle-value] OR [real-value] percent_of_true_range
elevation_error_sigma	[angle-value] OR [real-value] percent_of_true_range
range_error_sigma	[angle-value] OR [real-value] percent_of_true_range
range_rate_error_sigma	[speed-value]
hits_to_establish_track	M hits of N hits
hits_to_maintain_track	M hits of N hits
establish_track_probability	0.0 ... 1.0
maintain_track_probability	0.0 ... 1.0
off/on	

A communication object connects different platforms via transmitters, receivers, and antennas. Messages pass over internal links within a platform that shows communications among platform components. Messages pass over external links when platforms wish to communicate with each other.[18] Communication objects follow the 7-layer Open Systems Interconnection (OSI) model; when an object sends a message, the message passes through the object's protocol stack until the platform processes the information. The communication object serves as a node in the network and has various functions to deny, degrade, or change the data flow, as Table 4 shows.

Table 4. AFSIM Comm Functions.[18]

Function	Parameters
Physical Layer	
propagation_speed	[random-speed-reference]
transfer_rate	[random-speed-reference]
packet_loss_time	[random-speed-reference]
Datalink Layer	
channels	[integer-value]
queue_type	fifo OR lifo OR priority
queue_limit	[queue-limit]
purge_interval	[time-value]
retransmit_attempts	[integer-value]
retransmit_delay	[time-value]

Propagation speed determines how fast a message transmits between nodes; by default, propagation speed is the speed of light. Transfer rate is the amount of data transmitting over a given time, such as 100 bits per second, while packet loss specifies the delay to any transmission. These commands can interrupt the decision-making process by restricting the flow of information through the commander. The data link commands handle the scheduling and delivery of messages. The channels command supports simultaneous channels of transmission and, therefore, multiple information paths. The rest of the commands provide a structure for limiting and ordering the transmission and retransmission of data. These commands interact with the higher-level network membership the communication object may use. Network objects use addressing and links to connect platforms internally or externally. The communication object may attach to a router and gateway to add to another network.

3. Methodology

This section provides a process for identifying fog elements and developing the Fog Analysis Tool (FAT) in AFSIM. Then it defines a scenario to use for plugin verification and analysis.

3.1. Identifying and Manipulating Fog Elements

This section defines the methodology for identifying fog elements in a wargame scenario relating to Command and Control (C2). The derived process in Figure 5 focuses on making C2 decisions in a wargame environment specific to this research. The process derives from van Creveld's eight functions of C2 and the Lawson model.

This loop begins with checking if the number or state of the assets and communication links have changed. The assets provide the commander with information to perceive the environment and make decisions based on mission objectives. The commander sends orders to effect-generating assets such as weapon systems or jamming platforms. A human player or a scripted agent may control the commander in the wargame environment. Fog is a broad topic that spans the uncertainty of all information the leader may or may not have.[1] The C2 contextual decision-making process scopes fog into the information gathered through sensor platforms

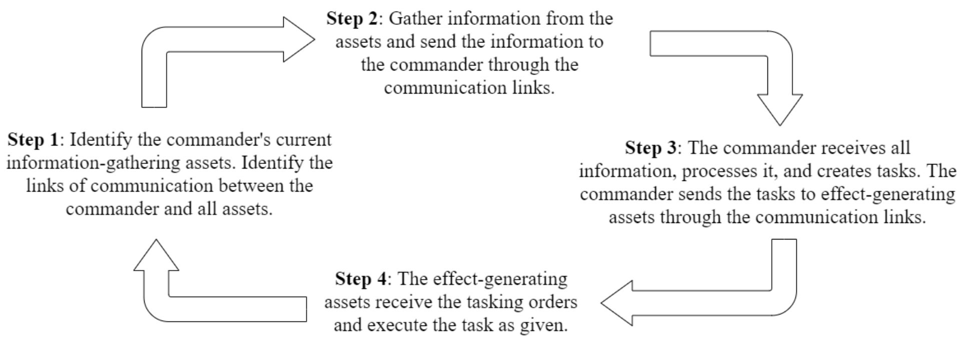

Step 2: Gather information from the assets and send the information to the commander through the communication links.
Step 1: Identify the commander's current information-gathering assets. Identify the links of communication between the commander and all assets.
Step 3: The commander receives all information, processes it, and creates tasks. The commander sends the tasks to effect-generating assets through the communication links.
Step 4: The effect-generating assets receive the tasking orders and execute the task as given.

Figure 5. C2 Decision-Making Process: The commander uses sensor platforms to gather information and effect-generating platforms to execute orders.

and the information the commander sends and receives through communication links.

Platforms perceive and collect data from the virtual environment via the sensors listed in Table 2. Sensor cutouts disrupt the platform's data flow, preventing the commander from receiving complete, usable, or accurate environmental data. Sensor noise or cutouts may provide the commander with an incomplete battlespace picture, allowing the decision-maker to fail the current objective. Sensor noise and cutouts create uncertainty about the enemy's actions and forces, the friendly units' behaviors, and the commander's environment

Communication links allow information to flow between the commander and subordinate platforms. The communication object uses message objects to contain a specific type of message for transit between nodes. Communication nodes connect through links that have limits. A severed link provides another type of information cutout and creates similar uncertainty to a sensor cutout. If a link becomes severed, a sensor needs redundant communication links to continue to function. The transfer rate affects the speed of the data flowing through the link. A dead link would fail to provide a commander with the critical information needed in the current decision cycle, and therefore the commander may form an ineffective decision. The delay in critical data may also occur when a communication buffer is full and waits to dispatch data.

Figure 6 illustrates the process for identifying and manipulating fog elements in a given C2 scenario. This model limits the process to a single commander in charge of sensor and effect-generating platforms. The information between the commander and subordinate platforms flows through communication links. Platforms may or may not hold links between them for multiple paths to the commander. The commander executes the decision-making process with the information perceived with the given configuration of cutouts, noise, and disruption on mission execution. The effect-generating platforms execute orders as they arrive via the communication link, and the post-analysis views how the commander accomplishes the mission objectives.

The Fog Identification and Manipulation Methodology (FIMM) uses the C2 decision-making process illustrated in

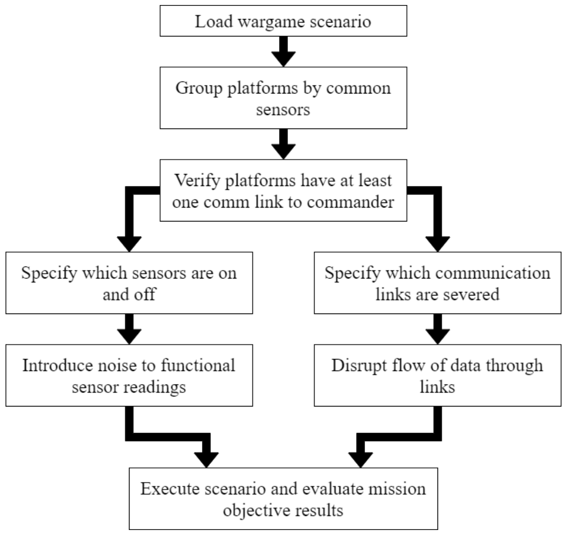

Load wargame scenario
Group platforms by common sensors
Verify platforms have at least one comm link to commander
Specify which sensors are on and off
Specify which communication links are severed
Introduce noise to functional sensor readings
Disrupt flow of data through links
Execute scenario and evaluate mission objective results

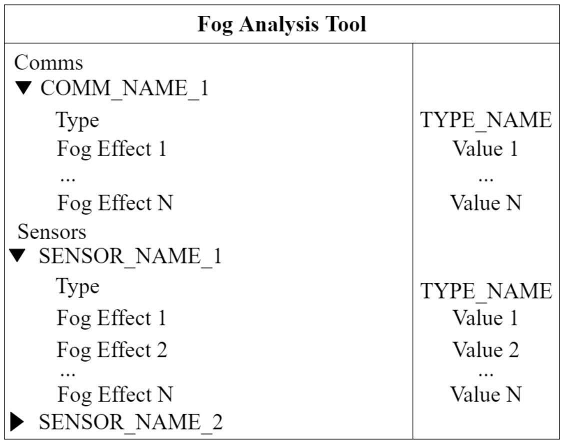
Figure 6. Fog Identification and Manipulation Methodology (FIMM): This process focuses on modifying sensor and communication objects in a wargame scenario. It focuses mainly on denial, disruption, and degradation of information flowing through the commander for decision making.

this section to focus on sensor platforms and communication links. The commander receives information from sensor platforms through communication links. This research assumes the effect-generating platforms execute orders as they receive them through the communication links. However, the information the sensors collect may be inaccurate, and the communication links may undergo issues to delay or corrupt the data. When subjected to inaccurate or delayed data, the subordinates execute unintended orders that may not match the commander's intent. The next section discusses an AFSIM plugin to implement FIMM from this section to allow an analyst to manipulate fog elements for sensor platforms and communication objects in an AFSIM wargame scenario.

3.2. Fog Analysis Tool

This section discusses the Fog Analysis Tool (FAT), an AFSIM plugin that displays the current sensors and communication components in a loaded scenario. Each component provides options for introducing and modifying fog effects.

The intent is for FAT to provide the wargame analyst a direct way in AFSIM to modify sensor and communication object values that relate to fog functions derived from FIMM. This implementation allows the analyst to modify multiple platforms and links at once while bypassing source file manipulation. The tool begins by parsing the current loaded scenario for all sensor and communication objects. The objects populate into a window in the AFSIM Wizard application in a tree view with a parameter-value scheme. The analyst directly modifies the parameter's value in the window, and the corresponding objects update accordingly. Figure 7 illustrates the conceptual view when interacting with the plugin.

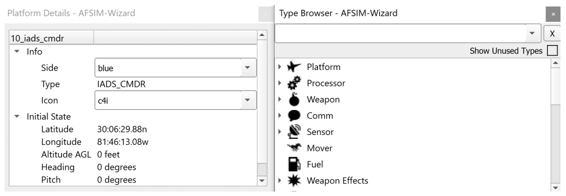


Fog Analysis Tool	
Comms	
▼ COMM_NAME_1	
Type	TYPE_NAME
Fog Effect 1	Value 1
...	...
Fog Effect N	Value N
Sensors	
▼ SENSOR_NAME_1	
Type	TYPE_NAME
Fog Effect 1	Value 1
Fog Effect 2	Value 2
...	...
Fog Effect N	Value N
▶ SENSOR_NAME_2	

Figure 7. FAT Conceptual View: The window displays two lists of communication and sensor objects. Each list displays all objects in use by the scenario. The analyst can change the values of each parameter instead of going through many source files.

The development of this plugin derives from the implementation of the Platform Details and the Type Browser plugins. Platform Details is a Warlock framework core class that implements into a Wizard child plugin class. The plugin searches for the currently selected platform of interest and displays a parameter-value list of information about the platform. Type Browser is a unique Wizard plugin that parses the project proxy object for all objects of all types in a given scenario. When the analyst changes anything in the scenario, the proxy and list of objects then update accordingly. The plugin separates objects into categories, such as platforms, weapons, sensors, and communications. However, this plugin contrasts with FAT because it does not display any information about the objects. The plugin only opens the file location of each object when activated. Figure 8 displays the graphical representation of the two plugins.

FAT displays all sensor and communication objects in a loaded scenario and provides options to modify their values. Therefore, the plugin updates when a new object



10_iads_cmdr

Info

Side	blue
Type	IADS_CMDR
Icon	c4i

Initial State

Latitude	30:06:29.88n
Longitude	81:46:13.08w
Altitude AGL	0 feet
Heading	0 degrees
Pitch	0 degrees

Figure 8. Platform Details and Type Browser: The left displays information about a currently selected platform. The right shows all objects in a given loaded scenario.[18]

is introduced or an existing object is modified. The plugin must also update when the project opens, closes, or switches to another project. FAT uses a custom interface object to retrieve data from the framework and update it to the source files' corresponding object. Figure 9 provides a Unified Modeling Language (UML) diagram of the plugin.

FAT implements framework functions that produce effects shown in FIMM. The functions of Tables 3 and 4 provide the effects needed. Table 3 shows that the azimuth, elevation, range, and range rate error sigma functions directly affect the sensors' measurements when perceiving another platform. The hits and probability functions affect the difficulty of creating and maintaining a perception of another platform. The off/on function allows for simulating cutouts of sensor feeds. In Table 4, the functions affect information flow and jamming perception. The functions that affect information flow determine when the commander or subordinates receive information. The jamming perception commands assist an operator to detect when communications are being jammed. FAT implements these functions to simulate noise, cutouts, and information flow. The following section uses an AFSIM scenario to verify the usefulness of the fog effects.

3.3. Verification Scenarios

This section examines an AFSIM scenario to verify the chosen fog functions' effectiveness. The scenario integrates multiple platforms into a Blue Integrated Air Defense System (IADS) group. The group consists of a commander platform that manages a radar company and a SAM battalion. The radar company uses two early warning (EW) radar platforms to detect when the opponent bombers enter a defined area (see Figure 10). The commander perceives when the opponent enters the battlespace and uses an acquisition radar (ACQ) to determine the opponent's position and a target tracking radar (TTR) to fire missiles from one of the three SAM sites. The Blue team protects n Blue tanks from m Red bombers. The results measure n_d the number of destroyed tanks out of n total tanks after scenario execution. The scenario fails when the Red team destroys more than 50% of the Blue tanks. The commander platform receives sensor data from its sensor platforms through communication objects linked into a Blue network. This scenario allows for the manipulation of multiple sensor platforms and communication links. The sensor to manipulate in this scenario is the TTR platform since it directly tracks and sends coordinates to fire upon the Red bombers. Noise introduction to other sensors would not affect the TTR platform's targeting, but cutouts to other sensors would affect if the commander activates the TTR

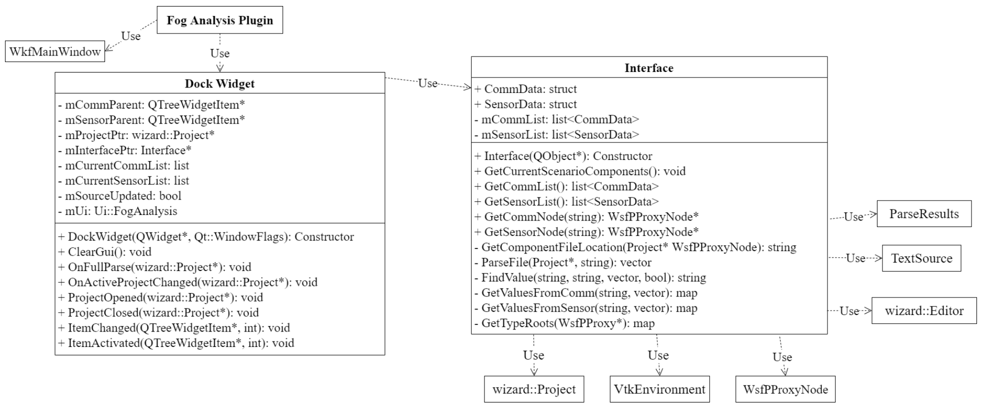

Fog Analysis Plugin
Use
WkfMainWindow
Use
Dock Widget
- mCommParent: QTreeWidgetItem*
- mSensorParent: QTreeWidgetItem*
- mProjectPtr: wizard::Project*
- mInterfacePtr: Interface*
- mCurrentCommList: list
- mCurrentSensorList: list
- mSourceUpdated: bool
- mUi: Ui::FogAnalysis
+ DockWidget(QWidget*, Qt::WindowFlags): Constructor
+ ClearGui(): void
+ OnFullParse(wizard::Project*): void
+ OnActiveProjectChanged(wizard::Project*): void
+ ProjectOpened(wizard::Project*): void
+ ProjectClosed(wizard::Project*): void
+ ItemChanged(QTreeWidgetItem*, int): void
+ ItemActivated(QTreeWidgetItem*, int): void
Use
Interface
+ CommData: struct
+ SensorData: struct
- mCommList: list<CommData>
- mSensorList: list<SensorData>
+ Interface(QObject*): Constructor
+ GetCurrentScenarioComponents(): void
+ GetCommList(): list<CommData>
+ GetSensorList(): list<SensorData>
+ GetCommNode(string): WsfPProxyNode*
+ GetSensorNode(string): WsfPProxyNode*
- GetComponentFileLocation(Project* WsfPProxyNode): string
- ParseFile(Project*, string): vector
- FindValue(string, string, vector, bool): string
- GetValuesFromComm(string, vector): map
- GetValuesFromSensor(string, vector): map
- GetTypeRoots(WsfPProxy*): map
Use
ParseResults
Use
TextSource
Use
wizard::Editor
Use
Use
Use
wizard::Project
VtkEnvironment
WsfPProxyNode

Figure 9. FAT UML Diagram: The plugin displays a dock widget in Wizard to the analyst. The widget uses an interface to retrieve and store data to and from the objects in the framework.

platform to fire upon Red bombers. AFSIM tasking messages are implemented as zero bit messages, and therefore the only communication fog effect to use in this scenario is cutouts of links.

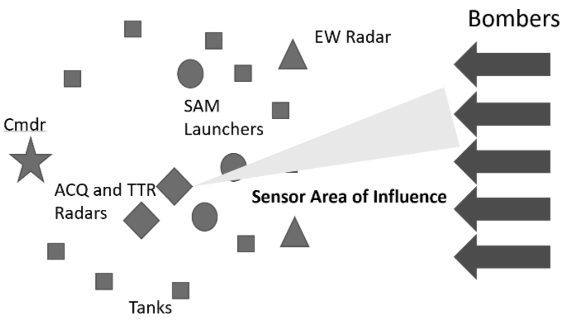


Figure 10. IADS Scenario: A Blue IADS group uses multiple SAM launchers to defend 10 Blue tanks against five Red bombers.

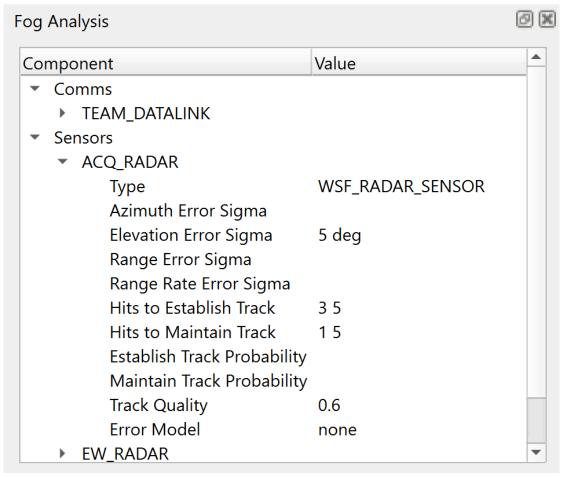
The next section provides plugin functionality results and scenario verification results. For each simulation run, a fog effect is measured at a specific level for x simulations as a different seed for random generation results. The result of n_d number of tanks destroyed is the mean taken over x simulations. The fog effects to tanks destroyed find the bounds of fog functions where the scenario passes and fails.

4. Results

This section discusses FAT functionality and current software development progress. The scenario designed to use FAT shows the scenario's success versus the specified level of fog effect.

4.1. FAT Functionality

The FAT plugin successfully compiles for AFSIM version 2.7.0 using CMake version 3.18.0 and Visual Studio Community 2019. The plugin populates sensor and communication objects in the window at project parse and loads each object's parameter-value pairs. When the name of the object is activated, the Wizard application opens up the editor file. The analyst may manipulate the values of each fog effect for each pair, and it updates accordingly. Figure 11 illustrates the plugin in the AFSIM Wizard application.


Fog Analysis

Component	Value
Comms	
TEAM_DATALINK	
Sensors	
ACQ_RADAR	
Type	WSF_RADAR_SENSOR
Azimuth Error Sigma	
Elevation Error Sigma	5 deg
Range Error Sigma	
Range Rate Error Sigma	
Hits to Establish Track	3 5
Hits to Maintain Track	1 5
Establish Track Probability	
Maintain Track Probability	
Track Quality	0.6
Error Model	none
EW_RADAR	

Figure 11. Fog Analysis Tool: The window provides options for the analyst to manipulate fog effects conveniently.

4.2. IADS Verification Scenario

AFSIM simulates the scenario 100 times for each level of fog effect and measures n_d the number of tanks destroyed for each run. The result of tanks destroyed is the mean taken across all runs. Over 50% of tanks destroyed is a failure for the Blue team and a Red team's success. The scenario uses 10 Blue tanks and 5 Red Bombers. Figure 12 displays the number of tanks destroyed versus the azimuth and elevation error sigmas as percent error increases.

The scenario entered failing conditions between 2.5 and 3 percent error for both the azimuth and elevation error sigma functions. These functions affect where the radar perceives the bomber platform is. The error skews the Red bomber's targeting and causes the SAM launcher to launch a missile to the wrong coordinates.

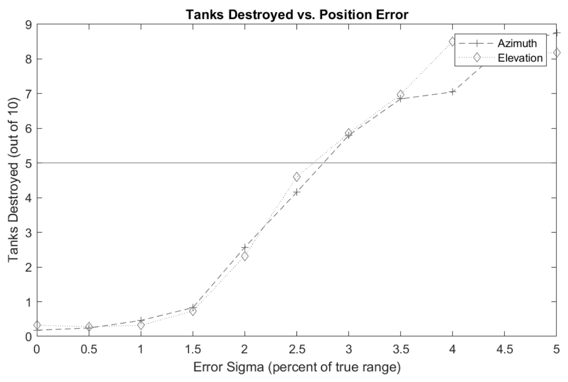


Figure 12. Azimuth and Elevation Error Sigmas: The tolerance for error is low due to the low viewing area of the TTR platform.

The AFSIM function for range error sigma did not produce results for percent values, but it produced unit values in Figure 13. The range error sigma function not producing results for percent values is a possible framework error. As the unit values for range error sigma increased, the tanks destroyed showed steady growth. As the range error increases, the missiles miss the Red bombers, and Blue tanks are destroyed. The AFSIM function for range rate error sigma did not produce any usable results.

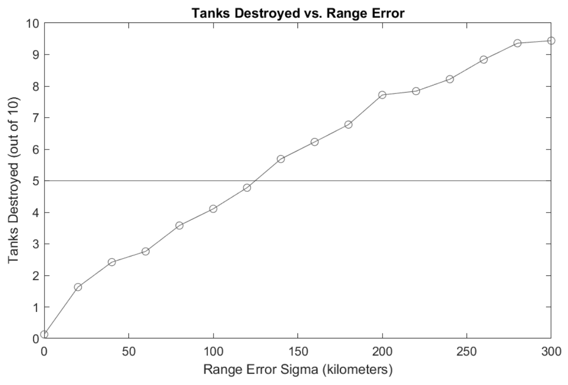


Figure 13. Range Error Sigma: As range error increases, the scenario enters the failure stage at a steady rate.

Figure 13 illustrates the scenario entering failure conditions around 125 km of error. A growth in range error leading to failing conditions seems consistent with the trend that more error correlates to a failing scenario, but the missile blast radius is 100 meters. It takes a very high amount of error compared to the blast radius, possibly due to a framework error or because the SAM launchers hold multiple missiles for redundancy.

AFSIM holds a maximum value for M of 32 hits to establish and maintain an object's perception or track. The hits to establish the track cause the SAM sites to fail, locking on to the Red bombers, and more Blue tanks are destroyed as the critical hits. In Figure 14, the hits to maintain track do not produce any significant effect.

The hits to maintain track function may not produce any statistical results when there are no other fog effects present in the scenario. Further analysis of a scenario with multiple fog effects is needed to verify the hits to maintain track function is a good fog effect.

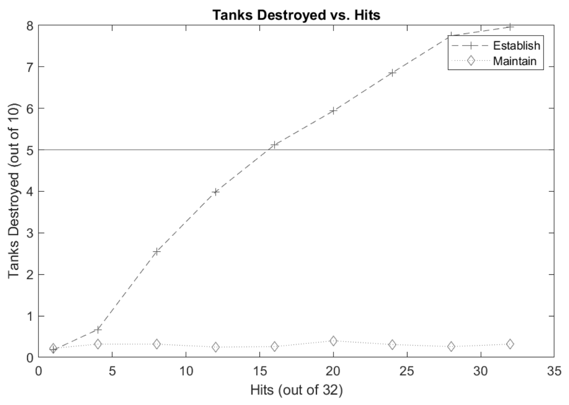


Figure 14. Hits to Establish and Maintain Tracks: The scenario reaches failure in the hits to establish track as the required hits increase out of 32.

A low probability of establishing and maintaining a perception or track causes the SAM sites to fail to lock on. As the probability increases in Figure 15, the Red bombers are destroyed, and the Blue tanks are protected.

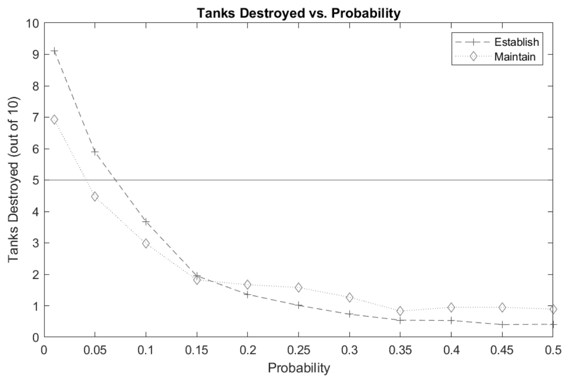


Figure 15. The probability to Establish and Maintain Tracks: The lower probabilities end up in scenario failure (over five tanks destroyed) but change to passing scenarios as they increase.

A slow propagation speed, 1 meter per hour compared to the light speed, simulates communication link cutouts. When no links fail, the simulation produces a mean of 0.25 tanks destroyed. The simulation produces 100% tanks destroyed when either the commander, a company leader, a sensor platform, all SAM launchers, or the full network links fail. When one out of three SAM launchers fails, the mean tanks destroyed increases to 0.75. When two of the three launchers fail, the mean tanks destroyed increases to over 50%, and the scenario fails.

The packet loss time function introduces a delay to every transmission by a communication object. This delay directly impacts the transmission of data from a node in the network. A higher delay, shown in Figure 16, produces a higher impact on the decision-making process and allows the Red bombers to destroy more Blue tanks.

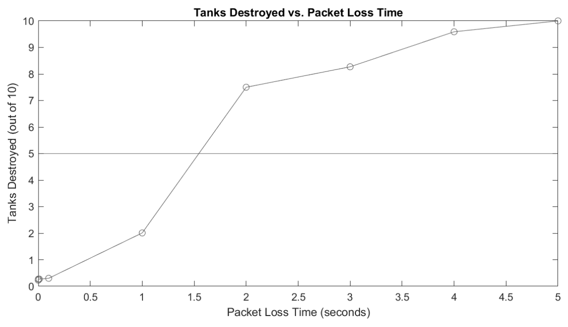


Figure 16. Packet Loss Time: This function introduces a delay to every transmission by the communication object. The delay produces more uncertainty as it increases.

The transfer rate function affects the flow of information between communication objects in the network. The function directly impacts the transmission of data through bottlenecking. This function requires a definition of message sizes. The communication objects transmit track, or perception, messages defined at a size of 50 kilobytes. Figure 17 illustrates the tanks destroyed versus transfer rate with the given size for track messages.

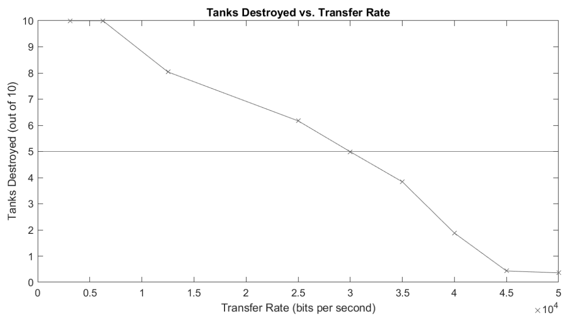


Figure 17. Transfer Rate: The communication objects transmit 50-kilobyte track messages throughout the network with the Red bombers' perception information. As the transfer rate increases to 50 kilobytes per second, the Blue IADS group detects and eliminates Red bombers before reaching the targets.

The queue limit function provides another bottleneck for transmitting messages. A low limit drops messages instead of slowing the transmission rate, resulting in retransmission and a longer wait time to reach the destination. Table 5 provides the results for varying the queue limit.

Table 5. Queue Limit Results.

Queue Limit (messages)	Mean Tanks Destroyed (of 10)
0	10
1	8.04
2	4.03
3	1.22
4	0.4
5	0.22

A zero queue limit rejects all incoming messages and simulates a cutout similar to the propagation speed. As the queue limit increases, more messages are transmitted through the communication links, and the IADS group functions successfully. The number of channels, purge interval, queue type, retransmit attempts, and retransmit delay functions do not produce any useful results for this scenario. An avenue for further analysis is to perform a combined analysis of these functions with the queue limit function's addition to view how the functions are dependent on each other. The next section gives a conclusion and provides a discussion for future work for FIMM and FAT.

5. Conclusions and Future Work

This research creates FIMM, a process for introducing and manipulating battlefield fog in a wargame scenario, and FAT, an AFSIM plugin that implements FIMM and allows the analyst to apply various fog effects to AFSIM scenarios. FAT implements into AFSIM 2.7.0 by parsing sensor and communication objects, populating a window with fog effects, and allowing the analyst to alter the effects. Each of the scenarios provides a bound of fog where the scenario stops succeeding and starts to fail. These results may differ for different types of scenarios.

FIMM provides a process to introduce uncertainty into the decision-making process through data collection and transmission. Data collection through sensor objects relates to the fog of war by controlling the commander's perception of the environment. More significant uncertainty in the collected data leads to greater obscurity in the perception. Data transmission through communication objects creates an issue where a commander may not receive the data fast enough to form an optimal decision. FAT verifies noisy data collection leads to wrong decisions. Also, data transmission link cutouts lead to incomplete or denied environmental data for the commander.

The results of the IADS scenario enforces the idea of redundancy in military operations. The multiple SAM launchers allow the commander to continue the mission when one fails. The results show that fog effects impact scenario execution by bringing the scenario to failure as the effects increase or decrease. The hits to maintain the track function does not seem to produce any noticeable results for the scenarios, but a scenario could exist where the function does. Introducing noise to the sensor feeds affects the targeting of enemy opponent platforms.

The U.S. military focuses on virtual decision-making to iterate possible courses of action (COAs) to choose the optimal path. However, fog affects decision-making in unpredictable ways. FIMM and FAT serve as a baseline to incorporate fog effects into decision-making analysis. The vision is the decision-making agent accounts for various fog levels in the process and uses the information to choose the optimal path. Military analysts may also use FIMM and FAT in their scenarios to view the bounds of fog a decision-maker can handle before failing the objectives.

This research requires more scenarios to test how other fog elements affect decision-making processes, such as jamming, delaying, or dropping network packets in a communication network. These scenarios prove the effectiveness of the fog functions in AFSIM, where the noise can overcome the decision-making process to cause the scenario to fail.

A possible research area for FIMM is to introduce tampering or intercepting information in transit through communication links. Tampering and intercepting information in transit requires opponent intervention and access to the network. This area may require a separate framework specialized for cyber warfare.

The purpose of FIMM is to provide an abstract method for identifying and quantifying levels of uncertainty for consideration in a decision-making agent. The agent would follow a decision-making process such as the OODA loop and uncertainty when forming decision paths to accomplish mission objectives. FIMM allows the analyst to help train an agent to overcome fog when subjected to various fog effects. FAT is one possible implementation, and the process spreads over multiple frameworks. Another possible research area is, therefore, implementing FIMM on another military simulation framework. A few possible frameworks are Next Generation Threat System (NGTS) by the U.S. Navy, One Semi-Automated Forces (OneSAF) by the U.S. Army, or the Unreal Engine by Epic Games.

Disclaimer

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Air Force, the Department of Defense, or the U.S. Government.

Acknowledgements

This research was supported by the Air Force Research Laboratory and the 711th Human Performance Wing. They provided the resources and support to conduct these experiments.

References

1. Clausewitz C, Howard M, Paret P, and Brodie B. *On War*, Princeton University Press; 1984.
2. West TD and Birkmire B. AFSIM: The Air Force Research Laboratory's Approach to Making M&S Ubiquitous in the Weapon System Concept Development Process, Journal of Cyber Security and Information Systems 2020, 7(3):50.
3. Setear J. *Simulating the Fog of War*, RAND Corp, Santa Monica, CA 1989.
4. Lawson JS. Command Control as a Process, *IEEE Control Systems Magazine* 1981. 1(1):5-11.
5. Creveld MV. *Command in War*, Harvard University Press; 1985.
6. Dixon R. Clausewitz, Center of Gravity, and the Confusion of a Generation of Planners, Small Wars Journal 2015; 20.
7. Naval Air Warfare Center Aircraft Division. NEXT GENERATION THREAT SYSTEM (NGTS). NAVAIR Public Release 2018-930.
8. Parsons D, Surdu J and Jordan B. OneSAF: A next generation simulation modeling the contemporary operating environment. In Proceedings of Euro-simulation Interoperability Workshop 2005 Jun 27.
9. U.S. Department of Defense. DoD Dictionary of Military and Associated Terms, *Joint Publication* 2020: 1-02.
10. Joint Staff. Joint Publication (JP) 1 Doctrine for the Armed Forces of the United States. Government Printing Office, Washington DC 2017.
11. Joint Staff. Joint Publication 3-13 Information Operations. Government Printing Office, Washington DC 2014.
12. Hagelback J and Johansson SJ. Dealing with fog of war in a real time strategy game environment. In: *IEEE Symposium On Computational Intelligence and Games*, Dec 15 2008. pp.55-62. IEEE.
13. Mason Z. *Trick of the Light: A Game Engine for Exploring Novel Fog of War Mechanics*. Master's Thesis. Worcester Polytechnic Institute, UK, 2018.
14. Rule JN. A Symbiotic Relationship: The OODA Loop, Intuition, and Strategic Thought. US Army War College; 2013 Mar.
15. Chunn C and Whitt JE. John Boyd and the "OODA" Loop (Great Strategists). https://warroom.armywarcollege.edu/special-series/great-strategists/boyd-ooda-loop-great-strategists/ (2019, accessed 16 November 2020).
16. Tighe T, Hill R and McIntyre G. A Decision for Strategic Effects: A conceptual approach to effects based targeting, Chronicles Online Journal 2000.
17. Clive PD, Johnson JA, Moss MJ, et al. Advanced Framework for Simulation, Integration and Modeling (AFSIM)(Case Number: 88ABW-2015-2258). In: *Proceedings of the International Conference on Scientific Computing (CSC)* 2015, p. 73. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).
18. Air Force Research Laboratory. AFSIM 2.7.0 Documentation, Advanced Framework for Simulation, Integration, and Modeling, October 2020.

Author Biographies

LIEUTENANT DILLON TRYHORN is a student at the Air Force Institute of Technology (AFIT) in Wright-Patterson Air Force Base in Ohio, USA. He is currently working for a master's degree in computer engineering and he has an interest in computer simulation software.

MAJOR RICHARD DILL is an Assistant Professor of Computer Science at the Air Force Institute of Technology (AFIT), Wright-Patterson AFB, Ohio USA. He received a B.S. in Computer Science from the University of Maryland at College Park in 2004, and both an M.S. in Computer Science in 2008 and a PhD in 2018 from AFIT. Major Dill's research interests include computer security, algorithms, and artificial intelligence.

DOUGLAS D. HODSON is an Associate Professor of Computer Engineering at the Air Force Institute of Technology (AFIT), Wright-Patterson AFB, Ohio USA. He received a B.S. in Physics from Wright State University in 1985, and both an M.S. in Electro-Optics in 1987 and an M.B.A. in 1999 from the University of Dayton. He completed his Ph.D. at the AFIT in 2009. His research interests include computer engineering, software engineering, real-time distributed simulation, and quantum communications. He is also a DAGSI scholar and a member of Tau Beta Pi.

MICHAEL R. GRIMAILA (BSEE 1993; MSEE 1995; PhD 1999, Texas AM University) is a professor and head of the Department of Systems Engineering and Management at the Air Force Institute of Technology, Wright-Patterson Air Force Base in

Ohio, USA. He is a member of Tau Beta Pi, Eta Kappa Nu, and the Association for Computing Machinery, as well as a Senior Member of the IEEE, and a Fellow of the Information System Security Association.

CHRISTOPHER MYERS is an experienced computationally oriented cognitive scientist who leverages modeling and simulation to improve the foundational understanding of human cognition and to develop intelligent systems capable of partnering with humans within teams. This research agenda is manifest across two broad areas of current research: synthetic teammates and physiocognitive models. Within each area, Dr. Myers has led multiple interdisciplinary research teams across multiple collaborations within AFRL, academia, and industry.

V. Journal of Information Warfare

Applying Fog Analysis Tool to AFSIM Multi-Domain CLASS scenarios

D Tryhorn[1], R Dill[1], D Hodson[1], M Grimaila[1], C Myers[2]

[1]Graduate School of Engineering and Management
Air Force Institute of Technology
Wright-Patterson AFB, Ohio, United States

E-mail: dillon.tryhorn@afit.edu; richard.dill@afit.edu; douglas.hodson@afit.edu; michael.grimaila@afit.edu

[2]711th Human Performance Wing: Cognitive Science, Models, & Agents Branch
Air Force Research Laboratory
Wright-Patterson AFB, Ohio, United States

E-mail: christopher.myers.29@us.af.mil

Abstract: Military leaders must consider uncertainty in decision-making to accomplish mission objectives. Wargames and military simulations model real-world scenarios to understand potential outcomes for alternative courses of action to support decision-making. This research applies the Fog Analysis Tool (FAT) (Tryhorn et al. 2021) to Cyber Land Air Sea Space (CLASS) scenarios for the Advanced Framework for Simulation, Integration, and Modeling (AFSIM). The CLASS scenario structure supports fog analysis in multi-domain operations within AFSIM scenarios. The CLASS result trends support the hypothesis that increasing fog effect levels disrupt the decision-making and tasking processes for all warfighting domains. A crossover event occurs from success to failure for each fog effect as the level increases for all Cyber Land Air Sea Space (CLASS) scenarios. Therefore, FAT should be used for incorporating fog in multi-domain wargame scenarios to aid commanders in course of action analysis and planners in modifying scenario difficulty.

Keywords: *Fog of War, Wargaming, Multi-Domain Operations, AFSIM*

Introduction

Military leaders must consider the information uncertainty when making decisions in the battlespace to accomplish objectives. Carl von Clausewitz (1873) compares this uncertainty to fog: meteorological fog physically obscures a person's view as war fog figuratively conceals the battlespace. Fog may cause a military leader to make decisions that might cause an unintended tactical effect. Therefore, identifying uncertainty sources enables the decision-maker to better account for battlespace fog when forming courses of action to execute a plan.

The decision-making process for this research involves command and control (C2) in military operations. A military leader follows a command chain structure where he or she reports to a higher authority and commands subordinates (Department of Defense 2020). The leader uses a decision-making process to accomplish objectives. Military analysts use wargames to simulate war scenarios and associate outcomes to commander decisions. Some wargames adopt a C2 structure where the teams each hold their C2 systems to form decisions and disseminate the information to subordinate platforms.

This research applies the Fog Analysis Tool (FAT) (Tryhorn *et al.*, 2021) for Multi-Domain Operations (MDO) and Joint All-Domain Command and Control (JADC2). MDO provides the United States (U.S.) Armed Forces with a strategy to execute simultaneous and sequential operations with continuous integration of capabilities across all warfighting domains to defeat adversaries in the 2025 – 2050 timeframe (Feickert 2020). JADC2 is the concept to connect sensors from all of the U.S. military services into a single network (Hoehn 2020). JADC2 supports emerging ideas, such as the sensing grid concept and unified joint C2 infrastructures (Williams and Sotiriadis 2019). This research analyzes FAT's impact on multiple warfighting scenarios to form a Cyber Land Air Sea Space (CLASS) set in AFSIM. This begins by setting up CLASS scenarios with Blue and Red teams that each leverage platforms for multiple domains. Then, FAT simulates the scenarios with different levels of fog for each run and records a success condition. The analysis includes viewing the percentage of success at a certain fog level and finding the crossover value if it exists. A crossover value reinforces the idea that fog disrupts the normal process of the scenario. The goal is to use FAT to introduce fog in wargame scenarios and use result trends to support fog effectively in disrupting the decision-making process across all warfighting domains.

Background and Related Work

This section provides a review of topics related to the information environment, the MDO and JADC2 concepts, C2 decision-making, and the AFSIM framework. The Fog Introduction and Manipulation Methodology (FIMM) and Fog Analysis Tool (FAT) are the tools used to introduce and manipulate levels of fog into AFSIM scenarios to view the impact on success rate. This section provides the necessary topics on fog for joint warfighting in the information domain so that FAT can use CLASS scenarios for multi-domain operations.

The information environment

The information environment interconnects the primary warfighting domains: land, air, sea, space, and cyberspace. It is responsible for collecting, processing, and disseminating information between the domains. Joint Publication 3-13 (2014) splits the information environment into three categories: the physical, informational, and cognitive (see Figure 1). The physical dimension comprises command and control (C2) systems, decision-makers, and supporting infrastructure that enable individuals and organizations to create effects. This dimension includes the physical platforms and interconnected communication networks. The informational dimension supports the storing and processing of information. Actions here affect the content and flow of information. The cognitive realm refers to individuals' or groups' information processing, perception, and judgment; it measures how ideologies, beliefs, morals, and education influence these elements. In turn, measuring their influences provides an understanding of how to influence the decision-maker's mind best and create the desired effects. The introduction of fog effects directly impacts the data collection of physical platforms and transmission of communication links in the physical dimension. The physical dimension's fog affects information data flow through the informational dimension and decision-making processes in the cognitive dimension. It is essential to understand that these layers build upon each other because introducing uncertainty at critical times and locations could negatively impact how the commander fights in the information environment.

MDO and JADC2

MDO, a U.S. Army concept, defines the joint force as the Air Force, Army, Marines, Navy, and Space Force and the warfighting domains as air, land, maritime, space, and cyberspace (Feickert 2020). MDO describes how the joint force could defeat a near-peer adversary in all domains. When deterrence fails, the joint force would then penetrate, disintegrate, disrupt, degrade, or destroy the enemy's anti-access and area denial systems. It exploits the resulting freedom of maneuver by defeating enemy forces in all domains and forces the battlespace, including the information environment, to a U.S favorable state. Dominance in all domains maintains enemy deterrence.

MDO emphasizes the joint force and all-domain dominance, but JADC2 provides a method to unify information across domains and aid commander decision-making to accomplish the goals of MDO. JADC2 is a U.S. Department of Defense (DoD) concept to support all U.S. military branches' sensor fusion. JADC2 envisions a cloud-like environment for the joint force to share intelligence, surveillance, and reconnaissance data, transmitting across many communications networks, to enable faster decision-making. The goal is to enhance commander decision-making by collecting data from numerous battlefield sensors, processing the data using artificial intelligence algorithms to identify targets, then recommending the optimal weapon to engage the target. Each U.S. military branch is attempting to implement aspects of JADC2, such as the Air Force's Advanced Battle Management System (ABMS), the Army's Project Convergence, and the Navy's Project Overmatch (Hoehn 2020).

MDO supports the use of assets from multiple domains to accomplish objectives. While the CLASS scenarios focus on a primary domain, they employ platforms from secondary domains. The CLASS scenarios use sensors and communciations to collect battlefield data and to send the data to a tasking unit similar to the concepts of JADC2. A joint force commander uses assets and sensor data from multiple domains to enforce adversarial deterrence. The FAT tool aids the simulation of war scenarios by introducing battlespace fog into data collection and transmission through sensors and communications. FAT uses uncertainty in CLASS scenarios to demonstrate how fog can affect MDO and affect the JADC2 process.

Uncertainty areas with C2

Military C2 is the exercise of authority and direction by a properly designated commander over assigned and attached forces to accomplish the mission (DoD 2020). Joint Publication 1 lists the tenets of C2, including clearly defined authorities, timely decision making, and information management (Joint Staff 2017). The tenets of C2 directly support the chain of command's goal to have clear, quick, and accurate communication between the commander and subordinates. These tenets also provide areas to introduce and view the impact of fog effects. The Van Creveld (1985) functions of C2 in Table 1 provide a detailed process of perceiving the environment and shaping it to accomplish objectives.

Function	Description
1	Collecting information on own forces, the enemy, the weather, and the terrain
2	Finding means to store, retrieve, filter, classify, distribute, and display the information
3	Assessing the situation
4	Analyzing objectives and finding alternative means for achieving them
5	Making a decision
6	Planning based on the decision
7	Writing and transmitting orders as well as verifying their arrival and proper understanding by the recipients
8	Monitoring the execution via feedback, at which the process repeats itself

Table 1. Van Creveld's (1985) eight defining characteristics of C2

The first and second functions provide an uncertainty where the commander may collect and store inaccurate information. Functions three through six depend on the commander's decision-making process and exceed the scope of this research. The seventh function introduces an uncertainty where the recipients may not receive the proper commands, and the eighth function relies on an accurate perception of the commander's environment. The functions of C2 provide options for identifying areas of uncertainty in C2 processes. Fog effects leverage these areas to disrupt the decision-making process of the commander. The uncertainty areas of C2 provide a structure to set up CLASS scenarios and entry points to manipulate fog in those scenarios.

Related frameworks to simulate fog across multi-domain operations

The U.S. military uses different simulations to analyze war scenarios specific to an agency's goals. The Advanced Framework for Simulation, Integration, and Modeling (AFSIM) is one of the Air Force Research Laboratory's (AFRL) solutions to simulate war scenarios in multiple warfighting domains and levels of fidelity (Clive *et al.*, 2015). AFSIM uses commands to affect perception data and communication link transfer. The U.S. Army utilizes the framework One Semi-Automated Forces (OneSAF) to achieve war scenario simulation similar to AFSIM. However, OneSAF has more support for land-based warfare with soldiers, equipment, and logistical supplies (Parsons, Surdu, and Jordan 2005). The U.S. Navy develops the framework Next Generation Threat System (NGTS) for multi-domain war simulation. NGTS provides rivaling capabilities similar to AFSIM, such as communications, sensors, and data visualization (NAWCAD 2018). The three software packages provide capabilities to simulate war scenarios in multiple warfighting domains. However, FAT is designed for AFSIM due to how its sensor and communication objects work. While FAT is implemented on AFSIM, it is important to understand the method can be applied to any war simulation framework.

Related methods for simulating and modeling fog

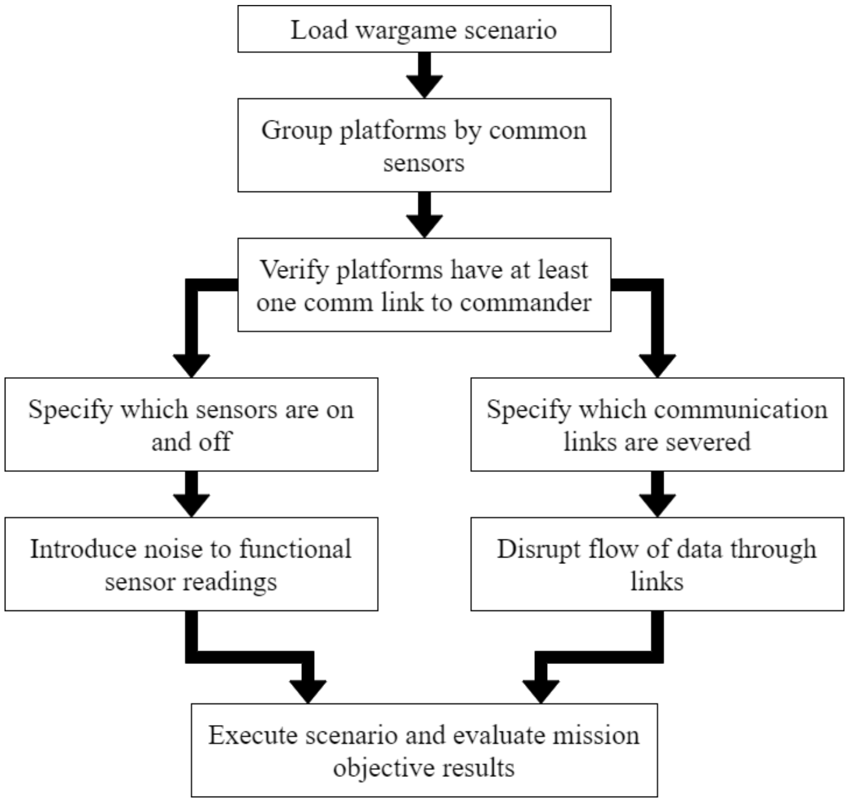
John Setear (1989) provides a general method for defining and simulating fog in wargames. The sources of battlefield fog are uncertainty about the enemy, such as intentions and forces. The natural environment facing the commander and the behavior of the friendly forces are also sources of fog. The last source is the uncertainty about the underlying laws of war that govern arms' clash on the battlefield. Setear stresses that the sources of fog affect commanders and environments at all levels. This result applies to strategic and tactical commanders and environments such as land, air, and the sea. Rules simulating uncertainty help the efforts of a wargamer to learn why events happen as they do and discover in the long run what characteristics a good commander needs to win in modern warfare.

Berry *et al*. (2000) focus on modeling information and decision-making with uncertainty in sensors. Their model catalogs the sources of uncertainty. The sensors have performance limitations relating to power, range, noise, and signal return strength. The environmental effects, such as clutter, cloud cover, or ionospheric state, determine if the system detects a target and the errors associated with the measurement. Multiple sensors may disagree with the estimates of speed or location. The uncertainty associated with the information increases over time unless the system continually receives truth evidence. The model uses probabilistic analysis to account for uncertainty and the unpredictable behavior of targets.

The Data Uncertainty Engine (DUE) is another software tool for assessing environmental data uncertainties for hydrological flow (Brown and Heuvelink 2007). The sources of uncertainty stem from imprecise measurements, sampling, interpolation, and positional errors. The engine incorporates the uncertainty levels into the probabilistic analysis. While these methods provide areas to introduce uncertainty, FIMM focuses on the wargame process with sensors and communication objects that disrupt the C2 process and commander decision-making.

Fog Identification and Manipulation Methodology

FIMM, as shown in Figure 1, is a process that provides options to introduce and manipulate fog effects with sensor and communication objects in a wargame scenario. It focuses mainly on denial, disruption, and degradation of information flowing through the commander for decision making. This model limits the process to a single commander in charge of sensor and effect-generating platforms. The commander receives information from sensor platforms through communication links. Platforms may or may not hold links between them for multiple paths to the commander. However, the information the sensors collect may be inaccurate, and the data in the communication links may become delayed or denied. The commander executes the decision-making process with the perceived environmental information.


Load wargame scenario
Group platforms by common sensors
Verify platforms have at least one comm link to commander
Specify which sensors are on and off
Specify which communication links are severed
Introduce noise to functional sensor readings
Disrupt flow of data through links
Execute scenario and evaluate mission objective results
Figure 1: The Fog Identification and Manipulation Methodology

FIMM provides an avenue for producing fog effects in a generalized C2 scenario that leverages sensors and communication links. The implementation FAT leverages AFSIM to introduce and manipulate fog effects in framework scenarios.

The information environment provides a domain, and Creveld's C2 functions provide a process for interfering with information collection and flow. MDO and JADC2 support multi-domain warfighting and using sensor fusion to aid decision-making. The military's current frameworks each have their capabilities for simulation war scenarios, but fog generating tools are scarce. Setear provides general methods for wargames, DUE focuses on hydrological flow uncertainty, and Berry models uncertainty for surveillance sensors. FIMM provides a unique process for introducing fog using sensors and communications to interrupt C2 processes. However, FIMM’s effectiveness in multi-domain scenarios is unverified and the methodology provides a system for evaluating FAT in multiple warfighting domains to support FIMM.

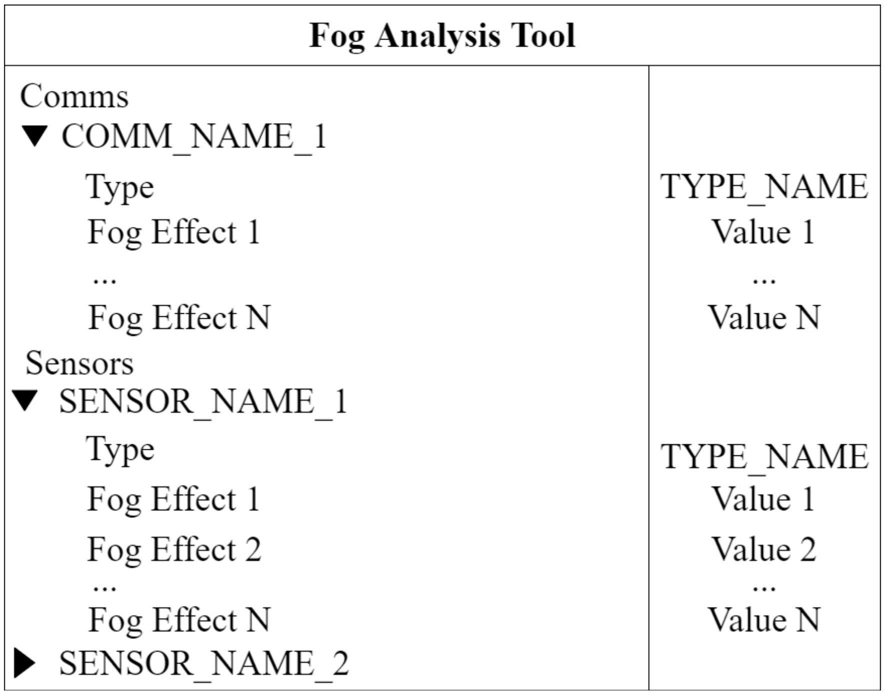
Methodology

This section describes the Fog Analysis Tool (FAT) for AFSIM and introduces the Cyber Land Air Space Sea (CLASS) set of scenarios for analyzing the impact of fog effects using FAT in AFSIM. FAT provides the necessary capabilities to generate and tune fog level for success rate analysis, while the CLASS scenarios represent a distinct combat domain so that analysts can learn FAT's impact on winning conditions. Each domain scenario represents a likely conflict, where a friendly team (e.g., Blue team) battles an adversary (e.g., Red team). The U.S. Air Force performs air superiority, interdiction, and strategic bombing missions regularly. Some weapon systems leverage the GPS for positioning coordinates, and the U.S. utilizes cyber operations for cross-domain operations (Mattis 2018).

Each CLASS subsection describes the problem, set up in AFSIM, and success conditions. FAT identifies the sensor and communication objects and measures the success rate at different fog levels. The expectation for each scenario is to produce a crossover event between success and failure as the fog effect's value increases. The goal is to introduce CLASS scenarios into the FAT process for multi-domain analysis.

Fog Analysis Tool

FAT is implemented as an AFSIM plugin that displays the current sensors and communication components in a loaded project, provides methods for an analyst to introduce and modify fog effects in both sensor and communication components. FAT intends to provide the wargame analyst a direct way to modify sensor and communication component values related to fog functions derived from FIMM. The tool begins by parsing a loaded project for all sensor and communication objects. The objects populate in an AFSIM window as a tree (see Figure 2) with a parameter-value scheme. The analyst can modify the parameter's value in the window, which updates the corresponding environment objects.



Fog Analysis Tool	
Comms	
▼ COMM_NAME_1	
Type	TYPE_NAME
Fog Effect 1	Value 1
...	...
Fog Effect N	Value N
Sensors	
▼ SENSOR_NAME_1	
Type	TYPE_NAME
Fog Effect 1	Value 1
Fog Effect 2	Value 2
...	...
Fog Effect N	Value N
▶ SENSOR_NAME_2	

Figure 2: The Fog Analysis Tool Conceptual View

FAT uses AFSIM specific features to produce effects to implement FIMM. The sensors affect information collection, including position error, track hit, and probability. The sensor affects the establishment of perceptions and other platforms' position reports (i.e., entities and players). The communication objects model information transmission between links, including transfer rate, queuing, and packet transmission delays. The CLASS scenarios view the impact of FAT fog effects in different combat domains.

FAT capabilities in AFSIM

FIMM provides a general process for manipulating fog in sensors, and communications and FAT leverages AFSIM to implement fog capabilities. The communication capabilities include link cutouts, transfer rate adjustment, and packet delays. Very low propagation speeds and queue limits of zero simulate link cutouts. The transfer rate capability allows a bit flow rate as the parameter. Packet delays introduce a time delay to every packet sent through the network. The sensor capabilities begin with positioning error for azimuth, elevation, and range shown in Equation 1. The positioning error is either a percent error P_e or a unit value V_e. The positioning error range R_p is standard deviation based on the truth value V_t of the platform perception.

$$\begin{gathered} R_p = V_t \pm V_t * P_e \\ R_p = V_t \pm V_e \end{gathered} \tag{1}$$

The next capability is the hits to establish a track or perception of another platform for M of N hits. AFSIM allows a maximum of 32 hits for M and N. The last two capabilities allow a probability for establishing and maintaining a track for another platform. The probability is a normalized value from 0.0 to 1.0. The communication capabilities affect cutouts for links and information flow between and through links. The sensor capabilities affect information collection for positioning and perception establishment and maintenance.

Validation and summary

The application of CLASS scenarios to FAT in Figure 3 is a direct process. The method begins by loading a scenario. Then FAT simulates a fog level configuration for *n* runs. A script parses the results and stores the percentage of success at the given fog level value. The results store the pairs for each fog level for analysis.

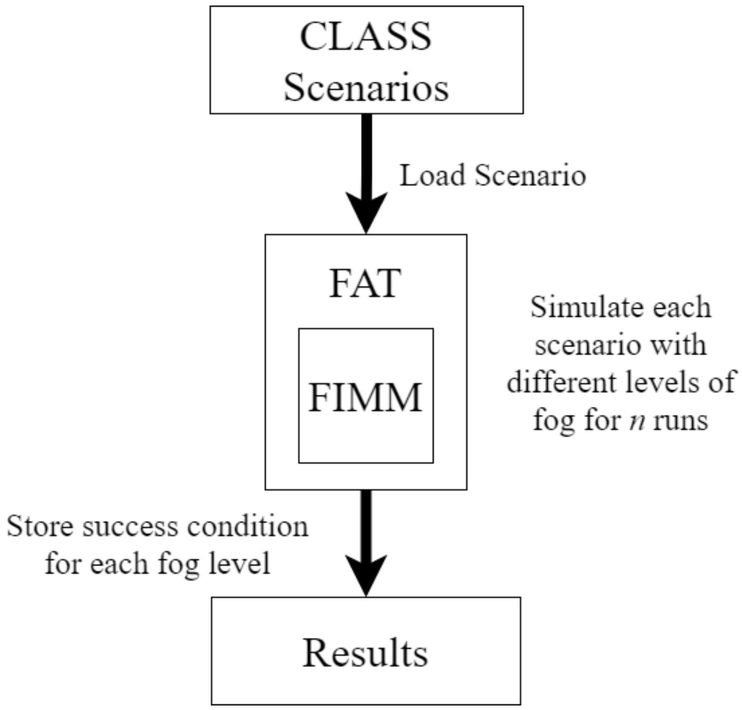

CLASS Scenarios
Load Scenario
FAT
FIMM
Simulate each scenario with different levels of fog for *n* runs
Store success condition for each fog level
Results

Figure 3: Applying FAT to CLASS Scenarios

FAT is a direct implementation of FIMM in AFSIM. FAT leverages AFSIM framework capabilities to produce fog effects. The CLASS scenarios application to FAT is a direct process that stores success percentage for each fog level. The next section provides the structure of each scenario and an analysis of the results they produce from FAT.

CLASS Scenarios and Results

The Cyber Land Air Sea Space (CLASS) set of scenarios showcase FAT usage in multiple warfighting domains to assess fog effects' impacts. These scenarios use AFSIM for simulation and results. Each domain scenario focuses on a primary domain but uses secondary elements from others to demonstrate MDO.

Each run uses a different seed, or number, for a pseudorandom number generator for result generation. The results illustrate the fail rate for each scenario over 100 simulation runs. When a scenario fails more than 50% of the time, the results produce a crossover event and label the fog effect level as the maximum fog tolerance level. The communication results show the different trends in the CLASS scenarios, excluding the air scenario. Track and tasking messages are defined as having a size t_s of 50 kilobits. The message size directly impacts the transfer of data through the link. The sensor results do not produce results in the land domain.

Cyberattack

The first scenario in Figure 4 occurs in the cyber domain scenario. The Blue team consists of a Blue cyber command post that uses a ground radar to detect incoming Red aircraft. The objective for the Blue team is to defend the Blue target. The Red team is a single Red uncrewed aerial vehicle (UAV) that attacks the Blue target with preset coordinates and air-to-ground missiles. Without interference from FAT, the Blue team locks on to the Red UAV when it comes within ten nautical miles of the cyber command post, and the post sends a cyber-attack to the Red UAV. The Red UAV has an altitude of 2500 feet and drops the air-to-ground missile at a set time no matter the altitude. The cyber-attack lowers the UAV to an altitude of 1000 feet. The UAV provides a cyber counter-measure to increase the altitude back to 2500 feet within 20 seconds, but it gives the Blue team enough time for the Red UAV to drop the air-to-ground missile and miss the Blue target. FAT introduces positioning error, hit, and probability effects in the radar sensor from the Blue radar site that locks on to the Red UAV. FAT tests cutouts, transfer rates, and packet delays for the communication link between the Blue radar site and cyber command post. The Blue team's objective is to defend the target. The target's destruction results in a mission failure.

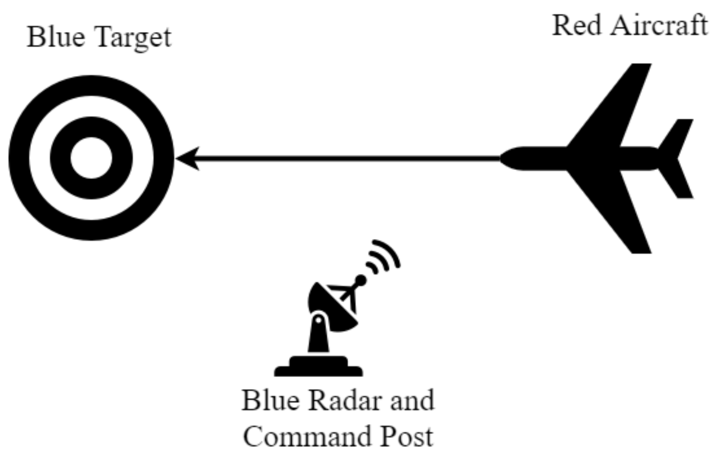


Figure 4: Cyber Scenario

The communication results impact data transfer, packet delay, and communication link cutouts (see Table 2). A link cutout between the Blue cyber command and ground radar results in the cyber command post not receiving the location and the Red aircraft's alert. The cyber command post does not launch a cyber-attack, and the simulation fails 100% of the time.

Capability	Parameter	Blue target destruction
No Fog Effect	N/A	0%
Link Cutout	Cyber Command to Ground Radar	100%
Transfer Rate	12500 bits/s	46%
--	6250 bits/s	69%
--	3125 bits/s	100%
Packet Delay	1 s	6%
--	2 s	14%
--	3 s	22%
--	4 s	35%
--	5 s	36%
--	6 s	43%
--	7 s	63%

Table 2: Cyber Scenario Communication Results

The transfer rate at $\frac{t_s}{16\ s}$ or 3,125 bits/s produces a fail rate of 100%. However, once the parameter reaches $\frac{t_s}{4\ s}$ or 12,500 bits/s, the fail rate reduces to 46%. The packet delay reaches failure between the parameters of six and seven seconds. The transfer rate directly affects the flow of information through the links, while packet delay affects the rate at which information enters the links. Both capabilities affect the time the command post receives the coordinates from the radar site, and therefore they affect the timing of the cyber-attack.

The sensor results impact the manipulation of positioning error, perception hits to establish, and perception probability of track establishment and maintenance (see Table 4). The azimuth and elevation error and the maintain track probability capabilities did not produce usable results.

Capability	Parameter	Blue target destruction
No Fog Effect	N/A	0%
Range Error	1.5 km	36%
--	2 km	49%
--	2.5 km	63%
--	3 km	71%
Hits to Establish Track	32 of 32 hits	38%
Establish Track Probability	0.03	41%
	0.02	53%
	0.01	74%
	0.001	99%

Table 4: Cyber Scenario Sensor Results

This scenario begins to fail around 2 kilometers of range error or 0.02 probability of establishing a track for the Blue radar sensor. The other positioning error capabilities did not produce results because the effect-generator is a cyber-attack. The cyber-attack in AFSIM does not require an accurate target position but requires a track of the target. The establishment of perception is essential for this scenario. If the Blue radar cannot establish a track, it fails to launch the cyber-attack.

Land strike

The second scenario in Figure 5 occurs on land, where the Blue team consists of a ground target evading destruction. The Red team uses a battalion commander that manages one 160mm rocket platform and one 240mm rocket platform. The Red commander uses preset coordinates for the target and sends tasking orders to the ground rocket platforms through communication links to launch rockets at the target. FAT manipulates the communication links of the Red team. The Blue team's objective is to evade destruction successfully. The Blue target's destruction is a mission fail. Zero sensors are used, and results are excluded from this scenario.

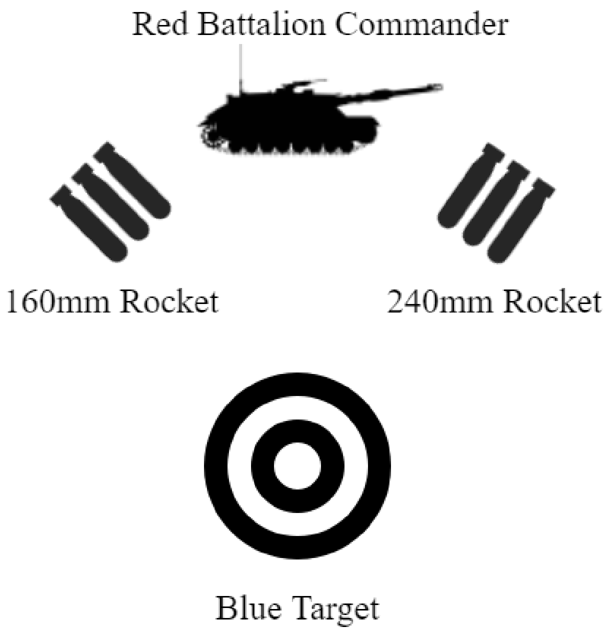


Figure 5: Land Scenario

The communication links between the commander and the subordinates transfer the position data. When a link is disabled, the effect-generating platform does not strike the target. This effect results in a zero percent fail rate. The Blue target remains stationary, so the packet delay delays the Red order, but the Red launchers still hit the stationary target. The transfer rate effects cause a 100% success rate at less than 100 bits per second. This scenario passes when the communication links are cut or similar to cut, such as queue limits of zero. The results show that stationary launchers primarily fail to hit a stationary target when information fails to transmit between platforms.

Air search-and-destroy

The third scenario in Figure 6, set in the air domain, happens during an aircraft search-and-destroy mission. The Blue team consists of a single surface to air missile (SAM) site and a high-value asset. The Red team is a single bomber that targets the Blue high-value asset with preset coordinates and GPS bombs. The Blue SAM site uses an attached sensor to detect and lock on to the Red bomber. When the Red bomber reaches the Blue SAM site radius, about 11.5 miles, the Blue SAM site fires a SAM to the Red bomber's coordinates and destroys the target. FAT analyzes the sensor of the SAM site. Since the site acts as the information gathering source and the effect generator, the Blue site does not use a communication link in this scenario, and FAT does not analyze communications. The Blue team's objective is to defend the high-value asset. The asset's destruction results in a mission failure. Zero communication links are used, and results are excluded from this scenario.

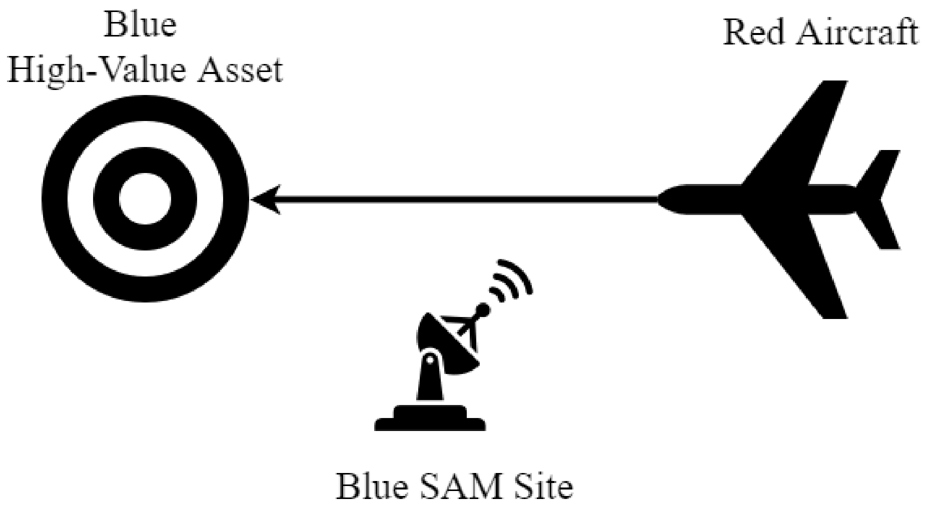


Figure 6: Air Scenario

The sensor results impact all positioning capabilities, the perception hits to establish, and the perceived probability of track establishment and maintenance (see Table 5). The positioning error capabilities increase the failure rate as the error increases, and this is the expected result. The error tolerance for failure is greater than 60% of the correct range. This value is large because the SAM site holds multiple missiles for redundancy.

Capability	Parameter	Blue target destroyed
No Fog Effect	N/A	0%
Azimuth Error	30%	2%
--	40%	17%
--	50%	46%
--	60%	59%
Elevation Error	60%	34%
--	70%	47%
--	80%	72%
Range Error	25 km	26%
--	30 km	57%
--	35 km	74%
Hits to Establish Track	32 of 32	36%
Establish Track Probability	0.05	25%
--	0.025	50%
--	0.01	74%
--	0.001	99%
Maintain Track Probability	0.25	39%
--	0.2	54%
--	0.1	88%
--	0.01	100%

Table 5: Air Scenario Sensor Results

The hits to establish track capability does not exceed the fail conditions of 50% and above. The establish and maintain track probability act as expected. A lower probability creates a lower chance for the SAM site to perceive the Red bomber and prevent the Blue high-value asset's destruction.

Sea evasion

The fourth scenario in Figure 7 focuses on the sea or maritime domain. The Blue team is a single ship evading the Red team that consists of a destroyer and a helicopter. The Red helicopter uses a search radar to detect the Blue ship and reports the information back to the destroyer. The destroyer processes the information and sends a missile to the Blue ship for destruction. FAT areas of analysis include the communication link between the Red destroyer and helicopter and the search radar sensor the helicopter utilizes for detecting the Blue ship. The Blue team's objective is to evade destruction from the Red destroyer. The Blue ship's destruction results in a mission failure.

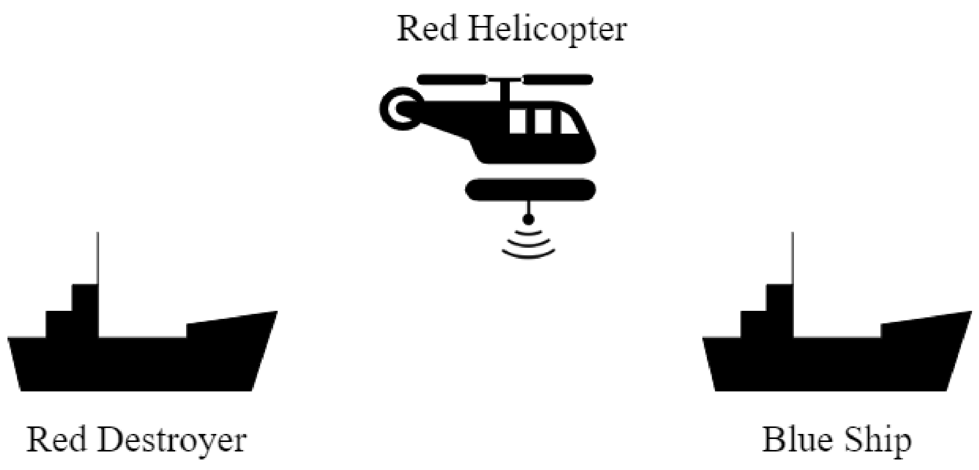


Figure 7: Sea Scenario

The communication results demonstrate an impact similar to the cyber scenario where FAT affects data transfer, packet delay, and communication link cutouts (see Table 3). A link cutout between the radar seeking helicopter and the destroyer causes the scenario to pass 100% of the time since the Blue ship evades destruction.

Capability	Parameter	Blue ship destruction
No Fog Effect	N/A	100%
Link Cutout	Helicopter to Destroyer	0%
Transfer Rate	1000 bits/s	68%
--	500 bits/s	0%
Packet Delay	1 s	100%
--	10 s	100%
--	20 s	74%
--	50 s	74%
--	60 s	0%

Table 3: Sea Scenario Communication Results

The sea communication results differ from the cyber scenario because the transfer rate capability requires less throughput for the simulation to fail. The simulation begins to fail between 500 and 1000 bits/s rather than 12500 bits/s in the cyber scenario. The packet delay does not affect the Red team until around 60 seconds of packet delay. The high packet delay and low transfer rates are required for mission success because the Blue ship moves at ten knots. A higher Blue ship speed would allow a lower packet delay and higher transfer rate to stay in success conditions.

The sensor results show an impact in the same areas as the air scenario (see Table 6), but the elevation error and maintain track probability capabilities did not produce any usable results. The sea scenario focuses on using the FAT fog effects to disrupt the Red team and protect the Blue ship.

Capability	Parameter	Blue ship destruction
No Fog Effect	N/A	100%
Azimuth Error	20%	62%
--	25%	42%
Range Error	100 km	59%
--	125 km	51%
--	150 km	43%
Hits to Establish Track	28 of 32	100%
	32 of 32	0%
Establish Track Probability	0.005	33%
--	0.01	60%

Table 6: Sea Scenario Sensor Results

The azimuth error tolerance is lower than the air scenario, but the range error tolerance is higher than the air scenario. The hits to establish track produces flips from a full fail rate to a zero percent fail rate between 28 to 32 of 32 hits. The established track probability shows a similar trend to the other scenarios.

Space GPS denial

The fifth scenario in Figure 8 uses assets from the space domain to simulate the global positioning system (GPS). The Blue team consists of a single radar site. The Red team consists of a surveillance aircraft and a striker aircraft. The Red team detects the Blue radar site and sends a striker that launches a GPS-guided missile at the Blue radar. The Blue radar detects the incoming missiles, jams the GPS signal, and causes the missile to miss the Blue radar. The atmosphere consists of many satellites that provide GPS. FAT manipulates the communication network of the GPS satellites and the sensor capabilities of the Blue radar site. The Blue team's objective is to deny the missile, so the Blue radar site does not become destroyed. The Blue radar site destruction results in a mission failure.

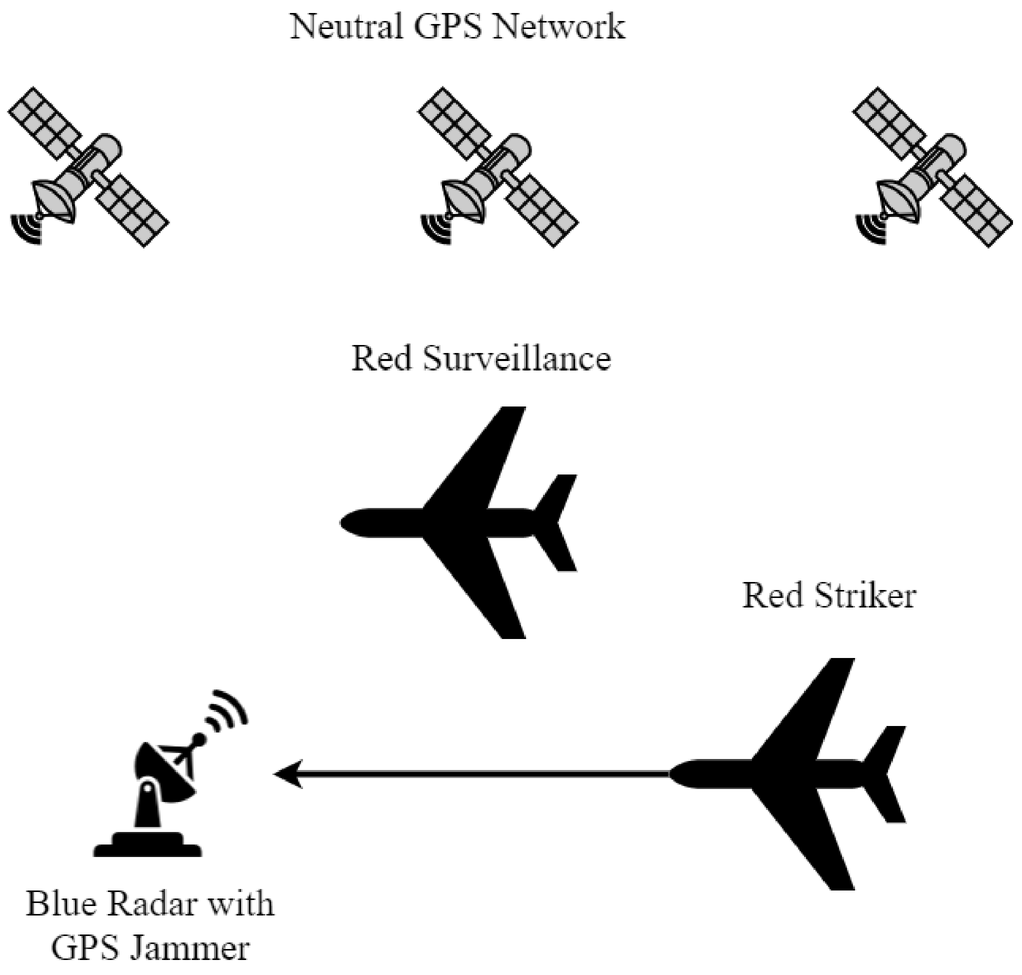


Figure 8: Space Scenario

The scenario communications use GPS satellites to provide GPS missile coordinates to a Blue target. The GPS network's failure causes the missile to miss the Blue radar 100% of the time and allows the scenario to pass. The packet delay and transfer rate capabilities do not produce any useful results.

The sensor results do not show any impact of fog effects for FAT except for a sensor cutout. Without the sensor, the Blue radar site cannot detect the incoming missile, and it fails to jam the GPS onboard the missile. Otherwise, the site jams the missile's GPS, and the site remains safe.

CLASS analysis

All results follow the trend where increased fog levels lead to disruption and degradation of tasking and decision-making where the Blue target destroys more than half the time. In the communication results, the two scenarios that impact positioning fog are the cyber and sea scenarios. The positioning fog capabilities demonstrate a low transfer rate, and a high packet delay disallows tasking messages to transfer in the link successfully. The air scenario does not use a communication link. The land scenario tracks a stationary target and therefore has an impact only from link cutouts. The space scenario impacts GPS failure since the missile depends on GPS to destroy the Blue target.

The sensor results from each domain scenario vary. The cyber scenario does not recognize an impact with some positioning capabilities, but it shows an impact with range error. The air scenario demonstrates an impact with all three positioning error capabilities. The sea scenario does not show an impact with the elevation positioning error. The land scenario does not use a sensor, but the space scenario only shows an impact with a sensor cutout. The hits and probability capabilities each show an impact for the cyber, air, and sea scenarios. Between the three, each fog effect's levels reach different values to achieve 50% of targets destroyed. The variation of these values depends on variables that were not accounted. The multi-domain nature of the CLASS scenarios introduces the need to account for variables in all warfighting domains. To reduce this complexity, FAT must analyze each scenario on a case-by-case basis. The fog level trends support the idea FAT effectively disrupts the tasking and decision-making process in multiple domains.

The results demonstrate that sensors and communication links may collect and transmit data independently of a platform's domain. However, the levels of fog effects required to disrupt and degrade the tasking depend on many environmental factors, including speed of target, viewing field of sensor, and quantity of ammunition.

Conclusion

The CLASS scenario analysis presented in this paper demonstrates the FAT tool's capability to vary multiple variables to analyze a wargame scenario from five warfighting domains. CLASS scenarios produce a structure for an analyst to focus on a specific operation domain while incorporating secondary domain assets to form a multi-domain operation. FAT provides options for introducing and manipulating fog effects in AFSIM specific to sensors and communication links for CLASS scenarios. The fog effects produce a similar trend in each domain where more fog or error interrupts a specific team's information collection and transmission and causes a crossover event to mission failure as fog increases. Similar trends reinforce the idea that manipulating information collection and transmission can disrupt C2 processes and cause mission failure. The results support the idea FAT is effective in multi-domain operations and should be used for future war scenario analysis.

Future Work

FAT is a manual process to test and evaluate different fog levels in AFSIM for all sensor and communication objects in a given scenario. FAT should be converted into a test suite to test all possible fog levels in AFSIM in an automated fashion. The test suite conversion would require a few scripts to change fog levels at given intervals over a specified number of simulation runs and use command-line arguments for the AFSIM mission application.

There is a large number of variables to consider when creating a scenario to evaluate with FAT. The tool must be used on a case-by-case basis to introduce fog effects and evaluate their impacts. There is a need to identify additional uncertainty variables and produce a standardizing battlespace fog beyond sensors and communication links.

FAT is an implementation of FIMM in AFSIM. FIMM should be implemented on other wargaming or war simulation frameworks to gain more results and verify that the method effectively introduces and manipulates fog in sensors and communication links.

FIMM intends to produce more realism for wargame scenarios. A wargame decision-making agent could incorporate the uncertainty capabilities from FIMM to account for battlespace fog when processing information and making decisions. The incorporation of FIMM into an agent's decision-making process would require a more probabilistic analysis of fog.

Disclaimer

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Air Force, the Department of Defense, or the U.S. Government.

Acknowledgments

This research was supported by the Air Force Research Laboratory and the 711th Human Performance Wing. They provided the resources and support to conduct these experiments.

References

Berry, P.E., Hall, D.L., Fogg, D.A.B. and Fok, V., 2000. Modelling information and decision-making under uncertainty for integrated surveillance operations. In *International Command & Control Research Technology Symposium, Canberra, Australia, Department of Defense Command & Control Research Program*.

Brown, J.D. and Heuvelink, G.B., 2007. The Data Uncertainty Engine (DUE): A software tool for assessing and simulating uncertain environmental variables. *Computers & Geosciences*, *33*(2), pp.172-190.

Clive, P.D., Johnson, J.A., Moss, M.J., Zeh, J.M., Birkmire, B.M. and Hodson, D.D., 2015. Advanced Framework for Simulation, Integration and Modeling (AFSIM)(Case Number: 88ABW-2015-2258). In *Proceedings of the International Conference on Scientific Computing (CSC)* (p. 73). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Feickert, A., 2020. *Defense Primer: Army Multi-Domain Operations (MDO)*. Congressional Research Reports.

Hoehn, J.R., 2020. *Joint All Domain Command and Control (JADC2)*. Congressional Research SVC Washington United States.

Williams, S. and Sotiriadis, J., 2019. *Air Force Enabling Concept: Multi-Domain Command and Control (MDC2)*. Air Force Warfighting Integration Capability.

Joint Staff. 2014. *Pub 3-13, Joint Doctrine for Information Operations*. Joint Chiefs of Staff.

Joint Staff, 2017. Joint Publication (J.P.) 1 Doctrine for the Armed Forces of the United States. *Department of Defense*. Government Printing Office, Washington DC.

Mattis, J., 2018. *Summary of the 2018 national defense strategy of the United States of America*. Department of Defense Washington United States.

Naval Air Warfare Center Aircraft Division (NAWCAD), 2018. *Next Generation Threat System (NGTS)*. NAVAIR Public Release 2018-930.

Parsons, D., Surdu, J. and Jordan, B., 2005, June. OneSAF: A next generation simulation modeling the contemporary operating environment. In *Proceedings of Euro-simulation interoperability workshop*.

Setear, J.K., 1989. *Simulating the Fog of War*. Rand Corp, Santa Monica, CA.

Tighe, T., Hill, R., and McIntyre, G., 2000. A Decision for Strategic Effects: A conceptual approach to effects based targeting, *Chronicles Online Journal*.

Tryhorn, D., Dill, R., Hodson, D., Grimaila, M., and Myers, C., 2021. Modeling Fog of War Effects in AFSIM, *Journal of Defense Modeling and Simulation*. Manuscript submitted for publication.

U.S. Department of Defense (DoD), 2020. *DoD Dictionary of Military and Associated Terms*. Joint Publication: 1-02.

Van Creveld, M., 1985. *Command in war*. Harvard University Press.

Von Clausewitz, C., 1873. *On war* (Vol. 1). London, N. Trübner & Company.

VI. Conclusion

Fog of war in wargames occurs when a decision-maker does not have complete information about the environment. Fog stems from a lack of knowledge about the enemy, enemy intentions, and enemy forces. Fog can also come from the natural environment and the behavior of friendly forces. For example, a unit that searches the entire map for a target may become exhausted after many searches. Multiple units that search for the target may divide the search time. The Fog Identification and Manipulation Methodology (FIMM) provides a novel method for manipulating fog effects in wargames related to sensors and communication links. The Fog Analysis Tool (FAT) provides an implementation of FIMM in the Advanced Framework for Simulation, Integration, and Modeling (AFSIM). Introducing fog into sensors and communication links affects the information the commander receives from information-collecting platforms and disseminates to effect-generating platforms. The introduction and manipulation of fog may cause the commander to fail the scenario. Higher levels of fog effects tend to lead to a higher fail rate. This trend also appears when analyzing various multi-domain scenarios.

6.1 Future Work

A test suite for the search algorithm would gather more results for analyzing how the map configuration can affect the parallel algorithm's speedup compared to the sequential algorithm. The test suite requires a map generator to randomly arrange the tiles on the map while keeping the assumptions in order.

FAT is a convenient tool for AFSIM analysts to change various levels of fog for a simulation run. However, the use of FAT is highly manual. The process could be converted into a test suite where the tool automatically takes an AFSIM scenario and finds the sensor objects and communication objects, simulates each object at different fog levels and the number of runs, and provides a statistical report on the success/fail rates. The test suite could eliminate the need for finding a standardized method to use FAT, and every scenario can be treated uniquely.

The integration of FAT into the Warlock application would allow for real-time analysis of fog effects. The analyst would be able to change a fog effect value in real-time and view how it affects the scenario's success. FAT with real-time analysis could be used for fine-grained tactical strategy testing.

FAT is an implementation of FIMM in AFSIM. FIMM should be implemented on other wargaming or military simulation frameworks to gain more extensive results and verify that the method effectively introduces and manipulates fog in sensors and communication links. A couple of possible frameworks are Next Generation Threat System (NGTS) by the U.S. Navy and One Semi-Automated Forces (OneSAF) by the U.S. Army.

FIMM intends to produce more realistic wargame scenarios. A wargame decision-making agent could incorporate the uncertainty functions from FIMM to account for 'battlespace fog' when processing information and making decisions. The incorporation of FIMM into an agent's decision-making process would require a more probabilistic analysis of fog.

Bibliography

1. Dillon Tryhorn, Laurence Merkle, and Richard Dill. Navigating an enemy contested area with a parallel search algorithm. *Journal of Parallel and Distributed Computing*, Manuscript submitted for publication, 2021.

2. Dillon Tryhorn, Richard Dill, Douglas Hodson, Michael Grimaila, and Christopher Myers. Modeling fog of war effects in afsim. *Journal of Defense Modeling and Simulation*, Manuscript submitted for publication, 2021.

3. Dillon Tryhorn, Richard Dill, Douglas Hodson, Michael Grimaila, and Christopher Myers. Applying fog analysis tool to afsim multi-domain class scenarios. *Journal of Information Warfare*, Manuscript submitted for publication, 2021.

4. Carl Clausewitz; Michael Howard; Peter Paret; and Bernard Brodie. *On War*. Princeton University Press, 1984.

5. Lt Gen Dash Jamieson. *Next Generation ISR Dominance Flight Plan*. 2018.

6. Matthew B. Caffrey Jr. *Wargaming 101: Why & How*, March 2020.

7. Decision Games. What Is Wargaming? Accessed August 2020.

8. Sea Air and Space Law. Operational law handbook, 2015.

9. Robert Dixon. Clausewitz, center of gravity, and the confusion of a generation of planners. *Small Wars Journal*, October 2015.

10. US Air Force. DoD Dictionary of Military and Associated Terms. *Joint Publication 1*, 2.

11. Joint Pub. Pub 1 Doctrine for the Armed Forces of the United States. *DOD US. July*, 2017.

12. Joint Pub. Pub 3-13 Information Operations. *DOD US. December*, 2014.

13. Martin Van Creveld. *Command in war.* Harvard University Press, 1985.

14. Joel S. Lawson. Command Control as a Process. *IEEE Control Systems Magazine*, 1981.

15. Clay Chun and Jacqueline E. Whitt. *John Boyd and the "OODA" Loop (Great Strategists)*, January 2019.

16. John K Setear. Simulating the Fog of War. Technical report, RAND CORP SANTA MONICA CA, 1989.

17. Johan Hagelback and Stefan J Johansson. Dealing with fog of war in a real time strategy game environment. In *2008 IEEE Symposium On Computational Intelligence and Games*, pages 55–62. IEEE, 2008.

18. Zackery Mason. Trick of the light: A game engine for exploring novel fog of war mechanics. 2018.

19. James B Olsan. Genetic algorithms applied to a mission routing problem. Technical report, AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING, 1993.

20. Margaret Rouse. *OODA Loop*, December 2018.

21. Lieutenant Colonel Jeffrey N. Rule. *A Symbiotic Relationship: The OODA Loop, Intuition, and Strategic Thought*. Technical report, Department of National Security and Strategy, U.S. Army War College, 122 Forbes Avenue, Carlisle, PA 17013, 2013.

22. Maj Thomas Tighe, Lt Col Raymond Hill, and Lt Col Greg McIntyre. *A Decision for Strategic Effects: A conceptual approach to effects based targeting*.

23. Air Force Research Laboratory. *AFSIM 2.6.0 Documentation*. 2020.

24. Infoscitex. *AFSIM*. Accessed: July 16, 2020. https://www.infoscitex.com/afsim/.

25. Timothy D. West and Brian Birkmire. AFSIM: The Air Force Research Laboratory's Approach to Making M&S Ubiquitous in the Weapon System Concept Development Process. *Journal of Cyber Security and Information Systems*, (vol. 7 no. 3):50—55, 2020.

26. Peter D Clive, Jeffrey A Johnson, Michael J Moss, James M Zeh, Brian M Birkmire, and Douglas D Hodson. Advanced framework for simulation, integration and modeling (afsim)(case number: 88abw-2015-2258). In *Proceedings of the International Conference on Scientific Computing (CSC)*, page 73. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2015.

27. AFRL/RQQD. *AFSIM 2.6.0 - Architecture Overview*, August 2019.

28. AFRL/RQQD. *AFSIM 2.6.0 - WKF*, August 2019.

29. AFRL/RQQD. *AFSIM 2.6.0 - Basic Agent Modeling*, August 2019.

30. AFRL/RQQD. *AFSIM 2.6.0 - Building AFSIM with CMake*, August 2019.

31. Andrew Feickert. Defense primer: Army multi-domain operations (mdo). *Congressional Research Reports*, 2020.

32. John R Hoehn. Joint all domain command and control (jadc2). Technical report, Congressional Research SVC Washington United States, 2020.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 25-03-2021	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* June 2019 - March 2021

4. TITLE AND SUBTITLE

Exploring Fog of War Concepts in Wargame Scenarios

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Tryhorn, Dillon, N., 2nd LT, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-21-M-086

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

AFMC/711 HPW
Building 852
WPAFB OH 45433-7905
COMM 937-938-4044
Email: christopher.myers.29@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

HPW/RHAC

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

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

This thesis explores fog of war concepts through three submitted journal articles. The Department of Defense and U.S. Air Force are attempting to analyze war scenarios to aid the decision-making process; fog modeling improves realism in these wargame scenarios. The first article "Navigating an Enemy Contested Area with a Parallel Search Algorithm" [1] investigates a parallel algorithm's speedup, compared to the sequential implementation, with varying map configurations in a tile-based wargame. The parallel speedup tends to exceed 50 but in certain situations. The sequential algorithm outperforms it depending on the configuration of enemy location and amount on the map. The second article "Modeling Fog of War Effects in AFSIM" [2] introduces the FAT for the AFSIM to introduce and manipulate fog in wargame scenarios. FAT integrates into AFSIM version 2.7.0 and scenario results verify the tool's fog effects for positioning error, hits, and probability affect the success rate. The third article "Applying Fog Analysis Tool to AFSIM Multi-Domain CLASS scenarios" [3] furthers the verification of FAT to introduce fog across all warfighting domains using a set of CLASS scenarios. The success rate trends with fog impact for each domain scenario support FAT's effectiveness in disrupting the decision-making process for multi-domain operations. The three articles demonstrate fog can affect search, tasking, and decision-making processes for various types of wargame scenarios. The capabilities introduced in this thesis support wargame analysts to improve decision-making in AFSIM military scenarios.

15. SUBJECT TERMS

Fog of War, Command and Control, Wargaming, AFSIM

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Major Richard Dill, AFIT/ENG
U	U	U	UU	101	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-3636, ext 3652; Richard.Dill@afit.edu