STRATEGIC EFFECTS OF AIRPOWER
AND COMPLEX ADAPTIVE AGENTS:
AN INITIAL INVESTIGATION

THESIS

Thomas R. Tighe, Major, USAF

AFIT/GOA/ENS/99M

19990409 037

DISCLAIMER

The views expressed in this thesis are those of the author and do not reflect the official

policy or position of the Department of Defense or the U.S. Government.

STRATEGIC EFFECTS OF AIRPOWER AND COMPLEX ADAPTIVE AGENTS:

AN INITIAL INVESTIGATION

THESIS

Presented to the Faculty of the Graduate School of Engineering and Management of the

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 Operational Analysis
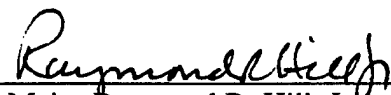
Thomas R. Tighe, B.S., M.A.S.

Major, USAF

March 1999

AFIT/GOA/ENS/99M-09

STRATEGIC EFFECTS OF AIRPOWER AND COMPLEX ADAPTIVE AGENTS:
AN INITIAL INVESTIGATION

Thomas R. Tighe, B.S., M.A.S.
Major, USAF
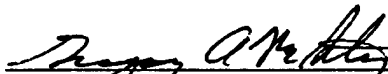
Approved:

_____   _____
Major Raymond R. Hill, Jr.   10 Mar 99
Co-Advisor   Date

_____   _____
Lt Col Gregory A. McIntyre   10 Mar 99
Co-Advisor   Date

iv

Acknowledgements

I would like to express my appreciation to my advisors, Lt Col Greg McIntyre and Major Ray Hill. Without their direction and expertise, this thesis would not have gotten of the ground.

Lt. Col J.O. Miller has also been very helpful as an unofficial advisor and reader. Mr. Kevin Healy of ThreadTec generously and patiently assisted me with Java multi-threading. I would also like to recognize my academic partner and mathematical mentor, Capt Lance Champagne. The AFIT experience would have been much more of a struggle without his insight and friendship.

Most importantly I want to recognize my lovely wife Christine, who has tirelessly dealt with daily household crises and kept our young family running smoothly. I would also like to thank Mitchell, Amelia, Anna, and Claire for helping me keep balance and perspective in my life, even if that wasn't apparent.

Finally, I dedicate this work to Capt Mark T. Todd and all those who have given their lives to make our country free.

<div align="right">Thomas R. Tighe</div>

Table of Contents

List of Figures

List of Tables

Abstract

US airpower theory and doctrine depend on the concept that the destruction of a few key targets or centers of gravity can unravel the enemy's physical ability to wage war or break his will to prosecute the war. This synergistic decimation of the enemy's effectiveness and resistance to our political will is known as Strategic Effects. These strategic effects are very difficult to quantify and are not directly accounted for in current DoD computer models. Since these computer models are used to aid with decisions about force structure and budget priorities, many believe that the Air Force's greatest potential contribution to modern joint warfare is going unrecognized and under financed.

This thesis explores military theory and current doctrine to define a method quantifying strategic effects. This method is based upon the Observe-Orient-Decide-Act (OODA) decision cycle. Next, current modeling techniques, and specifically the campaign level model, THUNDER, are examined for applicability to model strategic effects as defined. Finally, a proof of concept model is developed to study the advantage associated with OODA loop exploitation. This simple model uses Java-based, multi-threaded, autonomous, complex adaptive agents to demonstrate the non-linear (synergistic) results of OODA loop exploitation. These results are similar to the anticipated effects of strategic attack and provide a solid foothold from which the study and modeling of strategic effects can begin.

Strategic Effects of Airpower and Complex Adaptive Agents:
An Initial Investigation

CHAPTER 1

Airpower proponents advocate the decisive combat potential of modern airpower. Air Force leadership does not believe current models and simulations capture the true capabilities of modern airpower [SEAW notes]. This deficiency may put the Air Force in a disadvantageous position in joint force structuring, weapons system procurement and force employment decisions. The capability in question is known doctrinally as strategic effects. The strategic effect from the application of airpower has been intuitive to airpower advocates since air doctrine and theory was initially proposed by the likes of Douhet. However, this same intuitive effect has been nearly impossible to predict and is not directly accounted for in any mathematical models or simulations currently being used by the US Air Force. "Failure to properly analyze the mechanism that ties tactical results to strategic effects has historically been a failing of both airpower theorists and strategists."[AFDD 2-1, p3.]

In April of 1998, an Air Force wide workshop was held to examine the modeling of strategic effects. The Air Force leadership's concerns were summed up in the 6 March 1998 message announcing the event:

Workshop objectives are to provide actionable inputs to the modeling and simulation communities to support improved representation of airpower contributions in the current developmental suite of models...Current models and simulations do not adequately represent airpower contributions to joint warfighting capabilities beyond the tactical level. This situation is evidenced in the recent Quadrennial Defense Review (QDR) and the Deep Attack Weapons Mix Study (DAWMS).

This conference brought together those involved in the airpower theory debate and many simulation and modeling experts from the Air Force and DoD. Several new ideas were discussed, but the Air Force has not taken a position on the modeling of strategic effects of airpower.

This thesis attacks this problem in three ways. First, it defines strategic effects in a focused fashion amenable to modeling. Next, it examines the capabilities of the Air Force's current premier campaign-level model, THUNDER, to determine if it does or can model strategic effects. Finally, the thesis suggests a complexity-based autonomous-agent methodology for modeling strategic airpower effects. An experiment is performed using an initial model to test the concept of autonomous agents for modeling strategic effects.

This thesis is organized so that Chapter 2 is a stand-alone article suitable for submission to an academic journal. Chapter 3 outlines future extensions to this work. The appendices contain further details on the development of this methodology for modeling strategic effects. Specifically, Appendix A contains sample model output and calculations. Appendix B contains more doctrine framed in the perspective of OODA exploitation. Appendix C, is the documentation and source code for the model.

CHAPTER 2

2.1. Introduction, Overview, And Contributions

2.1.1. Introduction

Airpower is easy to speculate about, but often difficult to quantify. In World War II, airpower was massed in unparalleled magnitude against civilian-industrial and military targets with results that are still being studied. Airpower was used in the Vietnam War on targets of all descriptions with dubious results. These historical examples remind us that the nature of a war and the most appropriate application of airpower in that war are not always obvious.

The early airpower theorists and advocates, like Douhet and Mitchell, speculated about aerial weapons of mass destruction aimed at the enemy nation instead of the enemy's military forces. To these theorists airpower could be a decisive element in the application of military force. Attacking the nation instead of its armed forces could allow political objectives to be met while avoiding the horrific military confrontations typical of WWI. The concept of attacking an enemy's capability without having to engage its armed forces is now referred to as strategic attack. The massive conventional raids of WWII were strategic attacks, but it was the advent of atomic weapons that demonstrated the true strategic capability of airpower. These revolutionary weapons helped bring a hasty end to World War II. Despite the subsequent absence of their use, the deterrent

capabilities of nuclear weapons played an important role during the Cold War, yet are ineffective against the lesser adversaries of primary concern to US interests.

The world television audience witnessed the capabilities of modern precision-based airpower in Desert Storm. Despite the acute documentation of damage and the final results of that war, the exact influence of airpower on our military victory is still debated. Some claim that precision airdropped conventional munitions now have the strategic knockout capability of nuclear weapons without their political ramifications. In essence precision weaponry realized the strategic attack potential of airpower envisioned by Mitchell across the spectrum of conflict. Others claim that the premise behind strategic attack is flawed. This second group believes there is little or no historical evidence of the hypothesized catastrophic effects of strategic attack and these effects are neither predictable nor reliable. This would make strategic attack an ineffective use of airpower. These critics claim the most appropriate application of modern airpower, with its increased lethality, is to directly attack the enemy's fielded forces.

The place where these theoretical debates have become increasingly important is in the DoD budget. An aircraft optimized for strategic attack (B-2) does not usually have the same characteristics as an aircraft optimized for battlefield attack (A-10). It is often difficult to rationalize and justify the expense of developing and acquiring more than one new weapon system with modern austere defense budgets. The question arises then of which strategy and which aircraft is best suited to the airpower roles, missions, and doctrine. Computer models and simulations influence the decisions on force structure and weapons procurement issues like these. Curiously, these computer models do not

sufficiently demonstrate the strategic effects that our airpower theory predicts and doctrine expects.

The analysts and computer model caretakers do not necessarily refute the idea of strategic effects. The exact reason that one enemy chooses to continue to fight to the death while others sue for peace before the fight even begins is unclear. The subtle complexities of (enemy) social and political behavior are difficult to model mathematically, especially with enough certainty to program into a combat model. The targets and the level of destruction needed to produce these effects are also hard to quantify. There are currently too many uncertainties about strategic effects and their causes to validate and verify them and put them into a working model.

Computer model deficiencies not withstanding, modern strategic airpower advocates are confident that if the correct targets are chosen, the enemy will lose their will or ability to fight and accept our political alternatives. However, these advocates worry that computer-based mathematical models used to decide national military direction fail to capture the complex indirect military, social and political effects strategic attack are supposed bring to the fight. The major concern is that the Air Force's contribution to the US arsenal, which is arguably the most effective method of military-political force currently available in joint warfare, is understated and inadequately financed.

2.1.2. Overview of Research

This research contains four main thrusts. The first is defining strategic effects in a way that they can be observed, recognized and modeled. Primary sources for this area are

USAF doctrine and the military theories of Clausewitz [Clausewitz], Warden [Warden], Pape [Pape], Boyd [Fadok, McDonald], and Watts [Watts] as well as the modeling ideas of Ilichinski [Ilachinski], Zimm [Zimm], and Davis [Davis, Davis and Blumenthal]. The second examines THUNDER, and its capabilities and shortcomings for modeling strategic effects. Next, mathematical chaos and complexity theory, complex adaptive systems and autonomous agents are examined. These methods seem well suited for modeling strategic effects. Finally, a prototype complex adaptive agent simulation of strategic effects is built and used in an exploratory modeling fashion.

2.2. Defining Strategic Effects

The recently revised Air Force doctrine documents help clarify the meaning of strategic effects of airpower. This doctrine draws upon the theories of Clausewitz, Warden, Pape, and Boyd as next described.

<u>2.2.1. US Air Force Doctrine</u>

Three AF doctrine documents are important to the study of strategic effects. They are AFDD 1, Basic Doctrine; AFDD 2-1, Air Warfare; and AFDD 2-1.2, Strategic Attack.

AFDD 1 contains three "enduring truths" that describe the fundamental nature of war [AFDD 1, p. 6]. Each truth suggests a strategic effects approach to combat. The first truth is that, "War is an instrument of national policy." "Victory in war is not measured by casualties inflicted, battles won or lost, or territory occupied, but by whether or not political objectives were achieved" [AFDD 1, p. 6]. The second fundamental is, "War is a complex and chaotic human endeavor." "Human frailty and irrationality shape war's

nature. Uncertainty and unpredictability—what many call the 'fog' of war—combine with danger, physical stress, and human fallibility to produce 'friction,' a phenomenon that makes apparently simple operations unexpectedly, and sometimes even insurmountably, difficult" [AFDD 1, p. 6]. While there is no way to eliminate uncertainty or unpredictability, "...sound doctrine, leadership, organization, core personal values, technologies, and training can lessen their effects" [AFDD 1, p. 6]. The last fundamental truth is that "War is a clash of opposing wills." "War is not waged against an inanimate or static object, but against a living, calculating and highly unpredictable enemy" [AFDD 1, p. 6]. This result is a dynamic combination of action and reaction. While physical factors are crucial in war, the national will and the leadership's will are also critical components of war. The will to prosecute the war or the will to resist the enemy can be decisive elements.

The search for strategic effects starts with a definition of strategy. Strategy originates in policy and addresses broad objectives and the plans for achieving them. Military strategy helps achieve national political objectives [AFDD 1, p. 4]. Draft AFDD 2-1 [p. 1] defines strategy as "...a means to accomplish an end." In the context of global politics, the purpose or "end" of conflict is to bend an adversary's will to match our political goals. Strategy is our set of preconceived plans of how to deal with what we expect to encounter, using what we expect to have, to attain the objectives we *expect that we will* want to attain. Any significant change in these expectations will require a new plan. A strategic effect is anything that disrupts this strategy.

Strategic effect is the disruption of the enemy's *strategy, ability*, or *will* to wage war or carry out aggressive activity through the destruction or disruption of his center(s) of gravity or other vital target sets…Strategic effects can also indirectly result from the actions of aerospace or surface forces at the lower levels of war. [AFDD 2-1, p. 8]

Strategic attacks are, "…those operations intended to directly achieve strategic effects by striking at the enemy's centers of gravity (COGs)" [AFDD 2-1, p. 45]. A center of gravity is further defined as, "the characteristics, capabilities, or localities from which a force derived its freedom of action, physical strength, or will to fight" [AFDD 1, p. 51]. The point is also made that by attacking the COGs, we seek to avoid a direct, extended (high attrition), military-on-military engagement. Strategic attack should produce effects well beyond the direct physical damage of the attack. This synergistic property is often conceptualized as a cascading collapse of the enemy from within or a domino effect.

"Strategic attack is a function of objectives or effects achieved, not forces employed" [AFDD 1, p. 52]. Note strategic attack is not necessarily nuclear or long-range. This point is emphasizes in AFDD 1 [p. 52.] since "…many strategic actions tend to be nonnuclear conventional or special operations against more limited war or contingency operations objectives, and will increasingly include attack on an adversary's information and information systems." Command, control (C2) and communications (C3) targets are typically on the top of strategic target lists since the enemy's strategic plan is disrupted if instructions cannot be passed to those responsible for executing the plan. In fact, "…the enemy's C2 should always be a target of particular focus in strategic attack" [AFDD 1, p. 53].

Airpower strategic effects are not necessarily solely related to the air war. An effect is strategic if it impacts the entire conflict and that impact may be more in the political arena than on the battlefield. "The Berlin Airlift is a successful example of how aerospace power can profoundly shape and control events without necessarily having to resort to aerial attacks" [AFDD2-1.2, p. 3].

2.2.2. A Basis in Military Theory

Doctrine is based on military theory and lessons learned in combat. This section examines some of the ideas of various prominent military thinkers whose ideas have influenced the study and doctrine of strategic effects.

2.2.2.1. Clausewitz

Clausewitz [Clausewitz, p. 89] believed that the nature of war is not "...fundamentally constant with only minor adjustments for situational uniqueness." This contrasts with most contemporary combat modeling approaches in which combat is assumed to be controlled by rigid, constant rules. Instead Clausewitz views war as a "paradoxical trinity" of violence, chance, and reason which must be balanced in any effective theory on war [Clausewitz, p. 89]. Let's define each point of this trinity. Violence can best be exemplified by the people of a country and their desire for or *will* to support the war. Chance is often represented by the military and its commander whose lethal skills or *abilities* must be used creatively to capitalize on the circumstances of the battle at hand. Reason is commonly expressed as the government and the political limits on operations or *strategies* that keep war from outrageous escalation. Reason is also attributed to the individual combatants whose free will, fear, and morals must cooperate for them to be effective in combat. Clausewitz [Clausewitz, p. 89] explains this as:

9

The passions that are to be kindled in war must already be inherent in the people; the scope which the play of courage and talent will enjoy in the realm of probability and chance depends on the particular character of the commander of the army; but the political aims are the business of the government alone.

Clausewitz [Clausewitz, p. 90] defines three broad objectives in planning to target the entire enemy system. These three objectives relate to his "paradoxical trinity" and are the armed forces, the country, and the enemy's will. To be successful, the armed forces must be destroyed (physically or morally) so that "...they can no longer carry on the fight"[Clausewitz, p. 90]. It is important to understand that Clausewitz used the term moral not to mean virtuous, but to mean psychological or emotional. Clausewitz's definition is used throughout. Clausewitz states the enemy country must be occupied so that new forces cannot be built. This is a physical victory over the government. But the potential for reigniting the war exists "until the enemy's will has been broken: in other words, so long as the enemy government and its allies have not been driven to ask for peace, or the population made to submit" [Clausewitz, p. 90]. Breaking the will of the government or the population would be a moral victory over the enemy nation.

Clausewitz also explained that war on paper (strategy) differs from real war due to fog and friction [Clausewitz, p. 119]. These unpredictable and intangible concepts are critical to defining strategic effects. Clausewitz does not believe in underlying physical or mathematical laws of war and did not trust mathematical formulations of battle. Davis and Blumenthal [Davis and Blumenthal, p. 22] concluded, based on Clausewitz's writings, that models and calculations are not "...irrelevant, but rather that to be useful, models and modelers must include a wide range of qualitative factors and must also be

humble with respect to random factors and horseshoe nails." This suggests that models should become tools of exploration, not soothsaying answer machines.

2.2.2.2. Warden

Warden echoes the idea that the objective of war is not the defeat of the enemy's armed forces, but the peace (in a politically changed world) that follows. It is better if political objectives are met with limited or even no direct large-scale military conflict. As with Clausewitz, a key concept in Warden's targeting plan is identifying an enemy's COGs. These COGs are defined as "...the point where the enemy is most vulnerable and the point where an attack will have the best chance of being decisive" [Warden (1998b), p. 7]. Warden proposes a five-ring system as a general characterization of any enemy or "strategic entity." Analyses using this model are a basis for strategic planning. The five concentric rings of the model from the center (most important) outward are: Leadership, Organic Essentials, Infrastructure, Population, and Fighting Mechanism (see Figure 1). Elements of the enemy system are characterized into these rings from which further analysis defines the enemy COGs. Once these COGs are identified they should be attacked in parallel to disrupt the entire enemy system, or as Warden suggests, induce "strategic paralysis." This parallel attack depletes enemy resources, capabilities and alternatives faster than traditional serial attacks and is a key to successful strategic campaign. Today's precision weapons and stealth aircraft are the enablers of parallel attack.

Warden believes his five ring model may be employed to define strategic objectives which, when achieved, either change the enemy leadership's will to align with ours (coercive victory) or reduce the enemy's ability to physically resist our will through

our action (military-political victory). Warden emphasizes that strategic attack pressures the leadership, not the fielded military.
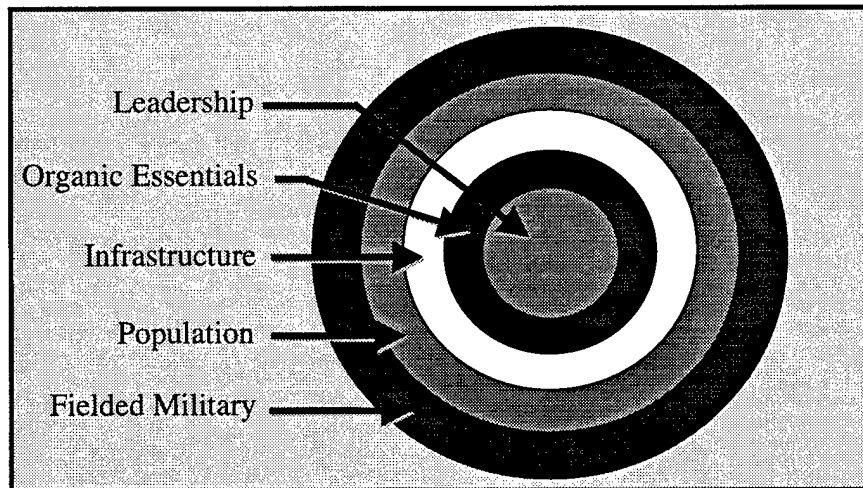
Figure 1. Warden's five ring model of any strategic entity

2.2.2.3. Pape

Pape is not an advocate of strategic air attack. He describes bombing for strategic purposes as coercion toward political goals and points out the difficulty in precisely measuring these "squishy" multifaceted (multivariate) objectives. He splits coercive airpower into three categories: decapitation, punishment, and denial [Pape, p. 97]. These three categories are roughly equivalent to Clausewitz's three objectives.

Decapitation is Pape's interpretation of Warden's emphasis on leadership and C3 as a center of gravity. Decapitation, in Clausewitzian terms, is an attack on the trinity elements of chance, violence, and reason. It is an attack on chance because it applies stress to the military C3 system. It could be an attack on violence by showing force against a prominent citizen. It could be an appeal to the element of reason, which may

be lacking in the targeted leaders but alive in the nation. Pape concludes that decapitation is not effective because of the political and physical difficulties of an attack for assassination. Also, competent military and political staffs can carry on with the loss of, or separation from, the leader(s). The redundancies of communication and information systems in modern industrialized countries make physical isolation of various elements of the command structure extremely difficult.

Pape concludes that punishment (of the enemy population) through the destruction of the Warden rings of organic resources, infrastructure, and of the population itself is also ineffective in altering the will of the nation unless a politically unacceptable percentage of the enemy population is directly targeted and killed. A punishment campaign does not inflict paralysis on the enemy military, but instead punishes the non-combatants (violence) who are denied dwindling national resources. Indigenous resourcefulness, which cannot be predicted or modeled, prevails to keep the military supplied despite the conditions of the general population.

Pape's last division of coercive attack is denial of the resources that the combat forces need to prosecute the war. This is the interdiction mission. Denial may be effective, but the elasticity of modern economies of war, especially if fought by alliances, can often overcome isolated or sporadic attacks on manufacturing, storage, and distribution facilities. Denial also usually only works in long wars of attrition where the enemy must rely on equipment manufactured during the conflict after stockpiled supplies are exhausted. Pape recommends tactical bombing of theater level military targets as a more effective alternative to a strategic air campaign.

2.2.2.4. Boyd

Boyd believes that military operations are basically a sequence of decision processes or cycles. These cycles have four main points: observe, orient, decide, and act (OODA). These cycles have been dubbed "OODA loops" and are consistent with the contemporary emphasis on situational awareness (SA). A smaller OODA loop is indicative of more SA as the information about the situation is synthesized and acted upon more quickly. If you can make your decision cycle shorter or "turn inside" the opponent's OODA loop, then you can foil the enemy's thought process and always be a step ahead of him. Boyd's aim for military operations was to first, "...create and perpetuate a highly fluid and menacing state of affairs for the enemy, and then disrupt or incapacitate his ability to adapt to such an environment" [Fadok, p. 15]. Key to this idea is adaptability, which is described as using friction to shape the conflict in your favor.

Boyd shares Clausewitz's view of the moral aspect of combat, noting that "...while considerations of time, space, and speed had an important impact on success and failure, the moral and psychological dimensions played a dominant role" [McDonald, p. 147]. Being "inside the OODA loop" generates the most important principal of war: surprise. Exploiting the OODA loop can cause two different kinds of surprise [McDonald, p. 149]. Moral surprise is being unaware of the attack or its location. Material surprise is knowing that the attack is coming, but being too logistically or physically unprepared to do anything about it.

Boyd believes that consistently operating inside your opponent's OODA loop will generate fear and anxiety (moral friction) and eventually destroy the

14

opponent's will to fight. McDonald [McDonald, p. 149] notes, "In this sense, operating inside the enemy's decision cycle has cumulative effects that surpass the linear impact of simple destruction of forces."

2.2.2.5. Asymmetric Force Strategy

The Air Force Doctrine Center's (AFDC) new concept of operation for the U.S. armed forces is called the Asymmetric Force Strategy [AFDD 2-1, p. 3]. This new approach to war is described as "US military forces now leverage sophisticated military capabilities to achieve national objectives and avoid bloody force-on-force engagements that characterized America's traditional strategy of attrition and annihilation" [AFDD 2-1, p. 3]. This is the classic concept of strategic effects. AFDD 2-1 [p. 8] lists two ways to achieve strategic effects; by direct strategic attack or by the cumulative indirect effects of non-strategic attacks. The strategic attack function "…is often aimed directly at producing the strategic effect of enemy defeat, with no intermediate level effects on enemy forces involved" [AFDD 2-1, p. 7]. This is the approach taken by Warden. Non-strategic attacks can attrit the enemy forces and resources to make him militarily ineffective. This is more in line with Pape's ideas.

USAF doctrine incorporates strategic effects in Asymmetric Force Strategy. Thus, we need models that quantify, measure, and study strategic effects in a way that logically ties back in to real world experience.

2.3. A Concept to Model

The moral elements of warfare are without a doubt a key to capturing strategic effects. They include fear, surprise, and will. These moral effects synergistically interact

15

when the physical attack diverges drastically in speed, intensity, or effectiveness from expectations. These elements are certainly manifested in many ways, but can be described in terms of their effect on the decision making of combatants at all levels of conflict.

2.3.1. The Moral Elements

Fear is anxiety about possible or expected unpleasant events in the uncertain future. An individual's reaction to fear shapes their decisions and corresponding actions. Surprise presents its victim with unfulfilled expectations or unforeseen occurrences. Surprise often results from incomplete or poor planning on the part of its victim. Surprise may be inevitable to the victim if planning information is inadequate. Will is determination expressed as a deliberate act of pursuing a course of action. Except for cases of martyrdom, will is bounded by an element of realism about the expected, though often optimistic, outcome of the situation to which it is applied. If the desired objective is clearly unattainable and the cost of failure is high, will may be broken and the course of action abandoned.

The moral elements comprising our definition of strategic effects address the ability to anticipate, correctly prepare for, and influence future events. Boyd's OODA loop construct offers a template for modeling these aspects of combat.

2.3.2. Boyd's OODA Loop

The OODA loop is recognized in Joint Publication 3-13.1 [Appendix A] as a decision model "...applicable to all C2 systems -- friendly or adversary." The OODA loop is also analogous to the Shewhart cycle and Ishikawa circle, which are general-purpose process improvement cycles taught in many "Quality improvement" techniques,

including the QAF initiative [Air Force Process Improvement Guide, p. 1]. Any entity

that makes systematic decisions has an OODA cycle.

Any OODA loop takes a measurable amount of time to complete, and this time

may vary between OODA loop cycles. The OODA loop model is represented graphically

as a circular connection of the four phases of the decision cycle (see Figure 2). A

common misconception to get from this image is that the OODA loop is a single series of

sequential events. Instead JP 3-1.3 describes the OODA process as continuous, meaning

an entity simultaneously has multiple concurrent OODA processes. Rather than a series

of OODA loops, conceptually we can think of a cable of OODA strands.
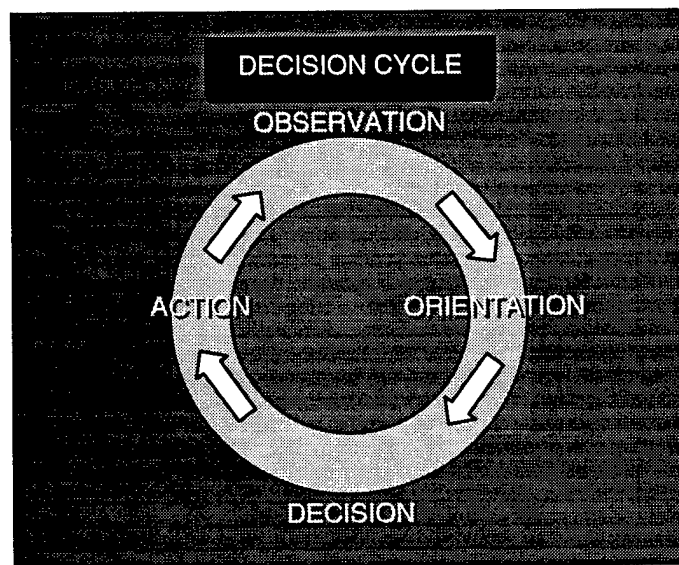
Figure 2. OODA loop as shown in JP3-13.1

2.3.3. The OODA Cable

Envision the "OODA cable" as four separate pieces of cable spliced together at

the OODA phases or nodes to form a loop (see Figure 3). Decisions flow through this

cable like charges of electrical current with many in the cable at one time. The piece of

cable going into the observe node is the thickest (most strands) with some loose or frayed

ends on the end away from the observe node. Each strand represents a data delivery

device or sensor available to the entity. The frayed ends represent sources of external

information while the unbroken strands are internal feedback. Not all incoming

information is relevant to the decision at hand, nor can all the information be processed,

thus at the observation node the cable is spliced into a thinner cable with a strand

representing all relevant inputs. The cable thickness of this section represents the

information capacity of the decision-making entity. Information overload occurs when

more information is presented to the entity that it can include in the decision process. At

the orient node all the strands of information are spliced into a thinner (fewer strands)
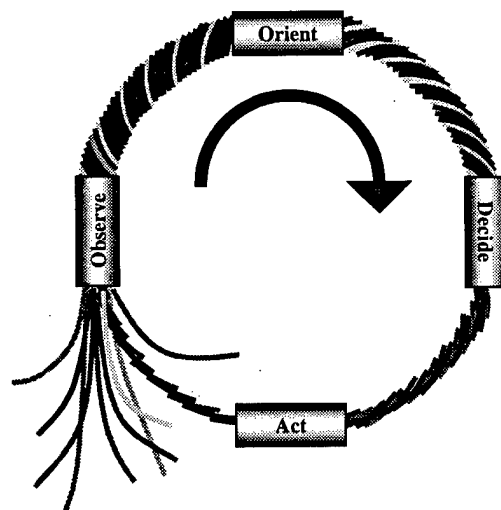
Figure 3. The OODA cable

cable with a strand for each plausible course of action. The number of strands in this

section of the cable represents the number of options of the entity can evaluate before a

decision must be reached. This section of cable is spliced into a smaller section at the

decide node. The strands leaving the decide node represent the actions the individual has

decided to take. This cable is spliced into a smaller cable at the act node in which there is

a strand for each act the entity can do simultaneously. These "act" strands lead directly

into the observe node as feedback from the action taken to be processed with new

external information from the loose or frayed strands.

In short, the OODA cable represents the inherent need to filter and consolidate

information through the decision cycle with thicker cables representing an increased

ability to carry information.
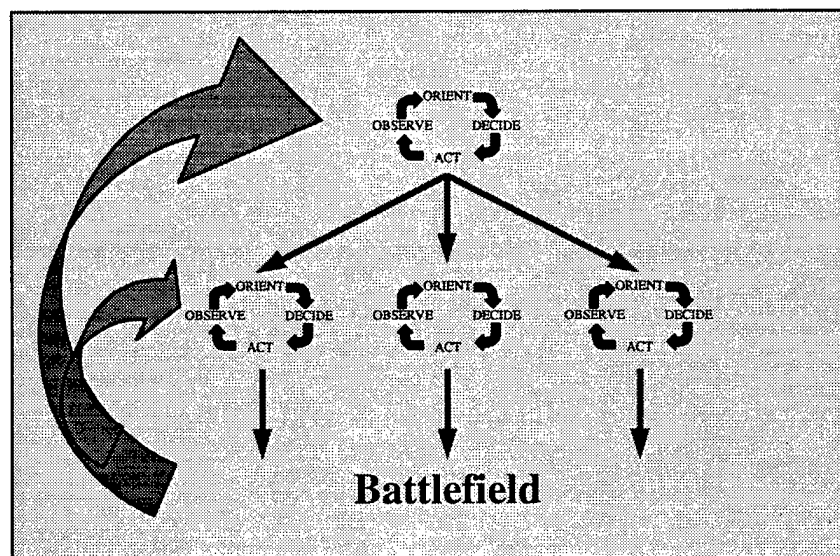
Figure 4. Beene's OODA interaction concept

2.3.4. A System of OODA Loops

In a complex system, OODA loops form a hierarchical interconnected system (see

Figure 4). OODA loops at one level in this hierarchical system may depend upon OODA

loops at the same or other levels for information (observations), instructions (orientation), or to carry out decisions (actions) [Beene, 1998]. OODA loops in a system can also be laterally dependent on adjacent "peer" loops.

OODA loop cycles at any level may differ in length from those at other levels. For example, a commander issues orders in the act phase. These orders feed into the orient phase of the subordinate. The outcome of the actions of the subordinate affect the commander's observe phase. The C3I challenge is to coordinate these OODA loops. "Though time frames vary at each echelon of command – according to the mission and battlefield perspective – decision cycles must be sufficiently synchronized to exploit both sequential and cumulative opportunity" [McDonald, p. 151].
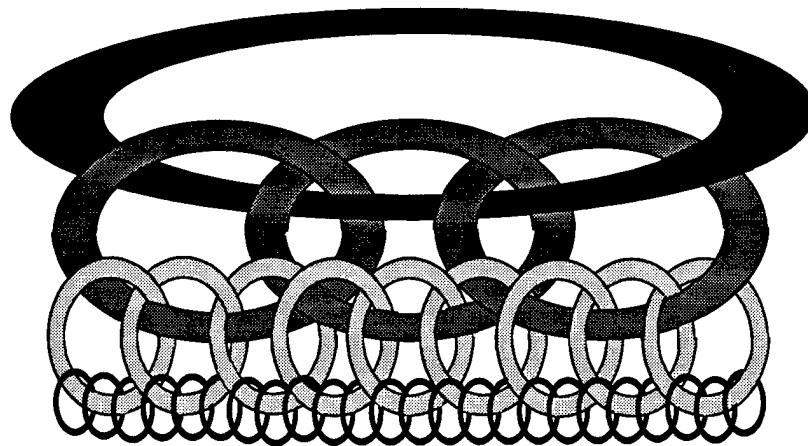
Figure 5. Stable OODA web

From an attack perspective, if the right OODA loops are exploited and made to collapse, the entire enemy system could become unstable and cascade into failure. This is one classically anticipated result of a successful strategic attack, and is Warden's vision of strategic paralysis.

From a decision making, or OODA perspective, envision the enemy system as a hierarchical web of dependent OODA links with irregular but successively smaller links towards the bottom. Each link is an OODA cable. The links of OODA cable are not uniform. Some links have many other links dependent on or connected to them. Others are dangling relatively free from adjacent links. The cables at the top are typically longer and thicker as commanders with their staffs are able to process a lot of information, but have relatively long decision cycles (a day or two). These larger links have many smaller
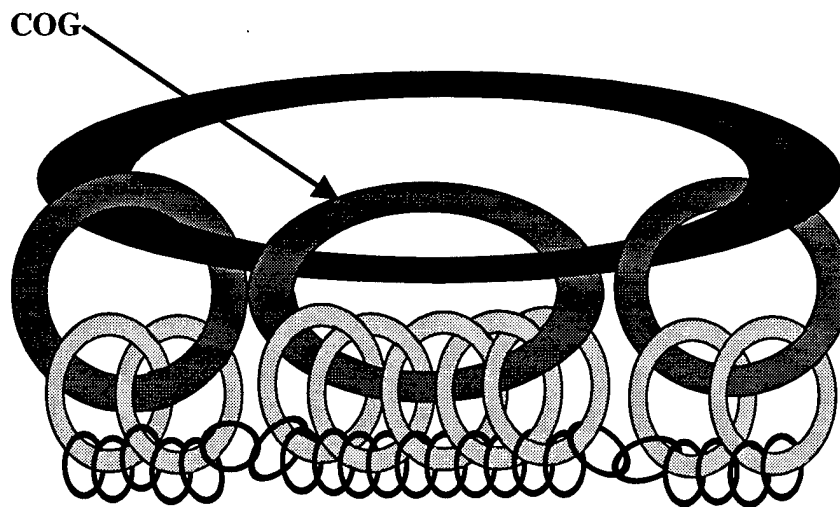
Figure 6. OODA web with identifiable COGs.

subordinate links attached to them as the lower echelons rely on command decisions and command staff relies on indirect information or feedback. Subsequent rows in the web are shorter, thinner cables fastened at the orient node to the commander's act node. The bottom row in the web is made of short, thin cables representing the individual combatant. These individuals have only a small amount of information to process, but need a decision every few seconds. If a bottom link is broken, the web is tattered and less effective. If an upper link is broken, the web may falter leaving the enemy exposed. If

21

enough links are broken, the web becomes ineffective. A link of any size that has a large number of dependent, attached links is a center of gravity.

2.3.5. OODA Exploitation

Boyd suggests "turning inside" the enemy decision process, or beating the enemy with faster, superior decisions and correspondingly devastating actions. This requires a shorter, faster OODA loop than the enemy's. This will add moral (mental) fog and friction to the enemy's situation. Warden's theory is similar [Fadok, p. 37], although he advocates making the enemy's OODA loop larger and slower by adding physical friction and fog through the destruction of key COGs, such as C3I. Further, continued attack prevents recovery or repair of the enemy's OODA web.

Pape uses historical examples to refute the strategic effects of bombing. For instance, the U.S. strategic bombing campaign in Europe during WWII did not sufficiently disrupt Germany's industrial capacity. The sequential versus parallel bombing campaign allowed Germany to reorganize, regroup and rebuild between attacks. In short, Germany repaired their OODA web before further US attacks could collapse it.

This is why Warden advocates parallel attack. Hit the system at all levels simultaneously and continuously and the system has neither the time nor the ability to recover. Stretch the links until enough of those links break preventing enemy recovery. Nuclear attack would (and did) halt the OODA process (cut the cables) on most levels simultaneously, giving a parallel attack with just one weapon. This explains why Pape calls a "king's X" on denouncing strategic effects of nuclear weapons.

Disrupting an enemy's OODA requires striking the proper COGs. Identifying

OODA processes that are vulnerable COGs can be a difficult task. Clausewitz's

"paradoxical trinity" of violence, chance, and reason, represent a valid starting point.

2.3.5.1. The Clausewitzian Trinity Revisited

Clausewitz's three elements are often misrepresented as military, population, and

government instead of chance, violence, and reason. Clausewitz gives the former three as

examples of the latter three, not substitutes. One reason for this substitution is that the

trinity example at the national level is easier with which to identify than the underlying

concepts. The trinity the Air Force Doctrine Center's proposes for the study of strategic

effects [AFDC, p. 6] is shown compared to Clausewitz's trinity (see Figure 7). The
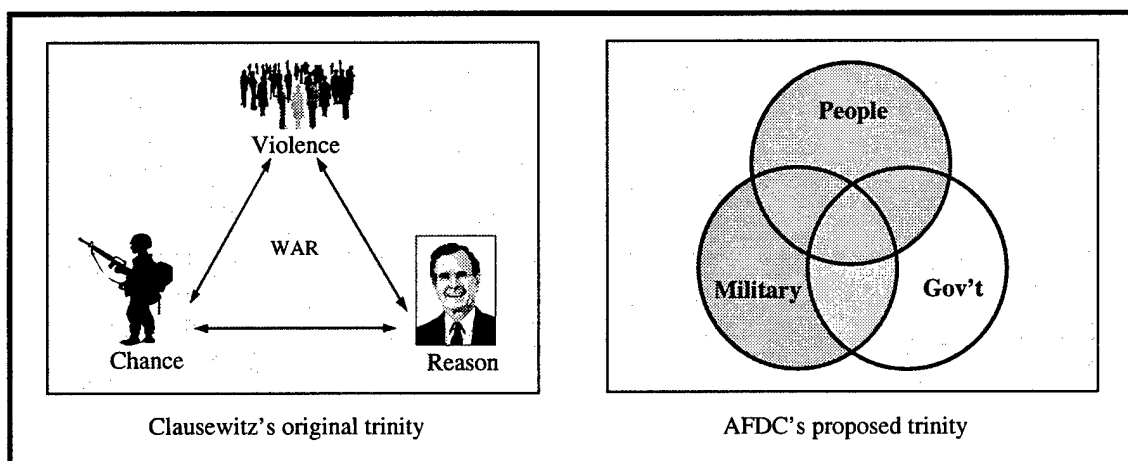
Figure 7. The Clausewitz trinity compared to the AFDC's interpretation

specific national level example of the trinity cannot be used as a replacement for the

general concept. All three original elements are actually likely to be found in each level

of the enemy system. Less specific examples of the trinity elements are given below.

23

The element of chance attempts to quantify the ability to take advantage of unexpected opportunities. It can be thought of as initiative. In the national example, the military is the only group that realistically is in a position to find opportunities against the enemy. Opportunities for initiative happen frequently at every level and in every aspect of the national system. International political alliances, increasing industrial production, and 'victory gardens' are all examples of chances to gain an advantage over the enemy. These chances just have to be seen to be acted upon. Chance can be interpreted as a function of the ability to observe a potential course of action.

Violence is the perspective from which events are seen. A population with violence will tend to attribute negative events to the enemy, strengthening its resolve or will against that enemy. A population with less violence may search for other alternative causes to the event before blaming the enemy. If a reasonable alternative can be found, the less violent population may not be inflamed to the point of supporting war. Violence frames the event in context of the conflict for decision-making entities. Violence *orients* observations toward a decision. This violence-based orientation can be found to some degree at every level of decision making in the enemy system. Reason is the element that keeps the war from escalating to the horrific level of 'total war'. The government takes this function at the national level because it usually makes the decision as to whether or not the country should attack, or conversely, should surrender. The government is also the body that will face any political repercussions for war crimes. Reason is applied at every level in the enemy system on decisions concerning the conflict. Individual soldiers must decide whether or not to shoot enemy combatants and enemy non-combatants. The

nation must decide whether or not to support military conscription and how young those

conscripts may be. Reason impacts the decide phase of decision cycles at every level

2.3.5.2. The Trinity and OODA Connection

If chance is a function of the ability to *observe*, violence helps *orients*

observations, and reason is an element of the *decision* process, then Clausewitz's trinity

can be viewed as factors that directly effect an OODA cycle. These three elements are

essential for the *action* of waging war. Clausewitz has given the formulation for an

OODA loop and an example that shows target classes for exploiting a national level

OODA. Figure 8 describes this alternative interpretation of the Clausewitz trinity. This

interpretation maps the three elements of the trinity to the three initial phases of the

decision process, instead of to components of the national system. This interpretation

will serve as a theoretical basis for targeting the OODA loop to achieve strategic effects.
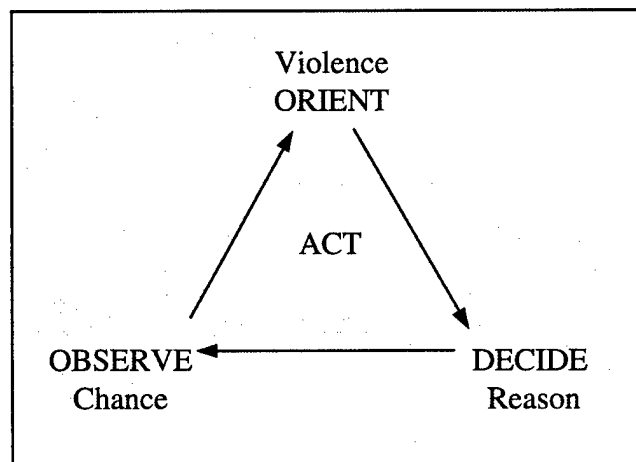
Figure 8. The Clausewitz trinity from the OODA perspective

2.3.5.3. Targeting the Trinity

Attacking Clausewitz's "chance" is striking the enemy nation's ability to wage an effective war by nullifying its preparations or strategy. This reduces the nation's ability to take the initiative if an opportunity is observed. The focus on rebuilding strategy after an initial strike can also put the enemy on 'the defensive' and blind them to immediate, tactical opportunities. Chance can be attacked by either foiling its military plans or directly diminishing its forces' abilities. The latter was the U.S. approach with the massive raids on German war industry in World War II.

Targeting Clausewitz's "violence" is targeting the enemy nation's will to fight. This is probably the most difficult component to prove existence, let alone target. Neither Germany's bombing of London nor Britain's bombing of Germany changed either nation's attitude or will to fight in WWII. In fact public resolve against the aggressor likely increased in each instance. On the other hand, swarms of Iraqi infantry surrendered to unarmed corespondents near the end of Desert Storm. "Violence" targets are entities with limited commitment to the political objectives. Propaganda and PSYOP may be the most useful weapon here.

Reason is a realization of the costs associated with war. A strategic attack that demonstrates the price a nation will have to pay is an attack on reason. Consider the El Dorado Canyon Raid in 1986 that targeted Libyan leader Kadafi. The attack demonstrated that the US was indeed willing to prosecute a war that would be costly to Libya and Kadafi. This attack was outside the bounds of Kadafi's expectations. It raised the cost of the Libyan course of action and apparently impacted Kadafi's will to pursue his policies.

2.3.7. Airpower and OODA Exploitation

Airpower is the force of choice for OODA exploitation. "Because of its speed, range, flexibility, and ability to maneuver as required to locate and precisely attack targets while neutralizing or avoiding threats, aerospace power is uniquely suited to conducting rapid, parallel attacks against the enemy" [AFDD 2-1, p. 5]. These characteristics of airpower allow simultaneous and continuous attacks on an enemy OODA system, preventing recovery. "A key difference between aerospace power and surface warfare is that aerospace forces can often strike directly at key target sets that have strategic results, without having to go through the process of drawn-out attrition at the tactical level of war" [AFDD 2-1, p. 8]. With airpower, all three corners of Clausewitz's trinity can be attacked at once. This was the case in Desert Storm and is fundamental to Warden's theory and current Air Force doctrine.

2.4. Current Combat Modeling Techniques

Most current combat models are excellent tools for their designed purpose, but are not appropriate for the study of strategic effects. As stated before, AFDD 1 delineates three fundamentals of war. The reason current models fail to capture strategic effects ties to each of these fundamentals, discussed separately in the following sections.

2.4.1. War is an instrument of national policy

This first fundamental says, "Victory in war is not measured by casualties inflicted, battles won or lost, or territory occupied, but by whether or not political objectives were achieved." Combat models use measures of effectiveness (MOEs) to measure victory with the most common MOEs being casualties inflicted and territory

occupied (forward line of own troops (FLOT) movement). This is the Lanchester equation paradigm.

2.4.1.1. Lanchester Equations

The Lanchester equations were published in 1914 ironically to try to determine the impact of aircraft on ground combat [Battilega and Grauge, p. 553]. Today some form of these equations is found in a significant percentage of combat models as the method for the adjudication of combat. They are based on the assumption that attrition is proportional to the size of the forces engaged in combat. These equations are deterministic differential equations. The unalterable outcome of combat adjudication is based on the starting troop strengths and their attrition rates (Pk).

Lanchester equations are still popular and widely used despite their many recognized flaws. The equations are simple, easy to compute, and give intuitive results. Unfortunately, the original equations do not accurately model many historical battles [Battilega and Grauge, p. 92]. Some historical battles have been "modeled" successfully by fitting the equation attrition coefficients to match the actual attrition of the battle or distinct segments of the battle.

Many people have made modifications in the past 50 years to shore up the original equations and have tried to validate these modifications with historical data. Helmbold is one of the most successful, but his formulation [Hartley] is designed and validated for "classic" battles that have approximately 10% attrition. If the battle varies from this type of conflict, the model becomes less reliable.

The reality is that attrition rates are not predefined or even stable. Thus, prediction with a Lanchester-based model is nearly impossible. It is even more difficult

in some future conflict where the nature of the battle is not yet known. Analysts can make the Lanchester equations fit the data but they cannot fit the future, especially when US doctrine seeks to avoid the "attrition" warfare approach.

2.4.1.2. Casualties vs. Causalities

The Lanchester type attrition ratios can be correlated to the outcome of the battle, but except in extreme cases of annihilation, attrition alone does not end battles. Political decisions based on military capability, national will, and the commander's intent are just as influential. Attrition is not a surrogate for victory. Davis and Blumenthal [Davis and Blumenthal, p. 8] point out some shortcomings of traditional types of models

> Typically, ground-combat simulations focus on complex calculations of attrition while treating command-control processes, tactics, and strategy in terms of stereotypes embedded in the data bases. This ignores the evidence of history that such matters (and other "soft factors") are first-order determinants of both deterrence and war outcomes, and should therefore be highlighted.

In other words, many important factors are not usually captured in our models.

Zimm [Zimm, p. 9] proposes "the causal model of warfare." His premise is that while attrition can be correlated to the outcome (victory) of a battle, it cannot predict it. Zimm [Zimm, p. 6] describes the benefits of tactical maneuver and how they are absent from today's method of attrition-based ground combat models. Zimm proposes that maneuver warfare attacks the enemy morale, cohesion and fighting spirit as well as the soldiers by operating inside the enemy's OODA loop [Zimm, p. 5]. He hypothesizes that maneuver warfare causes opponents to make decisions under stress in a fundamentally different way [Zimm, p. 25]. "If maneuver warfare confuses the enemy, and a confused enemy is easier to defeat, then we must model the capability to confuse and the results of the confusion" [Zimm, p. 22]. This sub-optimal decision making process causes errors

which become military opportunities and yield synergistic results. These hard to quantify, synergistic results are nearly identical to the Air Force's expectations of strategic effects.

To introduce political objectives into any model, those objectives and the maximum price the entities are willing to pay to achieve those objectives must be translated into militarily quantifiable terms. That cost may be an attrition number, a casualty exchange ratio, or an expenditure of a critical resource. It may be a loss of international prestige. The military objective would be to raise the price of that objective higher than the enemy is willing to pay while keeping our own costs within our own budgets. A full price does not need to be extracted from the enemy to achieve victory. The enemy just has to recognize that the price will be higher than he is willing or able to pay. That recognition is a strategic effect.

2.4.2. War is a complex and chaotic human endeavor

This element of the nature of war explains that "...uncertainty, unpredictability, and unreliability are always present" [AFDD 1, p. 6]. This directly conflicts with some of the more predominant model mathematical foundations. The traditional mathematical constructs of determinism and stochastisism are discussed next.

Many models, including the Lanchester-based models, are deterministic or have rigid rules and equations of war activity. If combat were indeed deterministic, there would be no need for war. Analysts would figure out who would win based on the capabilities each side was willing to commit, and sign the peace treaty. Deterministic models do not reflect the nature or causes of war. They only describe the symptoms. These models work well when trying to compare different hardware, etc., while holding the nature of the battle constant, but they are ineffective for studying strategic effects.

The next step up in mathematical modeling is stochastic models. These models have firm rules about combat like deterministic models, but some of the coefficients in these equations are represented by random variables instead of constant values. Stochastic variables have a strong place in models when examining events that behave like random variables such as hardware component life, ballistic errors, supply line waiting times (queue length). Human reaction and decisions in battle are not random fluctuations around an otherwise fixed law of war and representing decisions stochastically is not appropriate. Our doctrine and military experience offer flexible and situational *principles of war* that are to be applied to tactical decisions, not laws of war that prescribe the decisions. These stochastic models still assume underlying laws of war that are too rigid to meet the conditions set forth in the second element of the nature of war.

Many of the deterministic and stochastic models have linear characteristics, although combat is recognized a nonlinear. An experiment by RAND [Dewar, et,al.] studied a simple deterministic Lanchester based model of ground combat. This study showed that if the decision to send reinforcements to the battle was based on the condition of the battle through something like a force ratio threshold, that reinforcement decision introduced nonlinearities into the model. Mathematically, the reinforcement decision based on the state of the battle provided a feedback loop to the linear Lanchester attrition calculations. Real war has the potential for many such feedback loops. These types of feedback loops produced non-monotonic (unpredictable) outcomes to the simulated battle that met most of the criteria for mathematical chaos. Adding more

restrictions or inputs to the model did not eliminate this behavior. The significance of chaos is that the outcomes do not settle out to a steady state or a predictable cycle.

Dynamical systems, like the attrition model with feedback, can exist in three states [Rinaldi]. They can be non-chaotic or stable such as a linear system and absorb or dampen disturbances. Deterministic or stochastic models may be capable of modeling the system in this state. The system can be in a chaotic or unstable state where a small disturbance leads to unpredictable, often catastrophic results. The third state lies on the border between the stable and unstable region and is known as the complex region. The ideas of mathematical chaos and complexity are discussed later.

2.4.2.1. Chaos

Mathematical chaos theory examines the behavior of and interactions between a system of entities instead of studying the entities themselves. These systems are often nonlinear. Ilachinski [Ilachinski, p. 5] characterizes chaos as the study of how simple systems can generate complicated behavior. Chaos theory is used to describe dynamic systems that through simple interaction between elements provide distinct periods of very stable, predictable behavior. The transition between these stable periods often becomes erratic and unpredictable, i.e., mathematical chaos. These transitions can be seen as shifts in the tempo or momentum of the battle and can be caused by a single event. These chaotic transition states between periods may be the elusive overwhelming strategic effects. While this chaotic behavior is not predictable, it is also not random. Many of the defining qualities of chaotic systems [James, p. 38] such as nonlinearity, sensitivity to small disturbances, and mixing or interrelations between the variables are found in combat.

2.4.2.2. Complexity

Mathematical complexity is on the boundary of chaos. Ilachinski [Ilachinski, p. 6] describes complexity as "...the study of the behavior of collections of simple (and typically nonlinearly) interacting parts that can evolve and adapt to a changing environment." He contrasts chaos to complexity by noting that chaos involves the study of how simple systems can generate complicated behavior while complexity involves the study of how complicated systems can generate simple behavior [Ilachinski, p. 5]. The complex system is complicated because although it may have simple components, these components have interactions that generate a group behavior not characteristic of any individual's behavior. These components are usually hierarchically organized with a decentralized control or a lack of rigid external control. This is analogous to a military system with a hierarchical command and control system, but where each combatant ultimately decides upon and takes their own actions. Complex systems also have dynamics simultaneously from top to bottom and from bottom to top. This is true in military operations where a commander's orders influence the combatants' actions and the combatants' actions influence the commander's orders [Ilachinski, p. 11]. Mathematical complexity seems a very reasonable vantage point from which to approach strategic effects considering the second element of the nature of war.

2.4.2.3. Complex Adaptive Models

A special application of complexity theory is complex adaptive systems. The key to this type of model is that there is no "divine" omnipotent code that

33

controls the behavior of each combatant. Instead, the combatants are given a set of rules (doctrine) and an individual personality that favors following some rules over others. The individual agents are then responsible for making their own decisions as to how they should prosecute the battle. These decisions are influenced by the exact conditions surrounding the individual at the time of the decision

Ilachinski has developed a complex adaptive model, Irreducible Semi-Autonomous Adaptive Combat (ISAAC), to simulate the interactions between small groups of marines. He found that the Lanchester-based models did not simulate the "marine style" of fighting. That is, small independent well-trained units utilizing maneuver instead of a Lanchesterian consistent line of average soldiers. Ilachinski objects to the deterministic and aggregated Lanchester based models [Ilachinski (1997), p. iii] because they "...completely disregard the human factor in combat."

According to Ilachinski [Ilachinski, p. 39], complexity theory "...represents a shift of emphasis from force-on-force attrition calculations to consideration of high-level behaviors that emerge naturally from low-level rules." Complexity theory "...provides theoretical backbone to understanding aggregate behavior as fundamentally nonlinear and synergistic" [Ilachinski, p. 39]. It also introduces qualitative characteristics into combat. These characteristics include unit cohesion, morale, and leadership. A model based in complexity theory might meet the requirements of the second fundamental of the nature of war.

2.4.3. War is a clash of opposing wills

> An enemy can be highly unpredictable... Victory results from creating advantages against thinking adversaries bent on creating their own advantages. [AFDD 1, p. 6]

The models used to help answer the big questions in the U.S. military are the aggregated theater level models. The aggregation or averaging process strips this third element of the nature of war from combat models. These models try to simulate joint warfare with thousands of combatants employing hundreds of different weapon systems against capable if not similar opponents. Because of the vast amount of data and number of calculations required for such a broad based model, most entities are averaged or aggregated from higher resolution models that model smaller pieces of the battle in greater detail. The problem is that an average calculated from very high detail or fidelity loses detail.

2.4.3.1. Problems with Aggregation

Davis [Davis (1997), p. 28] points out that we can not work upwards in a model family (a chain of successively broader but less detailed models) with aggregated values or averages "without introducing errors that propagate with complex consequences." This is because the aggregate values are context dependent. The results from the higher (broader) models will only be accurate if the aggregate values are used in the same context from which they were created. This level of data verification is not common. Davis [Davis (1997), p. 41] suggests that, "Macroscopic behaviors have a coherence of their own that may not be readily understandable in terms of a more reductionist picture." In other words there may be large-scale influences that are inappropriate to model or unobservable at the lower level of the greater fidelity models.

Calculations are aggregate measures. Individual combatants are not actually modeled in most simulations nor is their initiative. Their contribution to lethality (firepower) and vulnerability is embedded in average values common to all other entities in the FLOT segment. The battle is waged as FLOT segment versus FLOT segment and attrition is uniformly calculated.

Another problem with aggregated models is that maneuver and other tactical advantages are not considered. One of Ilachinski's motivators for development of ISAAC was that Lanchester attrition calculations do not "account for spatial variation of forces" or advantage of maneuver. He explained this as "…the fundamental principles underlying modern land warfare with its general emphasis on maneuver and adaptation cannot be elucidated from analysis of force on force attrition alone" [Ilachinski (1997), pp. 3-4]. This is also the crux of the work by Zimm. Aggregated models do not model the variance in outcome due to brilliance or ineptness of individuals in the situation. These models have no way to capture this third fundamental of the nature of war.

2.4.3.2. Unable to capture modern Joint operations

Taylor doubts whether the aggregated Lanchester equations can capture modern warfare where the effectiveness levels of combatants may vary over the duration of the battle. Table 1 summarizes his opinions. Taylor considers the shaded area, below his "line of feasibility" least applicable to Lanchester equations. This region is representative of modern joint warfare with combined arms and fluctuating operations tempos and weapon firing intensities. A new modeling paradigm must be developed to support this region.

	No Replacements		Replacements	
	Constant Coefficients	Variable Coefficients	Constant Coefficients	Variable Coefficients
Two Homogenous Forces	Very Easy	Difficult	Easy	Very Difficult
Two Homogenous Forces With Supporting Fires Not Subject To Attrition	Easy	Very Difficult	Not too Easy	Very Difficult
Heterogeneous Forces (Several Combatant Types)	Difficult	Essentially Impossible	Very Difficult	Impossible
Heterogeneous Forces (Many Combatant Types)	Essentially Impossible	Impossible	Impossible	Impossible

Table 1. Taylor's classification of LANCHESTER-type equations for "modern warfare" and their ease of solution by analytical methods. [Taylor, p. 248]

Aggregate models use expected values or most probable outcomes, but strategic effects are anything but typical. Unfortunately, we lack the data to model strategic effects as probabilistic outcomes. Rather, strategic effects modeling should explore the range of possible outcomes in an attempt to better understand the phenomenon. To examine strategic effects, we must begin to model adaptive behavior and the decision processes (OODA loops) of the individuals at the appropriate levels of conflict.

2.5. THUNDER and Strategic Effects

THUNDER is the Air Force's premier theater-level combat model. THUNDER plans and executes 23 air missions that interact with ground combat. There is even limited capability for modeling naval and littoral combat. THUNDER accounts for most typical aspects and considerations in modern joint warfare. Most of the physical aspects

37

of a strategic attack and many of the entities expected to be effected by strategic attack can be modeled. THUNDER was designed to expressly model the air planning cycle [AFSAA]. This is effectively the theater-level joint forces air component commander's (JFACC) OODA loop and is the heart of THUNDER's capabilities.

The THUNDER air mission planning cycle takes apportionment guidance, available resources, intelligence estimates of enemy resources, and target prioritization doctrine as inputs or observations. THUNDER orients this information into an allocation of resources and a target list. The assignment of allocated resources to targets to form a mission is THUNDER's decision. THUNDER's air mission planning cycle acts by generating an air tasking order (ATO) and giving missions to the simulated units [AFSAA, p. 20]. This OODA cycle occurs at user-specified regular intervals, often every 12 simulation hours.

2.5.1. Strategic Attack Capabilities in THUNDER

THUNDER has a mission type devoted to strategic attack. In this mission area, strategic targets are defined, targeted, and attacked but the effects of attacks on these targets do not influence prosecution of the war. The strategic attack mission in THUNDER is basically a penalty mission that consumes resources that could otherwise be used against interdiction or tactical targets that directly impact typical MOEs. Thus, THUNDER does not specifically model direct strategic effects. The question then becomes whether or not THUNDER captures indirect strategic effects.

2.5.2. THUNDER's Air War

THUNDER's air planning OODA cycle is the same for both sides of the battle. The air planning OODA loop lengths can not be changed quantitatively to show or

exploit an advantage. The OODA cycle may be effected qualitatively as strategic effects can impact the quality of a decision. THUNDER allows for direct interdiction of C3I nodes and models a corresponding degrade on the targeted system. It also models the effect of poor intelligence, surveillance, and reconnaissance (ISR) on the command and control system and degrades mission effectiveness due to poor ISR. THUNDER allows for a very detailed supply distribution system that is fully interdictable. THUNDER also monitors the movement, use, and attrition of POL and any other designated critical resource. Interdiction attacks may have an OODA influence by logistically restricting options in the orient phase or foiling them in the act phase. Direct attacks on first and second echelon troops in missions such as close air support can also limit decision options and prevent decisions from being carried out. THUNDER also models simultaneous parallel attacks on targets in various strategic systems.

THUNDER's air war does not use OODA loops at any level of the simulation lower than the JFACC. The adjudication of the rest of the air war is stochastic. Again the results of the stochastic air war do feed and impact the air planning OODA cycle, but the air combatants do not have OODAs of their own.

2.5.3. THUNDER's Ground War

One of the common concerns with THUNDER is that while it is predominantly an air model, many of its results are explained in terms of the ground war. THUNDER's primary measure of effectiveness (MOE) is movement of FLOT based on attrition of the ground forces. This is not unreasonable since the ground war generates targets for a significant number of air missions. Without prosecuting a realistic ground war in the model, the air planning and targeting processes would be less realistic. This link between

39

air and ground also sounds very reasonable for a campaign level model aimed at joint operations.

The ground war in THUNDER is deterministic. Major movements of ground forces or the rules to make such moves are scripted into the campaign scenario. THUNDER simply uses the Army Concept Analysis Agency (CAA) ATCAL algorithms to adjudicate the attrition of forces and keep track of the munitions used [AFSAA]. ATCAL improves over Lanchester attrition algorithms in that it develops its own optimized attrition coefficients. The resulting attrition then drives force ratio determined reinforcement decisions and FLOT movement. These results are fed back into the air planning cycle for use in target selection and prioritization.

ATCAL attempts to address the recognized nonlinearities of battle in a method similar to blending the different forms of Lanchester equations to meet the situation [Hartman, p. 5-3]. Instead of averaging attrition across the entire force, ATCAL applies the attrition calculations on a weapon-by-weapon basis. ATCAL calculates these coefficients to make attrition computations for each weapon-target combination. Next, ATCAL uses a nonlinear iterative search until each weapon type has a near optimal target priority list (firing doctrine) to maximize the shooters' contribution to the battle. THUNDER then uses these optimized ground target priorities and weapon lethality scores to calculate the final result of the ground combat cycle. These priorities and attrition rates are recalculated within THUNDER every ground combat cycle (usually 12 or 24 hours of simulated combat).

Although ATCAL is a significant advance over traditional Lanchester equations, THUNDER does not adequately capture strategic effects. THUNDER remains a highly

40

aggregate model. The optimized units or combatants remain too homogenous and non-adaptive. Capabilities and results remain FLOT-segment oriented. Finally, there is no air planning OODA loop equivalent in the ground war, removing any strategic effect link between the air and ground components.

2.6. Developing a Model to Explore Strategic effects

One method for modeling the effect of an attack on a strategy is by modeling its impact on OODA cycles. To prove the viability of this concept a model is developed to examine whether OODA length differences produce the synergistic and catastrophic advantage expected from a strategic attack as predicted by current AF doctrine. Most current combat models are research models [Davis and Blumenthal] and are consolidative in nature. That is, the models attempt to embody all current knowledge about an event in order to be used as a surrogate for that event [Bankes]. However, little is known about strategic effects and field experience is limited. Instead a model is needed with which to explore the options and issues of strategic effects. Armed with such a model, one may gain insight into strategic effects by using the concept of exploratory modeling [Banks]. Complex adaptive models are inherently exploratory. A complex adaptive model in which combatant OODA loop lengths vary (presumably due to strategic targeting) is proposed, developed, and used for exploratory analysis of strategic effects.

STRATEGIC TARGET	Reason	Violence	Chance
Strategic Entity	Leader's will	Nation's will/ability	Military ability
System	Pol/Mil Leadership	Culture	Small unit (squadron)

System's Leadership	Laws
	Customs / Morals
	(civic/religious
	    doctrine)
		National leadership
		Customs / Morals
		(civic/religious
		    doctrine)
			Political Leadership
			Civic/religious doctrine
			Military Leadership
			Military Doctrine

Organic System's essentials	National Support National Resources National Need Military Capability	Belief in political goal Foreseeable improvement	Weapons Training/Doctrine Loyalty/cohesion Conviction in cause/leader
System's Infrastructure	Communications	Communications	Communications Logistic system Mobility system
System's Population	Citizens Allied nations	Impact on lifestyle Expectation of gain	Soldier or basic combat element
System's Fighting Mechanism	Military Political machine (UN) Media Propaganda	Personal sacrifice of citizen	Fielded weapons systems
Typical Agent	Courses of Action	Citizens	Basic combatants (or crew)

Table 2. Strategic systems appropriate for OODA exploitation experiments with autonomous agents.

2.6.1. A Proof of Concept Model

To prove that OODA advantage translates into strategic advantage the model

represents individuals with their own OODA decision processes. These individuals

represent any level of decision-making entities. Some likely choices are shown in Table 2.

For this effort these individuals will be modeled as basic combatants. The method

by which the OODA advantage is achieved is not critical at this time, only the fact that

the OODA advantage exists and that it effects the outcome. Since an OODA advantage is

a phenomenon of finite time, the model should have continuous exploitation, therefore a constant OODA advantage for one side is most appropriate for proving the concept.

2.6.2. Model Format

The prototype model for this research is influenced by the ISAAC model. ISAAC is a complex adaptive system of combat marine "agents" in a ground warfare environment. Most complex adaptive models consist of agents that independent actions and decisions at uniform time intervals. In these models, a master schedule or simulation clock controls these time intervals, which typically use a common timing scheme (OODA loop length) for all agents' actions. In contrast, multi-threading is used to keep the agents in this model as autonomous as possible. Multi-threading allows several programs to run on the computer simultaneously. The advantage in this case is that each agent is its own self-contained mini-program and runs relatively free of the typical control and overhead of the main program. The programming language *Java* was chosen because it incorporates multithreading. Each multi-threaded agent has an adjustable length OODA loop.

2.6.2.1. The Playing Field

The simulation battleground is rectangular and contains two solid squares representing the base of each force in opposite (diagonal) corners (see Figure 9). Each force, the blue and the red, is initialized with a typical strength of 20 combatants in front of their respective base. During the simulation, those agents that are killed are moved to the lateral boarder of the battlefield and turn pale in color. Those agents that reach their goal of the enemy base are moved to the vertical boarders and highlighted in white (see Figure 10). A shot is indicated by a solid line of the color of the shooter.

43
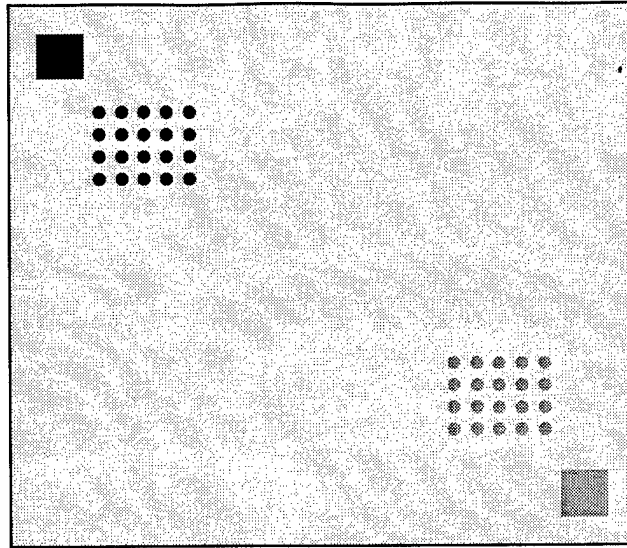
Figure 9. Sample simulation at initialization

2.6.2.2. Agent External Behavior

The agents initialize to positions near their base. When the simulation begins the agents move toward the enemy's base to attack. The agents try to keep a specified minimum distance from other agents. If enemy agents are encountered the agents must decide to either attack, fall back and regroup, or continue advancing toward the enemy base. These decisions are based upon the force ratio within each agent's field of view. User adjustable (doctrinal) force ratio thresholds define these decisions. If the base is attacked and the agent is within communications range of his base, he may be recalled to defend the base. An agent defending his base is assumed to have a prepared defensive posture and is able to withstand a few more hits from enemy weapons. If an agent decides
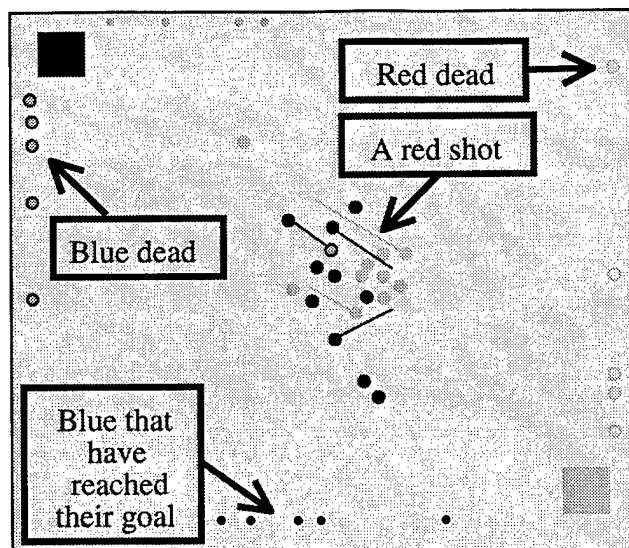
Figure 10. Sample simulation during run

to attack, he moves to a point on a line between the perceived center of mass of the enemy and his base. The position on that line is at his doctrinal best firing range from the perceived enemy's center of mass. The success of an agent's shot is based on the Pk specified for the agent. There is a linear accuracy degrade if the agent shoots from between his optimal firing range and his maximum firing range. If the agent decides to regroup, he moves to the perceived center of mass of his own troops. The agent's exact move is toward the average of the moves he should make for each activity (attack, regroup, etc.) weighted by his personality preferences (individual weightings or tendencies to favor one behavioral rule over another) for each activity. An agent's maximum speed or distance per OODA cycle also restricts the move.

2.6.2.3. Agent Internal Behavior

An agent's internal actions are processed as an OODA loop. Each agent observes all other agents within his field of view. The agent then orients by prioritizing his threats

45

and keeping track of a specified maximum number (default is 1) of threats. He computes the center of mass for the threats he is tracking and for the friendlies he can see or communicate with. He compares the force ratios for the friendly to the enemy and decides the most advantageous position to accomplish his objectives based on doctrine and personality. The agent then calculates a firing solution on the enemy threats. At this point the agent's thread is put to sleep to represent the finite amount of time required to think through the decision process and put the decision into action. While each step in the OODA loop has a finite amount of time associated with it, the sum of all those delays is represented at this one point in the model. When the agent thread wakes up or resumes operation, he moves to his precalculated optimum position. The actual position may be adjusted slightly to keep from stepping on another agent who has also chosen a "best" position close by. The agent then fires at the precomputed target coordinates of each enemy he has labeled a threat. This penalizes an agent with old information. A targeted agent can not return fire immediately, but must wait to complete his OODA cycle. There is no wounded status and it takes only one hit to kill an agent.

2.7. The Experiment

For the purposes of this experiment the exact behavior of the agents is not as important as the fact that the behavior is the result of an OODA process. The only concept for action that must be adhered to is that there must be a discernable penalty for acting with poor information. The behavior described above meets this criterion.

The purpose of the experiment is to study how two comparable forces fare given differing OODA loop lengths. Can an OODA difference provided by strategic attack

really deliver the knockout blow? A longer OODA process can be manifested in two ways. The first is that the decision and corresponding action get delayed. This could be extra time in gathering information, sorting through options (including waiting for orders to be issued), committing to a decision, or just putting the action into motion. This will be the first scenario in the experiment. The second manifestation is when a time critical decision must be made but the decision-maker has incomplete information. This will be the second scenario in the experiment. The first scenario in the experiment changes only the OODA length. The red agents will make fewer decisions and take fewer actions than the blue. In the second scenario, the speed of the red agent is increased proportional to the increase in OODA length. The impaired (red) agents can move at the same overall rate as the blue agents, but make longer range decisions and have less reaction capability.

2.7.1. Experimental Design

The experiment consisted of 4 scenarios, each with 17 cases. The first scenario is the baseline and kept the maximum speed the same for each agent. That is the agents moved the same maximum distance per OODA cycle. The red made fewer decisions and moved at a correspondingly slower average rate than the blue. The second scenario gave the red agents a speed increase proportional to their OODA length increase.

Scenario	Maximum speed		Initial Strength		Probability of Hit (SSPK)	
	Blue	Red	Blue	Red	Blue	Red
1	fixed	fixed	20	20	0.5	0.5
2	fixed	varies	20	20	0.5	0.5
3	fixed	fixed	20	40	0.5	0.5
4	fixed	fixed	20	20	0.5	1.0

Table 3. Summary of experiment scenarios

This gave the red fewer decisions, but further distance per decision and about the same miles per hour as the blue. The third scenario had the same max speed as in the first scenario, but red's initial force size is double that of the blue. The fourth scenario increased red's weapons effectiveness to twice that of blue. Each case of each scenario had a different OODA differential. The OODA differential was increased from no advantage to a 400% advantage in 25% increments. A 400% OODA advantage means blue will be completing 5 decision cycles to every one of red's. This may be unrealistic in a real situation, but was included in the experiment to study the nature of the phenomenon. Each case was run for 30 repetitions to collect statistics. A sample of the number of agents alive and the number of agents that penetrated the enemy base was taken at each second during the simulation. The simulation run time was restricted to 20 seconds, which allowed most of the agents alive after the fight to reach their goal (the enemy base). The two metrics used are the number of alive agents and the number of agents at the goal.

2.7.2. Results

The OODA differential has a significant impact on the outcome of the "battle." In all scenarios, the blue gained a significant advantage with OODA increase, although the advantage did not increase monotonically.

2.7.2.1. Scenario 1, Case 0: No OODA Advantage

The number of agents that are alive throughout the battle are shown in Figure 11. Red has an unexpected advantage due to the nature of the battlefield. Specifically, red is
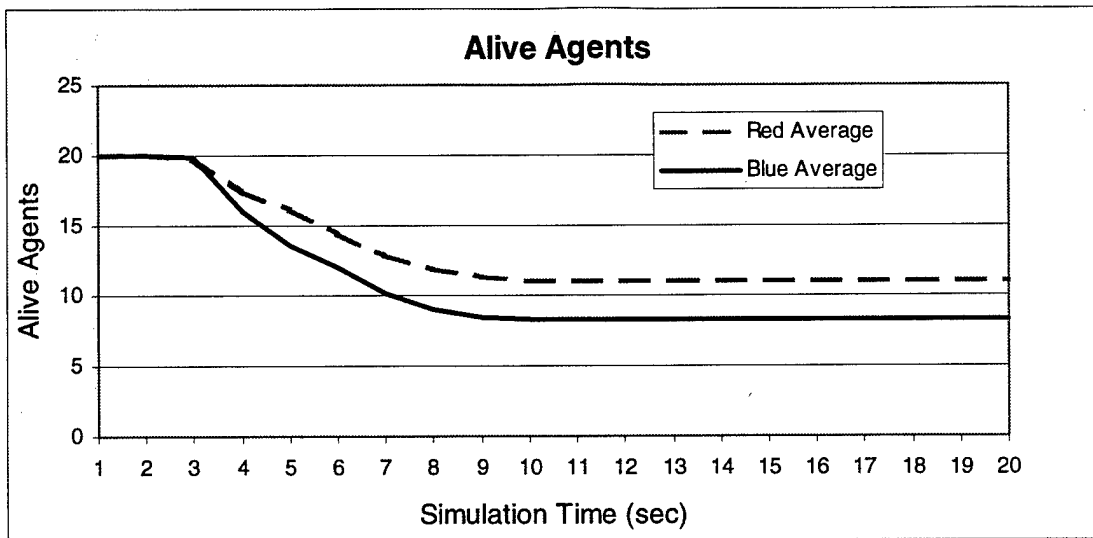
Alive Agents

Figure 11. Average number of alive agents with respect to time in Scenario 1, Case 1 (no OODA advantage)

in a position to fire first and this gives red an early advantage that it maintains. Both sides suffer attrition near 50% before disengaging to advance toward their respective goals.

2.7.2.2. Composite View of Scenario 1

Each of the other cases in scenario 1 had blue with an OODA advantage. The curves of attrition with respect to time for each case can be placed next to each other to produce a 'landscape' surface of the number of agents alive. These landscapes are shown for the blue in Figure 12 and for the red in Figure 13. The case 1 (no OODA advantage) attrition curves from Figure 11 are shown as the left edges of the landscapes. When blue has a 50% OODA advantage it has a clear advantage in both attrition and reaching the goal. Blue is in complete control of the situation when there is greater than a 50% OODA advantage. There blue attrition falls to almost nothing. Reds attrition continues to fall,

49

but is marked by a somewhat regular cycle of ridges and valleys. These ridges and

valleys are from the relative firing advantage derived from the specific physical

characteristics of the simulation. There are two causes for these ridges and valleys, which

is discussed in terms of a red ridge. The first is that a red move left many of the agents

just outside of blue's firing range and put red in a good position to fire on the next move,

giving a red first strike advantage. The second is that both sides were advancing to fire

and charged passed each other without much of a fight and fewer casualties. The nature

of these ridges and valleys change with the average step size (speed) and weapons range

of the agents and their relative position at initialization.

Similar landscapes showing the number of agents to reach their goals are given in

Figure 14 and Figure 15 although the orientation of the landscapes is changed to better

see the results.
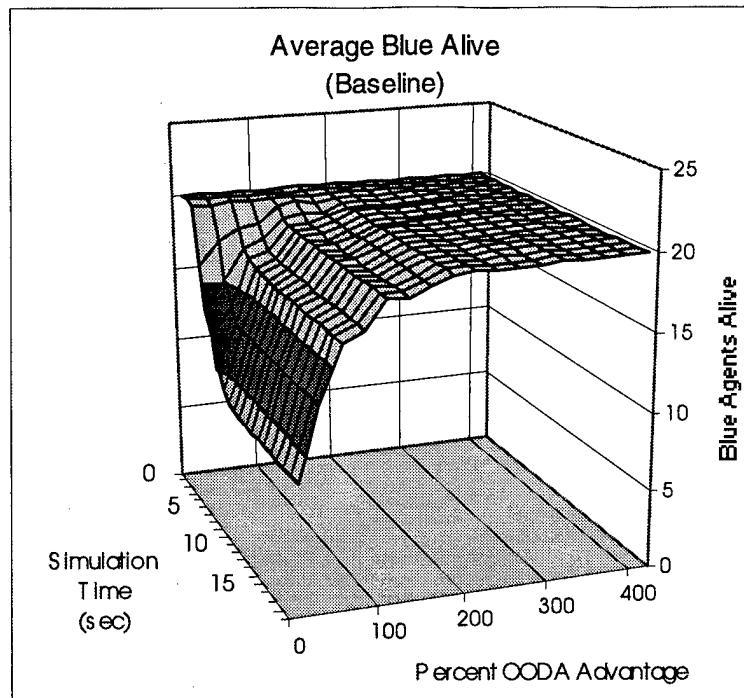
Figure 12. Landscape of Alive Blue Agentsfor Scenario 1
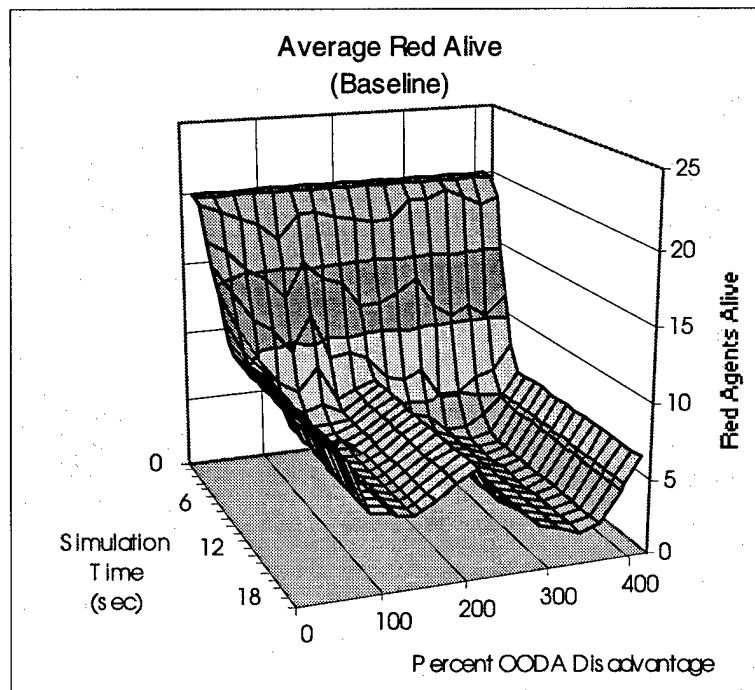
Figure 13. Landscape of Alive Red Agents for Scenario 1

51
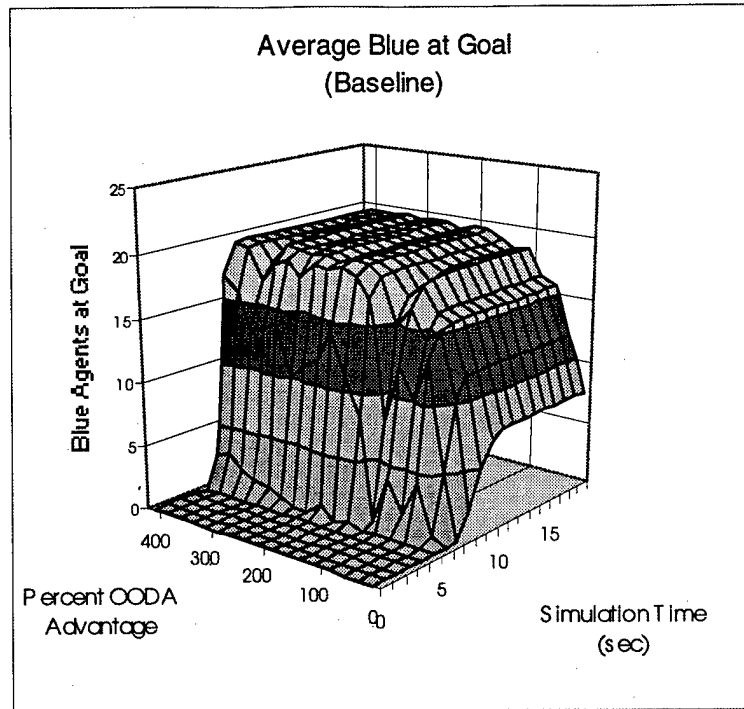
Figure 14. Landscape of Blue Agents at Goal in Scenario 1
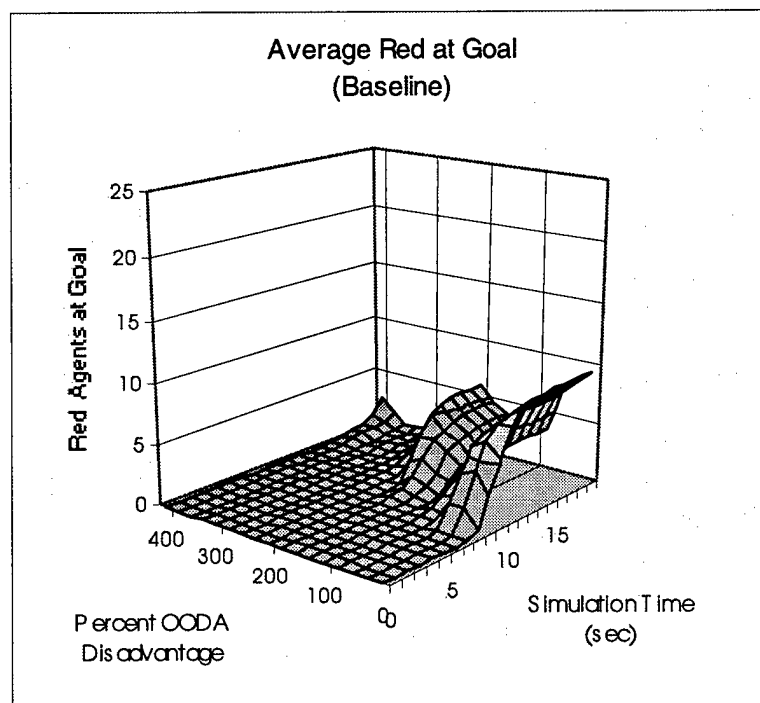
Figure 15. Landscape of Red Agents at Goal in Scenario 1

A comparison of the average final attrition for both sides can be seen in Figure 16.
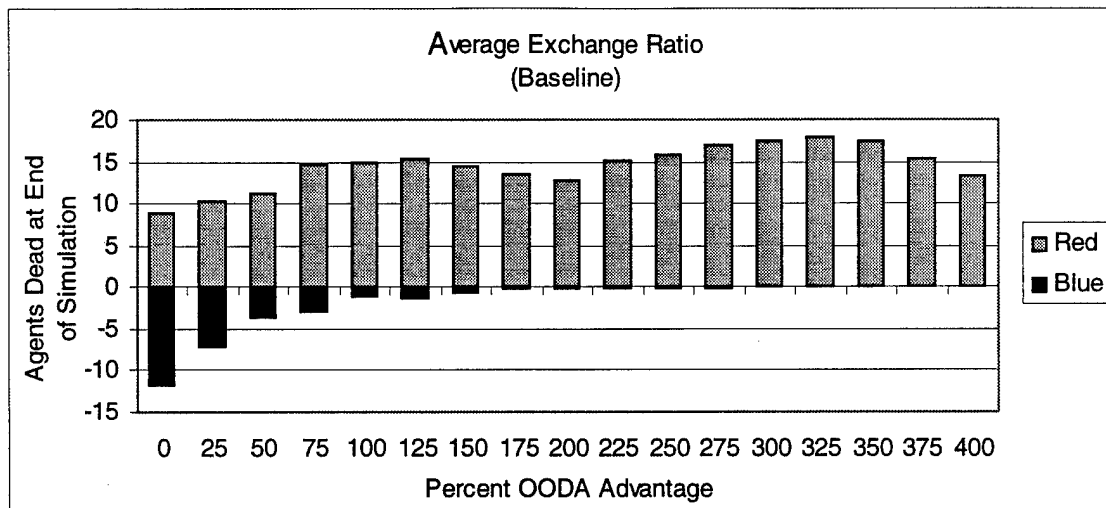
Figure 16. Exchange Ratios for Scenario 1

Lanchester predictions where made to test if the OODA advantage results differ from traditional model predictions. The Lanchester equations have two main variables. These are the attrition rate coefficients for the two opposing sides. This coefficient is defined as the rate of fire (ρ) times the single-shot probability of kill (SSPK). The SSPK is fixed in this scenario at 0.5 for both sides. The rate of fire can change with OODA advantage since the blue have an opportunity to shoot once in the act phase of each OODA. This rate of fire is the only variable that is actively changed between cases. The ratio of the blue attrition rate to the red attrition rate is expected to be no greater than the OODA advantage. This was not the case. In fact, the observed ratio of attrition coefficients differed so much from the OODA advantage that a logarithmic transformation was needed to view results (see Figure 18). This view clearly shows that the results of the simulation with an OODA advantage are drastically (synergistically)

53
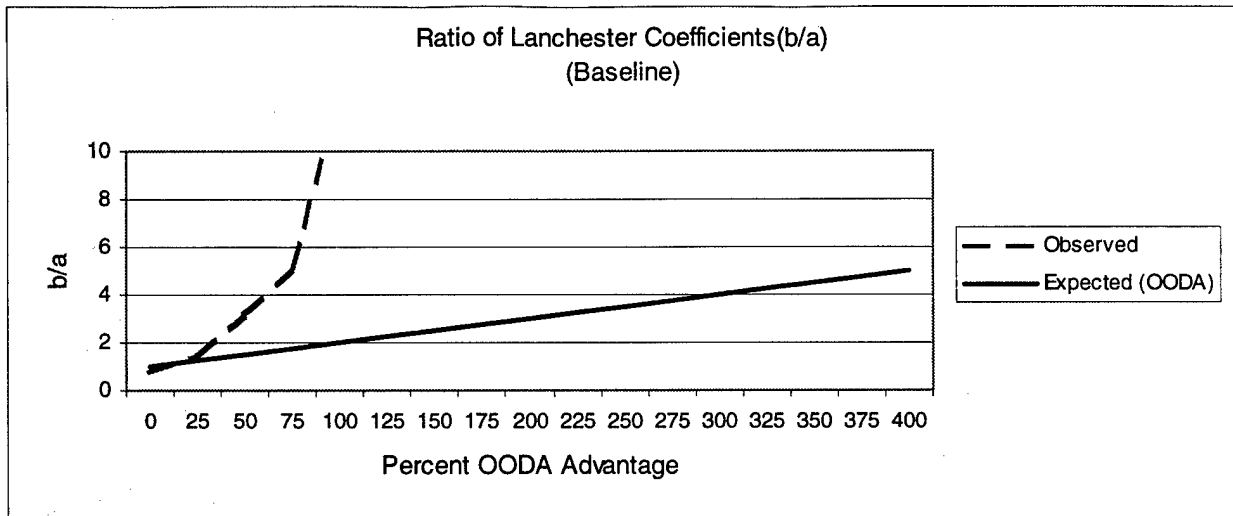
Figure 17. Observed versus expected ratios of Lanchester attrition rate coefficients

different than the Lanchester prediction. If on the other hand, the actual attrition data is

put into the Lanchester equations in an effort to derive attrition coefficients, the
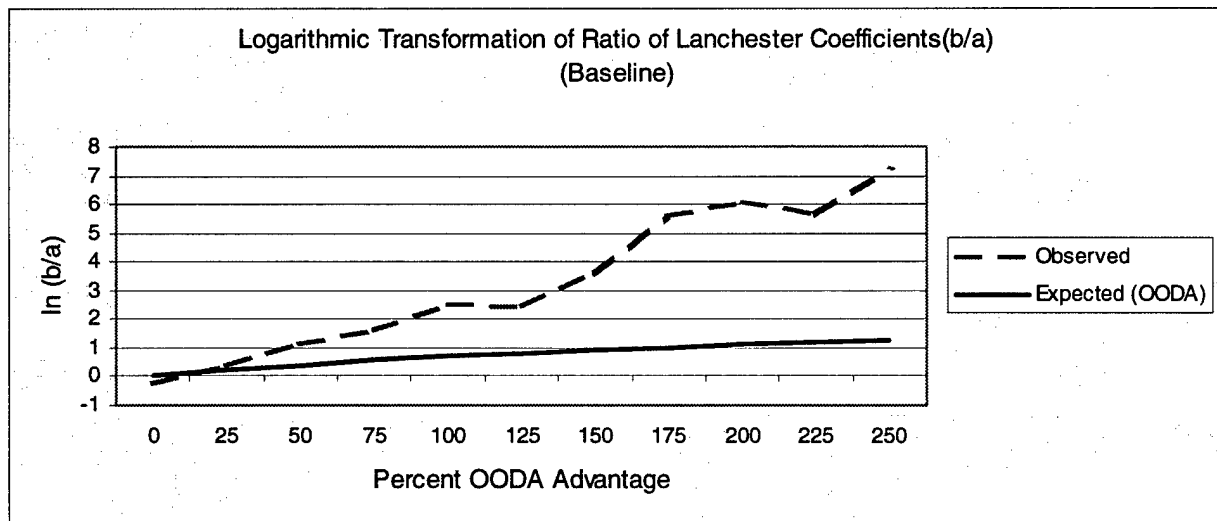
Figure 18. The ln transformation of the ratios of Lanchester coefficients in scenario 1

Lanchester equations can be made to 'fit' the data. In fact in Figure 19, the Lanchester

square, linear, and mixed laws all follow the general shape of the 'real' data. This

demonstrates Hartley's point that many forms of the Lanchester equations can be made to

fit data [Hartley, p. 448], but not reliably predict outcomes.
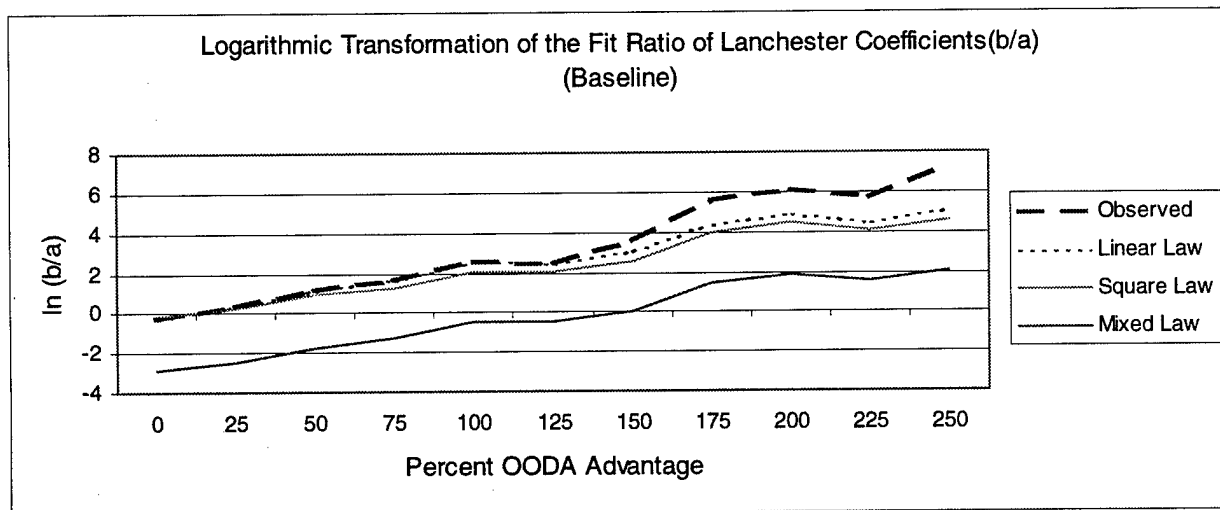
Figure 19. The ratios of coefficients of various Lanchester laws 'fit' to scenario1 data

2.7.2.3. Reasons for the Effects

Recall that the two components of the Lanchester attrition rate coefficients are

rate of fire and SSPK. Even though the SSPK was held constant, it effectively changed in

the execution of the model. The SSPK can be broken down into the probability that the

shot hits the target (P_H) times the probability that the shot kills the target if it was a hit

($P_{K/H}$). The probability that the shot hits the target is primarily due to the accuracy of the

weapon system, especially if the target is fixed. If the target is a non-cooperative target

and maneuvers during the shot, the target effects the probability that it will be hit. This is

the case demonstrated in Figure 20 and contributes to the rapid increase in blue

survivability. This effect can be seen in the effective SSPK calculated from the model

output (see Figure 21). Blue's SSPK stays constantly near the specified value of 0.5,

while red's SSPK begins to drop significantly at around the 150% OODA advantage

point.
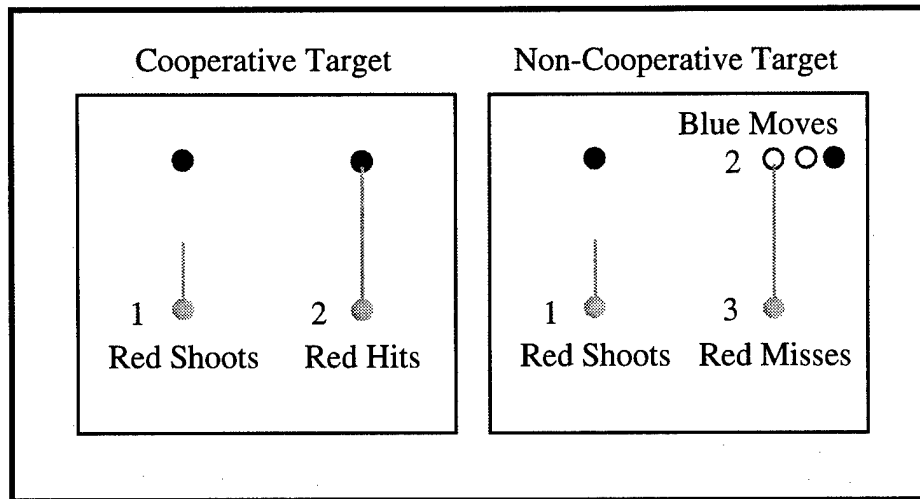
Figure 20. Cooperative and non-cooperative targets
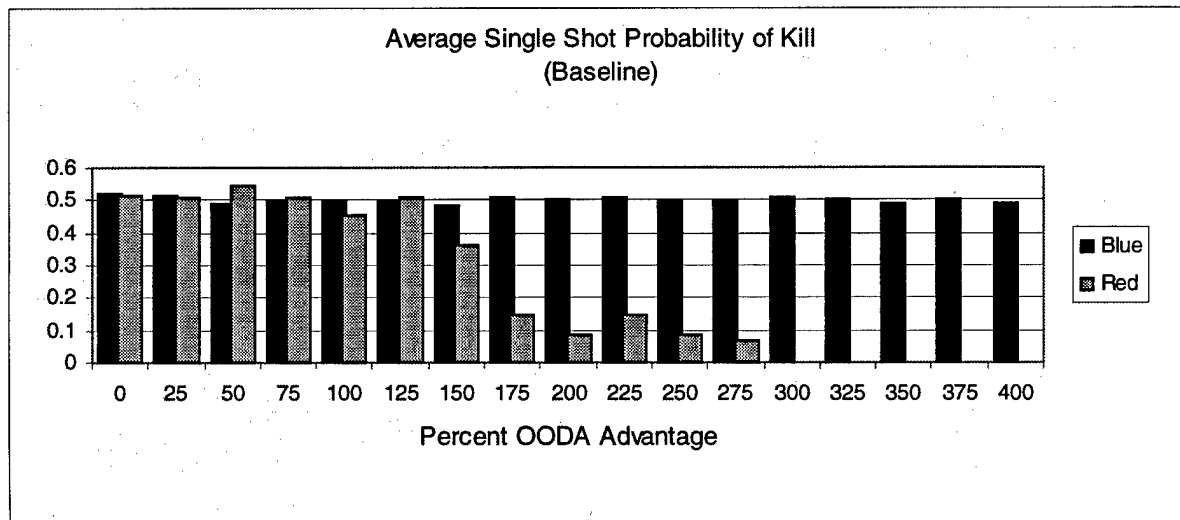
Figure 21. Average Measured SSPK in Scenario 1
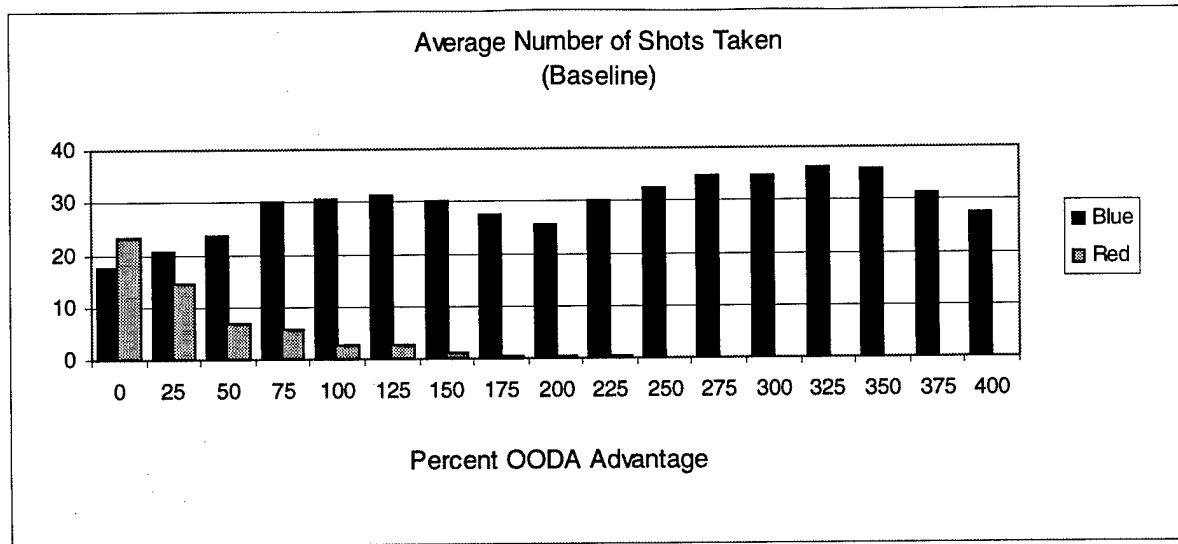
Figure 22. Average Number of Shots Taken in Scenario 1

The maneuver advantage given by the quicker OODA loop also allows blue to get into a firing position faster. The OODA advantage allows blue to fire before red can respond (surprise). The effect is that those red preparing to fire are killed before the reaching the act (shoot) phase and the 'second echelon' red are not yet preparing to fire when the blue engage. The result of this is that the red make fewer shots in the first few moments of combat leaving blue with an insurmountable numerical superiority. Figure 22 shows the average number of shots taken for each case in scenario 1. Fewer shots taken and a lower actual SSPK for those shots quickly drive the red attrition rate coefficients to near zero.

2.7.2.4. Scenario 2: Red Speed is Proportional to OODA Difference

In this scenario the distance a red agent can go on a single move is increased proportionally to the reds' OODA length disadvantage. The result is the red move at the same average speed as the blue even though it completes fewer OODAs. This 'forces'

57

red to make decisions further in advance of their moves, giving them less current information. The results (see Figure 23 and Figure 26) are very similar to those in scenario 1, although the ridges and valleys on the landscapes (see Figure 24 and Figure 25) have shifted due to the change in red step size. The agents especially have more of a
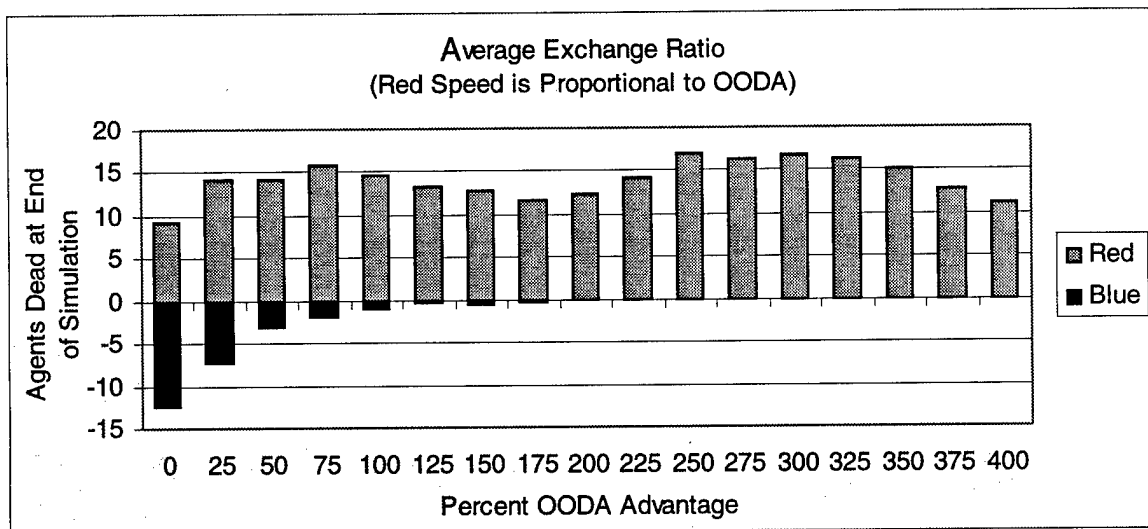
Figure 23. Exchange Ratio in Scenario 2

tendency to run past each other in this scenario as the OODA length and red maximum speed get large. The SSPK drops quickly for red, but shows a region of improvement linked again to the time-space relationships of the battle. This point is where the blue OODA cycle effectively "laps" the red, putting the red in a more advantageous position although an inconsequential cycle behind (see Figure 27). This demonstrates one criticism of Boyd's theory, that faster OODAs are not always better [Fadock, p. 18]. Instead, the best OODA pace is one that is comfortable for you, but not for the enemy. The number of shots fired in this scenario drops very fast for the red showing again that blue's first strike advantage is intensified by red's long decision lead time (see Figure 28).
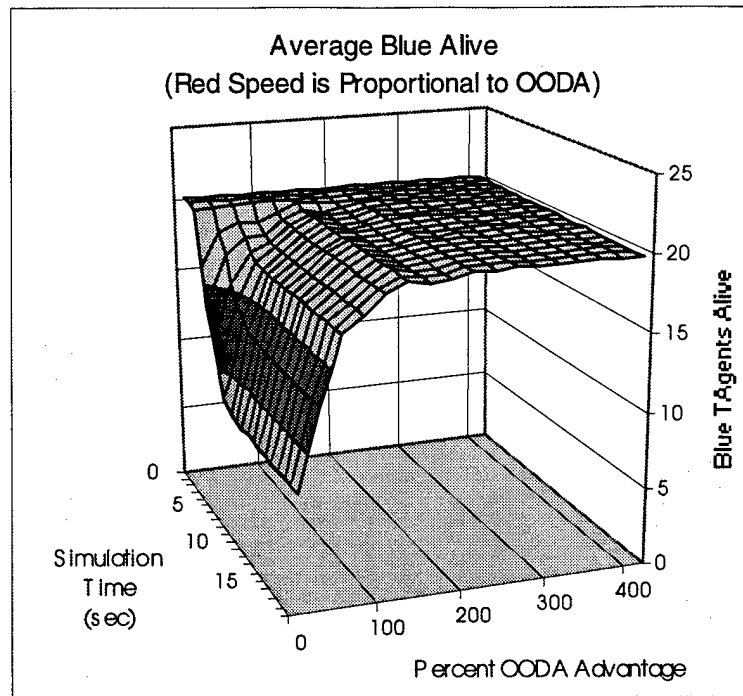
Figure 24. Landscape of Alive Blue Agents for Scenario 2
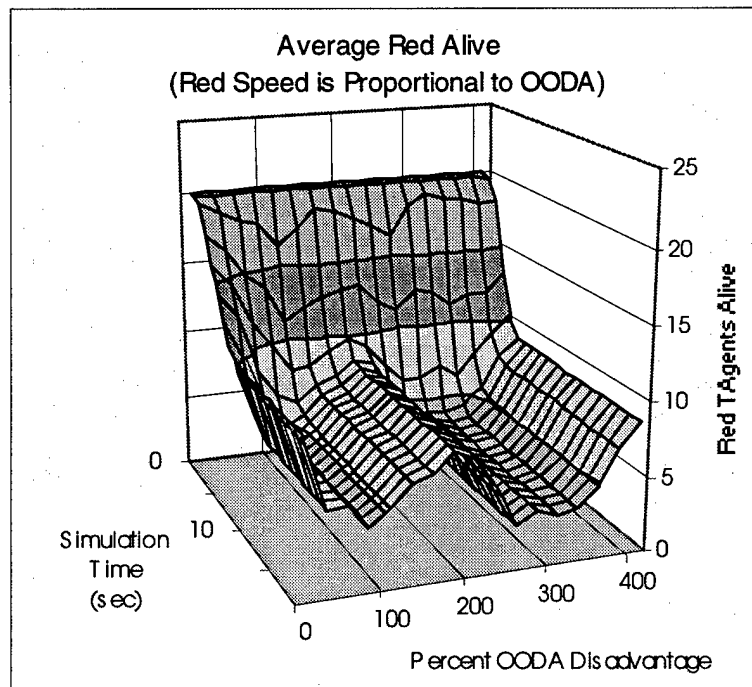
Figure 25. Landscape of Alive Red Agents for Scenario 2

59
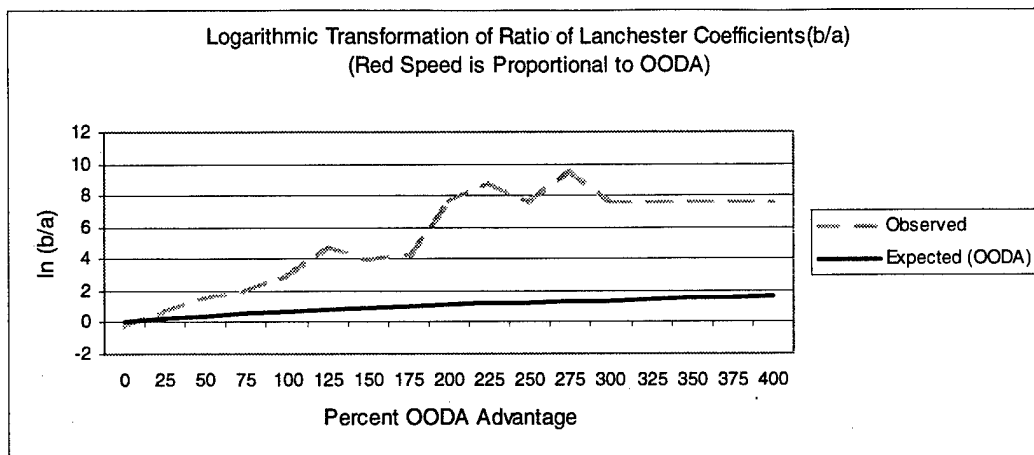
Figure 26. The ln Transformation of the Ratios of Lanchester Coefficients in Scenario 2
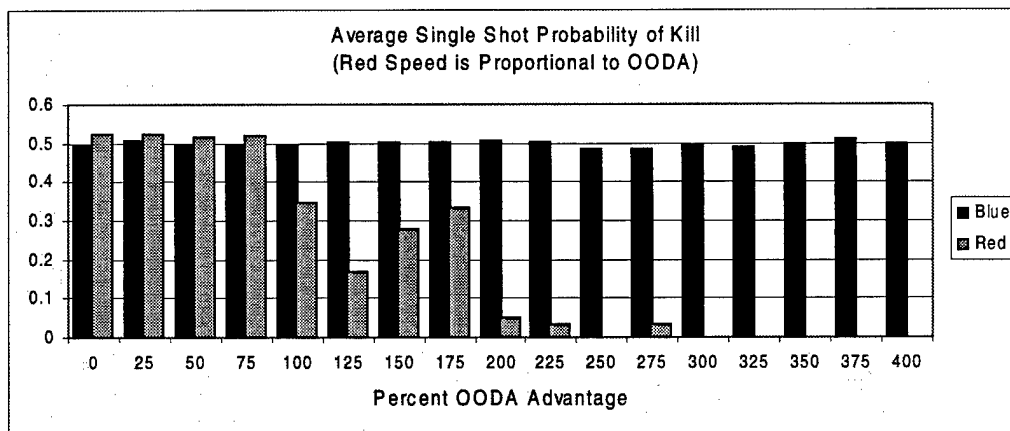
Figure 27. Average Measured SSPK in Scenario 2
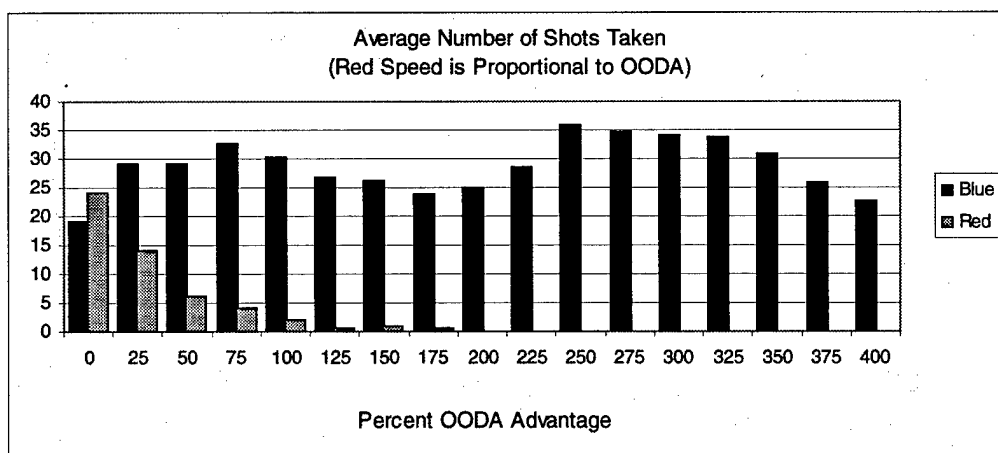
Figure 28. Average Number of Shots Taken in Scenario 2

2.7.2.5. Scenario 3: Red initial force size is double that of the blue

Scenario 3 initializes red with 40 agents and blue with 20. Red's initial size gives it quite an advantage over the other scenarios. Blue is just as effective as before, but the number of red nullifies the blue first-strike, leaving a fairly even match for the remainder of the battle. With only a 25% OODA advantage, blue matches red attrition and maintains its relative force size (20 fewer agents). This turns the apparent square law situation into a linear law result. With an OODA advantage of just over 200%, blue overcomes its initial deficit and ends the battle with a greater number of surviving agents (see Figure 29).
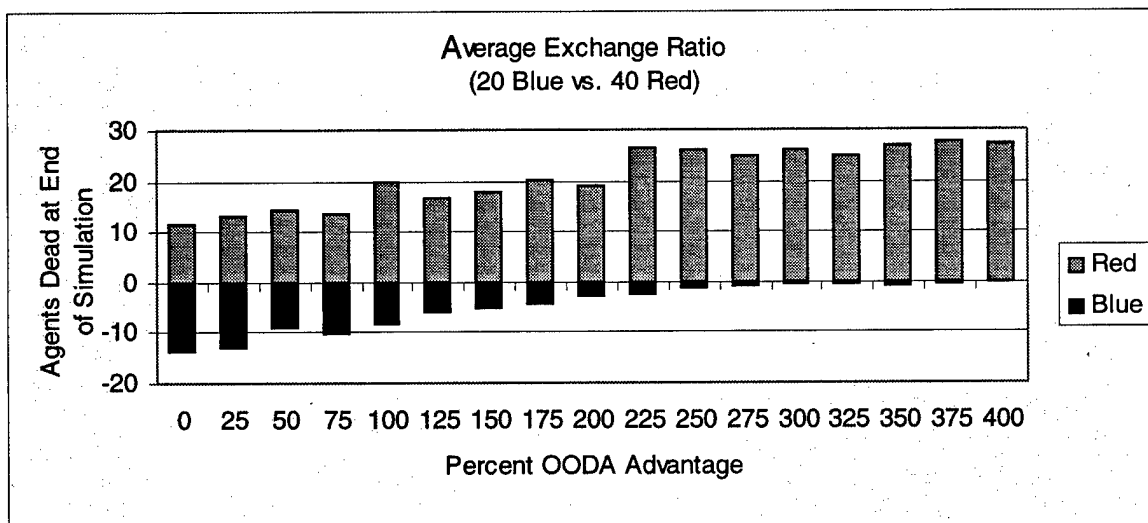
Figure 29. Exchange Ratio in Scenario 3

The red SSPK (Figure 31) is not as sensitive to the OODA advantage in this scenario. This is because the blue first strike does not kill most of the 'ready' red shooters. The higher number of surviving red shooters are in a good firing position with vulnerable blue agents that are just beginning the observe phase. The number of red shots

61
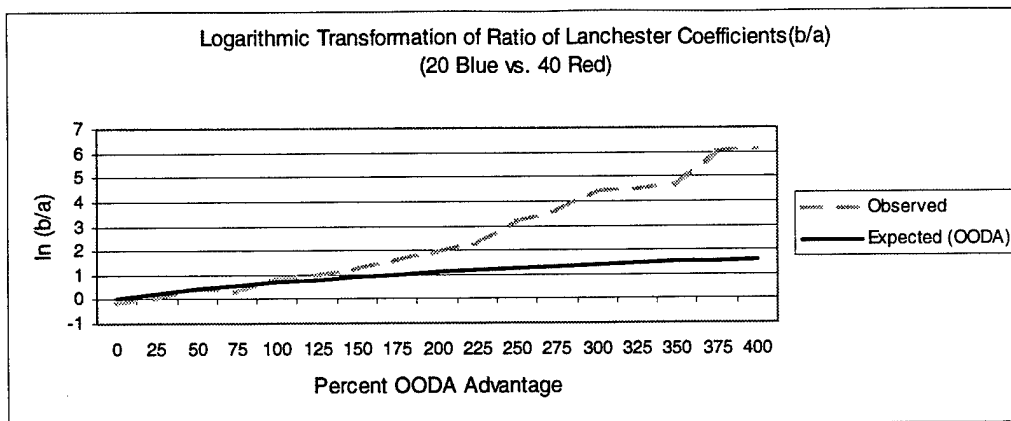
Figure 30. The ln of the Ratios of Lanchester Coefficients in Scenario 3
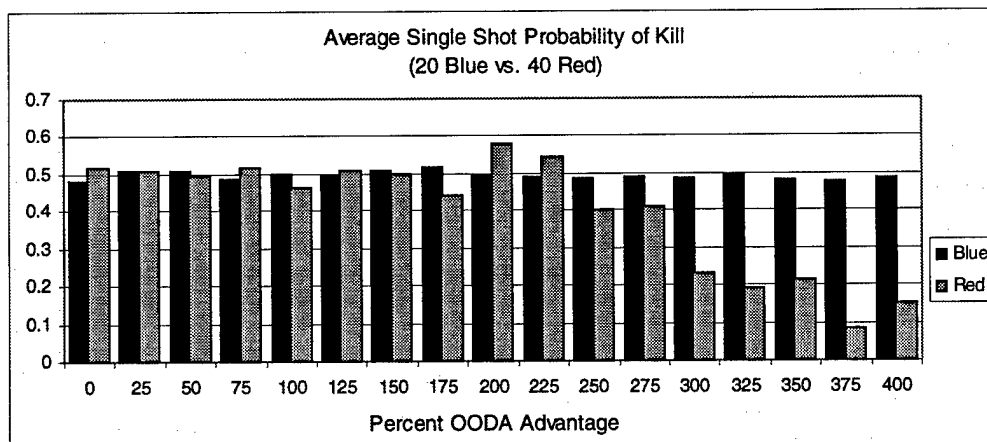
Figure 31. Average Measured SSPK in Scenario 3
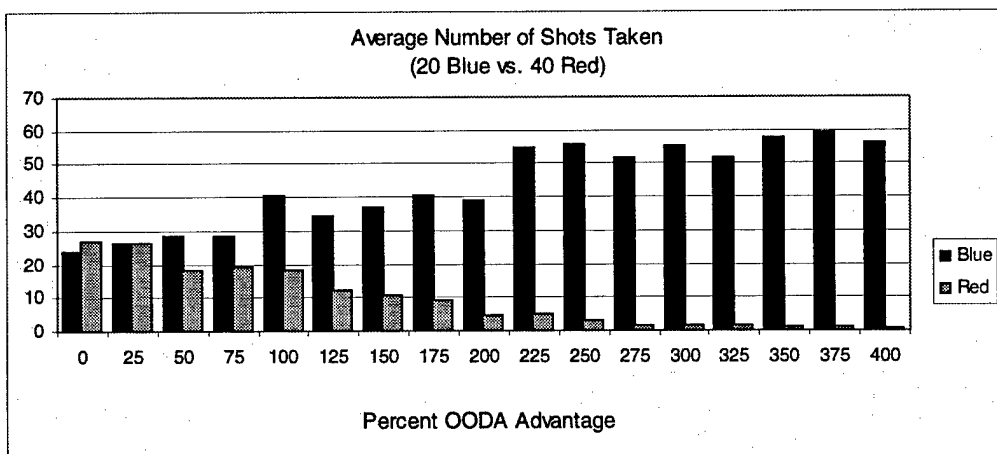
Figure 32. Average Number of Shots Taken in Scenario 3
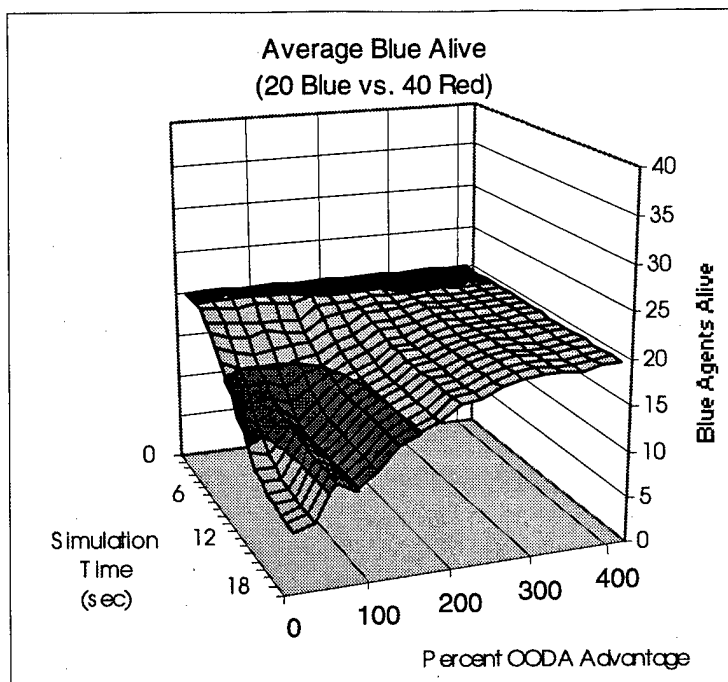
Figure 33. Landscape of Alive Blue Agents for Scenario 3
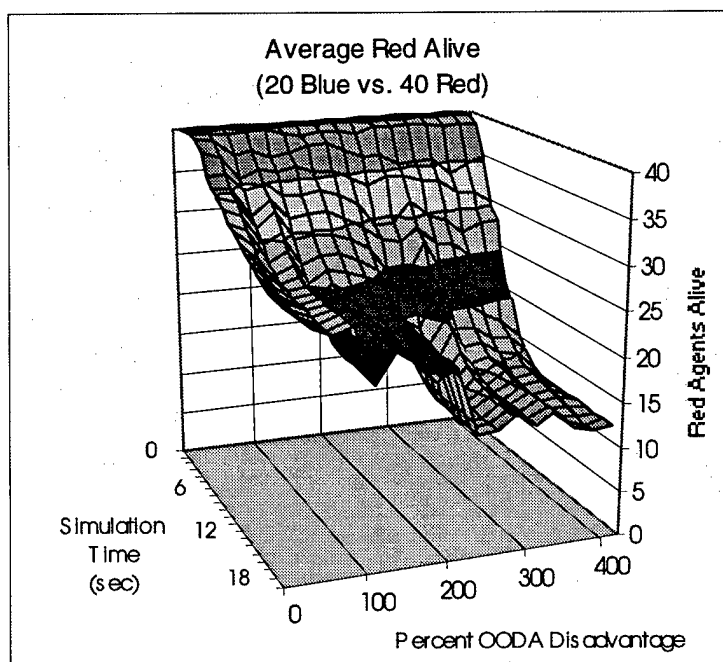
Figure 34. Landscape of Alive Red Agents for Scenario 3

(Figure 32) also tapers of much more gradually in this scenario as more alive red agents

take opportunities to return fire.

2.7.2.6. Scenario 4: Red weapon effectiveness is double that of the blue

Red enjoys a clear attrition advantage with low OODA differential due to their

increased firepower. This advantage is lost with an OODA differential of 75% or greater

(see Figure 35). The results of this scenario (see Figure 36, Figure 39, and Figure 40)

resemble those of the first two scenarios despite the higher red SSPK (Figure 37). This is

because red doesn't have many opportunities to take shots (see Figure 38) to capitalize on

its strength when blue has an OODA advantage.
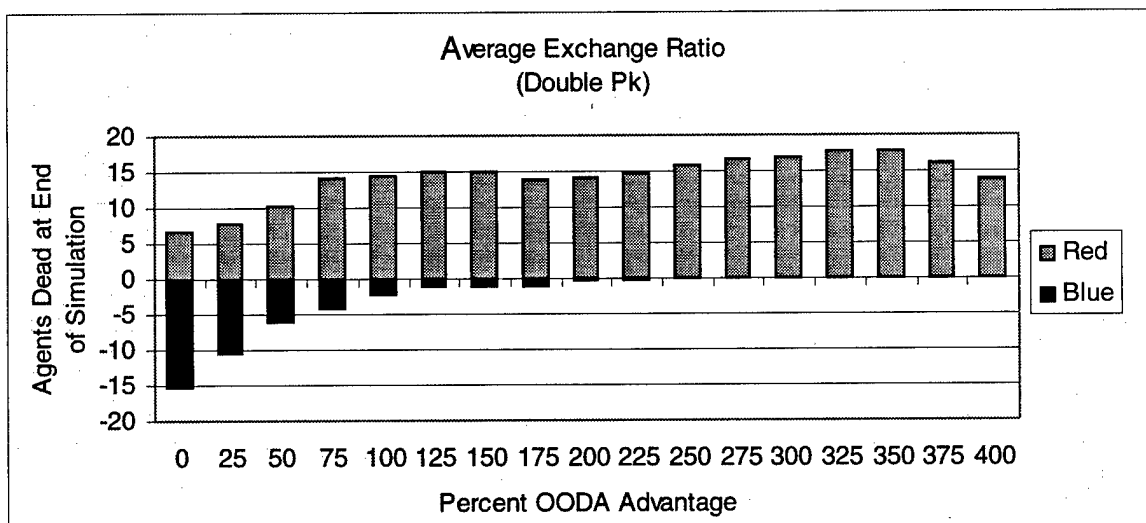
Figure 35. Exchange Ratio in Scenario 4
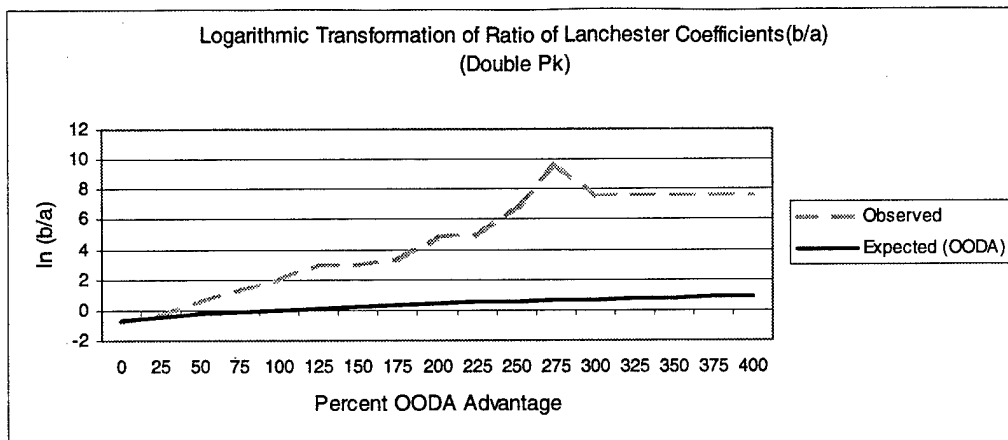
Figure 36. The ln of the Ratios of Lanchester Coefficients in Scenario 4
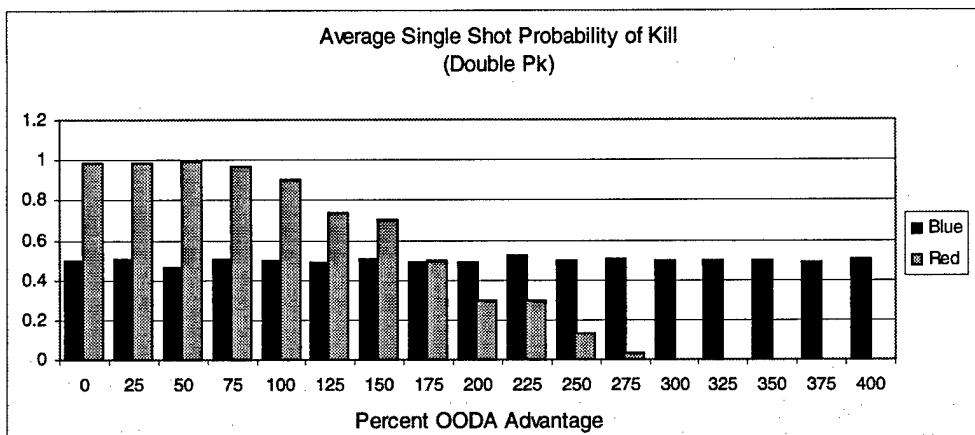
Figure 37. Average Measured SSPK in Scenario 4
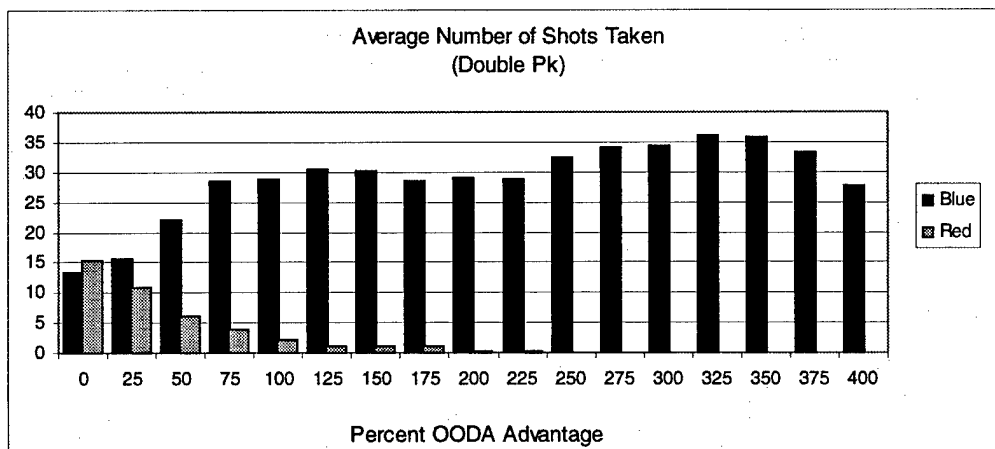
Figure 38. Average Number of Shots Taken in Scenario 4
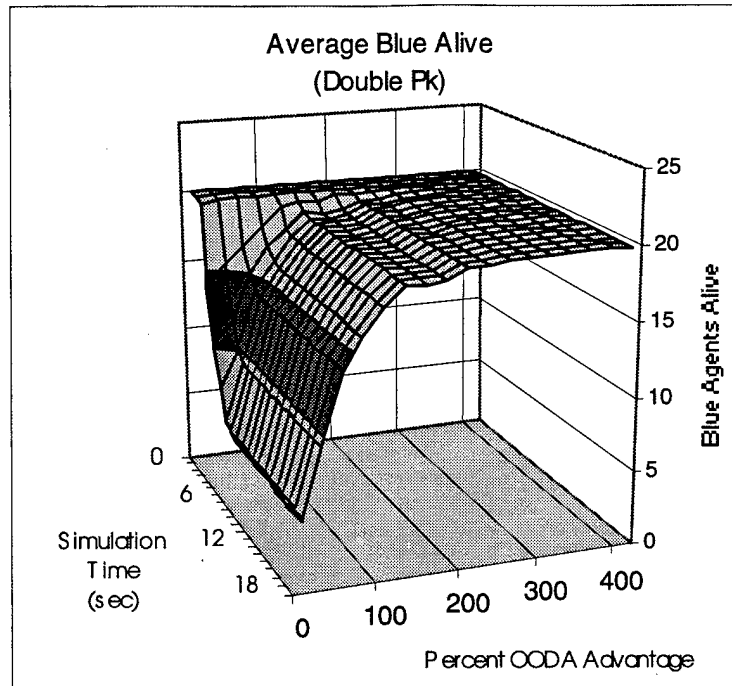
Figure 39. Landscape of Alive Blue Agents for Scenario 4
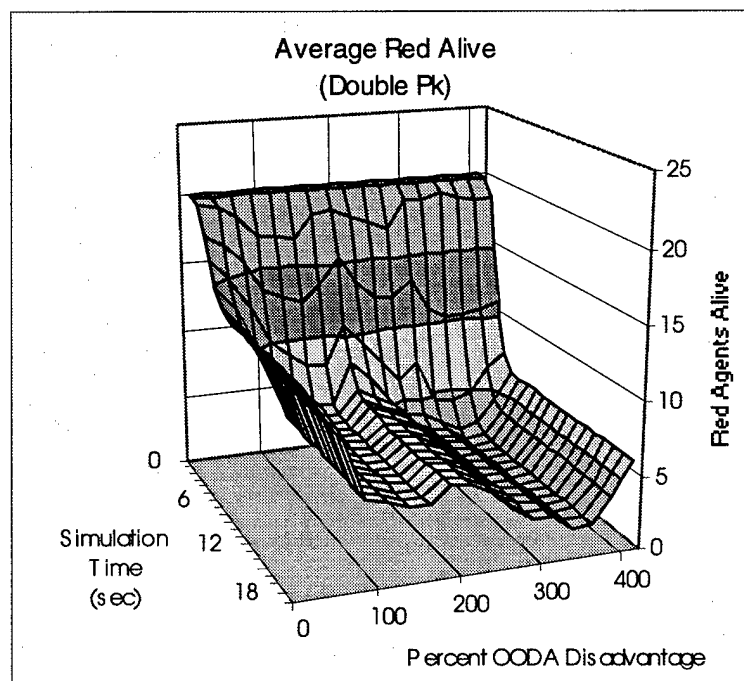
Figure 40. Landscape of Alive Red Agents for Scenario 4

2.8. Conclusions

The OODA exploitation model of strategic effects is consistent with the doctrinal foundation set forth in Air Force doctrine documents. It attempts to focus on the nature of war instead of the mechanics of attrition. The insight to the nature of war comes not only from AF doctrine, but also from Clausewitz. The model combines the modern theories of Warden and Boyd and explains some discrepancies established by Pape.

The model is different than most contemporary combat models because it models simple 'situational decisions' made by autonomous agents instead of adjudicating theoretical and elusive rigid laws of combat. Deterministic and stochastic models of combat are too inflexible to show the impact of human decisions on the battle. Aggregation robs initiative and other tactical advantages from the combatants. THUNDER is founded on a decision cycle, but cannot carry the decision process modeling down to the level of combat to show OODA exploitation.

All four scenarios of the experiment show a beneficial effect from OODA exploitation. A blue OODA advantage was not only decisive in otherwise even fights, but also quickly eroded a red military advantage. This is consistent with the doctrinal expectations of strategic effects. The advantage comes from the 'initiative' of the blue, who capitalize on a tactical opportunity for a first-strike attack before red is prepared to respond. This initial attack puts red at an insurmountable disadvantage for the rest of the battle in three of the scenarios and equalizes the initial blue deficit in scenario 3. The quicker reaction time of the blue also makes them less cooperative targets, confounding red's situation. The effect of OODA exploitation is often much greater than the Lanchester attrition equations would reasonably predict. The effect is synergistic

in that the advantage is disproportional to the actual increase in expenditure of firepower. This is a classically anticipated characteristic of strategic effects.

The OODA exploitation model of strategic effects is consistent with doctrine and theory, explains the shortcomings of other models for explaining strategic effects, and produces results consistent with strategic effects. Autonomous agents provide a vehicle to test and validate this approach to modeling the moral aspects of combat. The model presented in this thesis for proof-of-concept does provides a reasonable starting point. The results of the proof-of-concept model are sufficient to warrant further investigation of this concept by this means.

The outcome of battle is not predetermined. War is not random. It is a sequence of discrete, deliberate, irreversible, interrelated events. It is characterizable but unpredictable. It is complex.

CHAPTER 3

Contributions of Research

In this thesis, I define a tangible method for modeling strategic effects. While this definition may not be the only way to approach to the study of strategic effects, it does provide a beachhead for others interested in this topic and the rest of my research. I also explain the reason that strategic effects are difficult if not impossible to observe in typical attrition driven combat models. I then provide a simplistic exploratory model using autonomous agents as a proof of concept for studying strategic effects.

Areas for Continued Research

Future work may include behavior rule modifications to change the agents behavior be realistic with various actual missions. See if OODA exploitation seems feasible in the context of these real world missions. Also sensitivity analysis could be accomplished to find how much OODA advantage is needed to overcome various tactical or physical advantages in different situations. A hierarchical model could be developed to study the cascade effect. This could be done by linking OODAs together vertically and laterally with different agent classes, then studying the end result of OODA exploitation at each of the various levels. Non-military attack applications could be studied by looking for strategic effects from civilian and political population models using an OODA based model. Drawing a hard link between target sets and

69

OODA disruption would also be a logical continuation of this research. This will better define (strategic) target sets for OODA exploitation.

Internal OODA stability is a key point to Boyd's work and promises even more synergistic effects. This moral aspect is a very important part of OODA exploitation that is not covered by this work. The premise is that internal confusion or dismay can clog an OODA cable with doubt. Confidence can lead to decisions with less thought. These ideas are fundamental in the discussion of the moral aspect of combat. Study the effect of poor performance or results slowing the OODA and, conversely, good performance quickening the OODA. This compounding of OODA advantage could cause an acceleration of collapse in the enemy system and demonstrate a bound for realistic OODA differentials.

Bibliography

Applegate, Jeffery A. *The Combat Simulation of Desert Storm with Applications for Contingency Operations.* Warfare Modeling. Edited by Jerome Bracken, Moshe Kress, and Richard E. Rosenthal. Published by the Military Operations Research Society in cooperation with John Wiley & Sons, 1995, pp. 565-570.

Air Force Doctrine Center (AFDC). AFDC Issue: Modeling and Simulation of Strategic Effects. Unpublished briefing slides available at http://www.hqafdc.maxwell.af.mil/

Bankes, Steve. Exploratory Modeling. *Encyclopedia of OR and MS.* Edited by Saul I Gauss and Carl M. Harris. Norwell, MA Kluwer Academic Publishers, 1996, pp. 203-204.

Battilega, John A. and Grauge, Judith K. The Military Applications of Modeling. Wright-Patterson AFB, OH: Air Force Institute of Technology Press.

Beene, Eric A. Calculating a Value for Dominant Battlespace Awareness. AFIT Thesis. Air Force Institute of Technology, 1998.

Clausewitz, Carl von. On War. Edited and translated by Michael Howard and Peter Paret. Princeton, N.J: Princeton University Press, 1976, pp. 89-91.

Davis, Paul K (1997). Implications of Complex-Adaptive-System (CAS) Research for Defense Analysis. Briefing slides from a presentation given at the MORS mini-symposium and Workshop "Warfare Analysis and Complexity" at The Johns Hopkins University (APL), (Sep 15,1997).

Davis, Paul K. and Blumenthal, Donald (1991). The Base of Sand: A White Paper on the State of Military Combat Modeling. N-3148-OSD/DARPA RAND, 1991.

Deitel, Harvey M. and Deitel, Paul J. Java: How to Program, Second Edition Prentence Hall. Upper Saddle River, N.J, 1998.

Department of the Air Force. Air Force Process Improvement Guide. *Total Quality Tools for Teams and Individuals.* Maxwell AFB, AL: AF Quality Center, undated.

Department of the Air Force. Basic Aerospace Doctrine of the United States Air Force. AFDD 1. Washington: HQ, USAF, 1997.

Department of the Air Force. <u>Air Warfare.</u> AFDD 2-1.(Draft) Washington: HQ, USAF, (downloaded from AFDC home page) June 1998.

Department of the Air Force. <u>Strategic Attack.</u> AFDD 2-1.2. Washington: HQ, USAF, 1998.

Department of the Air Force. <u>(SEAW notes)</u> *Terms of Reference.* Unpublished notes from the *Strategic Effects of Airpower Workshop.* Maxwell AFB, AL, Air University, 1998.

Dewar, J.A, Gillogy, J.J., Juncosa, M.L. Non-Monoticity, Chaos and Combat Models. <u>Military Operations Research</u>. Vol 2, No 2 (1996) pp. 37-49.

Edmonds, David K. In Search of High Ground: The Airpower Trinity and the Decisive Potential of Airpower. <u>Airpower Journal.</u> Vol XII, No. 1, Spring 1998, pp. 4-21.

Fadok, David S. <u>John Boyd and John Warden: Air power's Quest for Strategic Paralysis</u>. Maxwell AFB, AL: AU Press (Feb. 1995).

Hartley, Dean S. III. *A Mathematical Model of Attrition Data.* <u>Warfare Modeling</u>. Edited by Jerome Bracken, Moshe Kress, and Richard E. Rosenthal. Published by the Military Operations Research Society in cooperation with John Wiley & Sons, 1995. pp. 443-465.

Hartley, Dean S. III and Helmbold, Robert L.(1995). *Validating Lanchester's Square Law and Other Attrition Models.* <u>Warfare Modeling</u>. Edited by Jerome Bracken, Moshe Kress, and Richard E. Rosenthal. Published by the Military Operations Research Society in cooperation with John Wiley & Sons, 1995. pp. 467-489.

Hartman, James K. *Lecture Notes in Aggregated Combat Modeling.* Edited by Jack Kloeber Jr. and Jack Jackson. Printed as courseware by AFIT/ENS Wright-Patterson AFB, OH (May 1998).

Ilachinski, Andrew. <u>Complexity and Warfare: Some Possible Approaches.</u> Center for Naval Analyses Briefing, <u>http://www.cna.org/isaac/lwcomp.pdf</u>.

Ilachinski, Andrew (1997). <u>Irreducible Semi-Autonomous Adaptive Combat (ISAAC): An Artificial-Life Approach to Land Warfare.</u> CRM 97-61.10 Center for Naval Analyses. 1997.

James, Glenn E. <u>Chaos Theory: The Essentials for Military Applications</u>. *Newport Paper Number Ten* (Newport, RI) Naval War Collage Press 1996.

Joint Staff, Department of Defense. <u>Joint Doctrine for Command and Control Warfare (C2W)</u>. JP 3-13.1, Washington: Director for Operations, Joint Staff, 1996.

Lockwood, Stephen D. and Siddalingaiah, Madhu. <u>Java API for Dummies</u>. Foster City, CA: IDG Books Worldwide, Inc, 1997.

McDonald, John W. *Exploiting Battlespace Transparency: Operating Inside an Opponents Decision Cycle.* <u>War in the Information Age: New Challenges.</u> Edited by Robert L. Pfaltzgraph, Jr. and Richard H Shultz, Jr. Washington: Brassey's pp. 143-168.

Pape, Robert A. (1998a). <u>The Limits of Precision-Guided Air Power</u>. Reprinted from *Security Studies, Vol 7, No 2* for use in the *Strategic Effects of Airpower Workshop* London: Frank Cass and Co, 1998.

Pape, Robert A. (1998b). <u>The Air Force Strikes Back: A Reply to Barry Watts and John Warden</u>. Reprinted from *Security Studies, Vol 7, No 2* for use in the *Strategic Effects of Airpower Workshop* London: Frank Cass and Co, 1998.

Rinaldi, Steven M <u>Complexity Theory and Airpower: A new Paradigm for Airpower in the 21st Century</u>. http://www.ndu.edu/inss/books/complexity/ch10a.html. http://www.ndu.edu/inss/books/complexity/ch10b.html.

<u>USAF Studies and Analysis Agency (AFSAA). THUNDER Analyst Manual version 6.5. Prepared by System Simulation Solutions, Inc. Alexandria, VA.</u> also at http://info@s3i.com.

van der Linden, Peter. <u>Just Java and Beyond</u>. Sun Microsystems Press, Palo Alto, CA: Prentice Hall 1998.

Warden, John A. III.(1998 a). <u>Success in Modern War: A Response to Robert Pape's</u> *Bombing to Win*. Reprinted from *Security Studies, Vol 7, No 2* for use in the *Strategic Effects of Airpower Workshop* London: Frank Cass and Co, 1998.

Warden, John A. III.(1998 b). <u>The Air Campaign: Planning for Combat.</u> Washington: Brassey's, 1989. (First published by NDU Press, 1988).

Warden, John A. III. (1995). The Enemy as a System. <u>Airpower Journal.</u> Vol IX, No. 1, Spring 1995, pp. 40-55.

Watts, Barry D. <u>Ignoring Reality: Problems of Theory and Evidence in Security Studies</u>. Reprinted from *Security Studies, Vol 7, No 2* for use in the *Strategic Effects of Airpower Workshop* London: Frank Cass and Co, 1998.

Zimm, Alan D. <u>Modeling Maneuver Warfare<i>: Incorporating Human Factors and Decisionmaking in Combat Modeling</i></u><i>.</i> Unpublished briefing slides, November 1997.

Appendix A

Data and Calculations

A comparison was made with Lanchester equations to see how this OODA exploitation model compared to more traditional modeling methods. There are three basic variations to the original Lanchester equations and they will be examined separately.

Lanchester's Square Law

The square law is to be used in situations with direct aimed fire and is probably most appropriate for a comparison to the OODA exploitation model. This law states the change in force size with respect to time (attrition) is based on the number of shooters and their lethality or kill rate. In the equation below 'a' represents the rate at which red kills blue.

$$\frac{d(blue)}{d(time)} = -a * (red)$$

There is an equivalent equation for red attrition with a coefficient 'b' that is the rate blue kills red. When these two equations are integrated and combined, the result is the square law state or difference equation:

$$b(blue_{initial}^2 - blue_{final}^2) = a(red_{initial}^2 - red_{final}^2)$$

Normally the attrition rates and initial force sizes are input to find the final force sizes. The ending ratio of strength is used to determine victory. For our purposes the state equation can be rewritten as:

$$\frac{b}{a} = \frac{red_{initial}^2 - red_{final}^2}{blue_{initial}^2 - blue_{final}^2}$$

This allows the unitless ratio, $\frac{b}{a}$, to reflect the relative advantage in attrition with numbers greater than 1 favoring the blue. This ratio can be calculated with just the attrition data from the model with no other information about the agents behavior nor parameters from the battle

76

needed. This is the method by which the Lanchester laws are often 'fit' to an historical battle. The ratio of 'fit' attrition coefficients can then be compared to the ratio of the predicted attrition coeficients.

The attrition coefficient is composed of the rate of fire (ρ) and the single-shot probability of kill (SSPK):

$$b = \rho * (SSPK)$$

Linear Law

The linear law is typically used to model unaimed or indirect fire such as artillery. The premise is that attrition is based on the number of targets as well as the number of shooters. The basic equation is:

$$\frac{d(blue)}{d(time)} = -a * (red) * (blue)$$

The state law for the linear law is:

$$b(blue_{initial} - blue_{final}) = a(red_{initial} - red_{final})$$

and the ratio of attrition coefficients is:

$$\frac{b}{a} = \frac{red_{initial} - red_{final}}{blue_{initial} - blue_{final}}$$

The attrition coefficients for the linear law are similar in form to those of the square law. The only difference is that the probability of hit for any shot (PH/S) is a function of the percent of the target area covered by the lethal radius of the (artillery) shell instead of direct weapon accuracy.

77

Mixed Law

This law is used when one side (blue) uses aimed fire and the other (red) uses area fire. It is often applied to an ambush situation where the victims are unsure of the location of the attackers. The attrition coefficient ratio for this law is:

$$\frac{b}{a} = \frac{2 * (red_{initial} - red_{final})}{(blue^2_{initial} - blue^2_{final})}$$

Logarithmic Law

The logarithmic law is typically used to describe attrition due to factors other than combat. This law models events like training losses and transportation accidents and is considered inapplicable once combat begins [Hartman, p. 6-22.]. The 'log' law is described here, but not included in the analysis because the OODA exploitation model does not have a mechanism for losses other than enemy fire. The basic equation is a function of the pre-combat attrition rate and the force size:

$$\frac{d(blue)}{d(time)} = -a * (blue)$$

The state equation for this law becomes:

$$b * \ln\left[\frac{blue_{initial}}{blue_{final}}\right] = a * \ln\left[\frac{red_{initial}}{red_{final}}\right]$$

Helmbold's Law

Helmbold modified the Lanchester equations to account for the fact that not all weapons in a large battle can be brought to bear on all targets [Hartman, p. 6-23]. The basic form is:

$$\frac{d(blue)}{d(time)} = -a * \left[\frac{blue}{red}\right]^{(1-w)} * red$$

where "...*w* is a measure of the efficiency with which the large force can be brought to bear on a smaller force" [Hartman, p. 6-23].

Helmbold's final formulation does not use typical attrition coefficients, but instead is based on the following relationship:

$$LHELMRAT = \alpha * LFORRAT + \beta$$

The logarithmic Helmbold ratio (LHELMRAT) is defined as:

$$LHELMRAT = \ln\left[\frac{red_{initial}^2 - red_{final}^2}{blue_{initial}^2 - blue_{final}^2}\right]$$

and can be read directly from the "Square Law" curve in Figure 19. The logarithm of the initial force ratio (LFORRAT) is zero in all but scenario 3. The term α describes which Lanchester law, or mix thereof, provides a basis for the attrition of the battle. The β term is used to describe attrition influences other than direct combat, which could include morale, leadership skills, weather, etc [Hartley, p463]. In scenarios 1, 2, and 4 the LFORRAT is zero meaning that all attrition effects in the LHELMRAT come from the β term. This would imply that a direct attrition advantage from OODA exploitation is not described by Helmbold's formulation of the Lanchester laws but instead by the other 'soft' factors.

Model Output

An example of the output from the OODA exploitation model (from scenario 1, case 1) is shown in Table 4. This data has been imported to a spreadsheet from its original tab delimited format.

Time	Blue Alive	Red Alive	Blue at Goal	Red at Goal	Blue Shots	Red Shots	Blue Hits	Red Hits	Rep
0	20	20	0	0	0	0	0	0	Rep 1
1	20	20	0	0	0	0	0	0	
2	20	20	0	0	0	0	0	0	
3	17	18	0	0	3	3	2	3	
4	16	16	0	0	6	8	4	4	
5	14	15	0	0	7	11	5	6	
6	12	14	0	0	9	15	6	8	
7	9	12	0	0	15	20	8	11	
8	7	10	0	0	20	24	10	13	
9	7	10	4	0	20	24	10	13	
10	7	10	7	0	20	24	10	13	
11	7	10	7	6	20	24	10	13	
12	7	10	7	9	20	24	10	13	
13	7	10	7	9	20	24	10	13	
14	7	10	7	9	20	24	10	13	
15	7	10	7	9	20	24	10	13	
16	7	10	7	9	20	24	10	13	
17	7	10	7	9	20	24	10	13	
18	7	10	7	9	20	24	10	13	
19	7	10	7	9	20	24	10	13	

Table 4. Sample model output

Once all the model output data was imported into a spreadsheet, it was reorganized and used in various calculations to produce the graphs presented in this thesis.

Appendix B

Doctrine as seen through OODA

Decision cycle warfare, or OODA exploitation, is aimed at the fundamental nature of war. In this section each of the Principles of War and Tenets of Air and Space Power will be briefly framed in the context of OODA exploitation. The doctrinal definition of each principle of war and tenet of aerospace power from AFDD 1 is included for comparison and clarification.

The Principles of War

The principles are independent—but tightly fused in application. No one principle should be considered without due consideration of the others. These principles are not all-inclusive but provide a basis for judgment in employing military forces. The art of developing air and space strategies depends upon the airman's ability to view these principles from an aerial perspective and integrate their application with the airman's fundamentals. The principles of war—combined with the additional fundamentals of air and space power discussed later in this chapter—provide the basis for a sound and enduring doctrine for the air and space forces of America's joint force.

If OODA exploitation is indeed relevant to each of these recognized concepts about the nature of war then it may be the common thread that ties the Principles together. Then again, the Principles may just be elements of OODA exploitation.

Unity of Command

A unified command can be represented as a single large OODA link at the top of the web. This single link prevents subordinate links from being lengthened by a lack of

coordination or direction in resources (observe), purpose (orient), training (decide and

act)

AFDD 1 Description:

Unity of command ensures the concentration of effort for every objective under one responsible commander. This principle emphasizes that all efforts should be directed and coordinated toward a common objective. Air and space power's theater wide perspective calls for unity of command to gain the most efficient application. Coordination may be achieved by cooperation; it is, however, best achieved by vesting a single commander with the authority to direct all force employment in pursuit of a common objective. The essence of successful operations is a coordinated and cooperative effort toward a commonly understood objective. In many military operations other than war, the wide-ranging agency and non-governmental operations involved may dilute unity of command; nevertheless, a unity of effort must be preserved in order to ensure common focus and mutually supporting actions. Unity of command is important for all forces, but it is vital in employing air and space forces. Air and space power is the product of multiple capabilities, and centralized command and control (C2) is essential to effectively fuse these capabilities. Airmen best understand the entire range of air and space power. Theater and global ranging capabilities impose theater and global responsibilities, which can be discharged only through the integrating function of centralized command under an airman. That is the essence of unity of command and air and space power.

Objective

A clearly defined objective makes orientation easier and faster

AFDD 1 Description:

The principle of objective is concerned with directing military operations toward a defined and attainable objective that contributes to strategic, operational, or tactical aims. In application, this principle refers to unity of effort. Success in military operations demands that all efforts be directed toward the achievement of common aims. In a broad sense, this principle holds that political and military goals should be complementary and clearly articulated. A clear national military strategy provides focus for defining campaign or theater objectives. At the operational level, campaign or theater objectives determine military priorities. Importantly, particularly in peace support operations, the time and persistence required to attain the objective must be clearly understood by all. The objective is important to all military forces, but it is especially so in air, space, and information warfare due to the versatility of air and space forces. Unlike surface forces, modern air and space forces do not normally need to sequentially achieve tactical objectives first before pursuing operational or strategic objectives. From the outset, air and space forces can pursue tactical, operational, or strategic objectives, in any combination, or all three simultaneously. From an airman's perspective, then, the principle of the objective shapes priorities to allow air and space forces to concentrate on theater or campaign priorities and seeks to avoid the siphoning of force elements to fragmented objectives.

This principle is also one that has significant meaning to air warfare. Offensive is to act rather than react and dictates the time, place, purpose, scope, intensity, and pace of operations. The initiative must be seized as soon as possible. The principle of the offensive holds that offensive action, or initiative, pro-vides the means for joint forces to dictate battle-space operations. Once seized, the initiative should be retained and fully exploited. Air and space forces are best used as an offensive weapon. While defense may be

dictated by the combat situation, success in war is generally attained only while on the offensive. This is particularly true for air and space forces. Even highly successful defensive air campaigns such as the World War II Battle of Britain were based upon selective offensive engagements rather than fragmenting into small patrols everywhere. Air and space forces are inherently offensive at the tactical level, even when employed in operational or strategic defense. Control of air and space is offensive in execution. History has generally shown that a well-planned and executed air attack is extremely difficult to completely stop. The speed and range of attacking air and space forces give them a significant offensive advantage over surface forces and even over the defending air and space forces, since for air attack the defender often requires more forces to defend a given surface area than the attacker requires to strike a set of specific targets. Although all military forces have offensive capabilities, airpower's ability to mass and maneuver and its ability to operate at the tactical, operational, or strategic levels of warfare—or to simultaneously operate at all levels—provide JFCs a resource with global presence to directly and almost immediately seize the initiative. Whether rapidly deploying forces and supplies into a region, conducting combat operations, or providing information superiority over an enemy, air and space forces provide the JFC the means to take the offensive. From the beginning of an operation, air forces can seize the initiative by attacking the enemy directly by flying over enemy lines and flying around massed defenses. Through prompt and decisive offensive actions designed to attain operational and strategic objectives, air and space forces cause the enemy to react rather than act, deny the enemy the offensive, and shape the remainder of the conflict.

Mass

Mass is designed to produce a logistical surprise.

AFDD 1 Description:

The principle of mass calls for concentrating combat power at a decisive time and place. Concentration of military power is a fundamental consideration in all military operations. At the operational level, this principle suggests that superior, concentrated combat power is used to achieve decisive results. Generally, surface forces must mass combat power before launching an attack, whereas airpower is singularly able to launch an attack from widely dispersed locations and mass combat power at the objective. Moreover, from an airman's perspective, mass is not based only on the quantity of forces and materiel committed. Mass is an effect . . . not just overwhelming quantity. The speed, range, and flexibility of air forces—complemented by the accuracy and lethality of precision weapons and advances in command, control, and information gathering technologies— allow them to achieve mass faster than surface forces. Mass is an effect that air and space forces achieve through efficiency of attack. Today's air and space forces have altered the concept of massed forces. In the past, hundreds of airplanes attacked one or two major targets each day. Massed bomber raids revisited targets often, intending their attacks to gradually attain cumulative operational- or strategic-level effects over time. Today, a single precision weapon that is targeted using superior battlespace awareness can often cause the destructive effect that in the past took hundreds of bombs. Emerging information warfare (IW) capabilities also present new opportunities to attack critical targets. IW can, with precision, stealth, and the speed of light, affect a variety of functions and capabilities. The airman's perspective of mass must also include air-power's ability to assist in the massing of lethal and non-lethal surface forces. Airlift provides a significant and critical capability to mass lethal and nonlethal forces on a global scale. The rapid mobility of airlift enabled the airborne assault during Operation JUST CAUSE, which played a pivotal role in massing US forces in Panama. The capability of air forces to act quickly and mass effects, along with the capability to mass other lethal and nonlethal military power, combines the principle of mass with the next principle—maneuver.

Maneuver

Maneuver may be used to generate moral surprise and logistical surprise.

AFDD 1 Description:

The principle of maneuver calls for action to place the enemy in a position of disadvantage through the flexible application of combat power. Air and space power's ability to conduct maneuver is not only a product of its speed and range, but also flows from its flexibility and versatility during the planning and execution of operations. Like the offensive, maneuver forces the enemy to react, allows the exploitation of successful friendly operations, and reduces our vulnerabilities. The ability to integrate a force quickly and to strike directly at an adversary's strategic or operational center of gravity (COG) is a key theme of air and space power's maneuver advantage. Air maneuver allows engagement almost anywhere, from almost any direction, thus forcing the adversary to be on guard everywhere. Additionally, the principle of maneuver is not limited to simple weapons delivery. In 1994, during Operation Vigilant Warrior, airpower's global awareness, global reach, and global presence was clearly demonstrated. Air Force air mobility forces provided combat power to deter Iraqi movements into Kuwait. Whether it involves air mobility or attack aircraft, in small or large numbers, the versatility and responsiveness of airpower allow the simultaneous application of mass and maneuver. Air and space maneuver is uniquely able to achieve mass while moving with unmatched agility. Maneuvering ground forces to achieve military mass has historically taken a tremendous logistics effort and a great deal of time. Airpower, however, is extremely agile in providing military mass. Whether considering the airlift over the Himalayan mountains in 1944, the Berlin airlift of the late 1940s, airlift to Israel in 1973, or more recent operations such as SUPPORT HOPE in Rwanda, PROVIDE HOPE in the former Union of Soviet Socialist Republics (USSR), or PROVIDE PROMISE in Bosnia, airpower plays a critical role in American diplomacy by providing unmatched maneuverability. In applying the principles of mass and maneuver, air planners must also consider a related principle, that of economy of force.

Economy of Force

Economy of force allows the preservation of resources that can be used

offensively in parallel attacks or defensively to prevent logistical surprise.

AFDD 1 Description:

The economy of force principle calls for the rational use of force by selecting the best mix of combat power. To ensure overwhelming combat power is available, minimal combat power should be devoted to secondary objectives. At the operational level, this requires minimum effort be made towards secondary objectives that do not support the larger operational or strategic objectives. This principle requires airmen to exercise a broader operational view and requires clearly articulated objectives and priorities. Economy of force may require airpower in an area to attack, defend, delay, or conduct deception operations, depending on the importance of the area or the priority of the objective or objectives. Although this principle suggests the use of overwhelming force in one sense, it also recommends against "overkill" by guarding against unnecessary force. This is particularly relevant in military operations other than war in which excessive force can destroy the gaining or maintaining of legitimacy and support for an operation. Information

operations conducted by air and space forces enable the Joint Force Commander (JFC) to have dominant battlespace awareness in order to economically allocate forces for maximum effect. While this principle was well developed before the advent of air-power, it responds precisely to the greatest vulnerability of air and space power employment: the misuse or misdirection of air and space power, which can reduce its contribution even more than enemy action. Ill-defined objectives can result in the piecemeal application of air and space forces with the resultant loss of decisive effects.

Security

Offensive security operations help to achieve surprise while defensive security operations are counter-surprise operations. If a combatant knows the enemy's intent, the combatant can prepare and prevent OODA exploitation.

AFDD 1 Description:

The principle of security requires that friendly forces and their operations be protected from enemy action that could provide the enemy with unexpected advantage. The lethal con-sequences of enemy air or space attack make the security of friendly forces a paramount concern. This principle also enhances freedom of action by reducing the vulnerability of friendly forces and creating opportunities to strike the enemy where least expected. Gaining or maintaining control of the air, space, and information mediums pro-vides friendly forces a significant advantage. Airpower is most vulnerable on the ground. Thus, air base defense is an integral part of airpower deployments. Bases not only must withstand aerial and ground attacks, but also must sustain concentrated and pro-longed air activities against the enemy. This must be a particular focus of operations during peace support or crisis situations when forces operate from austere and unimproved locations, in small units, or in crowded urban settings and face threats to security from individuals and groups as well as possible military or paramilitary units. Importantly, security may be obtained by staying beyond the enemy's reach. Air and space forces are uniquely suited to capitalize on this through their global capabilities. Not only can they reach and strike at extended range, but they can also distribute data and analysis as well as command and control across a worldwide span. Security from enemy intrusion conceals friendly capabilities and intentions while allowing our forces the freedom to gather information on the adversary. Critical to security is the understanding that air; and space power is no longer just aircraft, missiles, and satellites but information warfare tools as well. Thus security embraces not only physical security, but also security of the information medium. Information has always been part of air, land, and sea warfare; now, with the proliferation of information technologies, it has become even more central to the outcome of a conflict. The instantaneous global reach of modern information systems is as vital to the Air Force's strategic perspective as any air or space weapon. Today, advanced microchips and communications allow the concept of information superiority to be a strategic component of warfare. Precise strategic attacks delivered against Iraq's central command and control structure during DESERT STORM validated this concept. Additionally, information technology can directly or indirectly affect national or group leadership, population, and infrastructure, bypassing direct military confrontation. Now, whoever has the best ability to gain, defend, exploit, and attack information, and deny the same capabilities to an opponent, has a distinct strategic advantage. By blinding the Iraqi leadership, air and space power allowed ground forces to move undetected to a point where the Iraqi army was least prepared to deal with a massive attack. Space-based and air-breathing ISR systems allowed the coalition command to direct air strikes against Iraqi troops moving south to assist in the battle of Khafji. Because the Iraqis lacked security, their troops were destroyed long before they reached their objective.

Surprise

OODA exploitation is synonymous with surprise.

AFDD 1 Description:

Surprise leverages the security principle by attacking at a time, place, or in a manner for which the enemy is not prepared. The speed and range of air and space forces, coupled with their flexibility and versatility, allow air forces to achieve surprise more readily than surface forces. Air- and space-based ISR systems enhance the ability to achieve surprise by providing information superiority. The choice of time and place of assault rests with the commander of air and space forces because terrain and distance are not inhibiting factors in the air and space environment. Historically, armies and navies massed large numbers of troops or ships to create significant impact on the enemy. Today, the technology impact of precision guided munitions enables a relatively small number of aircraft to achieve national- or theater-level objectives. When combined with stealth and information technologies, air and space forces today can provide shock and surprise without unnecessarily exposing massed friendly forces. Surprise is one of air and space power's strongest advantages. On 11 November 1940, Admiral Andrew Cunningham delivered a crushing air attack from the HMS Illustrious on the Italian naval base of Taranto. While the British lost 2 of 21 attacking aircraft, they left 3 battleships in sinking condition, badly damaged 2 cruisers, and sank 2 fleet auxiliaries. This attack may have inspired the successful Japanese attack on Pearl Harbor over one year later. The 1986 surprise raid against Libya persuaded Muammar Qadhafi to change his policy of open support of worldwide terrorism. In 1990, Saddam Hussein believed he had nothing to fear from the United States Air Force. What he failed to consider was the global presence of air and space forces. Airlift and air refueling provided global reach, while combat aircraft provided strategic power. When the first explosions rocked downtown Baghdad, the ability of modern airpower to strike without warning, and with great accuracy, proved the Iraqi dictator wrong. Saddam Hussein grossly misjudged the power of an integrated air attack. He saw firsthand the principle of surprise in practice. Air and space forces can enhance and empower surface forces to achieve surprise. The rapid global reach of airpower also allows surface forces to reach foreign destinations quickly, thus seizing the initiative through surprise. Air and space power allowed the coalition to achieve an overwhelming surprise and also ensured the coalition forces themselves would not become victims of surprise.

Simplicity

Complex plans and operations have more parameters and variables to track and have more contingencies to filter in the orientation phase. These extra parameters clog and slow the OODA process. A simple plan is more likely to yield a shorter OODA loop.

AFDD 1 Description:

The final principle, simplicity, calls for avoiding unnecessary complexity in organizing, preparing, planning, and conducting military operations. This ensures that guidance, plans, and orders are as simple and direct as the objective will allow. Simple guidance allows subordinate commanders the

freedom to creatively operate within their battlespace. Military operations, especially joint operations, are often complex. Common equipment, a common understanding of Service and joint doctrine, and familiarity with procedures through joint exercises and training can help overcome complexity, but straightforward plans and unambiguous organizational and command relationships are central to reducing it. The premise that airmen work for airmen and that the senior airman (the commander of Air Force forces) works for the JFC is central to simplicity.

Tenets of Air and Space Power

Air and space power is intrinsically different from either land or sea power, and its employment must be guided by axioms different than those of surface forces. Both the air and space mediums involve operations in three dimensions. While airpower is primarily affected by aerodynamics, space power is guided by the principles of orbital mechanics and is not limited by the vertical extent of the atmosphere. Both share the advantages of three-dimensional maneuver such as the overlook of enemy positions and the ability to maneuver beyond enemy surface forces, and both are inextricably linked by warfighting principles. The fundamental guiding truths of air and space power employment are known as tenets, which in addition to the principles of war, should be under-stood by every airman. They reflect not only the unique historical and doctrinal evolution of airpower but the specific current under-standing of the nature of air, space, and, increasingly, information power. The tenets of airpower complement the principles of war. While the principles of war provide general guidance on the application of air and space forces, the tenets provide more specific considerations for air and space forces. They reflect the specific lessons of air and space operations over the history of powered flight and highlight the way integrated air and space forces differ from surface forces in providing global strategic air and space power. As with the principles of war, these tenets require informed judgment in application. They require a skillful blending to tailor them to the ever-changing operational environment. The seemingly conflicting demands of the principles and tenets, especially the demands of mass, economy of force, concentration, and priority, require an airman's expert understanding in order to strike the required balance. No two operations are alike; therefore, in the last analysis, the commander must accept the fact that war is incredibly complicated. The application of the principles and tenets must be left to commanders and their professional knowledge and experience as they strive to craft the most effective employment of air and space power for a given situation [AFDD 1].

Centralized Control and Decentralized Execution

This tenet is the blueprint for building an effective OODA web. The big links at top influence and coordinate the links at the bottom, but are isolated from the details and risks of battle.

AFDD 1 Description:

Centralized control and decentralized execution of air and space forces are critical to force effectiveness. Air and space power must be controlled by an airman who maintains a broad strategic and/or theater perspective in prioritizing the use of limited air and space assets to attain the objectives of all US forces in any contingency across the range of operations. During the initial engagements of World War II and

through the entire Vietnam conflict, command of US airpower was fragmented and controlled by competing commanders. The results taught airpower leaders that centralized control was the best way to effectively employ airpower. The outcome of the Gulf War stands in stark contrast to that of Vietnam.

The lesson is clear: attempts to fragment the control and planning of air and space power will ultimately cost blood and treasure by diverting effort and impact. Centralized control allows commanders to focus on those priorities that lead to victory. Through centralized control, commanders give coherence, guidance, and organization to the air and space effort and maintain the ability to focus the tremendous impact of air and space power wherever needed across the theater of operations. Just as central to the proper application of airpower is the concept of decentralized execution. Delegation of execution authority to responsible and capable lower-level commanders is essential to achieve effective span of control and to foster initiative, situational responsiveness, and tactical flexibility. Centralized control and decentralized execution were illustrated by the 2,000–3,000 sorties a day in the Gulf War. The single command intent of the JFC was centrally planned and then distributed and executed across an entire theater battlespace by over 500 flight leads; mission, crew, and flight commanders; and support teams in a continuous application against an entire range of separately engaging, thinking, reacting enemies.

Flexibility and Versatility

The duration of time in which an OODA is vulnerable is limited. These tenets are essential to the ability to exploit an OODA once it has been target while it is still vulnerable.

AFDD 1 Description:

Air and space power is flexible and versatile. Although often used inter changeably, flexibility and versatility are distinct in meaning. Flexibility allows air and space forces to exploit mass and maneuver simultaneously to a far greater extent than surface forces. At the operational level, flexibility allows air operations to shift from one campaign objective to another, quickly and decisively. The A-10, usually considered a close air support aircraft, took on many interdiction missions during DESERT STORM, while one wing of F-111s, optimized as long-range, deep-interdiction aircraft, destroyed hundreds of tanks and armored fighting vehicles with precision weapons. During the Vietnam conflict, B-52 heavy bombers provided highly effective support as close as 1,000 yards from the Marines defending Khe Sanh. Versatility in air and space power stems from the fact that it can be employed equally effectively at the strategic, operational, and tactical levels of warfare. Unlike other forms of military power, air and space forces have the versatility to be employed globally with unmatched responsiveness in support of strategic, operational, or tactical objectives and can simultaneously achieve objectives at all three levels of war—in parallel operations. Air and space attacks can be simultaneous and continuous against a broad spectrum of targets and with sufficient force to overwhelm the enemy. The versatility of air and space power, properly executed in parallel attacks, can attain parallel effects which pre-sent the enemy with multiple crises occurring so quickly that there is no way to respond to all or, in some cases, any of them. Such a strategy places maximum stress on both enemy defenses and the enemy society as a whole. Parallel operations can be conducted at the strategic, operational, and tactical levels of war and either symmetrically against the adversary's air and space forces or asymmetrically against the enemy's surface forces—often simultaneously. Parallel force-application theory is not new, but its recent emphasis is essentially a product of the efficiency of high technology precision weapons, command and control techniques, ISR systems, and the resultant synergistic application. For parallel strategic operations, the swift, massive, and precise application of air, space, and information power against several critical COGs may be sufficient to produce

shock and may result in organizational paralysis that provides the leverage to dominate surface as well as air and space operations.

Synergistic Effects

The disruption of the OODA associated with the physical destruction of a target can negate enemy plans and effect future decisions. This influence on future events is the synergistic effect.

AFDD 1 Description:

Air and space forces produce synergistic effects. The proper application of a coordinated force can produce effects that exceed the individual contributions of the individual forces employed separately. The destruction of a large number of targets through attrition warfare is rarely the key objective in modern war. Instead, it is the precise, coordinated application of the various elements of air, space, and surface forces which brings disproportionate pressure on enemy leaders to comply with our national will. Our overwhelming ability to observe our adversaries allows us to counter their movements with unprecedented speed and agility. Air and space power is unique in its ability to accomplish this and thus dictate the tempo and direction of an entire warfighting effort from MOOTW operations through major conflict.

Persistence

Persistence in attack ensures that an impaired OODA cannot recover and may lead to collapse.

AFDD 1 Description:

Air and space systems are uniquely suited to persistent operations. Persistence suggests continued efforts. Unlike surface power, air and space power's inherent exceptional speed and range allows its forces to visit and revisit wide ranges of targets nearly at will. Air and space power does not have to occupy terrain or remain constantly in proximity to areas of operation to bring force upon them. Space forces in particular hold the ultimate high ground, and as space systems advance and proliferate, they offer the potential for "permanent presence" over any part of the globe. The goal of persistent operations may be to maintain a continuous flow of materiel to peacetime distressed areas; surveil adversaries constantly to ensure they cannot conduct actions against our wishes; assure targets are kept continually out of commission; or ensure that resources and facilities are denied to an enemy or provided to an ally during a defined time. The end result would be to deny the opponent an opportunity to seize the initiative and allow friendly forces to directly accomplish their assigned tasks. Persistence is a critical element in ensuring the prolonged effect of air, space, and information operations. It is the intention of most modern air and space operations to quickly attain objectives through swift, parallel, and decisive blows to the adversary's operational and strategic

COGs. However, on some occasions, factors such as enemy resilience, effective defenses, or environmental concerns prevent this from happening. Realizing that for many situations, air and space operations provide the most efficient and effective means to attain national objectives, commanders must persist in air and space operations and resist pressures to divert resources to other efforts unless such diversions are vital to attaining theater goals or to survival of an element of the joint force. Given sufficient time, even the most devastating strategic effects can be circumvented by resourceful enemies; the goal is to keep pressure on and not allow the enemy that time.

Concentration

Concentration represents air power's ability to apply adequate stress on an OODA to stretch or break the link.

AFDD 1 Description:

Air and space operations must achieve concentration of purpose. The very versatility of air and space power makes it attractive in almost every combat task. Airmen must guard against the inadvertent dispersion of air and space power effects resulting from high demand. One of the most constant and important trends throughout military history has been the effort to concentrate overwhelming power at the decisive time and place. The principles of mass and economy of force deal directly with concentrating overwhelming power at the right time and the right place (or places). With forces as flexible and versatile as air and space power, the demand for them will often exceed the available forces, and may result in the fragmentation of the integrated air and space effort in attempts to fulfill the many demands of the operation. Depending on the operational situation, such a course of action may court the triple risk of (1) failing to achieve operational-level objectives, (2) delaying or diminishing the attainment of decisive effects, and (3) increasing the attrition rate of air forces consequently, risking defeat in detail. Importantly, concentration of purpose must not confuse "mass" with "purpose." A vital concept of air and space forces is its inherent ability to accomplish simultaneous strategic, operational, and tactical effects—to conduct parallel operations—and attain over-whelming effect (concentration of purpose) through carefully dispersed applications.

Priority

Since all entities in an enemy system have an OODA loop, those loops that will have the greatest impact on the enemy web must be targeted first. The rest of the system may become ineffective after collapsing these COG loops and the enemy may submit to our political will.

AFDD 1 Description:

Air and space operations must be prioritized. Given their flexibility and versatility, demands for air and space forces will likely swamp air commanders in future conflicts unless appropriate priorities are established. Only theater-level commanders of land and naval components can effectively prioritize their individual air component support requirements to the joint force commander, and only then can effective priorities for the use of air and space forces flow from an informed dialogue between the JFC and the air component commander. The air commander should assess the possible uses of air and space forces and their strengths and capabilities to support (1) the overall joint campaign, (2) air operations, and (3) the battle at hand. Limited resources require that air and space forces be applied where they can make the greatest contribution to the most critical current JFC requirements. The inherent strategic application of air and space forces must be balanced against their ability to conduct operations at all levels of war, often simultaneously. The principles of mass, offensive, and economy of force, the tenet of concentration, and the airman's strategic perspective all apply to prioritizing air and space force operations.

Balance

Critical OODA loops can be found at all levels in the OODA web. Fixating on a specific type of mission or target set will not usually cover the most critical or vulnerable enemy OODAs while maintaining our OODAs in tact.

AFDD 1 Description:

Air and space operations must be balanced. Balance is an essential guideline for air commanders. Much of the skill of an air commander is reflected in the dynamic and correct balancing of the principles of war and the tenets of airpower to bring air and space power together to produce a synergistic effect. An air commander should balance combat opportunity, necessity, effectiveness, efficiency, and the impact on accomplishing assigned objectives against the associated risk to friendly air and space forces. An air commander is uniquely—and best—suited to determine the proper theater wide balance between offensive and defensive operations, and among strategic, operational, and tactical applications. Technologically sophisticated air and space assets will be available only in finite numbers; thus, balance is a crucial determinant for an air commander.

Appendix C

The Model

Technical Administrative Information

This program was written using Semantics' Visual Café Professional Development version 2.5. JDK 1.1.6 was also used. All of the data runs were done in appletviewer through Visual Café on a 200MHz PC. The program will run in a browser (IE 4.0) if the Executive.printout switch is set to false.

Model Operation

The model is not difficult to set up or run, but it is not convenient either. The parameters for the model must be entered into the code and recompiled before running the simulation. The good news is that almost all the parameters that need to be changed on a run by run basis are in either the Executive class (simulation parameters) or TAgentAttributes class (for agent parameters). These parameters are well documented and easy to find in the code, usually at the beginning. The Executive class is the applet. It controls the rest of the simulation by controlling the graphics, holding input variables, printing output, and instantiating a series (loop) of manager classes. A different Manager class is established to control each case and performs a loop of iterations of that same case. Its purpose is to load and initialize the agents, collect statistics, and terminate the run. 30 iterations of each case were run for statistical purposes. The bulk of the program is the TAgent class. This class holds the OODA loop and personalities of each individual agents. The TAgents' parameters are set with the class TAgentAttributes. The observation class is used by the TAgents to hold the information needed about the other agents they can 'see'. The BattleGround class holds the methods to draw the graphics.

Lessons on Multi-threading

One aspect of multi-threading is that the threads do not actually run simultaneously. Each thread is allotted a timeshare on the CPU, but only one thread at a time is actually active. The

TAgent class threads all have the same priority so the computer picks which of the threads to execute first. Other threads on the computer, such as the operating system, screen savers, etc., also compete for CPU time. This means the simulation will run differently on machines with different CPU speeds and different software.

Another important point is that the agents are not entirely independent. To keep the multithreading from overtasking (locking up) the simulation, the agents are put into a sequence once per OODA to help the simulation determine which thread has precedence (for the CPU time). This is accomplished by making the decide method a synchronized method. This means that only one thread (TAgent) at a time can use this method. The effect is that the agents decide (and delay), act, observe, orient, and then wait in line to get access to the decide method again. This does dampen the system slightly, but does not force an order or sequence on the agents as the queue for the decide method is first-come-first-serve and based on the speed of the agents' previous OODA.

<u>Planned modifications</u>

If I had more time these are the changes I would make to the model. I don't believe any are that difficult, just time consuming.

1. Build a GUI to initialize the battle and modify the parameters on the fly

2. Display battle results someplace in applet so they can be seen in a browser.

3. Put in a weapon reload time (in TAgent.act or TAgent.shoot) to prevent simultaneous shooting at multiple targets.

4. Debug the last two straggling agents that just can't get into the goal. Check to see if this is because of a condition with the advance switch (if < unitsize and no enemies).

Source code for Executive class

```
// Major Tom Tighe's Air Force Institute of Technology Thesis Project
// last revised on 29 Jan 1998

import java.applet.*;
import java.awt.*;
import java.io.*;
import java.lang.*;
import java.util.*;

/** The Executive class runs the simulation.  It is responsible for overseeing the graphics,
stepping through the iterative runs, taking the user input variables dealing with the scope
and nature of the battle, and writing output to a file (in appletviewer).  The vitual machine
creates an instance of this class and calls it's init( ) method using a thread created by the
virtual machine.  This thread starts an Executive thread called "boss" that oversees the
remainder of the simulation.  The program's multithreading starts in the Manager class which
is also controlled by the boss thread.
*/

public class Executive extends Applet implements Runnable
{
/**  The first thread of execution that the program creates.  It will oversee
     the remainder of the simulation*/
     public static Thread boss;

     /** The playing field */
     static BattleGround field;

     /** The top of the screen */
     Panel north;

     /** The center of the screen */
     Panel center;

     /** The title caption */
     Label caption;

     /**  An instance of the class Manager that will control one repetition*/
     static Manager VP;

     /** a vector of dataPoints */
     public Vector data;

     /** An iteration counter for the repetitions to be run in a case of the simulation*/
     static int rep = 0;

     /** This will generate common random numbers for weapons effectiveness (kills) */
     static Random randomGenerator;

     /** An iteration counter for the number of cases to be run in the simulation */
     static int caseCount = 0;

     /** This switch will prevent a "null pointer exception in battlefield.paint()during
initialization*/
     static boolean drawTAgents = false;

 // These are the variables needed to initialize the forces ==================================
 // ==========================================================================================
     /** This switch will write output to a file in the same directory that the
     executive class is in.  This does not work if the simulation is run from a
     web browser.  Use appletviewer to run the simulation if you want output*/
     static boolean printout = true;

     /** This switch will use common random numbers for each case*/
     static boolean randomSeed = true;

     /** Time limit of simulation in seconds */
     static int timelimit = 20;

     /** Number of repetitions per case*/
     static int repetitions = 30;

     /** The number of cases to be performed in the simulation*/
     static int cases = 17;

     /** This value will be used to seed the common random number generator if "randomSeed =
true"*/
     static long seedValue = 252;
```

96

```
// The following variables are required to initialize the simulation ++++++++++++++++++++++++

    /** This is the initial force size of the blue side */
    static int numberOfBlue = 20;

    /** This is the initial force size of the red side */
    static int numberOfRed = 40;

    /** This sets the number of TAgents per row of the initial formation of blue TAgents */
    static int blueSquadSize = 5;

    /** This sets the number of TAgents per row of the initial formation of red TAgents */
    static int redSquadSize = 5;

    /**  The predefined TAgentAttribute used to initialize the blue force*/
    static int bluePersonality = 0;

    /**  The predefined TAgentAttribute used to initialize the red force.  It is adjusted
    iteritively in the run method of this class to step through multiple cases each with a
    different red personality defined in TAgentAttributes*/
    static int redPersonality = 1;

// end of user required inputs +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/**  The X cartesian coordinate of the door to the blue base */
    static int blueBaseX = 12;

    /**  The Y cartesian coordinate of the door to the blue base */
    static int blueBaseY = 12;

    /**  The X cartesian coordinate of the door to the red base */
    static int redBaseX = 388;

    /**  The Y cartesian coordinate of the door to the red base */
    static int redBaseY = 288;

    /** This controls the dimensions of the graphic (square)for the TAgents' bases*/
    static int baseSize = 10;

    /** This is the instance of the class TAgent Attributes used to initialize the blue force*/
    static TAgentAttributes blueAttributes;

    /** This is the instance of the class TAgent Attributes used to initialize the red force*/
    static TAgentAttributes redAttributes;

    /**  This string is the name of the output file*/
    static String outFileName;

    /** dataOut is the output stream that writes the "dataPoints" to a file*/
    static FileOutputStream dataOut = null;

// METHODS $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

    /** The init( ) method sets up the applet graphical output.  The init( ) method executes on
        the thread designated by the JAVA vitual machine, not one you've created.  After
        executing the init( ) method, that thread returns and then invokes the start( ) method.
    */
    public void init( ) {
        setLayout(new BorderLayout());
        setBackground(Color.white);

        north = new Panel();
        north.setLayout(new FlowLayout());
        caption = new Label("An Experiment in OODA Advantage");
        north.add(caption);

        center = new Panel();
        center.setLayout(new FlowLayout());
        field = new BattleGround();
        field.init(this);
        center.add(field);

        add(north, BorderLayout.NORTH);
        add(center, BorderLayout.CENTER);

    } // end init() ================================================================
```

```
    /** This method starts a thread of execution called "boss" to control the simulation.
        Like init( ), this method is executed by the thread designated by the virtual
        machine.  Once, this method is done executing, the thread created by the virtual
        machine is done. Instead of using that thread, you'll create a new thread called boss.
        "boss" executes the Executive's run( ) method  which will, in turn, create
        intances of other classes that will execute their run( ) method (Manager, TAgent)
        on their own threads.  The "boss" thread is done as soon as it's run( ) method
        completes executing.
    */
    public void start(){

        boss = new Thread(this);
        boss.setPriority(Thread.MAX_PRIORITY);
        boss.start();
        System.out.println( "\nAll done with the thread the virtual machine created to
instantiate Executive and execute its init( ) and start( ) methods.\n" );

        } // End of start() ==========================================================

    /** This method is what the "boss" thread created in the Executive's start( ) method does.
      You passed to the Thread's constructor, the instance of Executive originally created
      by the virtual machine.  Consequently, it's that same instance of Executive
      executing the run( ) method but doing it on a new/different thread of execution than
      was used to execute the init( ) and start( ) methods.
    */

    public void run( ){

        randomGenerator = new Random(seedValue);

//  This loop runs the specified number of cases
    for (int c = 0; c < cases; c++){
        rep = 0;
        caseCount = c;
        blueAttributes = new TAgentAttributes(bluePersonality);
        redAttributes = new TAgentAttributes(redPersonality);
        randomGenerator.setSeed(seedValue);



//  This will initialize the output file for each case
        if (printout == true){
            outFileName = new String("TigheB" + bluePersonality + "R"+ redPersonality +".out");
            try { dataOut = new FileOutputStream (outFileName);}
            catch (IOException e){System.out.println(" Error opening " + outFileName+"\n");}
            String header = new
String("Time\tBLive\tRLive\tBGoal\tRGoal\tBShot\tRShot\tBhit\tRHit\tRep");
            PrintStream ExecPS = new PrintStream (dataOut);
            ExecPS.println (header);
        }

 // This will run the specified number of repetitions in each case
        System.out.println( "\nStarting Case" + (c+1));
        for (int r = 0; r < repetitions; r++) {
            rep ++;
            VP = new Manager();
            VP.setPriority(Thread.MAX_PRIORITY);
            drawTAgents = true;
            VP.run();
            while (VP.finished == false){}
            VP.stop();
        }
        redPersonality+=1;   // updates the red's TAgentAttribute to be used in the next case
    }

        Label endCaption = new Label("All done");//adds a label when the simulation is complete
        north.add(endCaption);
        field.repaint();
    } // End of run() ==========================================================

/**  This method returns a random number used to determine the result of a TAgents shot*/
static double getRandom(){
    if (randomSeed == true){return randomGenerator.nextDouble();} // Common random numbers
    else {return Math.random();}   // Just random numbers
} // End of getRandom() ==========================================================

}// End of Executive Class   mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
```

Source code for Manager class

```
// Major Tom Tighe's Air Force Institute of Technology Thesis Project
//
// last revised on 29 Jan 1999
import java.io.*;
import java.awt.*;
import java.util.*;
import java.lang.*;


/** The Manager class controls one repetition of the simulation.  Manager reads the values
it needs to initialize the repetition directly from the Executive variables of the same name.
The Executive class creates an instance of this class for each repetition.
*/

public class Manager extends Thread {

//VARIABLES   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

/**  A vector of all the TAgents involved in this repetition*/
    static Vector v;

/** A vector of dataPoints.  This is to be used to collect and compute statistics, but this
    has not yet been implemented*/
    public Vector data;

/** This is a switch that will shut the simulation down when the time limit expires */
    public boolean finished = false;

//   These are statistic counters
    static int totalBlueShots = 0;
    static int totalBlueHits = 0;
    static int totalRedShots = 0;
    static int totalRedHits = 0;


 // These are the variables needed to initialize the forces and are read directly from Executive
    int timelimit = Executive.timelimit;
    int repetitions = Executive.repetitions;

    int numberOfBlue = Executive.numberOfBlue;
    int numberOfRed = Executive.numberOfRed;
    int blueSquadSize = Executive.blueSquadSize;
    int redSquadSize = Executive.redSquadSize;
    int bluePersonality = Executive.bluePersonality;
    int redPersonality = Executive.redPersonality;

    int blueBaseX = Executive.blueBaseX;
    int blueBaseY = Executive.blueBaseY;
    int redBaseX = Executive.redBaseX;
    int redBaseY = Executive.redBaseY;
    int baseSize = Executive.baseSize;

    public TAgentAttributes blueAttributes = Executive.blueAttributes;
    public TAgentAttributes redAttributes = Executive.redAttributes;

// CONSTRUCTOR $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

/** The only constructor is this default constructor */
    public Manager(){}


// METHODS $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

/**  This method initializes, runs, times, and shuts down the simulation*/
    public void run( ){
    System.out.println(" Manager is starting case "+Executive.caseCount+", rep "+Executive.rep);

        int totalCombatants = (numberOfBlue + numberOfRed);
        v = new Vector(totalCombatants); // This vector will hold all the TAgents in the
simulation

        // This is where the Agent threads are created and stuffed into the vector "v"
        for (int i = 0; i < totalCombatants; ++i) {
            TAgent temp;

        // This instantiates each TAgent and alternates the type in position in "v" (blue/red)
            if (i/2 < numberOfRed && i/2 < numberOfBlue){
```

99

```
                    if ((i%2) == 0){temp = new TAgent(i, 0, blueSquadSize, numberOfRed,
blueAttributes);}
                    else {temp = new TAgent(i, 1, redSquadSize, numberOfBlue, redAttributes);}
              }
       //  This handles the type and loading of 'v' if the force size is asymmetric
          else if (i/2 >= numberOfRed) {
                temp = new TAgent(i, 0, blueSquadSize, numberOfRed, blueAttributes);
              }
          else if (i/2 >= numberOfBlue) {
                temp = new TAgent(i, 1, redSquadSize, numberOfBlue, redAttributes);
                }
       //  This will catch any leftovers to complete the logic requirements
          else {temp = new TAgent(i);}

       // Set the TAgent threads to the lowest priority so Manager will be able to control them
            temp.setPriority( 1 );
            v.addElement(temp);
            Executive.field.repaint();
       }

       // Reset the statistic counters
            totalBlueShots = 0;
            totalBlueHits = 0;
            totalRedShots = 0;
            totalRedHits = 0;

       //  This will begin the TAgent threads running now that they have been initialized
        for (int k =0; k < v.size(); k++) {
            ((TAgent)v.elementAt(k)).start( );
        }

        int j = 0;    // This iteration variable is a clock to limit run time
        data = new Vector(timelimit);   // This is for stats collection and has not been
implemented yet


        String header = new String("Time \t BA \t RA \t BG \t RG \t BS \t RS \t BH \t RH for REP
"+Executive.rep +"\n");
      //    System.out.println(header);    //Prints header to dos prompt


        // Instantiate the printstream for data output
        PrintStream ps = new PrintStream (Executive.dataOut);

        /** This has the Manager's thread wake up every second to time the simulation,
        collect data and repaint */
        while( j < timelimit ){
          Executive.field.repaint( );
          if (v.size() > 0){
            dataPoint tempData = new dataPoint();
            tempData.zero();
                for (int i = 0; i < v.size(); ++i) {
                    tempData.addData((TAgent)v.elementAt(i));
                }
            data.insertElementAt(tempData, j); // This is for the future statistics computations

            String dataLine = new String (j + "\t"+((dataPoint)(data.elementAt(j))).blueAlive
+"\t"+ ((dataPoint)(data.elementAt(j))).redAlive +
"\t"+((dataPoint)(data.elementAt(j))).blueAtGoal +"\t"+
((dataPoint)(data.elementAt(j))).redAtGoal +"\t"+ totalBlueShots +"\t"+ totalRedShots +"\t"+
totalBlueHits +"\t"+ totalRedHits );
            if (j==0){ dataLine = (dataLine + "\tRep "+Executive.rep);} // This adds a marker to
the first rep of a run

      //        System.out.println(dataLine);  // prints data (dataLine) to the dos prompt
            if (Executive.printout == true){ps.println (dataLine);}


          }
            try {sleep(1000);}  // This sleeps the Manager's thread for one second, giving the
clock effect
            catch (InterruptedException e){}
            j++;
        } // end of "J-loop"

        //  This kills each TAgent at the end of the timeLimit
        for ( int k=0;  k < v.size( ); ++k ) {
            ((TAgent)(v.elementAt(k))).notDone = false;
        }

        //  This will clear the screen for 2 seconds, then terminate the Manager's thread
```

```
        System.out.println("END SCORE "+ ((dataPoint)(data.elementAt(j-1))).blueAlive +" to "+
((dataPoint)(data.elementAt(j-1))).redAlive );
        Executive.field.repaint();
        try{sleep(2000);}
            catch(InterruptedException e){}
        finished = true;
} // End of run()=========================================================================

}// End of Manager Class mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
```

Source code for BattleGround class

```
// Major Tom Tighe's Air Force Institute of Technology Thesis Project
//
// last revised on 29 Jan 1999

import java.awt.*;


/** The battleground class is responsible for presenting all the graphics.  It is initialized to
the center panel in Executive's "boarder layout"
*/

public class BattleGround extends Canvas{


// VARIABLES $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

// METHODS $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

/**  This will initialize the battleGround for use in the Executive */
    public void init(Executive app)
    {
        resize(400,300);
        setBackground(Color.lightGray);
        setForeground(Color.green);
    }// End of Init()=============================================================

/** This will draw all the TAgents by invoking the TAgent's draw() method.  It also draws the
    red and blue bases. */
    public void paint(Graphics g){

        int baseSize = Executive.baseSize;

        // draw each TAgent
        if (Executive.drawTAgents == true){
            for (int i =0;i < Executive.VP.v.size(); i++){
                TAgent pilot;
                pilot = (TAgent)Executive.VP.v.elementAt(i);
                if (pilot.notDone){
                    pilot.Draw(g);
                }
            }
        }
        // draw the bases
        g.setColor(Color.blue);
        g.fillRect(Executive.blueBaseX, Executive.blueBaseY, baseSize,baseSize);
        g.setColor(Color.red);
        g.fillRect(Executive.redBaseX-baseSize, Executive.redBaseY-baseSize,baseSize,baseSize);

    } // End of paint()===========================================================


/** battleGroundInit( ) is a placeholder for control initialization.
You should call this function from a constructor or initialization function if needed */

        void battleGroundInit() {
            System.out.println("Battleground Test1");
        } // End of battleGroundInit ===========================================


}// End of BattleGround mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
```

Source code for TAgent class

```java
// Major Tom Tighe's Air Force Institute of Technology Thesis Project
//
// last revised on 01 Feb 1999

import java.awt.*;
import java.lang.*;
import java.util.*;

/**
@author Thomas R. Tighe
    The TAgent or "Thread Agent" class is the autonomous agent.  Each TAgent is a
separate thread or program that runs independently of the other TAgents.  This
class contains all the behavior rules and most of the variables needed for the
agents to work.  The main function of the TAgent is to perform sequential OODA
loops in a combat decision environment.  The key to exploring OODA loop
exploitation lies in the ability to pause or "sleep" each agent for a specified
and changeable time period.  This quantifies the time the agent spends on each
OODA loop and allows the results of direct competition between agents with
different length decision cycles to be studied.
    These agents are called and monitored by the Manager class which is called
and monitored by the Executive class.
*/
public class TAgent extends Thread  {

  // CLASS VARIABLES AND PARAMETERS $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  // Trait variables specific to this agent ----------------------------------

  /** This thread's position in the vector of TAgents in Executive */
     public int id;

  /** This is the side or team of this Agent (Blue or Red)*/
     public int type;

  /** This sets the fill color of this agent that marks side and alive or dead*/
     Color inColor;

  /** This sets the border color of this agent to mark if the agent has made the goal*/
     Color outColor;

  //Cartesian coordinate variables ---------------------------------------------

  /** The old X position*/
     int xold;

  /** The old Y position*/
     int yold;

  /** The current X position*/
     int xnew;

  /** The current Y position */
     int ynew;

  /** The calculated change in the X direction to move on this turn */
     float deltaX;

  /** The calculated change in the Y direction to move on this turn */
     float deltaY;

  /** The calculated adjustment in the X direction to keep from stepping on other agents*/
     float delX;

  /** The calculated adjustment in the Y direction to keep from stepping on other agents*/
     float delY;

  /** The X coordinate of the enemy base*/
     int goalX;

  /** The Y coordinate of the enemy base*/
     int goalY;

  /** The X coordinate of the agent's base*/
     int baseX;

  /** The Y coordinate of the agent's base*/
     int baseY;
```

```java
/** The X coordinate of the target agent*/
    int targetX;

/** The Y coordinate of the target agent*/
    int targetY;

/** A temp variable for computing threat center of mass*/
    int myThreatX;

/** A temp variable for computing threat center of mass*/
    int myThreatY;

/** The X coordinate of the threat's center of mass*/
    int myThreatCgX;

/** The Y coordinate of the threat's center of mass*/
    int myThreatCgY;

/** A temporary variable for computing friendly center of mass*/
    int myFriendsX;

/** A temporary variable for computing friendly center of mass*/
    int myFriendsY;

/** The X coordinate of observed friendlies' center of mass*/
    int myFriendsCgX;

/** The Y coordinate of observed friendlies' center of mass*/
    int myFriendsCgY;

// Individual personality trait variables ---------------------------------------

/** The length of delay in the decision cycle*/
    public int sleepTime;

/** How many hits until this agent dies*/
    public int health = 3;

/** The single shot probability of kill at best range*/
    public double pk = .7;

/** The max number of enemy of which the agent can keep aware*/
    public int SA = 3;

/** The largest step an agent can take in one turn*/
    public int maxSpeed = 20;

/** The personality weight to advance to goal*/
    public int IAdvance = 2;

/** The personality weight to attack if favorable force ratio */
    public int IAttack = 6;

/** The personality weight to regroup if unit gets too small*/
    public int IRegroup = 4;

/** The personality weight to defend base if called*/
    public int IDefend = 5;

/** The personality weight to avoid enemy if not attacking*/
    public int IAvoidEnemy = 1;

/** The personality weight to maintain spacing if unit size is OK*/
    public int IAvoidFriendlies = 1;

/** The personality weight to avoid base if not under attack*/
    public int IAvoidBase = 1;

// Doctrinal personality trait variables ---------------------------------------

/** This is this side's visual or sensor range*/
    public int sensorRange = 50;

/** This is this side's max weapons range*/
    public int weaponsRange = 40;

/** This is this side's doctrinal best firing range range*/
    public int bestFiringRange = 20;
```

103

```java
/** This is this side's communications range*/
    public int commRange = 60;

/** This is a range at which other agents become noticable or threatening*/
    public int thresholdRange = 30;

/** The force ratio advantage an agent needs before attacking*/
    public int ratio2Attack = 1;

/** The force ratio needed to stop attacking and advance to goal*/
    public int ratio2Overrun = 3;

/** The minimum unit size an agent needs to advance toward goal*/
    public int min2Advance = 1;

/** The minimum number of nearby friendlies needed before the agent no longer tries to regroup*/
    public int minUnitSize = 1;

/** The distance the agent tries to stay away from the enemy*/
    public int minDist2Enemy = 7;

/** The shoulder spacing of a unit*/
    public int minDist2Friendly = 5;

/** The min distance an agent tries to stay away from his base*/
    public int minDist2Base = 20;

/** The number of rounds available to the agent*/
    public int ammo = 10;

/** The normal distance an agent will move in one turn */
    public int cruiseSpeed = 15;

// class wide variables ----------------------------------------------------

/** An iteration counter*/
    int j = 0;

/** The number of enemy in sensor range*/
    int enemyStrength;

/** The number of enemy within the threshold range*/
    int threatStrength;

/** The number of friendlies in sensor or comm range*/
    int friendlyStrength;

/** The number of friendlies within the threshold range*/
    int unitStrength;

/** A variable used to scale the effects of personality factors*/
    int divisor;

/** A variable used to scale the effects of personality factors*/
    int divsr;

/** A switch that gives an agent "armor" if defending his base*/
    public int preparedDefense = 0;

// Vectors of information maintained by the agent (his short term memory) -----------

/** The vector of type observation from agents in sensor range*/
    public Vector observedAgents;

/** The vector of type observation from agents that threaten this agent*/
    public Vector myThreat;

// Boolean state indicators -------------------------------------------------

/** A switch indicating agent is alive*/
    public boolean alive = true;

/** This indicates the simulation has not hit the time limit*/
    public boolean notDone = true;

/** This indicates the agent has fired his weapon*/
    public boolean takeShot = false;

/** This indicates the agent has hit his target*/
    public boolean hitTarget = false;
```

```java
/** This indicates the agent has reached the enemy base*/
    public boolean madeGoal = false;

/** A switch indicating conditions favorable for attacking*/
    public boolean attack = false;

/** A switch indicating conditions favorable for advancing*/
    public boolean advance = false;

/** A switch indicating conditions favorable for regrouping*/
    public boolean regroup = false;

/** A switch indicating conditions favorable for defending*/
    public boolean defend = false;


// CONSTRUCTORS  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

/** This is the default constructor used for logic tests and simple (1 v 1) cases.  It requires
most parameters to be "hardwired" within the TAgent class code.
*/

    public TAgent(int t){
        xnew = 200 + 50*(2*t - 1);
        ynew = 150 + 50*(2*t - 1);
        type = t;
        id = t;
        AssignColor(t);
    }

/** This is the constructor to be used in most cases.  It allows for preset personality traits
constructed in the TAgentAttributes class to be used.  The integer "i" is this agents position in
Manager's 'v' vector.     "t" is the type or side of this agent.    "s" is the squad size or
number of agents per row in the initial formation.    "e" is the initial number of enemy agents.
"A" is the personality this agent will have.
*/

    public TAgent(int i, int t, int s, int e, TAgentAttributes A){
        id = i;
        type = t;
        // s is squad size, a parameter defining the shape of the initial formation
        // e is the starting number of enemy
        int p; //this adjusts an assymetric force with ghost agents to make it symetric for
initialization.
        if (i<=(2*(e-1))+t){p = i;}
        else {p = (((i - (2*e))*2) + t +(2*e));}

        int squad = (int)Math.floor(p/(2*s));
        xnew = 200 + (100*(2*t - 1) + 3*(p-(s*squad*2))*(-2*t + 1));
        ynew = 150 + (100*(2*t - 1) + (6*squad)*(-2*t + 1));
        loadAttributes(A);
        AssignColor(t);

        if (type == 0) {
            goalX = Executive.redBaseX;
            goalY = Executive.redBaseY;
            baseX = Executive.blueBaseX;
            baseY = Executive.blueBaseY;
        }

        if (type == 1) {
            goalX = Executive.blueBaseX;
            goalY = Executive.blueBaseY;
            baseX = Executive.redBaseX;
            baseY = Executive.redBaseY;
        }
    }


// METHODS  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
// $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

/** The run() method is the main execution loop for the TAgent thread. It will call all major
sub-methods. Specifically, this is the TAgent's OODA loop. I have set decide() as a
synchronized method to deconflict the multithreading sleep processes as suggested by Kevin Healy
of Threadtech. Earlier attempts without the synchronization turned into multithreaded knots of
confusion. This does impair the free flow concept of multithreaded agents somewhat, but working
code is happy code.

105

```
*/

    public void run ( ) {
        while (notDone && alive){
            observe( );
            orient( );
            decide( );
            try { sleep(sleepTime);
            }
            catch (InterruptedException e){}
            act( );
            j++;

            if (notDone == false){stop();}
        }
    } // end run() ================================================================

    /** This is the agents observe portion of the OODA loop.  Here the agent scans to see all the
other agents within sensor range. The agent collects the information in the form of a vector of
type observation called observedAgents.
     */

    public void observe( ){

        observedAgents = new Vector( ); //This is the TAgent's short term memory of all agents
within sensor range
        //  Reset the following variables
        enemyStrength = 0;
        threatStrength = 0;
        friendlyStrength = 1;
        unitStrength = 1;
        myFriendsX = 0;
        myFriendsY = 0;

        //  The program finds the range to all other TAgents
        for (int i =0;i < Executive.VP.v.size(); i++){
            TAgent pilot;
            pilot = (TAgent)Executive.VP.v.elementAt(i);
            double range;
            range =  getDistance(pilot);

            // If an enemy agent is within range, he is observed
            if (pilot.id != id && pilot.alive == true) {
                if (range <= sensorRange && pilot.type != type){
                    observation bogey = new observation(pilot);
                    bogey.setRange(range);
                    observedAgents.addElement(bogey);
                    enemyStrength ++;
                    if (range <= thresholdRange) {
                        threatStrength ++;
                    }
                }

            // If an friendly agent is within range, he is observed
                if (pilot.type == type) {
                    if (range <= sensorRange || range <= commRange) {
                        observation bogey = new observation(pilot);
                        bogey.setRange(range);
                        observedAgents.addElement(bogey);
                        friendlyStrength ++;
                        myFriendsX += bogey.hisX;
                        myFriendsY += bogey.hisY;

                        if (range <= thresholdRange) {
                            unitStrength ++;
                        }
                    }
                }
            }
        }
    } // end observe ================================================================

    /** The orient portion of the OODA loop computes the center of mass for
observed enemy and friendly forces and prioritizes threates by position.
It also sets logic switches for possible courses of action.
    */

    public void orient( ){
        // reset the variables
        attack = false;
```

```
            advance = false;
            regroup = false;
            observation temp = new observation();
            observation EWO = new observation();
            myThreatX = 0;
            myThreatY = 0;
            divisor = 0;

/** Sort the observed enemy by range and keep track of highest threats (closest)*/
            int perceivedThreat;
            if (threatStrength < SA) {perceivedThreat = threatStrength;}
                else perceivedThreat = SA;

/** Put the highest threat enemies into the vector myThreat (short term memory)*/
            myThreat = new Vector(perceivedThreat);
            for (int i=0; i<perceivedThreat; i++) {
                temp.setRange(sensorRange +1);
                for (int k=0; k < observedAgents.size(); k++) {
                    EWO = (observation)observedAgents.elementAt(k);
                    if (EWO.team != type && EWO.range < temp.range) {
                        temp = EWO;
                    }
                }
                myThreat.insertElementAt(temp, i);
/** This will compute the center of mass of the threat */
                myThreatX = myThreatX + EWO.hisX;
                myThreatY = myThreatY + EWO.hisY;
            }

/** This will tell the agent his unit is undermanned to do the job and they should regroup*/
            if (unitStrength <= minUnitSize && friendlyStrength >= unitStrength) {
                regroup = true;
                divisor += IRegroup;
            }
            else {regroup = false;}

/** This checks to see if the agent's unit is doctrinally large enough to advance */
            if (unitStrength >= min2Advance || friendlyStrength == 1) {
                advance = true;
                divisor += IAdvance;
            }
            else {advance = false;}

/** This checks the force ratio to see if the agent should attack */
            if (threatStrength != 0) {
                if ((unitStrength/threatStrength) >= ratio2Attack) {
                    attack = true;
                    divisor += IAttack;
                }
                else {
                    attack = false;
                    if (regroup == false){
                        regroup = true;
                        divisor += IRegroup;
                    }
                }
                if ((unitStrength/threatStrength) >= ratio2Overrun) {
                    attack = false;
                    divisor -= IAttack;
                    if (advance == false){
                        advance = true;
                        divisor += IAdvance;
                    }
                }
            }


/** This will call all friendlies in commRange of base to defend the base if its threatened*/
            if ( getDistance2Base(this) <= commRange ) {
                defend = false;
                for (int i=0; i < Executive.VP.v.size(); i++) {
                    TAgent pilot;
                    pilot = (TAgent)Executive.VP.v.elementAt(i);
                    if (pilot.type != type){
                        double range;
                        range =  getDistance2Base(pilot);
                        if (range < sensorRange){
                            defend = true;
                            divisor += IDefend;
                        }
```

```
                    }
                }
            }
            else {
                defend = false;
                preparedDefense = 0;
            }

  /** This computes the center of mass (or CG, center of gravity) for the threat  */
            if (myThreat.size() > 0) {
                myThreatCgX = Math.round(myThreatX/myThreat.size());
                myThreatCgY = Math.round(myThreatY/myThreat.size());
            }

  /** This computes the center of mass (CG) for the friendlies in comm/sensor range */
            if (friendlyStrength > 1) {
                myFriendsCgX = Math.round(myFriendsX/(friendlyStrength-1));
                myFriendsCgY = Math.round(myFriendsY/(friendlyStrength-1));
            }

//System.out.println( "TAgent   " + id + " orients " );
//System.out.println( id +"'s attack is " + attack + " and regroup is  " + regroup);
        } // end orient =================================================================

    /** The decide part of the OODA loop picks a course of action for the agent.  It decides where
the agent should move and whether or not the agent should engage the enemy.  This is the
synchronized method so only one agent at a time can decide a course of action.  The thread is put
to sleep after this method to simulate the time required to complete the decision cycle including
physically executing the decision the agent makes here.
    */
    synchronized void decide( ){

    //  First, re-initialize the variables that will be calculated

            deltaX = 0;
            deltaY = 0;

    //  Next the agent calculates his best move

            if (advance == true) {   // Move toward goal
                    if (xnew < goalX){deltaX += (Math.min((cruiseSpeed),(goalX-xnew)))*IAdvance;}
                        else {deltaX -= (Math.min((cruiseSpeed),(xnew-goalX)))*IAdvance;}

                    if (ynew < goalY){deltaY += (Math.min((cruiseSpeed),(goalY-ynew)))*IAdvance;}
                        else {deltaY -= (Math.min((cruiseSpeed),(ynew-goalY)))*IAdvance;}
            }

            if (defend == true) {   // Move toward base
                if (xnew < baseX){deltaX += (Math.min((cruiseSpeed),(baseX - xnew)))*IDefend;}
                    else {deltaX -= (Math.min((cruiseSpeed),(xnew-baseX)))*IDefend;}
                if (ynew < baseY){deltaY += (Math.min((cruiseSpeed),(baseY - ynew)))*IDefend;}
                    else {deltaY -= (Math.min((cruiseSpeed),(ynew - baseY)))*IDefend;}
                preparedDefense ++;
                if(preparedDefense < 3){health ++;} // simulates an advantage for a prepared defense
            }

            if (attack == true) { // move to enemy cg wrt min range to enemy & friendly
                int distX = Math.abs(myThreatX - baseX);
                int distY = Math.abs(myThreatY - baseY);
                double scale = bestFiringRange/Math.max(1,getDistance(myThreatCgX, myThreatCgY,
baseX, baseY));
                int dx = Math.round(distX * (float)scale);
                int dy = Math.round(distY * (float)scale);

                if (myThreatCgX < baseX){
                    if(xnew < myThreatCgX+dx){deltaX+=(Math.min(cruiseSpeed, (myThreatCgX+dx)-
xnew))*IAttack;}
                    else{deltaX +=(Math.min(cruiseSpeed, xnew-(myThreatCgX+dx)))*IAttack; }
                }
                else {
                    if(xnew < myThreatCgX-dx){deltaX+=(Math.min(cruiseSpeed, (myThreatCgX-dx)-
xnew))*IAttack;}
                    else{deltaX +=(Math.min(cruiseSpeed,xnew-(myThreatCgX-dx)))*IAttack; }
                }
                if (myThreatCgY < baseY){
                    if(ynew < myThreatCgY+dy){deltaY+=(Math.min(cruiseSpeed, (myThreatCgY+dy)-
ynew))*IAttack;}
                    else{deltaY +=(Math.min(cruiseSpeed, ynew-(myThreatCgY+dy)))*IAttack; }
                }
                else {
```

```
                if(ynew < myThreatCgY-dy){deltaY+=(Math.min(cruiseSpeed,(myThreatCgY-dy)-
ynew))*IAttack;}
                else{deltaY +=(Math.min(cruiseSpeed,ynew-(myThreatCgY-dy)))*IAttack; }
            }

        }

        if (regroup == true) {   //  Move toward friendly cg wrt min range to enemy & friendly
                    if (xnew < myFriendsCgX){
                        deltaX += (Math.min((cruiseSpeed),myFriendsCgX))*IRegroup;
                    }
                    else {deltaX -= (Math.min((cruiseSpeed),myFriendsCgX))*IRegroup;}
                    if (ynew < myFriendsCgY){
                        deltaY += (Math.min((cruiseSpeed),myFriendsCgX))*IRegroup;
                    }
                    else {deltaY -= (Math.min((cruiseSpeed),myFriendsCgX))*IRegroup;}
        }

/** Normalize the agent's movement based on his personality */
            if (divisor != 0) {
                if (Math.abs(deltaX/divisor) < 1 && deltaX != 0) {
                    if ((deltaX/divisor)>0){deltaX = 1;}
                    else {deltaX = -1;}
                }
                else {deltaX = Math.round(deltaX/divisor);}

                if (Math.abs(deltaY/divisor) < 1 && deltaY != 0) {
                    if ((deltaY/divisor)>0){deltaY = 1;}
                    else {deltaY = -1;}
                }
                else {deltaY = Math.round(deltaY/divisor);}
            }

/**  This will limit the distance moved in one step to "maxSpeed" */
            if (deltaX >= deltaY) {
                if (deltaX > maxSpeed){
                    deltaY = maxSpeed*(deltaY/deltaX);
                    deltaX = maxSpeed;
                    }
                if (deltaX < -maxSpeed){
                    deltaY = -maxSpeed*(deltaY/deltaX);
                    deltaX = -maxSpeed;
                    }
            }
            if (deltaX < deltaY) {
                if (deltaY > maxSpeed){
                    deltaX = maxSpeed*(deltaX/deltaY);
                    deltaY = maxSpeed;
                    }
                if (deltaY < -maxSpeed){
                    deltaX = -maxSpeed*(deltaX/deltaY);
                    deltaY = -maxSpeed;
                    }
            }

//  This will keep the agents from running off the edge of the visible battlefield
            if (deltaX + xnew >= 395){deltaX = 395 - xnew;}
            if (deltaX + xnew <= 5){deltaX = xnew - 5;}
            if (deltaY + ynew >= 295){deltaY = 295 - ynew;}
            if (deltaY + ynew <= 5){deltaY = ynew - 5;}

//  This will stop the agent at the goal when he gets there
            if (xnew == goalX && myThreat.size()==0){deltaX = 0;
            }
            if (ynew == goalY && myThreat.size()==0){deltaY = 0;
            }
//  This will park the agent at the top or bottom of the screen when he reaches his goal
            if ((getDistance2Goal(this) <= 3) && myThreat.size()==0){
                deltaX = 0;
                deltaY = 0;
                xnew = 200 - ((150-5*id)*(2*type - 1));
                ynew = 150 - 145*(2*type - 1);
                goalX = xnew;
                goalY = ynew;
                outColor = Color.white;
                madeGoal = true;
            }
//  This will adjust the new normalized position to keep desired distance from base
            int min2Base = 3;
            if (defend == false) {min2Base = minDist2Base;}
```

```
                if (getDistance2Base(this) <= min2Base) {
                    if ((xnew - baseX) > 0) {delX += (min2Base - (xnew - baseX))*IAvoidBase;}
                    else {delX -= (min2Base - (baseX - xnew))*IAvoidBase;}

                    if ((ynew - baseY) > 0) {delY += (min2Base - (ynew - baseY))*IAvoidBase;}
                    else {delY -= (min2Base - (baseY - ynew))*IAvoidBase;}

                    divsr += IAvoidBase;
                }

        } // end decide ========================================================================

/** The act method simply executes the decision made before the thread's sleep cycle.
First, the agent moves into his selected position, then he fires at his greatest threats
based on the information (threat location) gathered in the observe method.
*/
    public void act( ){

/*  This will adjust the TAgent's next position for restrictions on being too close to others
that may have moved while he slept */
                int myNextX = (int)deltaX + xnew;
                int myNextY = (int)deltaY + ynew;
                delX = 0;
                delY = 0;
                int divsr = 0;
                int min2Enemy = 5;
                int min2Friendly = 5;

                if (observedAgents.size()>0) {
                    for (int s = 0; s < observedAgents.size(); s++){
                        observation pilot;
                        TAgent copilot;
                        pilot =(observation)observedAgents.elementAt(s);
                        copilot = (TAgent)Executive.VP.v.elementAt(pilot.index);
                        double nextRange = getDistance(copilot);

                        if (attack == false){min2Enemy = minDist2Enemy;}

                        if ((copilot.type != type) && (nextRange <= min2Enemy) && (copilot.alive ==
true)) {

                            if ((myNextX-copilot.xnew)>0){
                                delX += (min2Enemy - (myNextX - copilot.xnew))*IAvoidEnemy;
                            }
                            else {delX -= (min2Enemy - (copilot.xnew - myNextX))*IAvoidEnemy;}

                            if ((myNextY-copilot.ynew)>0){
                                delY += (min2Enemy - (myNextY - copilot.ynew))*IAvoidEnemy;
                            }
                            else {delY -= (min2Enemy - (copilot.ynew - myNextY))*IAvoidEnemy;}

                            divsr += IAvoidEnemy;
                        }

                        if (regroup == false) {min2Friendly = minDist2Friendly;}
                        if (getDistance(myNextX, myNextY, goalX, goalY) < 5) {min2Friendly = 1;}


if((copilot.type==type)&&(nextRange<=min2Friendly)&&(copilot.madeGoal==false)){
                            if ((myNextX-copilot.xnew)>0){
                                delX += (min2Friendly - (myNextX - copilot.xnew))*IAvoidFriendlies;
                            }
                            else {delX -= (min2Friendly - (copilot.xnew -
myNextX))*IAvoidFriendlies;}

                            if ((myNextY-copilot.ynew)>0){
                                delY += (min2Friendly - (myNextY - copilot.ynew))*IAvoidFriendlies;
                            }
                            else {delY -= (min2Friendly - (copilot.ynew -
myNextY))*IAvoidFriendlies;}

                            divsr += IAvoidFriendlies;

                        }
                    }
                }
```

```
                if (divsr != 0) {
                    if (Math.abs(delX/divsr) < 1 && delX != 0) {
                        if ((delX/divsr)>0){delX = 1;}
                        else {delX = -1;}
                    }
                    else {delX = Math.round(delX/divsr);}

                    if (Math.abs(delY/divsr) < 1 && delY != 0) {
                        if ((delY/divsr)>0){delY = 1;}
                        else {delY = -1;}
                    }
                    else {delY = Math.round(delY/divsr);}
                }

//  This will keep the agents from running off the edge of the visible battlefield
                if (delX + myNextX >= 395){delX = 395 - myNextX;}
                if (delX + myNextX <= 5){delX = myNextX - 5;}
                if (delY + myNextY >= 295){delY = 295 - myNextY;}
                if (delY + myNextY <= 5){delY = myNextY - 5;}


            if (alive == true){
/* This is needed to keep from acting if the agant was killed while the thread slept */

                xold =  xnew;
                yold =  ynew;
                xnew = xnew + (int)deltaX + (int)delX;
                ynew = ynew + (int)deltaY + (int)delY;
                Executive.field.repaint( ); // *** repaint the screen

                for (int s = 0; s < myThreat.size(); s++){
                    observation navigator;
                    navigator =(observation)myThreat.elementAt(s);
                    if (navigator.range<weaponsRange && ammo>0 && navigator.isAlive==true){
                        targetX = navigator.hisX;
                        targetY = navigator.hisY;
                        takeShot = true;
                        ammo--;
                        shoot(navigator.index);
                        Executive.field.repaint( ); // *** repaint the screen
                    }
                    takeShot = false;    // reset these switches for the next iteration
                    hitTarget = false;
                }
            }
        }

    } // end act() ============================================================


//   LESSOR METHODS $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
// $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

/** This method assigns the appropriate color to each side */
    public void AssignColor (int n){
        if (n == 0) {
            inColor = Color.blue;
            outColor = Color.blue;
        }
        else if (n == 1) {
            inColor = Color.red;
            outColor = Color.red;
        }
        else {
            inColor = Color.green;
            outColor = Color.green;
        }
    } // End sleep ============================================================================

/** This method draws the agent and tracer fire on the screen */
    public void Draw(Graphics g){
        g.setColor(outColor);
        g.drawOval(xnew-2, ynew-2, 4, 4);
        if (takeShot == true && alive == true) {
            g.drawLine(xnew, ynew, targetX, targetY);
        }
        if (hitTarget == true) {
            g.drawOval(targetX-3, targetY-3, 6, 6);
        }
        g.setColor(inColor);
        g.fillOval(xnew-2, ynew-2, 4, 4);
```

111

```java
        int b = Executive.baseSize;
        g.setColor(Color.blue);
        g.fillRect(Executive.blueBaseX, Executive.blueBaseY, b, b);
        g.setColor(Color.red);
        g.fillRect(Executive.redBaseX-b, Executive.redBaseY-b,b,b);




    } // End of Draw =========================================================================

    /** This calculates the range to another agent */
    public double getDistance(TAgent t) {
        double squareDistance;
        squareDistance = (xnew - t.xnew)*(xnew - t.xnew)+(ynew - t.ynew)*(ynew - t.ynew);
        return Math.sqrt(squareDistance);
    } // end getDistance (TAgent)=============================================================

    /** This calculates the distance between any two cartesian coordinates */
    public double getDistance(int x, int y, int X, int Y) {
        double squareDistance;
        squareDistance = (x - X)*(x - X)+(y - Y)*(y - Y);
        return Math.sqrt(squareDistance);
    } // end getDistance (int,int,int,int)===================================================

    /** This finds the distance an agent is from his base */
    public double getDistance2Base(TAgent t) {
        double squareDistance;
        squareDistance = (baseX - t.xnew)*(baseX - t.xnew)+(baseY - t.ynew)*(baseY - t.ynew);
        return Math.sqrt(squareDistance);
    } // end getDistance2Base =============================================================

    /** This finds the distance an agent is from his goal */
    public double getDistance2Goal(TAgent t) {
        double squareDistance;
        squareDistance = (goalX - t.xnew)*(goalX - t.xnew)+(goalY - t.ynew)*(goalY - t.ynew);
        return Math.sqrt(squareDistance);
    } // end getDistance2Goal =============================================================

    /** This method shoots at target coordinates.  The shot is penalized by distance.  A
    hit is determined by a stochastic random number draw against the probability of hit (Pk) */
    public void shoot(int id){
        TAgent victim;
        double targetRange;
        victim = ((TAgent)Executive.VP.v.elementAt(id));
        if (victim.xnew==targetX && victim.ynew==targetY) {//can hit only if TGT doesn't move)
            targetRange = getDistance(victim);
            if (type == 0){Executive.VP.totalBlueShots++;}
                else {Executive.VP.totalRedShots++;}
            if
(Executive.getRandom()<(pk*(bestFiringRange/Math.max(bestFiringRange,targetRange))) ){
                victim.gotShot();
                hitTarget = true;
                if (type == 0) {Executive.VP.totalBlueHits ++;}
                    else {Executive.VP.totalRedHits ++;}
            }
        }
    } // end shoot ==========================================================================

    /** This method decrements the agent's health if he gets shot and puts him in
    the morgue if he dies */
    public void gotShot(){
        health--;
        if (health == 0){
            alive = false;
            xnew = 200 + 195*(2*type - 1);  //This puts the dead agent on the side of the field
            ynew = 150 + (100 - 5*id)*(2*type - 1); //This adjusts vertical position of the dead
            if (type == 0) {inColor = Color.cyan;}  // This makes the dead pale in color
            else if (type == 1) {inColor = Color.pink;}
        }
    } // end gotShot ========================================================================

/** This method is called by the constructor and will load personality and doctrinal attributes
from an object of type TAgentAttributes into a newly created agent */
    public void loadAttributes(TAgentAttributes A){
        sleepTime = A.sleepTime;
        health = A.health;
        pk = A.pk;
        SA = A.SA;
        maxSpeed = A.maxSpeed ;
```

112

```
        IAdvance = A.IAdvance;
        IAttack = A.IAttack;
        IRegroup = A.IRegroup;
        IDefend = A.IDefend ;
        IAvoidEnemy = A.IAvoidEnemy;
        IAvoidFriendlies = A.IAvoidFriendlies;
        IAvoidBase = A.IAvoidBase;

        sensorRange = A.sensorRange;
        weaponsRange = A.weaponsRange;
        bestFiringRange = A.bestFiringRange;
        commRange = A.commRange;
        thresholdRange = A.thresholdRange;
        ratio2Attack = A.ratio2Attack;
        ratio2Overrun = A.ratio2Overrun;
        min2Advance = A.min2Advance;
        minUnitSize = A.minUnitSize;
        minDist2Enemy = A.minDist2Enemy;
        minDist2Friendly = A.minDist2Friendly;
        minDist2Base = A.minDist2Base;
        ammo =  A.ammo;
        cruiseSpeed = A.cruiseSpeed;
    } // end loadAttributes=================================================================


}// end of TAgent mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
```

Source code for TAgentAttributes class

```
/*  Major Tom Tighe's Air Force Institute of Technology Thesis Project

    last revised on 29 Jan 1999
*/


import java.awt.*;
import java.lang.*;

/**This class should contain most of the parameters needed
to build a TAgent with the exception of the index, the side,
and the initial position.  In other words this class will hold
the personalities of the agents and should help keep the Executive,
Manager, and TAgent initialization cleaner and easier.  */

public class TAgentAttributes {

//VARIABLES $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
// These are individual personality traits: ========================================

/**  The length of delay in the decision cycle*/
    public int sleepTime = 200;

/**  How many hits until this agent dies*/
    public int health = 1;

/**  The single shot probability of kill at best range*/
    public double pk = .5;

/** The max number of enemy of which the agent can keep aware */
    public int SA = 1;

/** The largest step an agent can take in one turn*/

    public int maxSpeed = 20;

/** The personality weight or desire to advance to goal */
    public int IAdvance = 2;

/** The personality weight or desire to attack if favorable force ratio*/
    public int IAttack = 10;

/** The personality weight or desire to regroup if unit gets too small*/
    public int IRegroup = 5;

/** The personality weight or desire to defend base if called */
    public int IDefend = 7;

/**  The personality weight or desire to avoid enemy if not attacking*/
    public int IAvoidEnemy = 2;

/** The personality weight or desire to maintain spacing if unit size is OK */
    public int IAvoidFriendlies = 2;

/** The personality weight or desire to avoid base if not under attack */
    public int IAvoidBase = 2;

// These are doctrinal standards: ============================================

  /** This is this side's visual or sensor range*/
    public int sensorRange = 70;

  /** This is this side's max weapons range*/
    public int weaponsRange = 60;

  /** This is this side's doctrinal best firing range range*/
    public int bestFiringRange = 40;

  /** This is this side's communications range*/
    public int commRange = 80;

  /** This is a range at which other agents become noticable or threatening*/
    public int thresholdRange = 40;

  /** The force ratio advantage an agent needs before attacking*/
    public int ratio2Attack = 1;
```

114

```
/** The force ratio needed to stop attacking and advance to goal*/
    public int ratio2Overrun = 3;

/** The minimum unit size an agent needs to advance toward goal*/
    public int min2Advance = 2;

/** The minimum number of nearby friendlies needed before the agent no longer tries to regroup*/
    public int minUnitSize = 3;

/** The distance the agent tries to stay away from the enemy*/
    public int minDist2Enemy = 30;

/** The shoulder spacing of a unit*/
    public int minDist2Friendly = 20;

/** The min distance an agent tries to stay away from his base*/
    public int minDist2Base = 30;

/** The number of rounds available to the agent*/
    public int ammo = 100;

/** The normal distance an agent will move in one turn */
    public int cruiseSpeed = 15;



//CONSTRUCTORS $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


/** This is the default constructor.  It accepts the default value for each variable */
    public TAgentAttributes(){} // end of default constructor =====================


/** This constructor contains many predefined personalities or sets of attributes.  Any number of
personalities can be built and stored here.  This is especially useful in multiple case
comparison runs when an integer iteration counter can be used to step through many of the
predefined personalities.  The integer argument "i" is the label given to a personality defined
in this constructor.
*/
    public TAgentAttributes(int i){
        if (i == 0){               // This is the baseline personality
            sleepTime = 200;
            maxSpeed = 20;
        }

        else if (i == 1){
            sleepTime = 200;
        }
        else if (i == 2){
            sleepTime = 250;
        }
        else if (i == 3){
            sleepTime = 300;
        }
        else if (i == 4){
            sleepTime = 350;
        }
        else if (i == 5){
            sleepTime = 400;
        }
        else if (i == 6){
            sleepTime = 450;
        }
        else if (i == 7){
            sleepTime = 500;
        }
        else if (i == 8){
            sleepTime = 550;
        }
        else if (i == 9){
            sleepTime = 600;
        }
        else if (i == 10){
            sleepTime = 650;
        }
        else if (i == 11){
            sleepTime = 700;
        }
        else if (i == 12){
            sleepTime = 750;
```

115

```
        }
        else if (i == 13){
            sleepTime = 800;
        }
        else if (i == 14){
            sleepTime = 850;
        }
        else if (i == 15){
            sleepTime = 900;
        }
        else if (i == 16){
            sleepTime = 950;
        }
        else if (i == 17){
            sleepTime = 1000;
        }
        else {
            System.out.println("Please select a TAgentAtttribute between 0 and 17");
        }


        if (i!=0){
        //    maxSpeed = sleepTime/10;
        //    cruiseSpeed = (int)Math.round(.75*maxSpeed);
        //    pk = 1.0;
        }

    }// end of constructor with argument (int i)=================================



/** This constructor is to be used to adjust sleepTime from user input without having to build a
predefined personality.  The float value "f" most be between 0 and 5.0 and will be the sleep time
or OODA delay in seconds.
*/
    public TAgentAttributes(float f){
        if ( (f > 0) && (f < 5000)){
            sleepTime = Math.round(f*1000);
        //    maxSpeed = Math.round(f*100);
        //    cruiseSpeed = (int)Math.round(.75*maxSpeed);
        }
        else{
            System.out.println("Please reinitialize with a OODA delay between 0 and 5.   \nThe
base line is 0.2");
        }
    }//End of constructor with float argument =====================================


//  METHODS $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

/**  This can be used to set the variable "sleepTime" in the TAgentAttribute */
public void setSleepTime(int s)              { sleepTime = s;}

/**  This can be used to set the variable "sensorRange" in the TAgentAttribute */
public void setSensorRange(int o)            { sensorRange = o;}


/** This method will return the value of the variable "sleepTime" in the TAgentAttribute */
public int getSleepTime()                    { return sleepTime;}

/** This method will return the value of the variable "sensorRange" in the TAgentAttribute */
public int getSensorRange()                  { return sensorRange;}

} // end of TAgentAttributes   mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
```

Source code for Observation class

```java
// Major Tom Tighe's Air Force Institute of Technology Thesis Project
//
// last revised on 06 Jan 1999

import java.awt.*;

/**  This class is used to store relevent information about agents that have
been observed.  This information is used by the TAgent in decisions about posture
and movement.
*/
public class observation {

//VARIABLES $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

/**  This is the range from this agent to the observed agent*/
public double range;

/**  This is the observed agents id or position in the vector of agents in the Manager*/
public int index;

/**  This is the type of the observed agent*/
public int team;

/**  This is the x coordinate of the observed agent*/
public int hisX;

/**  This is the y coordinate of the observed agent*/
public int hisY;

/**  This indicated if the observed agent is alive*/
public boolean isAlive;

/**  Indicates the observed agent has reached the enemy base*/
public boolean hasMadeGoal;

//CONSTRUCTORS $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

    /** This is the default constructor.  All variable values are set to zero or false */
    public observation(){
        index = 0;
        team = 0;
        hisX = 0;
        hisY = 0;
        range = 0;
        isAlive = false;
        hasMadeGoal = false;
    } // End of default constructor ============================================


    /**  This constructor makes an observation of the TAgent passed to it*/
    public observation(TAgent t){
        index = t.id;
        team = t.type;
        hisX = t.xnew;
        hisY = t.ynew;
        range = 0;
        isAlive = t.alive;
        hasMadeGoal = t.madeGoal;
    }// End of constructor ==================================================

// METHODS $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
// I built these methods, but only used setRange() in my thesis project.

/**  */
public void setIndex(int i)     {  index = i;}

/**  This can be used to set the variable "team" in the observation */
public void setTeam(int t)       {  team = t; }

/**  This can be used to set the variable "hisX" in the observation */
public void setHisX(int x)       {  hisX = x;}

/**  This can be used to set the variable "hisY" in the observation */
public void setHisY(int y)       {  hisY = y;}

/**  This can be used to set the variable "range" in the observation */
public void setRange(double r)   {  range = r;}
```

117

```java
/** This method will return the value of the variable "index" in the observation */
public int getIndex()            {   return index;}

/** This method will return the value of the variable "team" in the observation */
public int getTeam()             {   return team;}

/** This method will return the value of the variable "hisX" in the observation */
public int getHisX()             {   return hisX;}

/** This method will return the value of the variable "hisY" in the observation */
public int getHisY()             {   return hisY;}

/** This method will return the value of the variable "range" in the observation */
public double getRange()         {   return range;}



} // End of observation   mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
```

Source code for DataPoint class

```
// Major Tom Tighe's Air Force Institute of Technology Thesis Project
//
// last revised on 21 Jan 1999

import java.awt.*;

/** The dataPoint class is used to gather and hold sample data.  It
currently keeps track of the number of agents still alive and those
at their goal in the simulation at the time dataPoint is called
*/
public class dataPoint {

//VARIABLES $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

/** The number of alive agents of type 0 (blue) */
static int blueAlive = 0;

/** The number of alive agents of type 1 (red) */
static int redAlive = 0;

/** The number of agents of type 0 (blue) that have reached their goal */
static int blueAtGoal = 0;

/** The number of agents of type 1 (red) that have reached their goal */
static int redAtGoal = 0;

// CONSTRUCTORS $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

/** This class only has a default constructor */
    public dataPoint(){}

// METHODS $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

/** This method loads the data about one agent into the dataPoint object.
This method will normally be called iteratively to sample all the agents controlled
by the current Manager.  This method is called from within Manager.
*/
    static void addData(TAgent t){
        if (t.type == 0){
            if (t.alive == true) {blueAlive++;}
            if (t.madeGoal == true) {blueAtGoal++;}
        }
        if (t.type == 1){
            if (t.alive == true) {redAlive++;}
            if (t.madeGoal == true) {redAtGoal++;}
        }
/** The number of alive agents of type 0 (blue) */
    } //end of addData() =====================================================


    /**  This method resets all the values help in the dataPoint to zero */
    public void zero(){
        blueAlive = 0;
        redAlive = 0;
        blueAtGoal = 0;
        redAtGoal = 0;
    }// End of zero() =====================================================


} // end of dataPoint   mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
```

Vita

Major Tom Tighe was born on 25 April 1963 in Billings, Montana to Nancy and Robert Tighe. He graduated from Memorial High School in Tulsa, Oklahoma in 1981 and attended the United States Air Force Academy. He graduated with a Bachelor of Science in Engineering Sciences in May of 1985.

He completed Undergraduate Pilot Training at Vance AFB, Oklahoma in 1986. His first operational assignment was flying B-52Hs at Carswell AFB, TX. There Major Tighe progressed from copilot to instructor aircraft commander status and performed duties including 7BW standardization and evaluation. While at Carswell, he won the Mitchell Trophy, LeMay Trophy, and Ryan Trophy and was the aircraft commander of the Strategic Air Command's 1992 B-52 crew of the year.

He was reassigned to Castle AFB, CA in 1993 and served as a Formal Training Unit (FTU) flight instructor. In 1994, the FTU was relocated to Barksdale AFB, LA and Major Tighe assumed Chief Pilot duties before instructing at the Combat Flight Instructor Course.

Major Tighe was selected for the Operations Analysis program at AFIT in 1997. Upon graduation he will be assigned to the Air Force Studies and Analysis Agency, Pentagon, Virginia.

Major Tighe married the former Christine Henriksen in 1990. Together they have four children: Mitchell, Amelia, Anna and Claire, the last two of which were born during this thesis effort.

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1999	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE
STRATEGIC EFFECTS OF AIRPOWER AND COMPLEX ADAPTIVE AGENTS:
AN INITIAL INVESTIGATION

5. FUNDING NUMBERS

6. AUTHOR(S)
Thomas R. Tighe, Major, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
2950 P Street
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOA/ENS/99M-09

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Mr. Allen Murashige
HQAF/XOC
1070 Air Force
Pentagon, VA

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

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

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

US airpower theory and doctrine depend on the concept that the destruction of a few key targets or centers of gravity can unravel the enemy's physical ability to wage war or break his will to prosecute the war. This synergistic decimation of the enemy's effectiveness and resistance to our political will is known as Strategic Effects. These strategic effects are very difficult to quantify and are not directly accounted for in current DoD computer models. Since these computer models are used to aid with decisions about force structure and budget priorities, many believe that the Air Force's greatest potential contribution to modern joint warfare is going unrecognized and under financed.

This thesis explores military theory and current doctrine to define a method quantifying strategic effects. This method is based upon the Observe-Orient-Decide-Act (OODA) decision cycle. Next, current modeling techniques, and specifically the campaign level model, THUNDER, are examined for applicability to model strategic effects as defined. Finally, a proof of concept model is developed to study the advantage associated with OODA loop exploitation. This simple model uses Java-based, multi-threaded, autonomous, complex adaptive agents to demonstrate the non-linear (synergistic) results of OODA loop exploitation. These results are similar to the anticipated effects of strategic attack and provide a solid foothold from which the study and modeling of strategic effects can begin.

14. SUBJECT TERMS
Strategic Warfare, Airpower, Computerized Simulation, Adaptive Systems

15. NUMBER OF PAGES
133

16. PRICE CODE

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