3-2005

A Knowledge Matrix Modeling of the Intelligence Cycle

Kevin J. Whaley

A KNOWLEDGE MATRIX MODELING

OF THE INTELLIGENCE CYCLE

THESIS

Kevin J. Whaley, Capt, USAF

AFIT/GOR/ENS/05-18

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

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

AFIT/GOR/ENS/05-18

A KNOWLEDGE MATRIX MODELING OF THE INTELLIGENCE CYCLE

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Kevin J. Whaley, BBA

Capt, USAF

March 2005

AFIT/GOR/ENS/05-18

A KNOWLEDGE MATRIX MODELING OF THE INTELLIGENCE CYCLE

Kevin J. Whaley, BBA
Capt, USAF

Approved:

_____	_____
John O. Miller, PhD (Chairman)	Date
_____	_____
Stephen P. Chambal, Maj, USAF, PhD (Reader)	Date

Abstract

This effort models information flow through the United States Intelligence Community's Intelligence Cycle using a knowledge matrix methodology. The knowledge matrix methodology takes explicit data from multiple sources and fuses that data to measure a current level of knowledge about a target, or situation. Knowledge matrices are used to develop a measure of user-needs satisfaction. User-needs satisfaction compares requested levels of knowledge to a probability of collecting that knowledge within a designated timeframe. This effort expands the work done by Captain Carl Pawling in his March 2004 thesis, Modeling and Simulation of the Military Intelligence Process, by modeling intelligence as an opportunistic, multi-source, multi-entity system of systems. The value of intelligence fusion is compared, and analyzed between three different algorithms; no fusion, a mixed forward and fuse strategy, and strict fusion strategy. These fusion algorithms are then applied to competing intelligence collection architectures in varying intelligence activity scenarios to determine which architectures will most improve the probability of satisfactory collection. Satisfactory collection is measured in terms of quantity, timeliness, and user-need satisfaction of completed intelligence reports.

.

Acknowledgements

Thank you to the entire GOR-05M class and all of my professors for helping me complete this course of study. Special thanks to Capt Clinton Clark, 1Lt Tim Cook, 1Lt Jim Markham, and1Lt Earl Bednar you all helped me through this course when I needed help most. May those who walk this path after me realize their fortunes and not become enmeshed in a quixotic academic quagmire. Learn what you can while you attend AFIT and then go forth and do good work.

The most thanks go to my family for they have given the most and asked the least of me during my time at AFIT.

Table of Contents

Page

Abstract iv

Acknowledgements v

List of Figures viii

List of Tables x

1. Introduction 1-1
   - 1.1 Background 1-1
   - 1.2 Problem Statement 1-2
   - 1.3 Research Objective 1-2
   - 1.4 Research Focus 1-3
   - 1.5 Methodology 1-4
   - 1.6 Assumptions and Limitations 1-5
   - 1.7 Preview 1-6
2. Literature Review 2-1
   - 2.1 Introduction 2-1
   - 2.2 Defining Intelligence 2-2
   - 2.3 Knowledge and Intelligence 2-3
   - 2.4 The Nature of Information 2-4
   - 2.5 The Nature of Data 2-8
   - 2.6 Knowledge Management 2-11
   - 2.7 The Value of Knowledge and Intelligence 2-14
   - 2.8 A Brief History of the American Intelligence Enterprise 2-16
   - 2.9 The U.S. Intelligence Community and the Intelligence Cycle 2-18
   - 2.10 Quantifying and Modeling Intelligence and Knowledge 2-26
   - 2.11 Summary 2-33
3. Methodology 3-1
   - 3.1 Introduction 3-1
   - 3.2 Knowledge Matrix Concept 3-3
   - 3.3 Modifying the Knowledge Matrix Method 3-5
   - 3.4 Model Creation with Arena 3-11
   - 3.5 The Intelligence Cycle - A Needs Driven Model 3-13
     - 3.5.1 Model Concept 3-13
     - 3.5.2 Planning & Guidance Submodel 3-13
     - 3.5.3 Collection Submodel 3-25
     - 3.5.4 Need Assessment 3-30
     - 3.5.5 Processing & Exploitation Submodel 3-38
     - 3.5.6 Analysis and Production Submodel 3-42
     - 3.5.7 Dissemination Submodel 3-45
   - 3.6 Validation and Verification 3-49
4. Data Analysis 4-1

4.1 Introduction 4-1
4.2 Overcoming Initialization Bias 4-2
4.3 Statistical Measures Defined 4-5
4.3.1 Measuring Quantity 4-5
4.3.2 Measuring User Satisfaction 4-6
4.4 Creating a Baseline Scenario 4-8
4.4.1 Baseline Results 4-9
4.4.2 Baseline Inferences 4-13
4.5 Effect of a Heightened Perceived Threat Environment 4-14
4.5.1 Heightened Perceived Threat (WAR) Results 4-14
4.5.2 WAR Quantity 4-15
4.5.3 WAR User Satisfaction 4-16
4.5.2 Heightened Perceived Threat Inferences 4-18
4.6 Comparing Intelligence Architectures 4-19
4.6.1 Comparing Architectures Results 4-19
4.6.2 Comparing Architecture Inferences 4-24
4.7 Analysis Summary 4-25
5. Conclusions 5-1
5.1 Model Capabilities 5-1
5.2 Model Strengths 5-2
5.3 Model Weaknesses with Suggested Improvements 5-3
5.4 Future Research 5-4
Appendix A: Acronym Listing A-1
Appendix B: User-Defined Model Attributes B-1
Appendix C: Resource Parameter Tables C-1
Appendix D: Resource Schedules D-1
Appendix E: Paired T-Tests E-1

Bibliography 171

Vita 173

List of Figures

Page

1. Intelligence Cycle (JP 2-0, 2000:V).....2-19
2. Conceptual Model Flow-Chart 3-2
3. Partial Wire Diagram - Planning and Direction Submodel 3-14
4. Entity.Type Animated Figures.....3-17
5. Post Library Needs Assessment Logic 3-24
6. RADINT1 Capacity & Availability Schedule 3-27
7. Wire Diagram - Collection Submodel 3-28
8. Wire Diagram - Time Management Subprocess.....3-29
9. Wire-Diagram Activity_State Needs Parser and Counts Logic.....3-31
10. Wire Diagram - Collection Decision Tree Logic.....3-32
11. Wire Diagram - Resource Selection Modeling.....3-36
12. Wire Diagram - Processing & Exploitation Submodel.....3-39
13. Wire Diagram -Analysis & Production Submodel 3-44
14. Wire Diagram - Dissemination Submodel.....3-47
15. Wire Diagram - No Fusion Statistics Collection 3-48
16. Initialization Bias.....4-3
17. WIP Spiking Details 4-4
18. Discrete Vs. Continuous Satisfaction Ratios Measured 4-8
19. Baseline Quantity Measured.....4-10
20. Baseline Total User Satisfaction.....4-11
21. Baseline Timeliness Comparison by Algorithm.....4-12
22. Baseline User-Needs Satisfaction by Algorithm 4-13

23. Mixed Fusion Baseline Vs. War Quantity 4-15
24. War Total Satisfaction Algorithm Comparison 4-16
25. Mixed Fusion Base Vs. War Timeliness Comparison 4-17
26. Mixed Fusion Baseline Vs. War User Needs Satisfaction 4-18
27. Mixed Fusion Quantity Architecture Comparison 4-20
28. Scatter-Plot of MF Quantity Architecture Comparison 4-21
29. Total Satisfaction Architecture Comparison 4-22
30. Timeliness Comparison between Architectures 4-23
31. User Needs Satisfaction Comparison by Architecture 4-24
32. Location Satisfaction Comparison by Architecture 4-25
33. IMINT1 Resource Schedule D-2
34. IMINT2 Resource Schedule D-3
35. SIGINT1 Resource Sched D-4
36. SIGINT2 Resource Schedule D-5
37. RADINT1 Resource Schedule D-6
38. RADINT2 Resource Schedule D-7
39. MASINT2 Resource Schedule D-8

List of Tables

Table Page

40. Intelligence Disciplines (JP 2-0, 2000: 25) 2-21

41. Fusion Level Definitions (JDL, 2004:1) 2-24

42. Original Knowledge Matrix Example 3-4

43. Satisfaction Level Definitions 3-6

44. User-Needs Vector in a Knowledge Matrix 3-7

45. User-Needs Generation Distributions 3-8

46. Sample Collection Matrix, IMINT1 3-9

47. Comparison of Collection Vector to a User Needs Vector 3-10

48. Second Sensor Collection Vector Compared to Original User-Needs Vector 3-10

49. Maximum Satisfaction Vector 3-11

50. Entity Arrival Rate Distributions 3-16

51. Attributes Assigned at Matrix _Creation Node 3-20

52. Attributes Assigned at Opportunity_Matrix Node 3-21

53. Library_Search Example of Collection Vector Distributions 3-22

54. RADINT1 Parameters Modeled 3-26

55. IMINT Parameters C-1

56. SIGINT Parameters C-2

57. RADINT Parameters C-3

58. MASINT Parameters C-4

59. OSINT Parameters C-5

60. Counter -Intelligence Parameters C-6

61. HUMINT Parameters C-7

A KNOWLEDGE MATRIX MODELING OF THE INTELLIGENCE CYCLE

1. Introduction

1.1 *Background*

The United States Intelligence Community (IC) and the Department of Defense (DoD) did not thwart the September 11th 2001 Al Qaida terrorist attacks because of a series of intelligence failures. The 9/11 commission stated that the attacks highlighted four kinds of intelligence failures: in imagination, policy, capabilities and management (9/11 Commission Report, 2004: 339.). All four of these failures involved a central theme, lack of community cohesion and therefore a dearth of information sharing

The 9/11 Commission Report proposed that information be shared horizontally, across new networks that transcend individual agencies. A decentralized network model, the concept behind much of the information revolution, is one of the suggested solutions to horizontal integration. In the decentralized network model, agencies would still have their own databases, but those databases would be searchable across agency lines. In this system, levels of classified information could still be protected through the design of the network and an "information rights management" approach that controls access to the data, not access to the whole network." The new term used to describe this network is a "trusted information network." Since the idea of a trusted information network was

proposed, the IC and DoD have endeavored to incorporate horizontal integration into intelligence planning and operations. One key portion of horizontal integration is the complicated question of what investments should be made today to prepare for intelligence collection in the future. What intelligence architectures will improve Intelligence Community cooperation, coordination, and communication? What architecture will give the United States the best probability for successful intelligence collection?

1.2 *Problem Statement*

The National Security Space Organization (NSSO) was chartered by the Decision Support Center (DSC) Senior Steering Group (SSG) – USD(AT&L), VCJCS, and ASD(C3I), to report on the performance of multi-INT fusion. The Mulit-INT Fusion report produced illustrated the value of fusing intelligence data and modeled how imagery data might be fused to create new knowledge. This report together with the aftermath of September 11th spawned numerous questions concerning intelligence fusion. How do we attain greater fusion not only between military organizations, but across the entire Intelligence Community? How can fusion be measured? How can we use artificial intelligence to assist in fusion, and lessen some of the information overload experienced by human intelligence analysts? Finally, the two questions at the crux of this effort, does fusion exist, and if so, what intelligence architecture(s) will best support fusion and be most likely to meet end-user information needs?

1.3 *Research Objective*

The ultimate goal of this effort is to accurately model the flow of intelligence information through a multi-INT system and provide an output measure of total

satisfaction. Total satisfaction is defined as achieving a requested knowledge level within a specified timeframe. Given differing intelligence systems architectures (i.e. different sets of resources), the approximate collection distributions in a 24-hour time period, and the appropriate processing delays this model will determine which architecture will most likely meet user information needs satisfaction. The model must be both generic and flexible enough to analyze notional scenarios based on future needs estimates and future collection capabilities.

The secondary objective is to provide insight into the intelligence fusion process using a knowledge matrix methodology. The knowledge matrix method is a simple, straightforward, database management solution to the problem of modeling intelligence fusion. The method primarily uses probabilistic Monte Carlo simulation to deal with modeling knowledge and satisfaction levels. Although this work is only a model, some of the intelligence fusion concepts espoused here could be directly applied to real world intelligence processes. Compiling centralized data in generalized data categories to build automatically updating databases could assist intelligence professionals across the Intelligence Community better discover, analyze, and employ data.

1.4 *Research Focus*

Model development focuses on three areas of improvement; the creation of user-needs as knowledge matrices, the design of the collections sub-model, and the flow of knowledge matrices through the intelligence cycle. The preponderance of research concentrates on developing a methodology to model and quantify satisfaction. In pursuit of this goal the primary task was to develop a creation scheme and fusion methodology for knowledge matrices. Once the mechanics of creation and fusion were developed an

unclassified but accurate abstraction of the U.S. intelligence cycle had to be formulated. The model thus created had to be easily modified so what-if analysis and varying scenarios and parameter levels could be tested and compared.

1.5 *Methodology*

The kernel concept of the knowledge matrix method was developed in the NSSO report titled *Multi-Intelligence Fusion Performance,* and was further espoused in the RAND study, *Measuring the Value of High Level Fusion.* The knowledge matrices generated in this model are 6x6 matrices. The six rows represent information quality levels and the five columns define generalized knowledge areas commonly used in intelligence reporting. A modified tasking method was developed so that knowledge matrices could be used to model information flow. Intelligence tasking usually takes the form of Requests for Information (RFIs). The RFIs generated in this model are knowledge matrices which specify a user-needs satisfaction level within a specified time limit. Once the initial RFIs are generated, the model hands the requests off to the collections submodel.

The collections sub-module is comprised of six primary user-needs satisfaction collection submodels. A number of generic collection resource platforms supports each user-needs submodel. These platforms represent the array of intelligence collection platforms owned by agencies across the Intelligence Community. Each collection platform acts as a data source to update knowledge matrices flowing through the intelligence cycle. When collection is complete, or a pre-specified request time limit occurs, the available sensor reports are fused using a knowledge matrix fusion algorithm to create a maximum satisfaction vector. The resulting maximum satisfaction vector is

scored to determine the level of satisfaction attained. This will create a high level model of what has become known as the Intelligence, Surveillance, and Reconnaissance (ISR) System of Systems. Given the correct number of collection resource platforms, and accurate collection and processing distributions this model should adequately model the complex interactions of ISR platforms in the intelligence cycle. This work's final product will be an analysis of competing intelligence resource architectures using notional request levels, and notional collection distributions. The measures of interest for this analysis remain essentially the same as the previous work; quantity, timeliness, and user-needs satisfaction. The ultimate measure of effectiveness, Total Satisfaction, will have more fidelity in the current model than in past models due to the knowledge matrix method employed.

1.6 *Assumptions and Limitations*

Some assumptions must be established because we are working with a model of the real world. The first assumption is that this will not reflect all the intricacies of the real world system due to the level of abstraction. This model is an abstraction because it does not incorporate human intuition, initiative, or experience. Fusion by a human being includes deductions and inferences, which this model will not illustrate. A further abstraction beyond no human in the loop modeling is that this model is not a truth versus perceived truth model. Improving knowledge quality in an area will not drive improved collections capability. No information dependencies are simulated. This leads to two more assumptions, first that all RFI arrivals are independent of each other, and second that collection efforts by differing sensors are independent of each other. This model does not employ any actual sensor or collection data. The driving data in this model are

the statistical collection probabilities and process delay probabilities, which must be subjectively derived by Subject Matter Experts (SMEs). Collection probabilities and process delays are not included in this thesis because they are classified items. Also due to classification issues not all intelligence disciplines, collection platforms, or collection methodologies are included in this model. This classification issue extends to distributions used to create quality request levels, timeliness requirements, and delay times for processing data. Initially, all distributions used are notional; the thoughts and theories supporting each distribution used are noted in Chapter three, Methodology.

This model is designed to be generic so that classified analysis can be undertaken when the correct probabilities are placed in the model. It is assumed, that the model will accurately reflect real world performance when either empirical data or theoretically sound probabilities are employed.

The final assumption is that dissemination will occur as a generic time delay only, and that all data will be disseminated to the correct user without any need for further explanation. Despite its level of abstraction from reality this model could be used to facilitate an improved understanding of the intelligence cycle, its functions and responses

1.7 *Preview*

This thesis contains five chapters. This chapter, the Introduction, contains background information, a synopsis of the thesis concept, and the development of research goals. The second chapter, Literature Review, discusses the concept of data as the building blocks of knowledge; the connection between knowledge and intelligence; a brief history of the U.S. intelligence enterprise and the Intelligence Cycle, and a look at several different ways used to model and quantify knowledge in computer simulations.

Chapter three discusses the methodology used to develop the Arena model. A detailed discussion of model creation correlates how and why the model performs as it does compared to real world processes. The topics of matrix creation and how the matrices flow through the abstracted intelligence cycle is presented together with the ensuing modeling difficulties and solutions. The fourth chapter, Results and Analysis, presents the output and insights discovered from a baseline, peacetime scenario, a heightened perceived threat or wartime scenario, and a competing architectures scenario where a the decision must be made to purchase more UAV assets or a satellite upgrade. The fifth chapter, Conclusions, outlines the model strengths and weaknesses, recommends future model improvements and future theoretical research topics.

2. Literature Review

> Intelligence is probably the least understood and most misrepresented of the professions." "Your successes are unheralded, your failures are trumpeted. For obviously you cannot tell of operations that go along well. Those that go badly generally speak for themselves."
>
> **President John F. Kennedy, November 28 1961, inauguration of the new CIA Headquarters building and retirement of Allen Dulles as CIA director**

2.1 *Introduction*

To accurately model, any process one must obviously understand the system mechanics; how objects flow through a system and how processes relate to each other. However, understanding only the mechanical aspects of a problem is often shallow knowledge, which may reveal only the initial symptoms of a deeper problem. To understand the deeper issues and solve the true problems requires an understanding of the events that led up to the current system state. Learning the history of a system can provide enhanced insights, which can lead to system improvements and a solution to root problems instead of a series of cosmetic fixes. In the following section we seek to understand the nature of intelligence by: defining intelligence, delving into the discovery of information and data, the creation of knowledge; realizing the value of intelligence, reviewing the formation of the U.S. Intelligence Community; discussing the intelligence cycle and intelligence disciplines; and finally investigating some current methodology for quantifying and modeling the value of knowledge and hence intelligence. Therefore, the initial issue is naturally, what is intelligence?

2.2 *Defining Intelligence*

What is intelligence? What does it supposed to do? Many intelligence professionals have struggled to accurately answer these questions and define intelligence. Dr. Michael Warner of the CIA History Staff researched this question extensively and compiled a comprehensive list of intelligence definitions applied by credible organizations and individuals. The culmination of his work concludes in this definition, "Intelligence is secret, state activity to understand or influence foreign entities." (Warner, 2002: 7) This definition is curious because it handily avoids a keyword noted in almost all other definitions of intelligence. Dr. Warner's definition omits the word "knowledge". The majority of definitions he listed himself incorporate the concept of knowledge into intelligence. For example, the Department of Defense defines intelligence as "Knowledge of an enemy's capabilities and intentions" (JP 2-0, 2000: V), and the CIA defines intelligence as "Knowledge and foreknowledge of the world around us – the prelude to decision and action by US policymakers." (Warner, 2002: 2). Dr. Warner avoids using the term knowledge by substituting the phrase "understand foreign entities." His avoidance of the term knowledge may be his way of avoiding a fairly ambiguous and passive word. His choice of words implies action. The words understand and influence, describe intelligence in an active manner, and intelligence is, as we shall see, a very active process. However, the actions implicit in the word understand are the actions of gathering information, analyzing or thinking about the information, and then making inferences or deductions based on the pool of information presented. The "understanding" process is essentially the creation of knowledge from data, for that is what intelligence produces, knowledge. Understanding and knowing are synonymous

states of being therefore, for the purposes of this thesis we shall assume that intelligence seeks to gain knowledge. As previously mentioned the term knowledge is ambiguous and requires further elucidation.

2.3 *Knowledge and Intelligence*

Knowledge is a non-physical product, the result of complex combinations of cognitive algorithms unique to every human brain. The word knowledge has numerous shades of meaning dependent upon and individual's experiences. To give some frame of reference to this discussion of knowledge several definitions are presented (Dictionary.com:20 Aug 2004).

> Knowledge is information associated with rules which allow inferences to be drawn automatically so that the information can be employed for useful purposes.
> www.seanet.com/~daveg/glossary.htm
>
> The information context; understanding the significance of information.
> www.cio.gov.bc.ca/other/daf/IRM_Glossary.htm
>
> Information defines facts (A is B). Knowledge defines what one should do if certain facts apply. Thus, if A is B, then do C. There are many different ways knowledge can be encoded, but policies and business rules are popular formats.
> www.bptrends.com/resources_glossary.cfm
>
> Knowledge is part of the hierarchy made up of data, information and knowledge. Data are raw facts. Information is data with context and perspective. Knowledge is information with guidance for action based upon insight and experience.
> www.itilpeople.com/Glossary/Glossary_k.htm

We selected these definitions because they all indicate that knowledge consists of information, and that information is bits of specific data. The data to knowledge concept is well documented in many knowledge management texts. As computing power has improved so too has our ability to organize and access data. A number of knowledge management articles, books, and collegiate texts have come to the consensus that raw data "fused" in some fashion can eventually become new knowledge (Waltz, 2003: 3).

When used in the Intelligence arena we are most often interested knowledge about the nature of the physical world. We want spatial knowledge, where a physical object is located, and temporal knowledge, when an event/interaction between physical objects occurred.

2.4 *The Nature of Information*

Information in this thesis refers to items of intelligence value, usually a collection of related data that forms some kind of information. We can break down the term "items of intelligence value" into four distinct information categories; known facts, secrets, disinformation, and mysteries. (Berkowitz; 1989: 86)

A known fact is information open to discovery given the proper equipment or technology. The cost to discover this information may be prohibitive, but the information is available in the open given the proper resources. Known facts are not necessarily common knowledge. Most all people know that bombs exist, but not all people know how to make a bomb. If one wanted to build a bomb, the information could be found and applied because the ingredients needed to make a common bomb are known facts. Most people just choose not to know this piece of information. Not all facts can be discovered by open means. Some facts are protected by nation-states, or hidden by individuals as secrets.

Secrets, involve concealing or protecting information to some degree (Berkowitz, 1989: 88). We all know that nuclear bombs exist, but most people and even many nation-states do not know how to build a nuclear bomb. This is because of the protections placed upon this secret information. Because it can be difficult to obtain secret information, knowledge of secret information is often incomplete. This lack of

information is known as an intelligence gap. Filling intelligence gaps is one of the major challenges intelligence agencies face. Dealing with an intelligence gap creates uncertainty. We try to quantify our lack of information by making probabilistic estimates. One of the methods used to quantify uncertainty in intelligence was the Kent scale developed by Sherman Kent and Allen Foster Dulles, the first two Directors of the Central Intelligence Agency (DCI) (Kent, 1964: 4-5). The Kent scale was created because of a miscommunication between intelligence professionals and decision makers over the Invasion of Yugoslavia in 1951. An intelligence report, called a National Intelligence Estimate (NIE) at the time, concluded with the phrase "We believe that the extent of Satellite military and propaganda preparations indicates that an attack on Yugoslavia in 1951 should be considered a serious possibility." Speaking with some colleagues who read the report, Kent was shocked to discover that some readers had estimated the odds of invasion as low as a 20% chance of occurring. Kent himself felt that the chances of invasion were sitting at least 65% in favor of an invasion occurring.

Realizing that this issue of quantifying uncertainty would continually crop up he created a table (the Kent Scale), which broke uncertainty up into five different levels; Almost Certain (90% certainty), Probable (75% certainty), Chances are Even (50% certain), Probably Not (30% certainty), Almost certainly not (7% certainty). (Kent, 1964: 2-5) This chart, and its fuzzy quantification scheme, has been used by many intelligence agencies since its inception. The chart is not an exact measure but it does serve to put some kind of quantification or bound on the amount of uncertainty inherent in the intelligence gap being scrutinized. Here another problem becomes apparent, what

happens when information is deliberately tampered with to increase our level of uncertainty? What happens when we discover disinformation?

Disinformation is an active attempt to deceive or mislead intelligence collectors and hence distract intelligence resources and decision makers from the adversary's true actions/intentions (Berkowitz, 1989: 96). This type of information, when accepted as truth, can degrade knowledge by falsely increasing the certainty of an incorrect solution. Statisticians term the probability of accepting something as true that is actually false as a Type I error, or alpha error. Because statistics can be considered the science of extracting information from data it may make sense to view this type of data in a statistical context (Ramesh, 1998: 1). Adversaries wishing to distract or confuse a decision maker want to increase the chance that a Type I error is committed. Many misinformation techniques are used to create type I errors and deceive the unwary. Some commonly noted techniques are radio spoofing, visual decoys and double agents, but many more advanced techniques can be used as well. One famous and successful disinformation campaign was Operation Mincemeat, the man who never was, performed by the British Secret Service during WWII. (Berkowitz; 1989: 96). A corpse, dressed as a Royal Marine Captain with a number of falsified secret documents, was purposely released from a sub off the coast of Spain. The false documents stated that the current plan was to fool the German's into thinking that Sicily would be invaded next and not some other unnamed location. The German's not wanting to be fooled, redeployed units from Sicily to the island they thought would be invaded, Corsica (Berkowitz, 1989: 96). This ruse undoubtedly made the actual invasion of Sicily a bit simpler for the Allies. Intelligence professionals must

continually be wary that data discovered may be misinformation data (Berkowitz, 1989: 86).

The final information type that intelligence must deal with is the mystery. (Berkowitz, 1989: 103) Mysterious data can take two forms, paradox and uniform certainty. Paradoxes are circular logic, or impossible outcomes (i.e. 1+1 = 4). Paradoxes often appear as a piece of data or information that is anomalous but seems significant. This type of mysterious data forces intelligence agencies to ask more questions and search for corroborating data. If the data is corroborated but still cannot be put into proper context it is termed a mystery and a determination must be made if the information gap created by the mystery is worth the time and effort to pursue.

The second form of mystery, uniform certainty, is one where all the probabilities, or facts are known, but no answer is possible to extrapolate from the data. An example of this kind of mystery is the "voter's paradox" (Berkowitz 1989: 103). There are three different courses of action (COA) possible. Decision Maker#1 favors COA A, then COA B, and finally COA C. Decision Maker #2 favors COA B, then COA C, and finally COA A. Lastly Decision Maker #3 favors COA C, then COA A, and finally COA B. If all three Decision Makers have an equal vote in what COA will be taken then the sum of the three pieces of data negate each other and we are left with zero knowledge. We only know that there are three courses of action and that each has a uniform 33% chance of occurring. This uniform certainty is actually maximum uncertainty given a problem where all factors are known. Uncertainty is the antithesis of knowledge and an anathema to intelligence. The problem is that uncertainty abounds. It is ubiquitously present in

almost all intelligence problems because it is found at the very base of the knowledge chain, at the data level.

2.5 *The Nature of Data*

Because both knowledge and information can be broken down into data we must understand the nature of data. Data could be a spreadsheet of numbers, or a single number. It could be a string of textual characters (i.e. a language), or a single character. It could be any symbol to which meaning or value is assigned. Data has a multitude of forms and methods of transfer from oral, to print, to electronic. Shannon's information theory breaks data down into bytes or bits. These bits do not have to "mean" anything in particular. They can simply be a string of numbers whose only value or meaning is that they take up space on a computers hard drive. For the purposes of this thesis we seek a different form of data. We seek data that encodes or implies some meaning.

Unfortunately, data is often not readily available. Intelligence entities must seek out or collect data. In the course of collecting data four challenges are typically encountered. (Zack, 2004: 862)

1. *Uncertainty*: not having enough data/information

2. *Complexity*: having more data/information than one can easily process (i.e. information overload)

3. *Ambiguity*: not having a conceptual framework for interpreting data/information

4. *Equivocality*: having several competing or contradictory conceptual frameworks

The first challenge to overcome, uncertainty, we talked about at length in the nature of information section. We can say that we are uncertain about a situation when we do not know all of the factors affecting that situation. If all factors were known then making a

decision would be much simpler. In the IC great leaps have been made to overcome uncertainty. We now have platforms that collect tremendous amounts of data. Some would say that too much data is collected and nothing is done with it. This improved collection ability has not translated into improved understanding of the battlespace. We are overcoming data uncertainty, but we are now encountering the second challenge of data collection, complexity.

Complexity deals with the effective management of collected data. We now have more data than can be efficiently analyzed and correlated by the human resources currently available. Former Deputy Director of Intelligence (DDI) for the CIA, Jack Davis (1992:35) discussed this issue of data complexity at length. "The human mind is the most creative analytic tool in existence, but it reaches limits in three areas: the volume of information it can store, the number of variables that can be brought to bear on a problem coherently, and the ability to track the consequences of the whole set of variables in one of the factors under consideration." The current amount of data that can be collected confounds the ability of the human mind to rationally cope with it. Nobel Laureate Herbert Simon coined the term, information overload, in the mid 1950s.

> What information consumes is rather obvious: it consumes the attention of its recipients. Hence, a wealth of information creates a poverty of attention, and a need to allocate that attention efficiently among the overabundance of information sources that might consume it. (Tiwana, 2000: 55)

The latest military lingo for this deluge of data is analysis paralysis. We are paralyzed by too much data. That in itself is an issue, but we also become further paralyzed by the next issue, equivocaltiy. Equivocality means that we have conflicting data. One report states that troops are in-garrison at a certain time and another report states that they are out of garrison at the same time. One expert states that there is only a 30% chance of a

drought in a country this year while a second expert says that there is a 60% chance of a drought? Which data or probability is correct? Equivocality of data has become an area of great concern in the field of sensor fusion and as we gather more data the chances of receiving conflicting or erroneous data increases, but at least we understand what choices are when we have conflicting data. At least we have a choice of directions or avenues of inquiry to explore. What happens when we have data which may be significant, but we have not idea of it's meaning. We then face the problem of ambiguity.

Ambiguity means that we have some data, but we don't know what it means. For example, we may collect a signal that literally translates to "The fat man has a red lollipop." We have the transmission, we have the translation, but we do not have the meaning of the message. We do not have the necessary knowledge to understand the message's importance therefore the message (e.g. the data) is ambiguous. Both ambiguous data and equivocal data can fall into the information category of a mystery, which we may not have the time or resources to solve.

If data is the foundation of knowledge then these four data challenges are the cracks in our knowledge foundation. The cracks may not all stem from uncertainty, but a high degree of uncertainty does seem to simulate them. When we have a high degree of uncertainty we begin to collect more data in order to reduce our uncertainty about an outcome. Our attempts to mitigate our uncertainty have led us to collect massive amounts of data. This massive collection of data is not clearly organized or effectively managed. Due to this state of affairs, a complexity issue has been created. We have too much data and we do not know what to do with it all. This massive data collection effort has also inadvertently increased equivocality and ambiguity issues that we face. The

more data we have the more chance we have of that data conflicting or being just plain not understandable. Collecting data is not a bad thing in and of itself, but the real issue here is collecting the "right data". What is the right data? How do we identity it and how do we discover just the right data without being inundated with all the other interesting but irrelevant intelligence data? One answer to this question has been espoused for quite some time in the field of knowledge management.

2.6 *Knowledge Management*

Knowledge management texts have attempted to deal with uncertainty in knowledge by dividing knowledge into two categories, explicit and tacit. (Waltz, 20003: 63) Explicit knowledge is literally "book learning", knowledge that is written down, codified, scientific, and logical. Some examples would be, calculus, geometry, algebra, statistics, physics, orbital mechanics, geodesy, artificial intelligence just to name a few. Explicit knowledge is made up of known facts that can be measured and are concretely definable. Because it can be codified mathematically, explicit knowledge can be automated and transferred electronically (Waltz; 2003: 63). Due to the ease of automation, infallible logic, and completely researched nature of explicit knowledge it will be the primary type of knowledge and data presented in this thesis.

Tacit knowledge on the other hand is instinctual or intuitive. It is often described as a gut feeling, which is not easily described, or written down. It is learned by experience, and it is inherently uncertain. This type of knowledge is rife with uncertainty, but it is accepted as valid knowledge because the followers and leaders who adhere to the loose guidelines established by this type of knowledge appear to be successful. This knowledge appears to support correct decision making more than 50%

of the time so the uncertainties inherent in this knowledge do not overwhelm the value inherent in the knowledge. The real question with tacit knowledge is how unknowable is it? Is tacit knowledge really unknowable, or can tacit knowledge be made explicit by some concerted effort?

A RAND study conducted in the early seventies contended that not only should data and knowledge be explicit, but the questions or requests for information asked of intelligence sources should be explicit as well. The study, *Quantifying Uncertainty Into Numerical Probabilities for The Reporting of Intelligence*, noted that the statements passed between the intelligence system and decision maker may be divided into two categories: confirmable and non-confirmable (essentially explicit and tacit). A confirmable or explicit statement is one that can be judged as true, or false by any reasonable person, given that all the facts regarding the statement are known (Brown, 1973: 1). An example of an explicit question might be, how many were tanks are present at the An Najaf Republican Guard Barracks at noon today? The question is posed so that the information is attainable and confirmable. This question has a definite, finite answer. An example of a non-explicit question would be, between this guard unit and an Iranian Badr Corp tank unit which one would win a tank battle? Probabilities could be postulated, but actual ground truth might never be known. Intelligence units are often asked to answer or at least opine an answer to this type of non-confirmable question. The essential problem with these types of opinion questions is that too many variables are involved. There are too many interactions between variables, and too much variance within those variables and interactions. Any answer to this question would contain some inherent degree of uncertainty. The study noted that many superficially non-confirmable

statements are simply shorthand expressions for a bundle of confirmable statements. (Brown, 1973: 1). There are a number of confirmable questions that could be formulated regarding the tank battle. Which unit has more tanks? Which unit has trained more in the last year? Which unit has more modernized or newer tanks? Which tank type has a greater range? Which tank type has thicker armor or reactive armor? Which tanks' firepower is more penetrating? All of these questions are confirmable, and will give some indications as to who would most likely win the proposed scenario, but the answer would still be highly uncertain. If explicit questions are asked then explicit data can be accrued and exploited to create explicit knowledge. Essentially, it boils down to first asking the right questions. Once the right questions are asked intelligence assets can then review and/or collect the available pertinent data. The data can then be fused into information (i.e. groups of facts, which imply meanings) and eventually the information groups will form a larger picture, a picture of the true situation. This picture of truth is the spatial, and temporal knowledge intelligence seeks and it is the first of two intelligence goals.

The second intelligence goal beyond a picture of truth is the CIA's concept of foreknowledge, the ability to predict future situations based on inferences and deductions made concerning the current perceived truth. This foreknowledge goal deals with non-physical knowledge of intention, capability, and will. Because this type of knowledge is seldom available as factual data, we must postulate it using probabilities.

Ultimately these two intelligence products, a true picture of current events (based on data) and a prediction of the future picture (based on probabilities) have been valued

because they allow decision makers to act upon facts and constructive reasoning, not fears.

2.7 *The Value of Knowledge and Intelligence*

We cannot touch knowledge; we cannot feel knowledge, but we can feel or experience the effects of knowledge. Leaders throughout history have experienced the effects of knowledge first hand. Whether the situation has been tactical, or strategic, militaristic, economic or diplomatic the party with superior knowledge has held a recognized advantage over their adversaries. Using intelligence to gain knowledge and hopefully some kind of advantage is historically well documented. Two examples oft cited by intelligence academics are the writings of Sun Tzu and the Bible.

In 480 B.C., Sun Tzu wrote a chapter in his *Art of War* treatise called *The Use of Spies*. He noted that, "Foreknowledge cannot be elicited from spirits. It cannot be obtained inductively from experience, nor by any deductive calculation. It must be obtained from men who know the enemy situation" (Rudnicki, 1996: 86). Sun Tzu highly recommended the use of spies, the only source of intelligence available at the time of his writing. Another prime example of intelligence activity is in the bible story of Moses leading the Israelites through the wilderness. God told Moses to send one ruler from each of the twelve tribes of Israel to explore Canaan. He said, "Go up through the Negev and on into the hill country. See what the land is like and whether the people who live there are strong or weak, few or many. What kind of land do they live in? Is it good or bad? What kind of towns do they live in? Are they unwalled or fortified? How is the soil? Is it fertile or poor? Are there trees on it or not? Do your best to bring back some of the fruit of the land." (Numbers 13:17-20, NIV). Using human intelligence or any

other sort of intelligence to gain an advantage over an opponent is commonly seen throughout history. A plethora of successful intelligence exploits can be seen throughout the course of history. Some examples are: the Greek victory at Thermopylae to the Venetian embassies expanding merchant trade; the British defeat of the Spanish Armada; the decryption of the German Enigma machine by mathematicians at Bletchley Park; and the decoding of Japanese messages under the code name "Magic" during World War II to mention a few successful intelligence endeavors. These examples and many others like them support the notion that intelligence can be invaluable to the leader who knows how to use it.

Therefore, knowledge is valuable. How valuable? We have specific historical instances that highlight intelligence as the most valuable factor in a situation, but hindsight is 20/20. Seeing intelligence in a historical context where the critical elements of a situation are known, we are able to say that the intelligence acquired was a significant contributor to the success of a battle, a war, or a national objective. Based on these past successes we continue to invest in similar current intelligence endeavors. The problem with intelligence is that as events unfold in real time, not all critical factors are known or recognized, and the decision maker must proceed to allocate resources based on intelligence as it is understood at that time. So how, in the present tense, can the value of intelligence be quantified? How much time, effort and other resources (such as human lives) should be spent to create gains in intelligence? The answer usually seems to be situation dependent and seems to fall into the area of a tacit question that requires further categorization into explicit questions. Pursuing the marginal value of intelligence is outside the scope of this research. Suffice to say that current thought indicates that

intelligence is valuable and worth the investment of considerable resources. There has been only one voice of dissent in the past three hundred years.

Carl Von Clausewitz, the well known Prussian author of the treatise *On War*, generally did not look kindly upon the intelligence enterprise. Clausewitz stated that, "A great part of the information obtained in War is contradictory, a still greater part is false, and by far the greatest part is of a doubtful character" (Clausewitz, 1982: 162). Clausewitz's derision of intelligence is no doubt a product of his era (1780-1831). Intelligence in those days was not a technological endeavor. It consisted mostly of human intelligence (HUMINT) or "soft intel" (Waltz, 2003: 6). Technology, "hard intel", has mitigated much of the data which Clausewitz opined as false and of doubtful character, but it has not yet solved the problem of conflicting or contradictory data. Interestingly a reason for this mitigation of unreliable data might be that modern technology focuses on answering only explicit questions. In order to acquire data from a non-human source an explicit question or task must be asked of a machine. It is also interesting to note that Clausewitz wrote about intelligence just as the nascent American nation, which today has risen to the forefront of intelligence collection, appeared.

2.8 *A Brief History of the American Intelligence Enterprise*

The first state sponsored American intelligence officer appears to have been Alexander Hamilton (Dulles, 1963: 29). Hamilton and his two assistants, Tallmadge and Boudinot, gathered information, performed counter-espionage activities, and developed ciphers for George Washington and the revolution (Dulles, 1963: 29). Intelligence work was not formalized during the Revolutionary War, or in any other action until the Civil War. Most intelligence networks were setup out of a general's or a politician's pocket.

At the start of the Civil War President Lincoln hired Allen Pinkerton, a private detective, to setup a makeshift intelligence organization for the Union (Dulles, 1963: 38). Lincoln made Pinkerton an army major and allowed him free reign to gather intelligence as he saw fit. Most historians agree that Pinkerton's organization was ineffective and Pinkerton himself most likely realized this as he resigned before the war ended. The Lincoln administration decided to formalize military intelligence at this time by forming the Bureau for Military Information headed by Major George H. Sharpe (Dulles, 1963: 39). Following the Civil War, both the army and the navy established permanent intelligence agencies. The Army setup the Military Intelligence Division and the Navy created the Office of Naval Intelligence. The Military Intelligence Division was placed under the auspices of the Army's Second Division and was given the designation G-2 (Dulles, 1963: 41). To this day Army intelligence is known as G-2, while joint military intelligence is the J-2. This formalized structure withered away after the Civil War ended because no mission was clearly defined. Because of this American forces in World War I did not have a true intelligence cadre. Most intelligence was passed along by French and English forces (Dulles, 1963: 41). World War II was the defining moment for American intelligence. Following the catastrophe at Pearl Harbor, President Franklin D. Roosevelt called on Colonel William J. Donovan to establish the Office of Strategic Services (OSS). This precursor to the Central Intelligence Agency (CIA) performed admirably on a global scale (in North Africa, Europe, the Far East) throughout the course of World War II. If not for the actions of Soviet Union following WWII this organization might too have withered away. President Truman on March 12, 1947 signed the National Security Act which unified all military service intelligence activities under the Secretary of Defense;

created a civilian intelligence service, the CIA; and established the National Security Council to advise the President on intelligence matters. With this formalization of intelligence sanctioned by the President the structure of the current intelligence community was founded.

2.9 *The U.S. Intelligence Community and the Intelligence Cycle*

The U.S. Intelligence Community has come a long way since 1947. After President Truman created a formal organizational structure the federal government began funding intelligence in earnest. The resultant intelligence community is universally recognized as one of the largest and most technologically advanced in the world. A list of Intelligence Community Members, as outlined in the 9/11 Commission Report follows. (9/11 Commission Report, 2004: 405-406)

Members of the U.S. Intelligence Community:

- Office of the director of Central Intelligence, which includes the Office of the Deputy Director of Central Intelligence for Community Management, the Community Management Staff, the Terrorism Threat Interrogation Center, the National Intelligence Council and other community offices
- The Central Intelligence Agency (CIA), which performs human source collection, all-source analysis, and advanced science and technology

National Intelligence Agencies:

- National Security Agency (NSA), which performs signals collection and analysis
- National Geospatial-Intelligence Agency (NGA), which performs imagery collection and analysis
- National Reconnaissance Office (NRO), which develops, acquires, and launches space systems for intelligence collection
- Other National reconnaissance programs

Departmental Intelligence Agencies:

- Defense Intelligence Agency (DIA), of the Department of Defense
- Intelligence entities of the Army, Navy, Air Force, and Marines
- Bureau of Intelligence and Research (INR) of the Department of State
- Office of Terrorism and Finance Intelligence of the Department of Treasury

- Office of Intelligence and the Counterterrorism and Counterintelligence Divisions of the Federal Bureau of Investigation of the Department of Justice
- Office of Intelligence of the Department of Energy
- Directorate of Information Analysis and Infrastructure Protection (IAIP)
- Directorate of Coast Guard Intelligence of the Department of Homeland Security

As the intelligence community became formalized, so to did the processes and practices followed by intelligence professionals. Over time a generalized process model known as the Intelligence Cycle became accepted by the various military and civilian intelligence enterprises. Figure 2-1 displays the Intelligence Cycle in its current form.

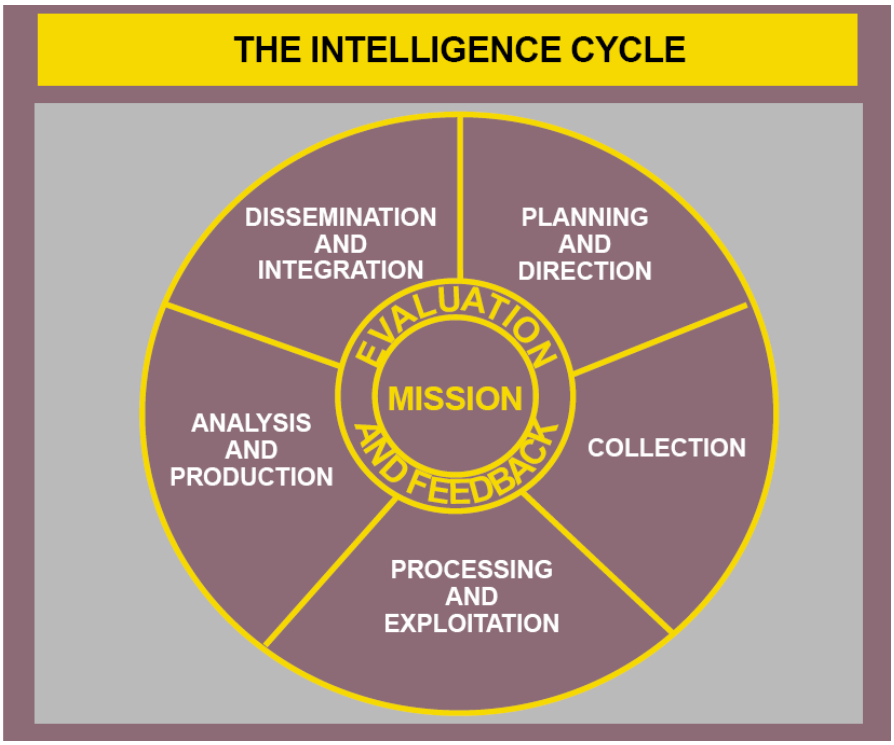

Figure 2-1 Intelligence Cycle (JP 2-0, 2000:V)

The Intelligence Cycle of Figure 2-1 is a broad conceptual model, which outlines an ideal information flow between the entities that makeup the Intelligence Community. The Intelligence Cycle displayed illustrates the operational mission at the model core. This shows that the operational mission is always central. Actual political or military missions are the reason that intelligence is gathered in the first place. This model

recognizes that intelligence is an operational support function, also known as a force multiplier. The next sub-central ring, evaluation and feedback used to be a sixth sector of the intelligence cycle pie, but was modified as a second ring in the late 1980's when Total Quality Management (TQM) revealed to the world that process improvement never stops. To represent the continual process improvement efforts that occur at every step of the intelligence cycle this ring was added. The five outer sectors are the true meat of the intelligence cycle. They outline the five basic cycle steps which structure the gathering of data and creation of knowledge in the U.S. Intelligence Community.

The cycle is generally begins in the Planning and Direction sector. In Planning and Direction, the decision makers and information end-users request information, identify information gaps, and generally define items of valuable intelligence. The process of planning and directing is continually evolving as the world situation evolves. However, the physical product of this planning is an annually reviewed and stratified list of valuable intelligence items called the Essential Elements of Information (EEI's). This master EEI list keeps the Intelligence Community in loose coordination to fulfill the listed desires of intelligence users. Given this list of EEI's the agencies of the IC's then go forth to collect data concerning these EEI's to lessen our uncertainty concerning them.

Intelligence collection, the next sector in this model, refers to data collection. The gathering of data involves the use of open sources, sensors, and spies to discover answers to specific questions or to monitor events. In the framework of intelligence collection there are currently seven recognized Intelligence disciplines illustrated in Table 2-1. Each discipline focuses on a different genre of intelligence collection. For example, IMINT deals primarily with obtaining visual evidence while SIGINT is concerned with

entity communications, any form of message sent between entities of interest. Each of the intelligence disciplines listed have a variety of sensors and collection techniques available to them. No one entity in the IC has total control over any one intelligence discipline. The Intelligence Cycle model assumes at this point that intelligence faces an uncertainty problem that can be solved by gathering more data. If the EEI in question does not require more data collection then this phase as well as the Processing and Exploitation can be omitted.

Table 2-1 Intelligence Disciplines (JP 2-0, 2000: 25)

INTELLIGENCE DISCIPLINES	
IMINT	**Imagery Intelligence**
HUMINT	**Human Intelligence**
SIGINT	**Signals Intelligence**
MASINT	**Measurement and Signature Intelligence**
OSINT	**Open-Source Intelligence**
TECHINT	**Technical Intelligence**
CI	**Counterintelligence**

Processing and Exploitation deal with the physical processes that must occur for a piece of data to be realized as intelligence information. The photographs must be developed and possibly annotated to highlight significant details that an untrained eye might miss. A signal report must be filed and its contents translated or deciphered. Once the necessary preparation has been completed and the data has notionally been converted into information the next process can begin, Analysis and Production.

Analysis and Production is the phase of the intelligence cycle where knowledge is created. It occurs at a fairly macro level and involves the fusing of various pieces of intelligence information available from the seven intelligence disciplines. Here in the

piecemeal jigsaw puzzle of facts, where intelligence disciplines attempt to compete instead of corroborate to show their worth, we see disjointed bits of the truth which leave decision makers and information users wondering about the effectiveness of intelligence as a whole. These battles over who has the actual true picture of a situation have been termed theologic wars. A poem by John Godfrey Saxe called the Hindu Parable illustrates this idea of a theologic war that can occur between intelligence collection agencies. (Saxe, 1880: 1)

It was six men of Indostan
To learning much inclined,
Who went to see the Elephant
(Though all of them were blind),
That each by observation
Might satisfy his mind.

The First approached the Elephant,
And happening to fall
Against his broad and sturdy side,
At once began to bawl:
"God bless me! but the Elephant
Is very like a wall!"

The Second, feeling of the tusk
Cried, "Ho! what have we here,
So very round and smooth and sharp?
To me 'tis mighty clear
This wonder of an Elephant
Is very like a spear!"

The Third approached the animal,
And happening to take
The squirming trunk within his hands,
Thus boldly up he spake:
"I see," quoth he, "the Elephant
Is very like a snake!"

The Fourth reached out an eager hand,
And felt about the knee:
"What most this wondrous beast is like
Is mighty plain," quoth he'"Tis clear enough the Elephant
Is very like a tree!

The Fifth, who chanced to touch the ear,
Said: "E'en the blindest man
Can tell what this resembles most;
Deny the fact who can,
This marvel of an Elephant
Is very like a fan!"

The Sixth no sooner had begun
About the beast to grope,
Than, seizing on the swinging tail
That fell within his scope.
"I see," quoth he, "the Elephant
Is very like a rope!"

And so these men of Indostan
Disputed loud and long,
Each in his own opinion
Exceeding stiff and strong
Though each was partly in the right,
And all were in the wrong!

Moral:

So oft in theologic wars,
The disputants, I ween,
Rail on in utter ignorance
Of what each other mean,
And prate about an Elephant
Not one of them has seen

The IC is attempting to rid itself of these theologic wars and concentrate on fusing data rather than fighting over whose intelligence is better. Both in initial collection and in the Analysis and Production phase more emphasis is being placed on fusing data.

"Fusion is the synergistic process of associating, correlating, and combining

Hostile, Friendly, and Neutral Forces data and environment factors to derive information and knowledge, tailorable to support the warfighter to effect and expedite command and control." (Keithley, 2000:2) Table 2-2 outlines the current definitions of the five fusion levels

Table 2-2 Fusion Level Definitions (JDL, 2004:1)

Fusion Levels Defined		
Level	**Title**	**Description**
0	Source Preprocessing	Normalize, order, compress, merge sensor data
1	Object/Entity Refinement	Refine position, track, identify objects by fusing individual sensor position & identity estimates
2	Situation Refinement	Interpret Relationship between objects & events: Order of Battle(OB), Common Operating Picture (COP), Situational Awareness (SA)
3	Threat Refinement	Estimate enemy capability & Intent; Predictive BattleSpace Awareness (PBA)
4	Process Refinement	Refine estimates, optimize ISR resources modify fusion processes to improve information
5	User Refinement	User visualization of fusion products & generate feedback/control to improve products

(JDL, Joint Directors Laboratories, Extract 2004, 1)

Data fusion uses deductive reasoning to correlate incoming reports with existing knowledge to improve the level of knowledge within each specified knowledge type (Waltz, 2003: 280). In Level 1 fusion the sensor/source reports are grouped geographically and temporally (Waltz, 2003: 280). This type of fusion is a mapping from

a database into the real world and some might call it a four dimensional mapping. This mapping or transformation creates a common time-space coordinate system to detect an association between multiple sensors about a common object (Waltz, 2003: 281). The associations will theoretically increase the current level of knowledge concerning the target object. The correlation metric, C, is: $C = \sum_{i1=1}^{n} w_i x_i$ where, w_i = weighting coefficient for attribute x_i , and x_i = ith correlation attribute metric (Waltz, 2003: 281). Level 1 fusion is present in the current intelligence cycle, but it is not distribute throughout the IC, but rather stove-piped by INT. Level 1 fusion decreases the workload on human analysts by auto-correlating databases into this real world mapping. The next step is correlating objects and/or events to each other.

Level 2 fusion associates objects with one another, and is currently mostly manual (Keithley, 2000: 3). A good example of this might be a terrorist network that is geographically separated but can still communicate via cellular phones, websites, and intermediaries. In this case geographic proximity would not show any correlation, time may also be an elusive correlation. This means that the current level 1 fusion method is not sufficient. A new method of correlating message content will have to be invented. Current real world fusion engines are used in the Defense Information Infrastructure (DII) Common Operating Environment (COE) in combination with the Modernized Integrated Database (MIDB). Only low level fusion, Level 1, is currently automated. Level 2 automation is underway, but the higher levels of fusion are still entirely in the realm of research and development.

Once the pieces of information are fused into knowledge, theologic wars aside, that knowledge must be given to the proper authorities so that decisions can be implemented. This is the Dissemination and Integration phase of the intelligence cycle. In this phase knowledge gained does not necessarily go directly back to the initial requestor. Knowledge must be given to the authority that can use it to best effect. For example, let us assume that a naval unit discovers some data that leads them to the knowledge that a land attack will occur against an Army post. That knowledge must flow to the Army. Then, through the dissemination chain, the Navy knowledge should flow back to the Commander of the Army post, not just back to the Captain of the ship or the Admiral in charge of the Fleet. Proper and timely dissemination of knowledge and the integration of that knowledge into a decision, or action validates the value of intelligence. We could collect all the data in the world, but if we do not use it; what was the point in the first place? The Intelligence Cycle provides a framework for discovering and moving data to create knowledge, but it does not measure or quantify that knowledge.

2.10 *Quantifying and Modeling Intelligence and Knowledge*

The short answer to the question of intelligence quantification is that no one, non-subjective methodology is used to measure intelligence. The most commonly used measure of intelligence, as we briefly discussed, is the Kent Scale. The scale is really a five-tiered stratification of uncertainty based on the consensus of a group of intelligence professionals. It is extremely subjective, does not lend itself to automation or a mathematical methodology, and it is inefficiently time consuming. Its only saving grace is that it seems to work and it is simple enough to be universally understood.

To delve more deeply into the question of quantifying intelligence and attempt to develop a less subjective quantification scheme, we must break down intelligence by asking some explicit questions. First, if we are to measure the value of knowledge, what is our base line, our starting point? Do we assume that we know nothing and start at zero?

This begs the question can knowledge be measured on a scale from zero to infinity? Can knowledge be bounded? Is zero a proper lower bound, or can we know less than nothing? When is knowledge considered 100% totally accounted for, understood, or "known? Can we assume that a certain amount of knowledge is enough, and once that level of knowledge is attained we are close enough, and have reached a near enough approximation of 100%? Does intelligence have diminishing amounts of return as we gather more of it? Can we say at some point that the cost of further gains is not worth the effort or resources required? All these questions and many more regarding quantifying knowledge are actually questions of probability because probability as we stated before is a mathematical form of representing knowledge.

To illustrate the concept that probability is a mathematical form of knowledge let's used the oft example of Polya's urn. Say that we know there are a total of five blue balls, three red balls, and one urn. If the balls are placed in the urn and we are given a chance to pick one ball out of that urn then we can postulate that we have a 5/8 chance of getting a blue ball and a 3/8 chance of getting a red ball. Those probabilities represent knowledge. If we pull a blue ball then our knowledge has changed, we know now that there are only four blue balls left and three red. Note that our level of knowledge hasn't improved. We still know and understand all the variables in the game. Having a simple

mathematically linear rule set and full knowledge of all variables means that we have 100% knowledge of our system. We have the true full picture of this game at any one point in time. Intelligence rarely has a rule set this simple, and all variables are almost never known. This means that probabilistic knowledge is difficult to attain, but it maybe the only mathematically tractable way to model knowledge. Even this mathematical tractability is quickly challenged by two simple changes to Polya's Urn game. Let's add another urn, and a random distribution of balls between the two urns. We still have eight balls total, five blue and three red, now however each of the two urns may have no balls, all the balls, or a proportion in between. Now when we pick an urn we do not know what our true chance of picking a blue ball will be. We can postulate that the chance of picking a blue ball is still higher because more blue balls are present, but our uncertainty level has increased. The problem with statistical knowledge and in particular the knowledge modeling done based on Bayes Rule (which is what we were using when we determined our probabilities in the Polya's Urn example), is an assumption of prior knowledge. In the Polya's Urn example the initial number of balls is given, eight. Then the number of blue and red balls are given, five and three respectively. With all this given data a problem is easily solved and probabilities can be determined. In the real world intelligence problem these given values are not easily discovered. The opposing force strength is often unknown and at best is estimated by Intelligence Preparation of the Battlespace (IPB). Despite the uncertainty present in modeling intelligence/knowledge based on a set of estimated points this seems to be the primary direction of all modeling and quantification efforts to date.

While no universally used methods for quantifying intelligence exist, there are four generally accepted theories for fusing knowledge have been postulated. The four theories are: Bayesian Networks, Information Theory, Boolean Logic, and Evidential Reasoning**.** (Kraiman, 2001: 2-3) All four theories assume a base starting knowledge of zero. Nothing is assumed either known. We start perfectly non-biased. All assume a linear approach to increasing the quantity of knowledge over time. All assume that knowledge can be measured on a scale from zero to one hundred, from no knowledge to total knowledge. Bayes' Rule is commonly stated as

$$P(B_j \mid A) = \frac{P(A/B_j)P(B_j)}{\sum_{i=1}^{k} P(A \mid B_i)P(B_i)}$$

We assume (B_1,B_2,…B_k) are partitions of the set of all possible outcomes, S, into k separate spaces and $P(B_i)>0$ for i=1,2,…k. (Wackerly, 2002: p68). The assumption that is commonly made in many intelligence models is that the total number of enemy is known. It the enemy strength is known then the probability of enemy troops being in an area is the probability of A given B (P(A|B)), and the probability of a sensor finding the enemy is estimated as the Probability of B. The issues here are that our calculations are based on two assumptions that may not be factual. First, we assume that our knowledge of the enemy strength is accurate, and second we assume that our probability of detecting the enemy is accurate. If both of these assumptions are true then this formulation will be accurate and provide a suitable model for quantifying the knowledge gains and fusing

intelligence. Obtaining accurate IPB and actual sensor readings is the challenge to this methodology.

Information theory quantifies knowledge using Shannon's Information Entropy Theory. Shannon's theory states that an event. a_i has a probability $p(a_i)$ of occurring. Knowing this probability the knowledge associated with the probability is $K(a_i)=-\log p_i$. Given n mutually exclusive and collectively exhaustive events, $(a_1,a_2,\ldots a_n)$, where each has a probability of occurring of $p_i=p(a_i)$, then the "average information, uncertainty or Shannon entropy" is,

$$H(p) = -\sum_{i=1}^{n} p_i \log p_i$$

Using this theory an interesting article "Modeling Knowledge in Combat Models" proposed a method of measuring the value of spatial knowledge in a combat model. The final algorithms generated concerning general knowledge K(X) and specific two dimensional spatial knowledge, K(x,y), are presented below.

$$K(X) = \frac{\ln(n+1) - H(X)}{\ln(n+1)}$$

or

$$K(X) = 1 - e^{-[\ln(n+1) - H(x)]} = 1 - \frac{e^{H(x)}}{n+1},$$

,where $H(X) = -\sum_{i=-\infty}^{\infty} \ln[f(x_i)] f(x_i)$ for the discrete case and,

$$H(X) = -\int_{-\infty}^{\infty} \ln(f(t)) f(t) dt$$

or the continuous case. (Perry, 2003; p46-54).

The two dimensional specific knowledge is modeled using the following equation:

$$K(x,y) = 1 - e^{-\ln\left[\left(\frac{\sigma}{\sigma_{\max}}\right)^2\right]} = 1 - \left(\frac{\sigma}{\sigma_{\max}}\right)^2.$$

The σ values noted relate to the Circular Error Probable (CEP) accuracy of the intelligence sensors being modeled. These algorithms were applied to the Combat Sample Generator (COSAGE). Which the Army uses to create "victim-killer scoreboards" for their Combat Effectiveness Model (CEM) and RAND also uses these scoreboards to run their Joint Integrated Contingency Model (JICM). This only one example of Shannon's entropy theory, many more exists.

The method of Evidential Reasoning is based on the Dempster-Shafer Theory of Evidence. The Dempster-Shafer Theory of Evidence takes a series of belief sets (Bel_1,…,Bel_n) and combines them if certain conditions are met. The conditions that must be met are presented in a series of three theorems, Theorem 3.1, 3.2, and 3.4, and the actual combination theorem is given as Theorem 3.3 (Shafer, 1976:57-64).

$$\text{As an extension of Dem}\, K = \left(\sum_{\substack{B \subset S \\ B \neq 0}} (-1)^{|B|+1} Q_1(B) \ldots Q_n(B) \right)^{-1}$$

pster-Shafer Theory, Alan Steinberg suggested the following formulation for stochastic simulations if two uniform sampling vectors g and h are to be fused.

$$f_j = 1-(1-g_j)(1-h_j)[1-\ln\{(1-g_j)(1-h_j)\}].$$

(Gonzales, 2003: 8). Dempster-Shafer Theory of Evidence was used in the JCOAT simulation tool as well as in a RAND Ground C4-ISR Assessment Model (GCAM)

The final method Boolean Logic unlike the other three commonly used methods does not deal in the realm of statistical probability. Boolean Logic allows for the creation of rule sets, which evaluate statements (factual or probabilistic). The statements are evaluated using up to seven different gate types; NOT, AND, OR, NOR, NAND, XOR, XNOR. The following is an explanation of each of the Boolean Logic gates mentioned (Schneeweiss, 1989: 6):

NOT – If the expression evaluated is not true then this switch is engaged, or this path is followed.

AND - This operator evaluates the truth of two or more elements in an expression. If both evaluate as true then this switch is engaged.

OR – This operator evaluates if either one or the other of two elements in an expression are true. If either is true then this switch in engaged.

NAND – This is an inversion of the AND operator. If neither of the operators in an expression evaluate as true then this switch is engaged.

NOR – This is an inversion of the OR operator. If either one or the other of the operators in an expression is false than this switch is engaged.

XOR – Known as the "exclusive or " because if either operator is true, but not both then this switch is engaged. This logic can be created with a combination of other Boolean logic, but creating this one gate is less cumbersome.

XNOR – Known as the "exclusive nor" because if one operator is false, but not both then this switch is engaged.

Using these nodal evaluations, a decision tree methodology can be established which will direct the flow of data into the correct knowledge end node. The gain in knowledge is determined by a linear increase in each end node. Another interesting note about Boolean Logic is that it can be used in conjunction with any of the three fusion

algorithms above to fuse data into knowledge once data has been sent to the correct knowledge bin.

All four fusion theories are applied in a number of combat models which incorporate intelligence into their calculations. Some Air Force specific combat models, which use one of the above methods for fusing data, are the Extended Air Defense Simulation (EADSIM), the THUNDER Air Campaign Model, and AFRL's ISR-TPED model. The primary issue with these models is that they assume that intelligence is really only spatial awareness, where an object (usually the adversary) is in space. Spatial awareness is level one fusion as previously defined in the Analysis and Production section of the Intelligence Cycle, but full fusion and full spectrum intelligence is much more than just spatial awareness. Captain Carl Pawling in conjunction with the NSSO began a new effort to evaluate the value of varying intelligence architectures. No model currently attempts to incorporate the broad range of intelligence processes into one model. Captain Pawling's work focused mostly on architecture issue of whether to Task, Process, Evaluate, Disseminate (TPED), or to Task, Process, Post, Use (TPPU) intelligence information. He modeled the TPED vs. TPPU issue as a single stream of intelligence with multiple users. In his work, he briefly mentions the possible use of a knowledge matrix methodology, but he did not pursue it.

2.11 *Summary*

The United States Intelligence Community has made tremendous advances since its inception in 1947. It continually strives to attain three goals simultaneously, a perfect picture of the past truth, present truth, and an accurate probabilistic view of the future.

Unfortunately, absolute knowledge and perfect prognostication are seldom attainable. The data collection issues of uncertainty, complexity, ambiguity, and equivocality combined with the challenges of always asking the right questions and battling internal theologic wars between intelligence sources all muddy the waters of the intelligence profession.

The right intelligence is invaluable and can be the most significant factor that contributes to a success, but there is no method currently available to quantify knowledge gained by intelligence in a non-subjective, and unbiased manner. This work attempts to put some bounds on the quantification issue using a knowledge matrix method to bind the intelligence questions into an explicit meta-data shell format, much like Captain Pawling suggests with his formatted intelligence input sheets. Once the problem is bounded the value of intelligence and fusion is tied to user-needs satisfaction. These concepts are discussed in detail in Chapter 3.

3. Methodology

3.1 *Introduction*

The model developed in this chapter is a high-level Arena process model, which simulates the flow and fusion of knowledge/intelligence through the U.S. Intelligence Cycle. The model's purpose is two-fold. First it can be used as an analysis tool to determine what intelligence architecture(s) will provide the greatest number of satisfied Requests for Information within set time constraints. The model measures of effectiveness (MOE) will be Quantity (number of requests completed), Timeliness (number of requests completed within the constraint time), and User Satisfaction (meeting a requested information satisfaction level). In addition to modeling various intelligence architectures the model will also illustrate the value of multiple source intelligence fusion using the same MOEs previously stated.

The model concept is based upon Captain Carl Pawling's 2003 thesis, "Modeling and Simulation of the Intelligence Cycle". Primarily two innovations were applied to the model concept to make it more closely simulate reality. First multiple intelligence resources were added to simulate the diverse platforms used to collect information. Second, a knowledge matrix methodology was implemented for modeling and measuring data, information, knowledge and user-needs satisfaction in the Intelligence Cycle. These two innovations considerably increased model complexity and resulted in a number of improved intelligence modeling techniques such as; Needs-driven intelligence modeling; opportunity intelligence collection vice strictly tasked collection; Standing Request for

Information (SRFI) looping logic, decision tree logic for modeling resource allocation, and a simple fusion algorithm.

As in Captain Pawling's original thesis, this model uses the Intelligence Cycle precept presented in Joint Publication 2.0, *Intelligence Support to Operations*, as an outline. The Intelligence Cycle Model is comprised of five sub-models; Planning & Guidance, Collection, Processing & Exploitation, Production & Analysis, and Dissemination, representing each phase of the Intelligence Cycle. Figure 3-1 displays a generalized flow chart of this process model. The actual model is far more complex than the figure depicted.

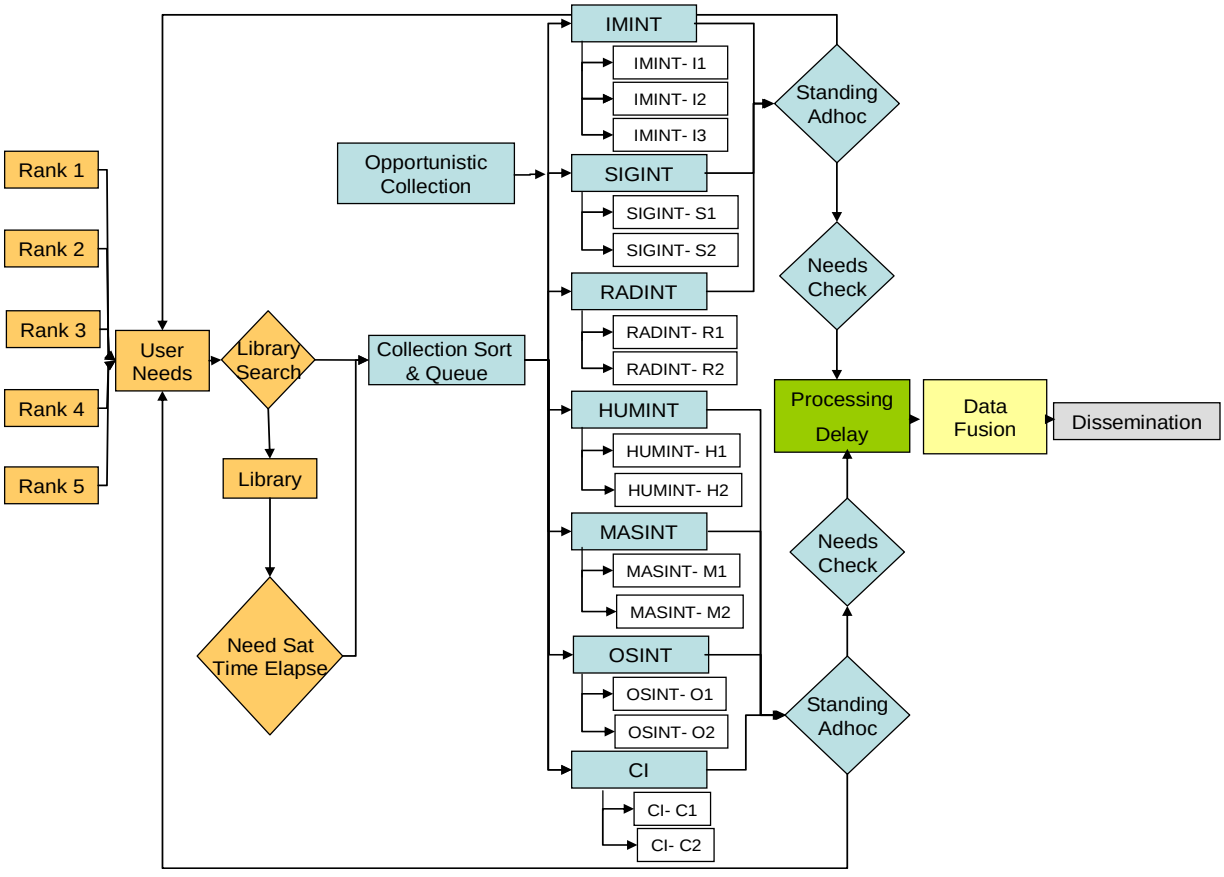

Figure 3-1 Conceptual Model Flow-Chart

The knowledge matrix method was a concept developed by the National Space Security Organization (NSSO) at the behest of the Decision Support Center (DSC) Senior Steering Group (SSG) in the spring of 2001. The methodology was presented to the DSC-SSG in the *Multi-Intelligence Fusion Performance* report in order to illustrate the value of data fusion in the intelligence enterprise. The report was based on an experiment conducted using Joint C4ISR Operations Analysis Tool (JCOAT) Campaign Model (Keithley, 2000: 7). Table 3-1, illustrates a generic knowledge matrix setup for the JCOAT model.

3.2 *Knowledge Matrix Concept*

The concept behind the knowledge matrix methodology is fairly intuitive. A knowledge matrix is essentially a state vector describing the present level of knowledge attained about an object (Keithley, 2000: 8). The columns of the matrix are generalized knowledge types while the matrix rows are increasing levels information quality for each knowledge type. The generalized knowledge types of the knowledge matrix method are: Location, Track, Identity, Activity/State, Intent, and Capability (illustrated in Table 3-1). The knowledge matrix essentially breaks down intelligence into the standard journalistic terms; who, what, where, when why, and how (i.e. who = Identity, what = Activity/State, where = Location and/or Track, when = implicit time of collection, why = Intent, how = Capability). The quality levels obtained in each of the six Knowledge areas can be collected by a variety of intelligence resources. In the original knowledge matrix each of the Knowledge Areas was broken down into six quality levels, Table 3-1 illustrates six levels of image quality from 10 Kilometers (Km) to 5 meters (m).

This initial knowledge matrix concept used in the JCOAT model focused on tactical intelligence and ISR assets. In the experiment each intelligence resource was assessed as having a certain probability of detecting at each of these levels shown above. The probabilities were normalized, based on the Central Limit Theorem, about an empirically determined mean. To apply this concept to the entire scope of Intelligence Cycle the Knowledge Matrix Methodology needed to be broadened to incorporate the ability to model other types of intelligence collection and model the flow of information throughout the entire intelligence system, not just the tactical sensors. Therefore, some conceptual adjustments were required.

Table 3-1 Original Knowledge Matrix Example

Knowledge Areas					
Location	**Track**	**Identity**	**Activity/State**	**Capability**	**Intent**
5 meters	Vectors & prediction	Specify Object & Hierarchy	Many actions, states & linkages	Many factors & influences	Desired end state & intent for future ops known
10 meters	Vectors & prediction	Specify Object	Many actions, states & several linkages	Several Factors & influences	Desired end state known & intent for future ops determined
20 meters	General Speed & direction	Type Object	Several actions, states & one linkage	Few factors and influence	Desired end state & intent for future ops determined
100 meters	Toward or Away	Distinguish Object	Few actions, states, no linkages	Few factors and no influence	Desired end state determined & intent for future ops inferable
1 Km	Stationary or not	Discriminate Object	Single action or state	One factor and no influence	Desired end state inferable & intent for future ops inferable
10 Km	Detect	Detect	Detect	Detect	Desired end state inferable & intent for future ops unknown

3.3 *Modifying the Knowledge Matrix Method*

The core knowledge matrix concept remains the same in this model. The same six abstracted knowledge areas; Location, Activity/State, Track, Identity, Intent, and Capability, are maintained. There are still six tiers associated with each knowledge area. The quintessential shift made is an abstraction of what each of these tiers represents. Instead of quantifying the quality level of a sensor, for example the 10Km to 5m for the imagery fidelity example used in the last section, the tiers now represent satisfaction.

Satisfaction is a subjective measure broken down much like the Kent Scale discussed in Chapter Two. There are six satisfaction tiers are associated with user-needs and collection vector responses. The user-needs and the collection vector responses are decomposed according to the scale shown in Table 3-2. This shift to measuring satisfaction instead of pure quality means that information applicability and not strictly sensor fidelity is our primary measure of effectiveness (MOE) for this model. An example is in order; let us assume that a request for information desires to know the country in which an individual is located. If an intelligence asset is able to produce the city where the individual is located then the user is satisfied at a level 3, according to the new information needs met scale. The country and the individual are matched as requested, not to a targeting level, but well enough to satisfy the end-user.

The actual sensor used to collect that intelligence may have a low fidelity rating (10Km for example), but the information still rates well and is usable. In other words, the resulting satisfaction may score fairly high, a 3 satisfaction for location even though the sensor report was of poor quality. Applicability does not always require a high degree

of sensor fidelity. This leads into two new concepts using knowledge matrices, creating a user-needs vector, and quantifying satisfaction.

Table 3-2 Satisfaction Level Definitions

Tier	User-need Satisfaction	Meaning to the User	Collection Vector	Meaning to the Collector
5	Excellent	Perfect Knowledge, Absolute Truth	Absolute Knowledge of a person/place/thing/event	No Uncertainty, Rarely Achieved
4	Very Good	Detailed information, with supported data	Satisfactory	Target quality, Immediate Actionable Information
3	Good	Functional information not highly detailed	Largely Satisfied	Can report and use
2	Satisfactory	Data with possible contextual issues	Meets minimum reporting standards	Data with minimal uncertainty
1	Marginal	Data with intelligence gaps, Little supporting evidence	Below Reporting threshold	Data with considerable uncertainty
0	No Need	No Need	Unsatisfactory	Cannot Report

A user-needs vector is a vector of six values each of which represent a required satisfaction in their respective knowledge areas. The user-needs vector for Table 3-3 would be (3,0,4,2,1,2). This assumes that an end-user can quantify their needs in a matrix construct.

If end-users could conceivably request information in this vector format an electronic request shell could be made to generate RFIs in a knowledge matrix format. A text section would also have to be included to highlight richer details that could not be transmitted using a strictly knowledge matrix method.

Table 3-3 User-Needs Vector in a Knowledge Matrix

Information Need		Attributes of Object					
		Location	Track	Identity	Activity/Status	Capability	Intent
Satisfaction Levels Needed	5						
	4			*Need*			
	3	*Need*					
	2				*Need*		*Need*
	1					*Need*	
	0		*Need*				

Modeling the creation of this user-needs vector is done using a set of discrete probability functions like the ones illustrated in Table 3-4. Table 3-4 requires some explanation. DISC is the acronym used by Arena for a discrete probability density function (PDF). The string of numbers in parentheses fully characterizes that PDF. The *Activity_State_Need* PDF is read; P(X=0) = 0.1, P(X=1) = 0.2, P(X=2) = 0.4, P(X=3) = 0.8, P(X=4) = 0.98, P(X=5) = 1.0. The final value, 22, defines the common random number (CRN) stream assigned to this distribution using Arena's common random number capability. The CRN variance reduction technique will be discussed further in this chapter in section 3.4 Modeling with Arena. The random vectors of needs generated represent explicit questions to which end-users desire answers. Each RFI generated in this manner may require more than one intelligence resource to satisfy its needs.

The satisfaction responses to this user-needs vector could then be tabulated using the knowledge matrix concept as a collection vector. This collection vector is the same conceptually as the original knowledge matrix kernel. The collection vector from each

intelligence resource used would be that sensor's current level of knowledge reported concerning a particular intelligence target. The sensors current knowledge level can then be compared to the user-needs requested knowledge level to derive a satisfaction level.

Table 3-4 User-Needs Generation Distributions

User Need Attributes	Requested Satisfaction Level Distribution
Activity_State_Need	DISC(0.1, 0, 0.2, 1,0.4,2,0.8,3,0.98,4,1.0,5,22)
Capabililty_Need	DISC(0.3, 0, 0.45, 1,0.6,2,0.9,3,0.98,4,1.0,5,23)
Identity_Need	DISC(0.18, 0, 0.3, 1,0.4,2,0.75,3,0.98,4,1.0,5,21)
Intent_Need	DISC(0.3, 0, 0.45, 1,0.6,2,0.9,3,0.98,4,1.0,5,24)
Location_Need	DISC(0.1, 0, 0.25, 1,0.35,2,0.75,3,0.98,4,1.0,5,20)
Track_Need	DISC(0.8,0,0.9,4,1.0,5,16)

Satisfaction is a continuous state of being; it is not black and white. We can be unsatisfied, somewhat satisfied, mostly satisfied, almost satisfied, and of course totally satisfied. We often state satisfaction levels in terms of percentiles. We were 100% satisfied with the product, or 80% satisfied, or 50% satisfied, and so on and so forth. This continuous level of satisfaction requires a continuous probability distribution for modeling purposes. To model a sensor's ability to satisfy User-needs each of the 15 sensor resources simulated in this model was given a notional triangular collection distribution in each knowledge area of the knowledge matrix. Table 3-5 is an example of what possible satisfaction levels a resource, IMINT1, might obtain. Refer to Appendix C for a listing of the 15 sensors simulated in this model and their associated collection distributions.

The triangular distribution was chosen and applied on all resources for three reasons: it is continuous; it is bounded; and a most likely value (the mode) can be easily assigned. The triangular distribution is a good example because it seems well suited to

our purposes here. If empirical data suggests that the appropriate distribution is something other than triangular than any distribution could be easily applied..

Table 3-5 Sample Collection Matrix, IMINT1

IMINT 1	Sat LvL	Loc	Activity/State	Track	Identity	Intent	Capability
	5	Max					
	4	Mode					
	3		Max		Max		Max
	2				Mode	Max	
	1		Mode	Max/Mode		Mode	Mode
	0	Min	Min	Min	Min	Min	Min

As we stated previously satisfaction is a continuous state of being therefore a continuous distribution should be used. Bounding is important because a sensor may only be able to satisfy a user-need request up to a maximum level (i.e. Sat Lvl 4, but not at a level 5). The sensor simply does not have the fidelity or capability for whatever reason. Another possible scenario is that a sensor always attains at least a level 2 satisfaction level (a picture is always taken, it just may not be the fidelity required). These user-need levels are based on the 0-5 scale outlined in Table 3-2. The collection vector for Table 3-6 would be (2,0,3,3,2,0). Integer values are used for simplicity in this example, but keep in mind that any of these integer values could easily be a real non-integer number within the bounds of its user-need probability distribution.

Comparing the dark Sensor X collection vector versus the grey User-needs vector it can be seen that the sensor did not provided 100% satisfaction in all knowledge areas. This sensor report then would not satisfy an end-user. Only 2 of 5 knowledge areas were satisfied therefore

Table 3-6 Comparison of Collection Vector to a User Needs Vector

Information Need		Attributes of Object					
		Location	Track	Identity	Activity/Status	Capability	Intent
Satisfaction Levels	5						
	4						
	3			*SensorX*	*Sensor X*		
	2	*Sensor X*				*Sensor X*	
	1						
	0		*Sensor X*				*Sensor X*

the end-user is only 2/5 or 40% satisfied with this sensor report. Often intelligence knowledge is improved by the fusion of two different data sources. To improve our knowledge level and hence our satisfaction level we would request another collection either by this sensor or by a different sensor. For example, let us assume that a second report, Table 3-7, arrives from sensor Y

Table 3-7 Second Sensor Collection Vector Compared to Original User-Needs Vector

Information Need		Attributes of Object					
		Location	Track	Identity	Activity/Status	Capability	Intent
Satisfaction Levels	5						
	4						
	3	*Sensor Y*					*SensorY*
	2			*Sensor Y*			
	1				*Sensor Y*		
	0		*Sensor Y*			*Sensor Y*	

We could fuse this information using any of the methods discussed in Chapter 2, or we could, if the multiple reports were received at approximately the same time, just

take the maximum satisfaction level from each report and derive a vector of maximum satisfaction as shown in Table 3-8. This maximum satisfaction vector fused from the two reports gives us a 4/5 or 80% satisfaction level. This model assumes that each user-need is best fulfilled by collection from one resource. The application of this algorithm is discussed in the section 3.7, the Production and Analysis submodel.

Table 3-8 Maximum Satisfaction Vector

Information Need		Attributes of Object					
		Location	Track	Identity	Activity/Status	Capability	Intent
Satisfaction Levels	5						
	4						
	3	*Max X,Y*		*Max X,Y*	*Max X,Y*		*Max X,Y*
	2					*Max X,Y*	
	1						
	0		*Max X,Y*				

3.4 *Model Creation with Arena*

The Intelligence Cycle model created is a discrete event simulation built with the Arena modeling software. The Arena software package was chosen for this simulation because the original model created by Captain Pawling was run in Arena, it was flexible, had a powerful and intuitive graphical user interface (GUI), and a superb random number generator. "The Arena generator has the facility for splitting the cycle of 3.1×10^{57} random numbers into 1.8×10^{19} sub-streams, each of length 1.7×10^{38}. Each stream is further divided subdivided into 2.3×10^{15} sub-streams of length 7.6×10^{22} apiece." (Kelton, 2003; p502) A robust random number generator, like the one in Arena was required for two reasons; first to drive the all of the random number draws for the

model's probabilistic processes, and second because over 200 separate random number streams were used to reduce model variance using the Common Random Numbers

Variance reduction technique (Kelton 2003; p512). CRN is a technique used to synchronize streams of random numbers when comparing model modifications. Some example modifications might be adding new resources, changing RFI arrival rates, or changing collection probability distributions. By keeping the random numbers synchronized, we can better determine if changes that occur in the system are due to the architecture change instead of simply a change in the random numbers drawn.

Note again at this point that all random number distributions used in this model are strictly notional. No empirical data or any other study currently supports the distributions used to model intelligence resources or processes. Should this thesis generate sufficient interest then empirical data could be collected from real world systems. A best fit could be applied to that data, and the proper distribution allocated to each process and resource in this model. As each phase of the Intelligence Cycle model is presented, a brief description of the notional distributions applied will be discussed.

Finally, Arena is an extremely flexible, attribute based modeling package. Users can define any number of attributes to characterize the entities created in a model. By characterize we mean that these attributes not only describe the "physical traits" of an entity, but also define that entity's interaction with the model logic, and other entities encountered in the simulation. Attributes are extremely powerful and allow for almost any process to be modeled using this software. There are 49 user-defined attributes and 26 user defined variables used to characterize each entity which flows through this model. For a listing of all user-defined attributes and variables, refer to Appendix B.

3.5 *The Intelligence Cycle - A Needs Driven Model*

3.5.1 Model Concept

It was mentioned in the introduction to this chapter that this model is needs driven and user-needs were discussed in the knowledge matrix methodology section. Needs driven means that information users are able to identify their needs, quantify those needs, and submit them in a knowledge matrix format. The knowledge matrices formed will be the Requests For Information (RFIs) which flow through this model. The assumption of a clearly stated need is reasonable as we discussed in Chapter II. If a question cannot be clearly and explicitly formed then a clear answer to that question most likely will not be forthcoming. It is the clear statement of user-needs in a knowledge matrix format which drives this Intelligence Cycle model.

3.5.2 Planning & Guidance Submodel

This first section of the model simulates two distinctly different "Need Events." Those two events are the planned submittal of RFI's, and the unplanned discovery of data which fulfills an already established EEI. Intelligence can be collected without a specific tasking. Sensors which are set to automatic search cycles or have human search operators may discover unasked for data. Non-specific searches may seem like a waste of resources but they can often provide valuable and unexpected insights. By including these collection opportunities in the model an attempt is made to more accurately capture resource utilization levels and successful/unsuccessful efforts in collecting non-tasked data.

In this model, the creation of RFIs represents a multitude of information users identifying their needs and submitting requests (RFIs). The untasked discovery of EEI

supporting data represents intelligence collectors that utilize resources to find data without a specific RFI directing search efforts. The user-needs which drive RFI creation, and the untasked discovery are simulated through the probabilistic creation of user-needs vectors as previously shown in Table 3-4. Figure 3-2 illustrates Planning and Direction starting with the end-users. The first six nodes, the arrow shaped creation nodes of Figure 3-2 represent all end-users, thousands of end-users, both military and civilian. These first six nodes generate all of the entities which flow though the model. For this model the first five nodes labeled *Rank1- Rank5* abstract the arrival of RFI requests into the intelligence cycle. The sixth node, labeled *Opportunity _Collect* represents information that is discovered, but not specifically requested

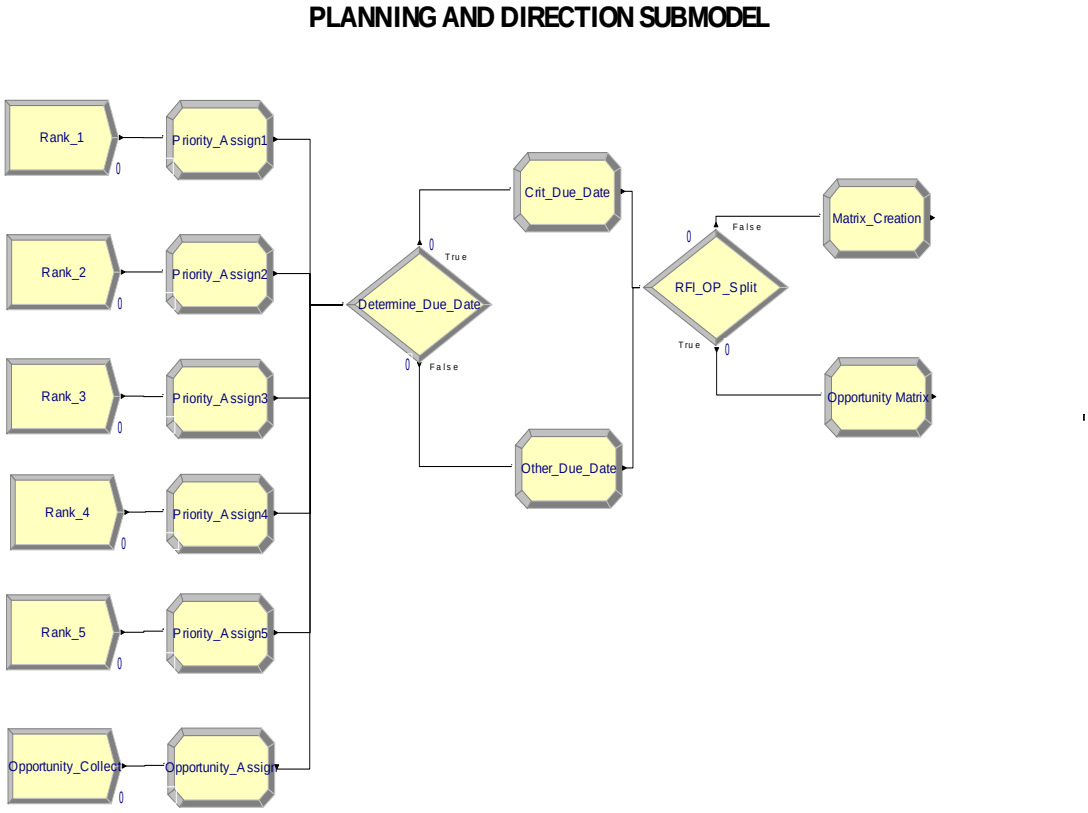

Figure 3-2 Partial Wire Diagram - Planning and Direction Submodel

Note that the five nodes titled Rank1 to Rank5 do not directly correlate to the social, political, or military rank of a requestor. These ranks refer to the priority of the requested EEI regardless if the actual requestor is an Army sergeant in the field or the President of the United States. In reality, the Army sergeant may have a higher ranked EEI need than the President of the United States because of that soldier's tactical situation. Five priority levels provide a sufficient number of choices for message ranks. The ranks represent (1) Critical, (2) Urgent, (3) Important, (4) Standard, and (5) Non-Time Critical RFIs. For the purposes of this model, Critical means that a mission cannot be completed without this request fulfilled. Urgent means that a mission may be completed without this information. It is not a show stopper, but mission degradation will occur. Important means that fulfilling this request will act as a force multiplier for friendly forces (i.e. give us an advantage over our opponents). Standard is usually an indications and warning type of request, something that must be monitored but does not constitute a higher priority. Finally Non-Time Critical requests are good to know bits of information, items that may be useful but are not considered directly actionable such as training data or scenario development data.

In modeling entity generation rates for each of the creation nodes (*Rank1-5* and *Opportunity_Collect*) the assumption is made that each entity, arrival rate can be modeled as a stationary Poisson process. This means that arrivals occur singly, are independent of each other, and the average rate is constant over time (Kelton, 2004:225). This assumption was made because no empirical data could be used in this unclassified thesis and stationary Poisson process are generally accepted as a good starting point for modeling discrete arrival rates (Law, 2000; 390)). Each entity type is assigned a Poisson

arrival rate based on their varying λ mean times of arrival (e.g. they all have varying exponential inter-arrival times of 1/λ). The arrival distributions used for the baseline scenario are shown in Table 3-9.

Table 3-9 Entity Arrival Rate Distributions

Baseline Scenario Arrival Rates	
Entity.Type	Distribution(λ,CRN)
Rank1	POIS(48,1)
Rank2	POIS(15,2)
Rank3	POIS(10,3)
Rank4	POIS(8,4)
Rank5	POIS(12,5)
Opportunity_Collect	POIS(6,493)

Once an entity is created, attributes are assigned to characterize that entity. Initial software attributes are automatically assigned by the Arena package upon entity creation. The software package tracks entities and establish fundamental model logic with these attributes. Two software assigned attributes, *Entity.SerialNumber* and *Entity.Type*, are used significantly in this application. *Entity.SerialNumber* is a unique number assigned to each entity upon creation. This number tracks each entity as it flows through the Intelligence Cycle model. An interesting property of the Entity.*SerialNumber* attribute is that entities duplicated throughout the course of the model all retain the same *Entity.SerialNumber*. This property is used extensively in the fusion logic portion of this model.

The *Entity.Type* attribute is linked to the animated images which flow through the model. The six images used are shown in Figure 3-3. The images represent the created entities ranked 1-5 and the opportunity collect entity.

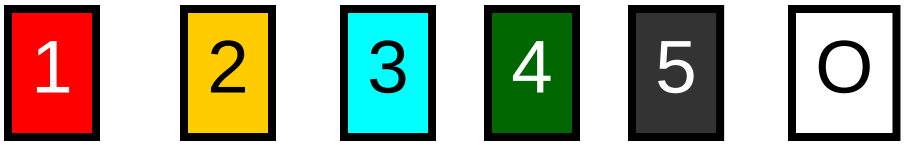

Figure 3-3 Entity.Type Animated Figures

Many more attributes are automatically assigned by the software package, but we are not concerned with them at this time. The attributes of prime concern are 55 user-defined attributes which characterize each entity as it flows through the Cycle. The first user-defined attributes are allocated at the assign nodes encountered immediately following each of the six create-nodes. These are the octagonal Assign blocks labeled *Priority Assign1* to *Priority Assign5*, and *Opportunity_Assign*.

These Assign blocks, and all others throughout the model, add or modify attributes for each entity that passes through them. The two attributes assigned at each of the first six-assign blocks are Priority and *RFI_Type*. Priority represents the Urgency of the RFI request as described above. The *Priority_Assign* blocks assign an integer value to the *Priority* attribute from 1 to 5. *Priority_Assign1* is set to 1, *Priority Assign2* is set to 2, …, *Priority Assign5* is set to 5. The *Opportunity_Assign* block applies a random *Priority* assignment based on a discrete distribution DISC(0.22, 1,0.35,2,0.7,3,0.9,4, 1.0,5,33). These *Priority* values are used throughout the model for queuing purposes. Their application will be discussed further when queuing logic is explained.

The *RFI_Type* attribute, like the *Priority* attribute, drives much of the critical logic in subsequent model performance. The *RFI_Type* attribute is divided into three specific categories: 1) Time-Critical; 2) Time Dependent; and 3) Fill At Will. The concept behind the *RFI_Type* attribute is simple, but essential to the question of

intelligence collection. Does the intelligence need to be collected in the fastest possible manner, only at a specific time, or filled as time permits?

The Time Critical *RFI_Type* is self-explanatory. These RFIs must be filled as fast as possible therefore this RFI type is mostly associated with a *Priority* = 1 request. A *Priority* = 5 would most likely not be time critical and this logic has been implemented in the model. The next *RFI_Type,* Time Dependent, means that collection must occur at a specific time, not earlier and not later. Often intelligence events are time dependent and assets must be allocated at the correct time or the data collected may not be usable. The final *RFI_Type*, Fill At Will means that this request can be filled early without any adverse consequences. These types of RFI's will most likely have a due date constraint, but it is not a pressing one. Once *Priority* and *RFI_Type* are assigned the entities flow into a diamond shaped decision node.

At a decision node, the model evaluates a logic statement to determine if the statement is true or false. Some decision nodes will have N different branches because more than two possible outcomes exist. This first decision node, Determine_Date_Due evaluates the statement, *RFI_Type* = =1 || *Entity.Type*= =*Opportunity* (the symbol = = means evaluate statement for truth and the || symbol is read as "or"). There are only two possible outcomes either the entity possesses at least one of the attribute types *Opportunity_ Collect Entity.Type* or *RFI_Type 1*, or it does not. If the statement evaluates to true then the entity is sent to *the Crit_Due_Date* assign node where a *Due_Date* within the next 2 to 48 hours is assigned (time assignment is based on a ~TRIA(,2,12,48) distribution, mean=12 and Standard Deviation=2). If the statement is not true then the entity is a Non-Time Critical RFI and it is sent to the *Other_Due_Date*

assign node where a *Due_Date* is assigned with an ~Exponential (10days) distribution, (λ=10days). This indicates that the mean value of this request is approximately a week and a half, but theoretically the elongated tail probabilities of the exponential distribution allows for a request that could be due years in the future. The exponential distribution works because theoretically such an intelligence collection request could exist.

Once a *Due_Date* attribute is assigned to each entity then the entities are once again broken back down into RFI entities and Opportunity collect entities by the logic in the RFI_OP decision node. The RFI entities flow to the RFI Matrix_Creation Node and the Opportunity entities flow to the Opportunity Matrix. These two nodes are the foundation on which this model is built. They assign the majority of the user-defined attributes which characterize all model entities. They generate three types of attributes; administrative attributes, decision logic attributes, and knowledge matrix user-needs vectors. Administrative attributes are model constructs used to help the entities flow through the processes within the model. Decision logic attributes determine what paths the entities will choose as they interact with the model. The knowledge matrix User-needs vectors have already been discussed. They drive collection and are the measuring sticks to which satisfaction is applied. Table3-10 depicts the *Matrix_Creation* attributes assigned to RFI entities generated in the model. Table 3-11 depicts the *Opportunity_Matrix* node attributes assigned to Opportunity entities generated in the

Table 3-10 Attributes Assigned at Matrix _Creation Node

TYPE	MATRIX_CREATION	VALUES ASSIGNED
Admin	Reoccurance	Due_Date-Entity.CreateTime
	RFI_Create_Date	Entity.CreateTime
	Sort_Rule	Due_Date*Priority
Decision	Abroad	DISC(0.5,0,1,1,13)
	Contine_Collect_After_Due	DISC(0.8, 0, 1.0, 1,18)
	Detectable_Emissions	DISC(0.5, 0, 1.0, 1,14)
	Emits_RF	DISC(0.5,0,1,1,15)
	Library_Search	TRIA(0,0.8,1,17)
	Quit_Collect	Due_Date+DaysToBaseTime(EXPO(30,32))
	Standing_or_Adhoc	DISC(0.6, 0, 0.61, 6, 0.66, 12, 0.74, 24, 0.80, 48,
		0.9,168,0.95 720,1.0,4368,19)
KM	Activity_State_Need	DISC(0.1, 0, 0.2, 1,0.4,2,0.8,3,0.98,4,1.0,5,22)
	Capabililty_Need	DISC(0.3, 0, 0.45, 1,0.6,2,0.9,3,0.98,4,1.0,5,23)
	Identity_Need	DISC(0.18, 0, 0.3, 1,0.4,2,0.75,3,0.98,4,1.0,5,21)
	Intent_Need	DISC(0.3, 0, 0.45, 1,0.6,2,0.9,3,0.98,4,1.0,5,24)
	Location_Need	DISC(0.1, 0, 0.25, 1,0.35,2,0.75,3,0.98,4,1.0,5,20)
	Track_Need	DISC(0.8,0,0.9,4,1.0,5,16)

model. The *Matrix_Creation* node and the Opportunity Matrix both assign the same attributes, but do so based on different distributions. For example the

Opportunity_Matrix will generate only zeros for the *Library_Search* attribute because *Opportunity Entity.Types* collect only new raw data.
They do not search through archives for previously discovered data. A regular RFI however, could have needs that are fulfilled by a library search of archived data so an archival search is modeled.

The most critical attributes assigned at these nodes are the attributes which form our User-needs vector. The possibility does exist that a vector of all zeros could be created in either of these two nodes.

Table 3-11 Attributes Assigned at Opportunity_Matrix Node

TYPE	OPPORTUNITY_MATRIX	VALUES ASSIGNED
Admin	Sort_Rule	Due_Date*Priority
Decision	Abroad	DISC(0.5,0,1,1,26)
	Contine_Collect_After_Due	DISC(0.8, 0, 1.0, 1,31)
	Detectable_Emission	DISC(0.5,0,1,27)
	Emits_RF	DISC(0.5,0,1,1,28)
	Library_Search	0 (No Library Search Ever)
	Quit_Collect_Date	Due_Date+DaysToBaseTime(EXPO(30,32))
	Standing_or_Adhoc	0 (Adhoc only, never a Standing RFI)
KM	Activity_State_Need	DISC(0.1, 0, 0.2, 1,0.4,2,0.75,3,0.9,4,1.0,5,36)
	Capabililty_Need	DISC(0.3, 0, 0.45, 1,0.6,2,0.9,3,0.95,4,1.0,5,37)
	Identity_Need	DISC(0.1, 0, 0.2, 1,0.4,2,0.75,3,0.9,4,1.0,5,35)
	Intent_Need	DISC(0.3, 0, 0.45, 1,0.6,2,0.9,3,0.95,4,1.0,5,38)
	Location_Need	DISC(0.1, 0, 0.25, 1,0.35,2,0.5,3,0.9,4,1.0,5,34)
	Track_Need	DISC(0.8,0,0.9,4,1.0,5,29)

To prevent one of these null RFIs from shooting through the model, drop logic is added in the form of a decision node (*Null_RFI_Decide*) and a dispose node (*Null_RFI*). Any null RFI created will trip the truth logic and be disposed of immediately. If an RFI is not a null RFI then it proceeds to the next decide node, the *Library_Search* decide node. This Library node represents the time an intelligence professional might spend looking for information archived in any number of intelligence databases or libraries. As such the Library process node enacts a delay, but does not utilize any intelligence collection resource. The delay is modeled as TRIA(0.5,1,8) distribution which means that a minimum time of a half hour and a maximum time of one work day (8 hours) could be spent searching for archived information. Note that this is the first delay built into the model, after this library search delay some information may have been gained. To model this information gain the Library assigns six new attributes, collect attributes, to the RFI as illustrated in Table 3-12.

Table 3-12 Library_Search Example of Collection Vector Distributions

Collect Attributes	Level of Satisfaction Probabilities
Activity_State_Collect	Tria(0,1,2,496)
Capabililty_Collect	Tria(0,1,2,499)
Identity_Collect	Tria(0,1,2,497)
Intent_Collect	Tria(0,1,1,498)
Location_Collect	Tria(0,1,3,495)
Track_Collect	0

With the Library collection vector created the RFsI flow into the Drop_Logic decision node. Drop Logic is a reoccurring theme through out the Intelligence Cycle. Drop Logic is a check to see if the information being requested is still worth collecting. The attribute *Due_Date* is used to simulate a report deadline, a time when information is required to be back to the end-user/requestor. This end-user established deadline is the time constraint mentioned at the start of this chapter. Often information must be collected despite the fact that its point of maximum usefulness is past. For these instances the *Continue_Collect _After_Due* attribute was added to the model. But even valuable information has a finite shelf life so there will come a time when collection is pointless and no further effort should be made, for this point in time the *Quit_Collect* attribute was added to the model. The model compares these three attributes; *Due_Date*, *Continue_Collect_After_Due*, and *Quit Collect* against the simulation clock time, TNOW, to determine entity timeliness and proper progression through the Intelligence Cycle.

Drop Logic does not imply that the RFI will drop out of the system entirely, possibly just out of the current submodel. Due to tardiness an entity may simply be forwarded to a point further along in the Intelligence Cycle (i.e. skipping fusion), it may be sent back to the end-user as a null collect report (i.e. collection not accomplished

because time ran out), or it may be recycled into the Intelligence Cycle if it is a standing RFI entity.

At this point the difference between a *Standing_RFI* and an *Adhoc_RFI* must be explained. An *Adhoc_RFI* is a one time request for information that will shoot through the Intelligence Cycle only once. Note that an *Opportunity Entity.Type* will always be modeled as an *Adhoc_RFI*. A *Standing_RFI* on the other hand is a reoccurring request for intelligence. A *Standing_RFI* will loop through the Intelligence Cycle a number of times, based on its periodicity (daily, weekly, monthly, semi-annually), until its *Quit_Collect* date occurs.

This *Adhoc* versus *Standing RFI*'s model logic is a significant improvement over the previous model. It more closely simulates the real world making the model a true Intelligence Cycle, rather than a straight line Intelligence Process. There is one drawback to this method that we will discuss further in Chapter IV, cycling RFI's complicates statistics collection by skewing our current work in progress (WIP) estimations for each *Entity.Type*.

If the entity is not dropped or recycled in the Drop Logic sequence then the final node in the Planning and Direction submodel is encountered, the *Needs_Check* submodel. This submodel forwards RFIs satisfied by the Library Search to the Process and Evaluation submodel, or submits the request for further collection because a library search did not reveal enough information to satisfy the end-user. Within the *Needs_ Check* submodel is the *Needs_Assess Library* subprocess block. The logic in this block, shown in Figure 3-4 counts the number of non-zero user-needs in the user-needs vector and assigns that value to an attribute called *Num_Needs*. The number of needs is

counted at this point because of an assumption made concerning collection. It is assumed that efficient collection is obtained by assigning each user-need above zero one-collection resource. This assumption drives both the collection and fusion algorithms used in this model. Note that there is an extra decision node in our needs check logic called *Library_ Check* even though library is not a need. The *Library_Check* node is included because we do not want to lose any information that was gained in a library search. If this logic were not included then the collection vector gained by the library search would not be included in the fusion process later in the model.

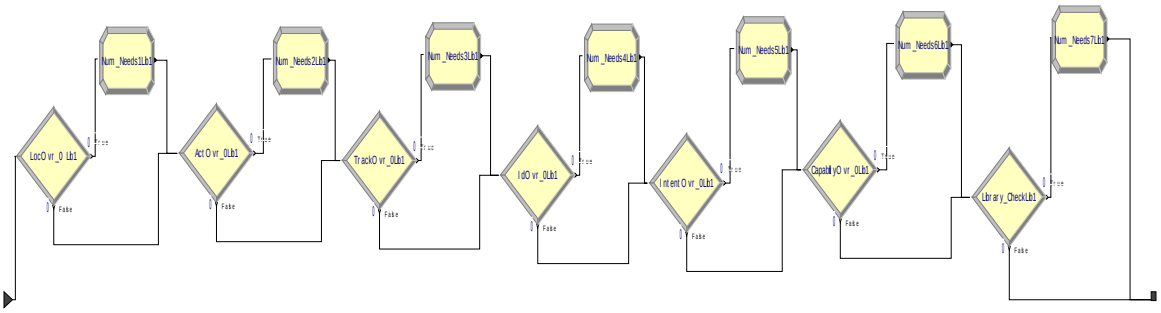

Figure 3-4 Post Library Needs Assessment Logic

If all needs were satisfied the RFI skips further collection and jumps to the Processing and Evaluation node. If not all needs were not satisfied the entity continues out of the Planning and Direction submodel into the Collections submodel.

In summary there are five ways to exit the Planning and Direction submodel:

1) Drop out of the Intelligence Cycle entirely because *Quit_Collect* time was surpassed

2) Jump from *Needs_Check* to the Processing & Exploitation submodel because all Needs were fulfilled in the Library search

3) Leave through the exit point without a needs check because no library search was done

4) Leave through the exit point after a *Needs_Check* identified that needs were not satisfied

5) Be sent to the Standing RFI hold node because *Due_Date* expired but not *Quit_Collect*

3.5.3 Collection Submodel

This submodel simulates entity prioritization, resource selection, resource allocation, and intelligence platform data gathering performance. The assumption that each non-zero user-need element requires one collection resource to satisfy its need is key to the logic used in this model. Original RFI's are duplicated based on *Num_Needs* and sent to each user-need area collection submodel for simulated collection. Resource capabilities and intelligence target characteristics are evaluated to determine what one resource is most likely to collect the requested data within the *Due_Date* time constraint. This evaluation is performed as decision tree logic within each user-need collection submodel. Collection is simulated as a time delay followed by the assignment of a collection vector, similar to that shown previously at the Library_Assign node. We hope that the collection vector for each user-need will satisfy the primary need in that vector (i.e. the need that resource was chosen to fulfill). Essentially an RFI is matched against the resources that will most likely result in a satisfactory collection vector. This section is heavily dependent upon proper intelligence resource modeling.

3.5.3.1 Resource Modeling

Fifteen different intelligence platforms are simulated in this portion of the model. Realistic resource simulation involves creating collection probabilities, estimating quantities of requests that can be handled at the same time, and creating resource schedules. Table 3-13 most of the parameters assigned to the RADINT1 Resource. For this model the RADINT1 resource models a real world Space Based Radar (SBR). The parameters shown characterize the probabilities that RADINT1 will capture data in each of the knowledge matrix-knowledge areas. The last three parameters characterize the systems ability to collect RF emissions, other emissions, and if the system can operate both (B) abroad and within the US. This system has and N for both RF and Find Emis because it does not perform those functions, but it can operate both abroad (outside the U.S.) and at home. Refer to Appendix C for a comprehensive list of all fifteen resources and all of their associated parameters.

Table 3-13 RADINT1 Parameters Modeled

Sat LvL	Loc	Activity State	Track	Id	Intent	Capability	System	RF?	Find Emis	Abroad
5			Max				SBR	N	N	B
4			Mode							
3	N/A				N/A					
2		Max		Max		Max				
1		Mode		Mode		Mode				
0		Min	Min	Min		Min				

Now that we know what the RADINT1 resource can do, we have to know when it is available for operations. Figure 3-5 depicts the scheduled availability and capacity for

RADINT1. Looking at Figure 3-5 we can see that the RADINT1 can simultaneously handle up to 250 entities. The resource is available 12 hours out of each day, but only in fifteen minute intervals, then it is down for a fifteen minute interval. For diagrams of all resource schedules used in this model, refer to Appendix C. More resources could easily be added to the model as needed.

Due to the complexity of this process, there are a number of assumptions that must be made. These assumptions are discussed as the section progresses. Also due to this section's complexity it contains a number of layers that are broken down into separate subsections. Figure 3-6 displays the first layer of the collection submodel

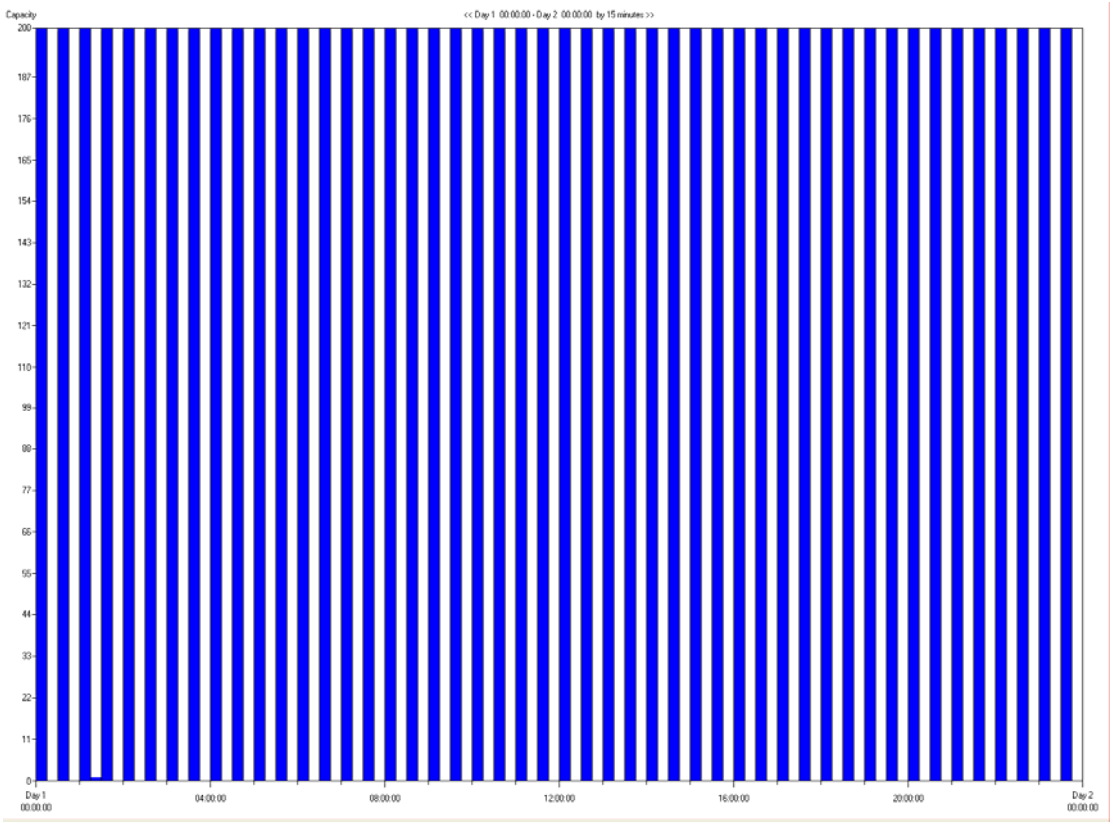

Figure 3-5 RADINT1 Capacity & Availability Schedule

Notice in Figure 3-6 that this collection submodel has two entrance points. The upper entrance point is the one through which first time requests will flow into the submodel, and the lower entrance is the one through which standing RFI's will loop through the collection submodel. Both entrances flow into the Time Management sub-process node.

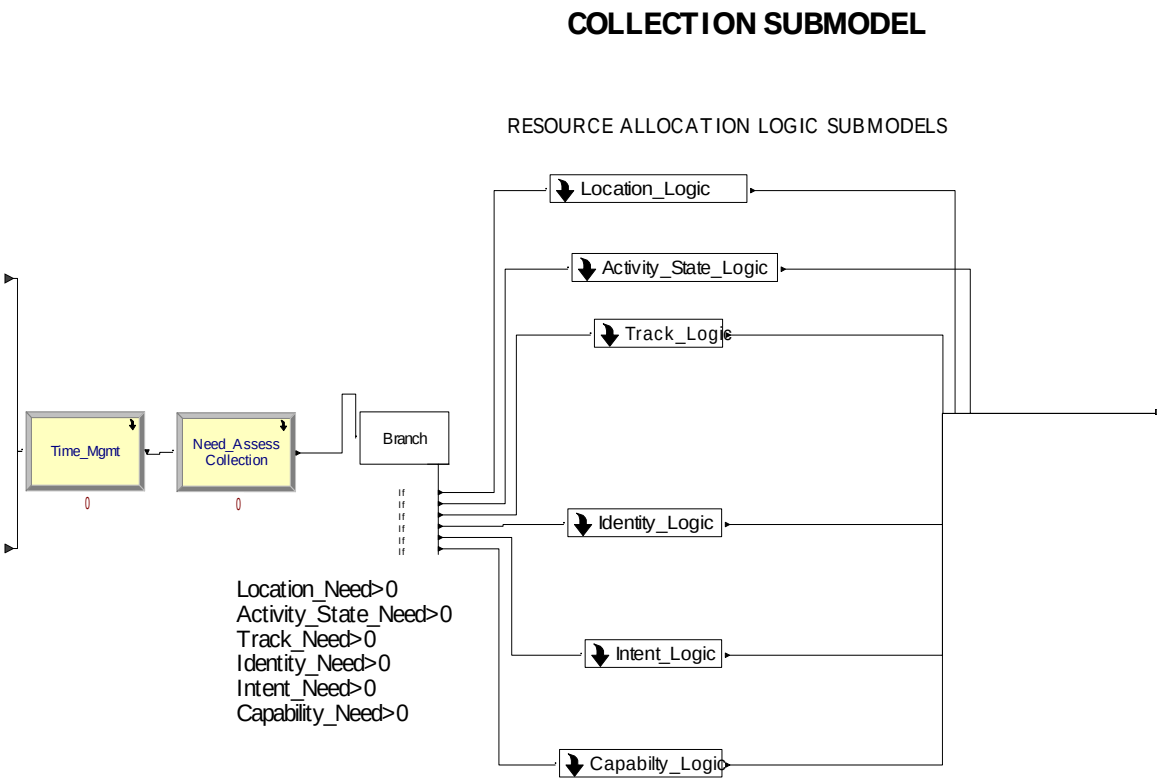

Figure 3-6 Wire Diagram - Collection Submodel

.

3.5.3.2 Time Management Subprocess

The purpose behind this node was outlined in the discussion of RFI_Types. A type 2 RFI means that the RFI is time dependent; it should only be collected at a certain time and not before. The logic applied in this submodel, Figure 3-7, will hold entities here until they are within 48 hours of collection, at which time they will be released back

into the Cycle. The first decide node *Not_T_Dependent* will determine if the RFI is type 2 or not. If not then it is allowed to continue on its way. If it is a type 2 RFI then it is sent to the next decide node, *Time_Parser* which will determine the proper Hold node to send the entity to based on that entities *Due_Date*. The nodes shown are RFI's six months out, a month to six months out, A week to a month out, a week to two days out, and then after this last hold node the entity is released from the *Time_Mgmt* subprocess.

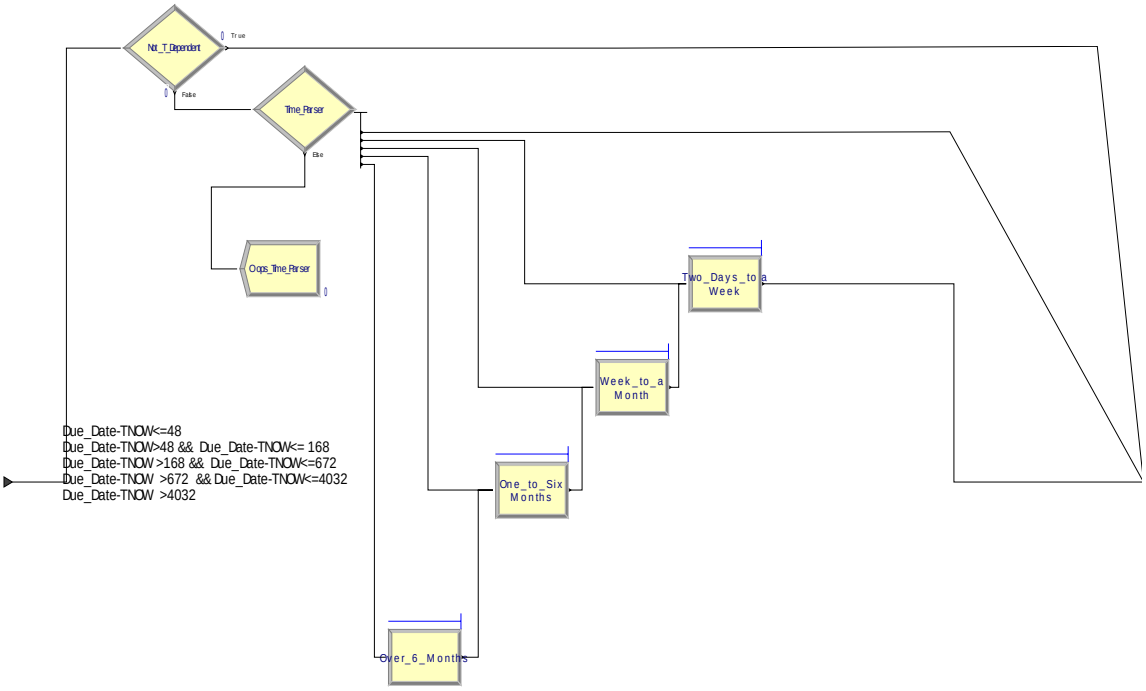

Figure 3-7 Wire Diagram - Time Management Subprocess

The prioritization scheme used in the queues for each of the hold nodes is a lowest attribute first scheme, with *Due_Date* as the attribute that is queued. Each hold node scans for a condition to be met. For example in the *Week_To_A_Month* hold node, when *Due_Date* – TNOW < = 168 then less than a week is left and the entity waiting is released to the *Two_Days_To_a_Week* hold node. This logic is followed until the entity

is released from the *Time_Mgmt* subprocess and continues on to the *Needs_Assesment Logic.*

3.5.4 Need Assessment

The logic in the *Need_Assess Collection* subprocess node is the same as that used previously in the *Need_Assess Library* subprocess node. This node once again counts the number of non-zero user-needs so that fusion and statistics collection is possible later in the model. This action is repeated because entities which either skipped library search or were not satisfied by the the library search never had a needs assessment completed. Once *Num_Needs* is established the entity will proceed out of the *Needs_Assess* subprocess and into a branch node. This branch node is first step in resource allocation. Based on the assumption that each need can be fulfilled best by one resource this node evaluates each entity's user-need vector and sends a duplicate of the original entity to each user-need logic submodel which requires a satisfaction greater than zero. For example if an entity has a user-needs vector with four needs above zero then four duplicates are created and each one is sent to its appropriate *Need_Logic* submodel for collection (i.e. *Activity_State_Logic*, *Location_Logic, Track_Logic* etc…).

3.5.4.1 ActivityState_Collect Node-by-Node Collection Example

This step in the model simulates logic that an automated Collection Manager might employ. We will use the *Activity_State_Logic* submodel as our example for this area of the model. Each of the *Needs_Logic* submodels are setup with logic similar to the *Activity_ State_Logic* submodel. Figure 3-8 illustrates the first logic layer.

The first decide node checks to see what kind of satisfaction level is required. This may be a question of sensor fidelity in the real world sense because many satisfaction levels can be based upon a degree of resolution desired. The decide node *Needs_Parser_Act* divides the entities into those that require a 3 to a 5 satisfaction level, and those that need only a 1 to 2 level of satisfaction. We will follow the track of entities that require a 3 to 5 level satisfaction. These entities are routed to the *Act_HiRes* submodel. Again the *Act_RegRes* has very similar logic.

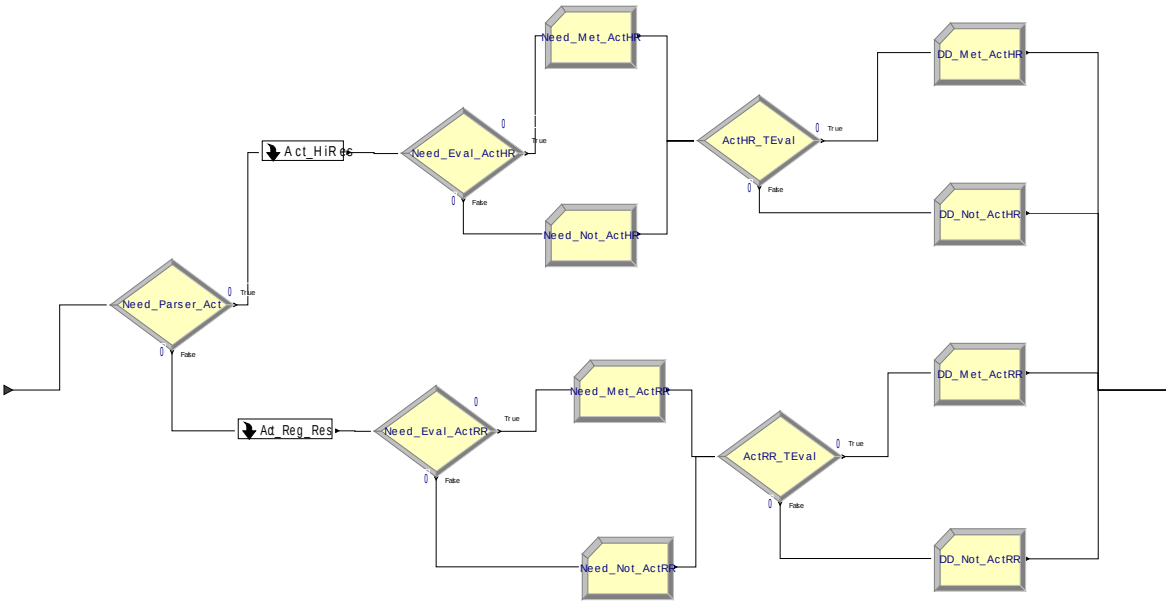

Figure 3-8 Wire-Diagram Activity_State Needs Parser and Counts Logic

3.5.4.2 Hi-Resolution Collection Decision Tree

The *Hi-Res* and *Reg_Res* collection submodels simulate intelligence resource selection, and data gathering. Figure 3-9 depicts a very simple decision tree used to determine what intelligence resource set should be selected. The basis for using a decision tree at this point in the model was stimulated by a discussion in the text “Making Hard Decisions with Decision Tools” (Clemen, 2001; 83) The decisions made are based

upon attribute information provided to the computer Collection Manager through the RFI input shell. In the case of Opportunity entities it is assumed that the seized resource found EEI supporting information in the course of a search.

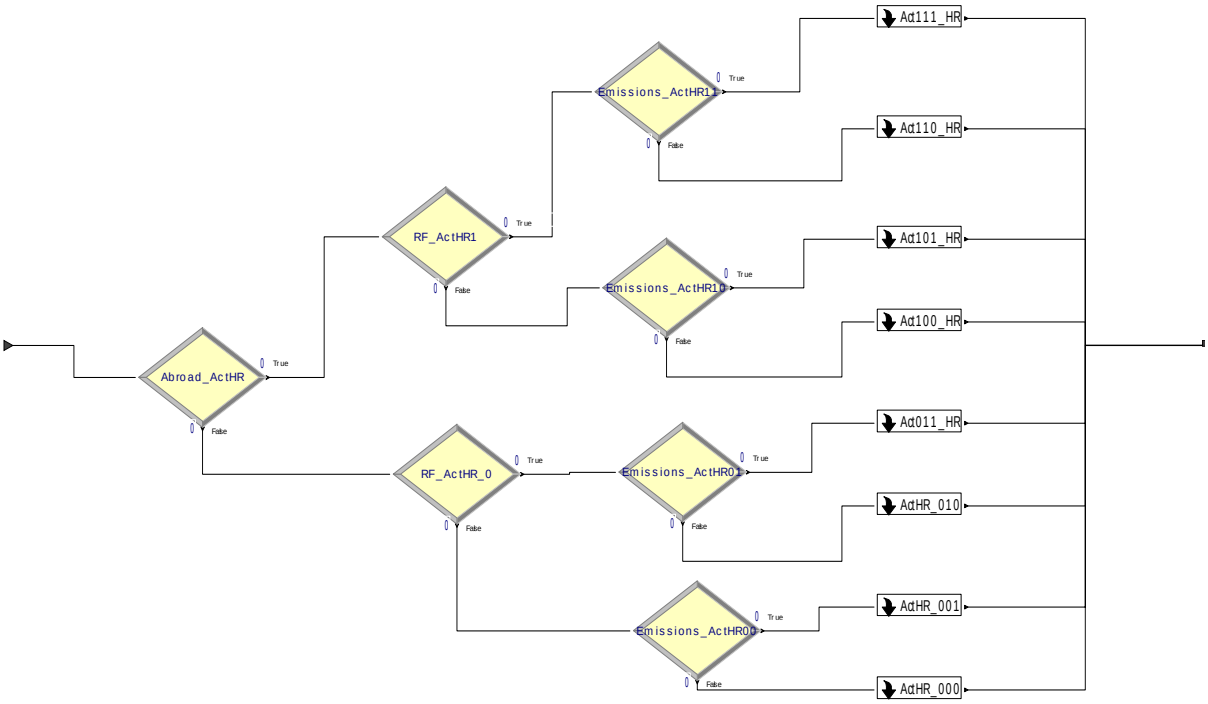

Figure 3-9 Wire Diagram - Collection Decision Tree Logic

In this simple sample problem the three attributes were used to create the decision trees; *Abroad*, *Emits_RF*, and *Detectable_Emissions*. These attributes are set to either 1 (true) or 0, false based on a user-defined probability distribution. The *Abroad* attribute determines if the intelligence target is within the United States, or on foreign soil. Certain intelligence resources are located only in the U.S. and others are located only abroad. For example, the FBI operates only within the United States, while a Rivet Joint aircraft usually operates only abroad. The next attribute, *Emits_RF*, refers to the type of communications conducted by the intelligence target. Does the target communicate using

any form of Radio Frequency, or other Electro-Magnetic communication (cell phone, fax, walkie-talkie, short wave radio, etc…). If electronic communication is used then another set of resources might be engaged against the target.

The final attribute used in this decision tree is *Detectable_Emmissions*. This attribute is added as a catch all for remaining intelligence resource allocation. For an example of a detectable emission consider a magnetic anomaly detector (MAD) which could be used to look for coastal submarine activity (example based on text of Nimrod MR2 description on the Federation of American Scientist website). These three attributes are very simple, explicit questions which when used in this decision tree format create a kind of a Ruled Needs based asset allocation strategy. Each entity will travel through the decision tree logic, following its "true" path until it reaches the appropriate collect submodel. For our example, we will look at the *Act110_HR* submodel.

3.5.4.3 Activity_State_Collection Resources

The *Act110_HR* submodel, Figure 3-10, displays the final stage of asset selection and tasking in the *Act110_HR* submodel. *Act110_HR* means that this submodel deals with intelligence targets that are abroad, emit and RF signal, but do not have other detectable emissions, the binary designator kept decision tree logic orderly as the model was created. Also of note at his point is the fact that decision tree logic could become cumbersome quite quickly. There will be 2^N final collect submodels like *Act110_HR*. Where N in this case is the number of attributes which make up the splits in our decision tree. We used three attributes therefore $2^3 = 8$ possible Collect submodels. If seven attributes are used to define the decision tree then $2^7 = 128$ different Collect submodels

similar to the one shown in Figure 3-10 *Act_110_HR*. There are more than seven attributes that could be used to characterize intelligence targets and some of those attributes may have an outcome space larger than simply true or false (0 or 1).

The collect submodel names like *Act110_HR* were designed to be easily understood one reads them as: Need(Abroad,Emit_RF,Detectable_Emissions)_ Need_Satisfaction Level(High Resolution (HR), or Regular Resolution(RR)). Once through the decision tree logic the collect subroutines at the ends of the decision trees are the very bottom layer, the foundation of this modeling effort. It is at this point that a human must determine which resource(s) which will most likely achieve the best collection vector based on the information and constraints presented by the RFI input data (i.e. the defining attributes such as *Abroad, Emits_RF, Due_Date,* etc…) and program those resources into the computers resource seize logic. This is the most hardwired portion of this model and would be the most tedious to change during architecture restructuring.

The first node encountered in the *Act110_HR* collect submodel is a queue node. This queue prioritizes entities based on a lowest attribute first rule. The attribute used for this prioritization is the *Sort_Rule* attribute. The *Sort_Rule* attribute value is determined by (*Due_Date*-TNOW)* *Priority*. This simple decision algorithm gives equal weight to priority and time remaining in a strictly linear fashion. Once entities are prioritized they will wait in queue until they reach the front of the queue. Once an entity is the first in line it will be pulled forward by its associated select node. In the select node the entity's *Sort_Rule* attribute is compared to the *Sort_Rule* attribute for all entities, in all other collect submodels throughout the entire model which are requesting use of the same

resource(s) because these select nodes use Arena's shared resource capability. The entity with the lowest *Sort_Rule* will be allowed to seize the desired resource. If multiple resources are available the *Select* node will determine which of the resources provided should be utilized based on a Resource Selection Rule (RSR). In Figure 3-10 because we are in the *Act110_HR* collect submodel and the three SIGINT assets have similar collection vector distributions the RSR is Smallest Number Busy (SNB). The asset least heavily tasked will pick up the slack from those that are heavily tasked.

After a resource is seized an immediate drop logic check is performed. This drop logic is implemented at this point because it is possible that the entity in question could have been held in its *Queue* node past its *Due_Date*. If it is past due (TNOW>=*Due_Date*) then the entity will flow to the *CC_Chk_Act111_HR* decision node, where a check is made to see if collection is still warranted. If the entity is past it's *Quit_Collect* date then it is immediately sent to the Release resource node, and the resource in question is release without incurring any time on the simulation clock. The entity is then given a *Null_Collect* report (which means all *Need_Collect* attributes are set to zero). The fact that the report was dropped before collection was done is counted by the count node and the entity is finally forward by the *Route* Node, "*ActHR111RptSkpS1*". A route node allows an entity to jump from one area in Arena to another without following the wire diagram. All entities which are sent through this drop logic sequence are sent to the Processing and Exploitation "*No_Resource_Siezed*" station. For a route node to be used it must have a destination station node associated with it.

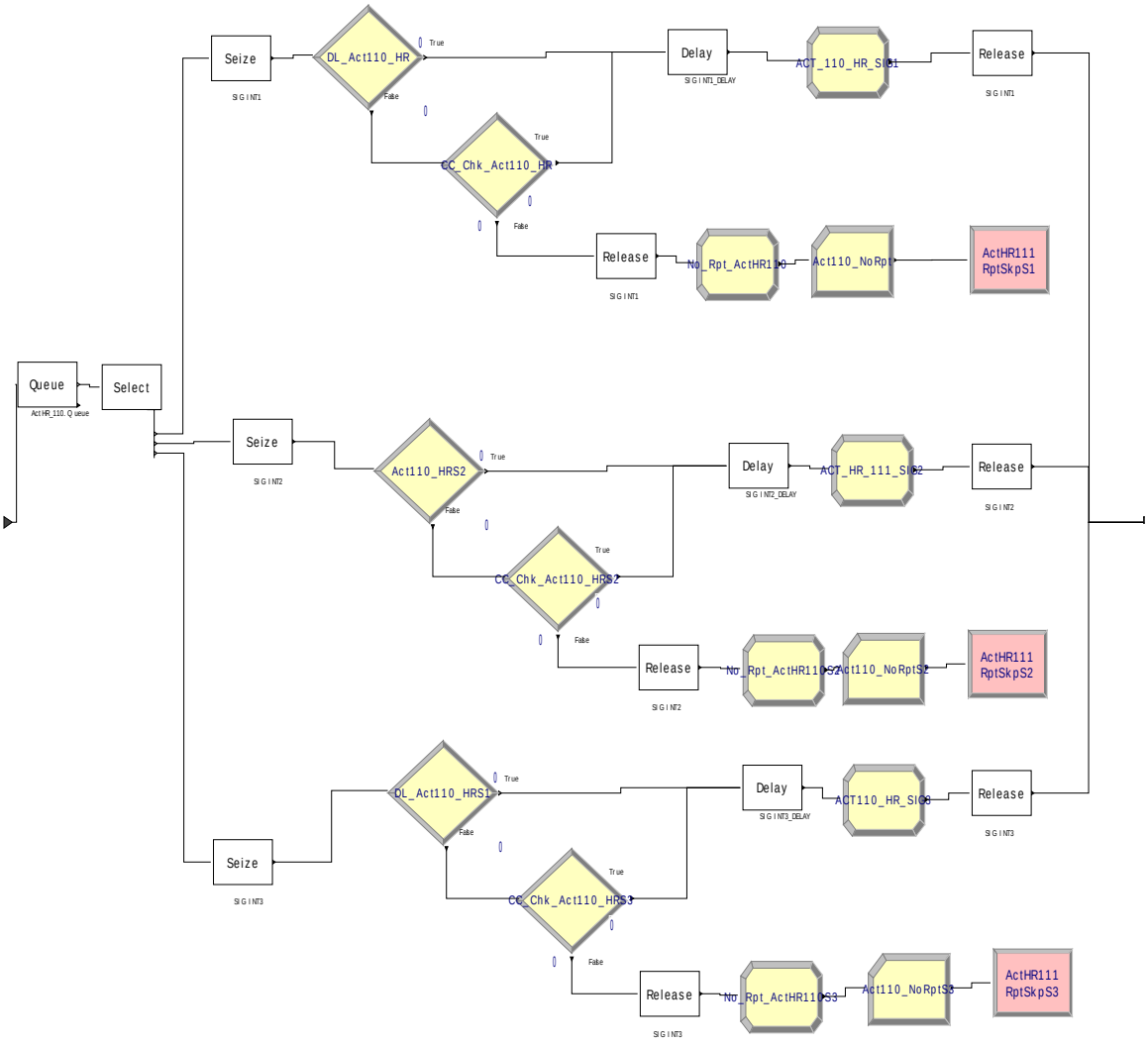

Figure 3-10 Wire Diagram - Resource Selection Modeling

Essentially this drop logic sequence ensures that timed out reports do not occupy resources without reason. They are not dropped from the system because all reports, even those with null collection vectors, are required for proper functioning of the fusion algorithm which begins in the Processing and Exploitation section of the model. All reports that are submitted for collection must be accounted for after collection or in this case non-collection have been performed. The fusion algorithm is expecting to receive a

specific number of reports equal to *Num_Needs/Num_Needs_Norming*. If any reports are dropped from the system at this point the fusion logic later in the model will hold all the other *Need_Collect* reports on this intelligence target waiting in the Processing and Exploitation's *First_Fuse* queue until its *Quit_Collect* date is reached.

If a *Need_Collect* report is not past its *Due_Date* or if collection is still warranted after the *Due_Date* then the entity, with its resource still seized, flows into its delay node. It is delayed based on its predetermined delay distribution (refer to Appendix C for delay distributions used) the two distributions used are Triangular and Exponential distributions with differing (min,mode,max) and λ values depending on the resource being modeled. Once the delay is completed the entity flows into its Collect assign node, *ActHR_110_ SIG1* in our example case. In this assign node a needs collection vector is created by the assignment of six new attributes: *Activity_State_Collect*, *Location_Collect*, *Track_ Collect*, *Identity_Collect*, *Intent_Collect*, and *Capability_Collect*. All of these *Need_Collect* attributes are assigned Triangular probability distributions. Once this needs collection vector is generated the entity flows into the resource *Release* node. The entity then exits the collect submodel and steps back into the decision tree submodel, from which it immediately exits up into the *Needs_Logic* submodel.

3.5.4.4 Exiting Collection

Looking back to the *Need_Logic* submodel, Figure 3-8, we see that two decide and count sequences follows both the *Hi_Res* and *Reg_Res* decision tree submodels. The first decision node after exiting simulated collection, the "*Needs_Eval_ActHR*" node for example, determines whether the specified need (*Activity_State_Need*) was successfully

satisfied by the *Activity_State_Collect* attribute. The entity is directed to the proper count node (the rectangle with the clipped upper left corner) for either satisfying the *Activity_State_Need* or not satisfying that Need. Then whether the Need was satisfied or not the entity enters the next decision node, *ActHR_TEval*, where timeliness is determined. If the *Due_Date* is still greater than TNOW, the entity did not spend too long in collection and run over time so the entity flows to the *DD_Met_ActHR* node and that counter is incremented. If timeliness fails at this point then the entity is sent to the *DD_Not_ActHR*. These two counter sequences serve as flags to alert us if either our user-needs MOEs and/or our timeliness MOEs are met at this point in the model. If a failing trend is noted then there may be a problem with the architecture that was established, or with the allocation of resources. These are the last nodes encountered in this submodel. After these counters, entities will exit the *Activity_State_Logic* submodel, and then exit the Collection submodel altogether and flow into the Processing and Exploitiation submodel.

3.5.5 Processing & Exploitation Submodel

The Processing and Exploitation submodel, Figure 3-11, simulates four events; preparation of finished intelligence products, implementation of logic to evaluate different forms of fusion, the decisions of if and/or when to fuse or forward information, and collection vector batching (i.e. pre-fusion). The three different fusion algorithms are formulated at this point in the model. They are, no fusion (NF), mixed fusion (MF), and strict fusion (SF). Each algorithm uses the exact same collection vectors to evaluate model MOEs.

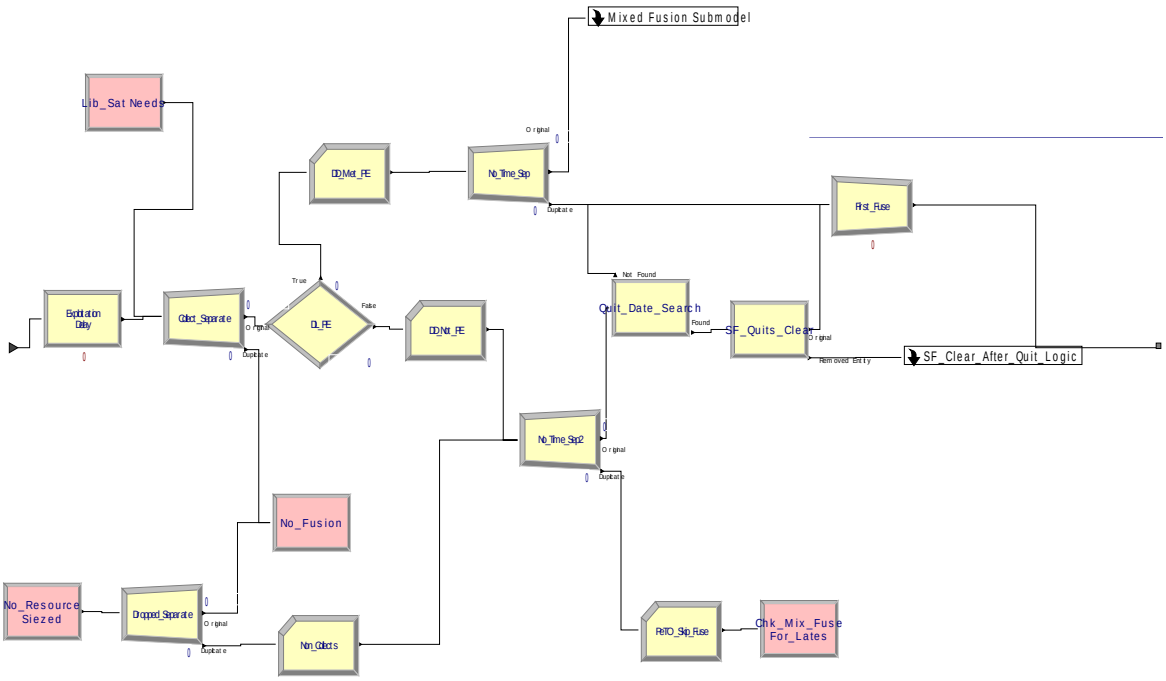

Figure 3-11 Wire Diagram - Processing & Exploitation Submodel

The first event, preparing a finished intelligence product, is simulated as a simple delay in the first node of the Processing and Exploitation submodel, *Exploitation_Delay*. The actual activities performed at this time might be annotating imagery, transcribing a SIGINT report, or debriefing a HUMINT source. The assumption made in this model is that a TPED (Task, Process, Exploit, Disseminate) architecture is in place and not a TPPU (Task, Process, Post, Use). Captain Pawling gives a good brief explanation of TPED vs. TPPU in his thesis (Pawling, 2004; 2-5). A TPED architecture assumes that all intelligence must be processed in some way, while TPPU assumes that some types of intelligence are finished products as soon as they are collected and can be used immediately without any processing. In this model no resources are seized during this processing time. If TPPU versus TPED architectural comparisons need to be run and if

processing resources needed to be modeled then the *Exploitation_Delay* node could be modified into a subprocess node and TPPU logic could be incorporated as needed.

The two other entry points into the Processing and Exploitation Submodel are the "*No_Resource_Seized*" station located just beneath the *Exploitation_Delay* node, and the Lib_Sat Needs station, from which RFIs satisfied in library search arrive. At this point the implementation of differing fusion algorithms begins.

Recall that the original RFI or Opportunity Collect was a vector of one to six non-zero user needs. Each non-zero user-need created a duplicate of that RFI to be sent to a Collection_ Logic submodel as an individual resource tasking. Therefore the number of collection vectors generated for each RFI is always equal to the value of the *Num_Needs* attribute (*Num_Needs*-1 if a library search was used as a resource as well). The first step in formulating the three different fusion algorithms is to duplicate every collection vector and send them all through the No Fusion route. The *No_Fusion* route node sends all of the collection vectors to the *Non_Fused* station in the Dissemination submodel, where statistical analysis will be performed on all of the collection vectors generated, without any fusion applied. The MOE evaluation done on the *Non_Fused* reports will be compared with a mixed fuse and forward strategy, and a strict fusion strategy. With the No Fusion collection vectors already disseminated, the original collection vectors are duplicated once again. The duplicated vectors sent to the mixed fusion submodel and the originals are sent through the strict fusion algorithm. This way all of the same collection vectors are fused in three different ways so that the value of fusion can be analyzed. At this point both the Mixed Fusion and Strict Fusion algorithms must decide to either fuse or forward the collection vectors they receive.

The decision to fuse and/or forward, is based upon the time remaining from the current time until entity *Due_Date*. The assumption made at this point is that time is the primary concern because an end-user has established a time constraint, the *Due_Date*. It is assumed that the information is needed by that time or it may not be useful. While fusion may improve user-needs satisfaction, it may also take too long and not meet end-user time requirements. The mixed fusion algorithm depends upon a user defined global variable, *Fuse_Limit*, which is established as the number of hours preceding *Due_Date* when fusion may make an entity go over *Due_Date*. In other words if *Due_Date*-TNOW <= *Fuse_Limit* then fusion should not be started and immediate forwarding should occur the same as in the no fusion case. If fusion has already begun then the collection vectors that are available should be fused and forwarded. The fusion algorithm used is very simple. All the collection vectors are reviewed and the maximum satisfaction value across all vectors is kept in each user-needs area. This will create a vector of maximum satisfaction that is sent to the Forwarded Reports station in the dissemination submodel. The logic needed to implement this algorithm in Arena is fairly complex and does not lend itself well to wire diagrams.

The strict fusion algorithm works much the same as the mixed fusion algorithm except that it does not use the *Fuse_Limit* variable. Strict Fusion will wait for all collection vectors to arrive until it reaches an entities *Quit_Collect* date. Once a quit collect date occurs then the fuse and forward functions are tripped just as they are in mixed fusion algorithm.

A brief discussion on how Arena performs batching may give the reader some more insight into the logic used within the model. What the model is really doing at this

point is putting the original RFI back together as one entity rather than allow all the separate collection vectors to process individually. For strict fusion the *First_Fuse* node queues up all of the original collection vectors. It does this by sorting all of the collection vectors in the queue according to the *Sort_Rule* prioritization scheme, the same one used in the collection submodel. What this will do is put all the collection vectors from one RFI in order in the queue. All the collection vectors generated by that RFI should be lined up next to each other in the queue. Then Arena matches up entities that have the same Entity.SerialNumber attribute. Recall that this number is unique to all newly created entities, but is retained during all the duplication processes that happened at the Collection submodel branch node, and at all other duplication nodes along the way. The *First_Fuse* queue will match entities with the same *Entity.SerialNumber* together until it has reached its matching quota, which in this case is equal to then *Num_Needs* attribute value. Once there are "Num_Needs" number of entities in queue with the same *Entity.SerialNumber* these entities are bundled together as one entity and released from the queue. In this case the entity created is a temporary one which means that all of the individual data from each of the collection vectors is still available for processing. Once a bundled entity is release from pre-fusion it is sent to the Analysis and Production submodel.

3.5.6 Analysis and Production Submodel

The Analysis and Production submodel is dedicated to data fusion and the resubmission of standing RFIs back into the Intelligence Cycle. This submodel simulates knowledge creation from the multiple data sources tasked in the collection phase. Given multiple collection vectors as our bits of data or information the fusion method used in

this model takes the maximum satisfaction value (*Need_Collect*) in each Need Column across all the collect vectors from each of the resources tasked. The example given in section 3.3, Modifying the Knowledge Matrix Method, discussed this fusion methodology in detail with an example provided. However in this model the fusion algorithm used is not as important as the process established to implement the algorithm. It is the iterative framework established to support this simple maximum satisfaction vector methodology that could be used to support any of the more complex fusion algorithms discussed in Chapter Two. With that in mind let's look at the way that the Arena software was used to model information fusion.

Fusion preparation began at the end of the Processing & Exploitation submodel in the *First_Fuse* node. Note that there is no fusion delay node in this submodel. It was determined that the time spent waiting in the *First_Fuse* queue for all collection vectors generated by a single RFI would account for delay spent fusing information. Also not adding a time delay takes out the model idiosyncrasy of delaying an entity that has only one user-need and really does not require any fusion. All other entities with more than one user-need are batched together in the *First_Fuse* node and still retain all of their separate collection vectors. The algorithm implemented here takes those temporarily bundled entities, separates them once again and does a pair-wise comparison of entities in the time order that they arrived at the *First_Fuse* node. As each of the entities is compared, the maximum value of the comparison is saved as a variable, the *Activity_State_Fuse* variable, for example, represents the maximum *Activity_State_ Collect* value achieved after all pair-wise comparisons are made. Thus, a maximum collection vector over all user-needs is obtained. The final entity for each set of entities

to be fused is directed to the *Final_Fusion* assign node, where the maximum satisfaction vector is assigned as an entity attribute. Then the final entity triggers the *ClearingHouse* assign node, which resets all the *Need_Fuse* variables back to zero so that the next set of collection vectors can be processed.

When the final entity of a set enters the *Second_Fuse* node it completes the same queueing and batching algorithm applied in *First_Fuse*. Note that there will be not time delay here because all of the entities flow into the node at virtually the same instant. All the entities in a set are reformed as one permanent entity with one maximum satisfaction vector (as assigned to the last entity that passed through when the RFI set was split apart). Now that the original RFI or *Opportunity_Collect* is back together and we enter the second phase of the Production and Analysis submodel, resubmitting of Standing RFIs.

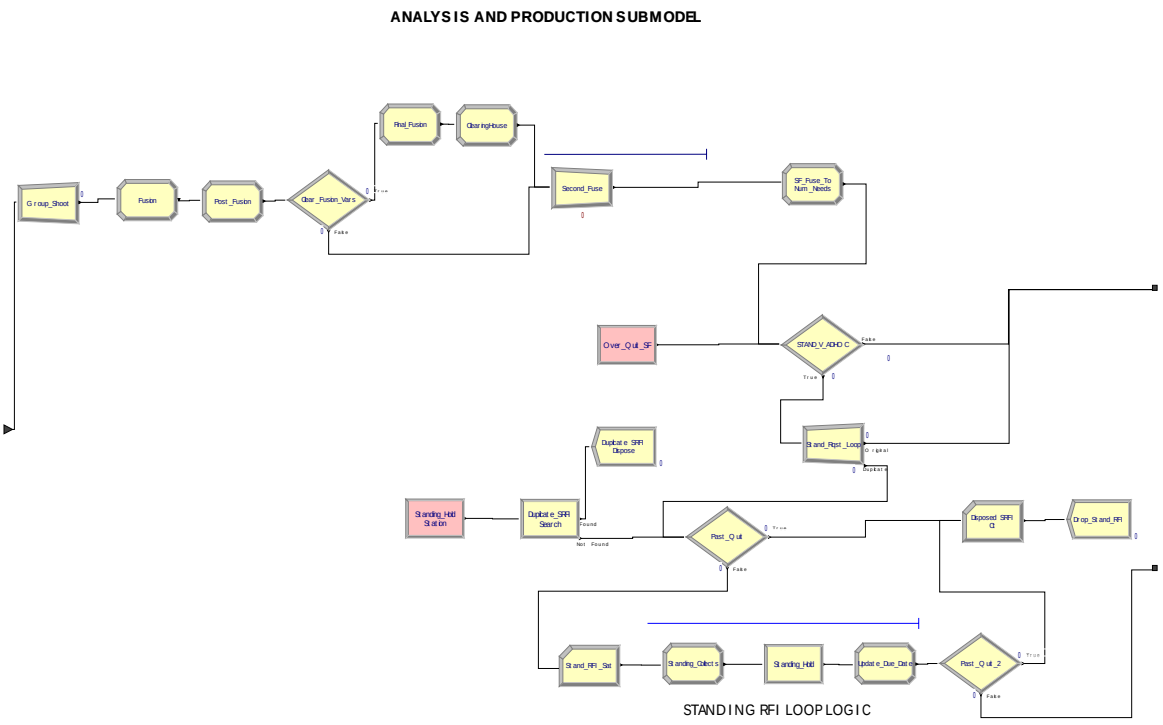

Figure 3-12 Wire Diagram -Analysis & Production Submodel

The first decision-block of this area, *Stand_V_Adhoc*, determines if the request is a Standing RFI or not. If the entity is a Standing RFI then it is sent to the *Stand_Rqst_Loop* separate node where it is duplicated. The original entity is sent out of the Production and Analysis submodel to the end-user(s). With the information sent on its way the duplicate will be sent to a *Past_Quit* decision node. The *Past_Quit* decision node determines if the RFI is no longer needed in the system. If TNOW>= *Quit_Collect* then the RFI has reached the end of its collection loop and is dropped out of the Intelligence Cycle at the *Drop_Stand_RFI* dispose node. If the RFI is still within its collection cycle then it is counted and assigned a *Looping_RFI_Count* attribute so that analysis could be done on the number of times that an RFI loops through the system. Once these administrative details are completed the RFI is held in *Standing_Hold* node until *Due_Date*. The entity is prioritized in the hold queue from lowest to highest Due_Date order. When TNOW >= *Due_Date* a Standing RFI's *Due_Date* the RFI is released from the hold queue. It's new *Due_Date* is assigned at the *Update_Due_Date* node and then it proceeds to the *Past_Quit2* decision node. This node checks to see if the RFI has exceeded its *Quit_ Collect* time while it was held in the *Standing_Hold* node. If this is the case then the RFI is sent to the *Drop_Stand_RFI* dispose node, if not then the RFI is looped back into the Intelligence Cycle at the beginning of the collection submodel. If the entity is not a Standing RFI then it immediately leaves the Analysis and Production submodel for the Dissemination submodel.

3.5.7 Dissemination Submodel

Dissemination is the final step before a user receives information. This model assumes that dissemination is correctly implemented meaning the correct end-users receive the correct information, Figure 3-13. Recall that the information requestor is not always the one who most needs the information acquired. This assumption of correct information delivery is simulated as a simple time delay in this model. The delay is set as a triangular distribution TRIA(0.1,0.5,1.5) this assumes time to type information into a computer and transfer a file, or dial a phone and relay a message.. Normally dissemination is the end of the Intelligence Cycle, or possibly a new beginning as more information supplied usually breeds more questions, (i.e. more RFI's). In this model however the majority of the Dissemination submodel is taken up by automated statistics collection. The Dissemination submodel breaks out statistics into three categories previously mentioned; No Fusion, Mixed Fusion, and Strict Fusion. All three statistics collection areas are setup identically. We will use the No Fusion submodel for our example, Figure 3-14.

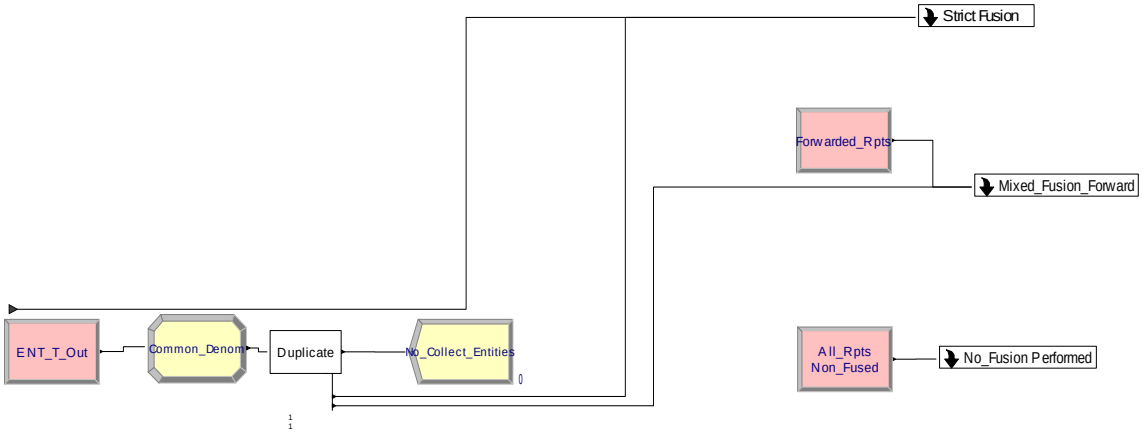

Figure 3-13 Wire Diagram - Dissemination Submodel

In No Fusion, statistics collection counts every non-zero user-need as a collection vector. Recall that one RFI or Opportunity collect may have multiple resource taskings because it can have up six User-needs to satisfy plus a possible library search This means that when the RFI or Opportunity entity was split up according to User-needs at the branch node in the collection submodel, all of the individual tasked need requests were counted separately. This includes all library searches, all requests with null_collection vectors, and all populated collection vectors. From this breakout of total tasking we can see exactly how much simulated collection took place over the course of the simulation run period. The counter *NF_Tol_Entities* at the beginning of the no fusion section will tell us how many total collections were requested.

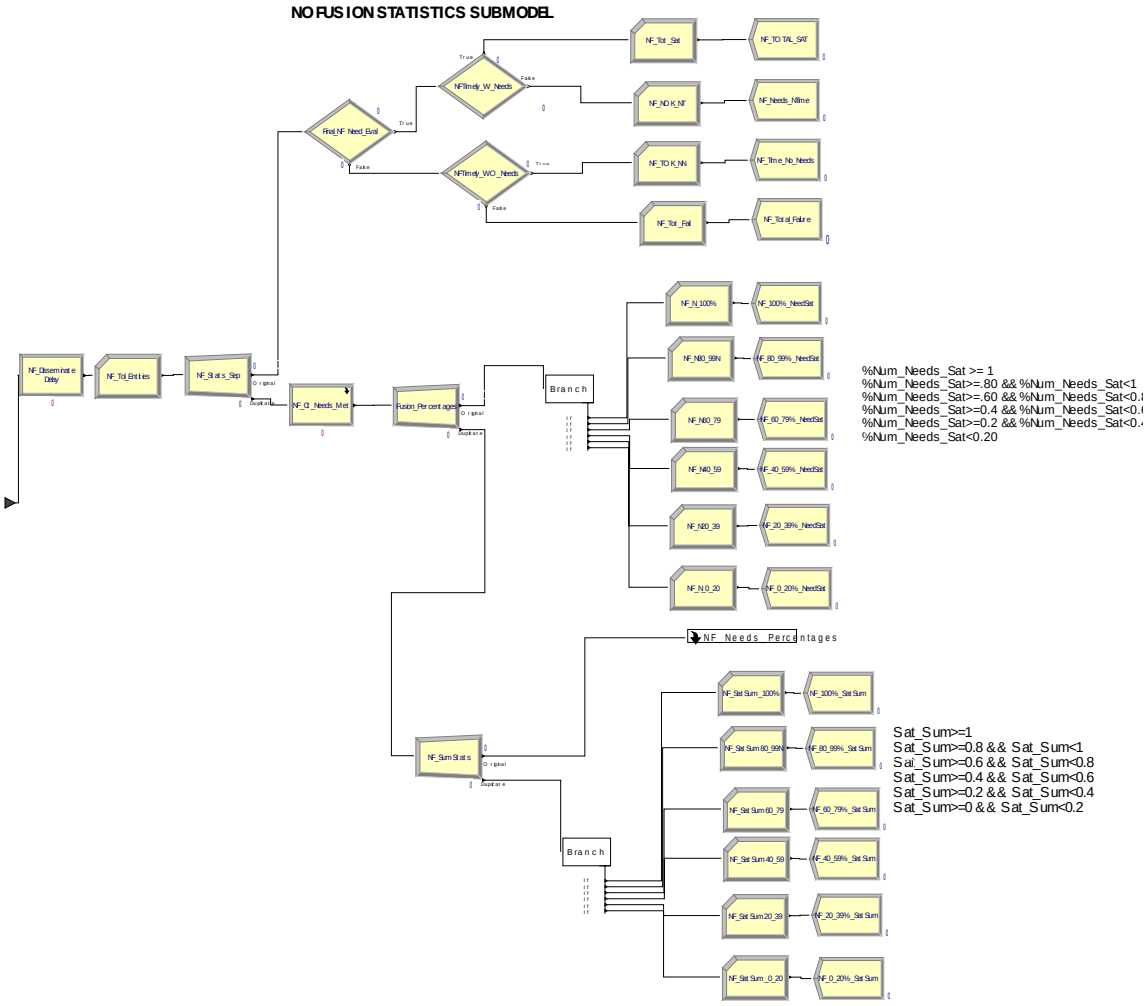

Figure 3-14 Wire Diagram - No Fusion Statistics Collection

Once a total count has been made the entities are duplicated three times so that three different levels of statistical evaluation can be made. The first split will send entities up to the highest level of statistical collection, our top level MOE's. The top four-dispose nodes, left pointing blocks, determine our four levels of general satisfaction:

1) Total Satisfaction, all user-needs satisfied and timely
2) All user-needs satisfied, but timeliness not satisfied
3) Timeliness satisfied, but user-needs not satisfied
4) Total Failure, neither timeliness nor user-needs satisfied

The only difference for Mixed Fusion and Strict Fusion are the types of entities that arrive at each node. Rather than all tasked user-needs requests Strict Fusion receives only fused collection vectors

3.6 *Validation and Verification*

Two specific tests verify this model. The first test was a series of logic tests within the model itself. Drop logic dispose nodes were placed in the model at key logic locations to determine if logic processes were working as anticipated. At the conclusion of each set of test runs, and actual scenario runs, all drop logic sequences returned zero values so no model logic in those key areas was broken. The second test conducted to verify the model's functionality was a series of common sense tests.

The common sense tests followed an entity through the model using single time steps on the model event calendar while monitoring model animation and calculations at each step. The model was run for approximately 10,000 hours. Then animation was paused, and the next entity generated was followed systematically through the entire process. Specific checks made as the entity progressed through the model included verifying appropriate queuing according to the *Sort* expression, routing through the library, time management, needs assessment and intelligence resource processes. Routing followed all assigned attribute values as specified in the Matrix Creation assign node. Finally, the entity was followed through each of the fusion algorithms to ensure that each algorithm was performing correctly. The model displayed no logic aberrations, such as entities stuck in queues indefinitely or entities not queuing correctly. There was only one model idiosyncrasy noted. After the statistics are truncated, see section 4.2 Overcoming Initialization Bias, the Mixed or Strict Fusion algorithms could become the

quantity leader over the No Fusion algorithm. At first, this appeared to be a double counting, error, but as further investigation revealed no double counting occurred. At the one-year truncation point, statistics clear but there are still entities from that initial year in the Mixed and Strict Fusion queues. Those first-year entities are processed together with the newly generated entities in the second year where statistics are collected. In this way the Strict and Mixed Fusion algorithms can become quantity leaders because they have more entities available to process compared to the No Fusion algorithm that have not been processed. This means that between fusion algorithms quantity is not a good performance measure for this model.

The model is subject to validation by the end-user, subject matter experts (SMEs) at the NSSO, Mr. James Kindle, and Dr. Hans Keithley. These two SMEs were consulted during model development to ensure that the model concept reasonably represented the real world processes modeled. The model flow follows the guidelines outlined in Joint Publication 2.0, *Intelligence Support to Operations.* Notional data was created based on the operational and intelligence experiences of the author. Where those experiences were lacking generic assumptions were made. The true validation of this model can be assessed when empirical data is loaded into the model and model runs are compared against real world results.

4. Data Analysis

4.1 *Introduction*

This model was built primarily as a "what if" analysis tool to discriminate between the perceived values of competing intelligence architectures. The discriminating values used were the MOE's; quantity of collections, and user satisfaction. User satisfaction is a combination of user-needs and timeliness requirements. User-needs are based on the knowledge matrix concept of six distinct knowledge areas divided into six levels of satisfaction (refer to Table 3-2 for descriptions). Timeliness is the date, and in the case of this model, the hour by which a request must be satisfied.

The secondary objective for this model was to investigate the value of data fusion. The value of data fusion is measured using the same MOE's as the architecture analysis against three different fusion algorithms; No Fusion, Mixed Fusion, and Strict Fusion. All algorithms operate simultaneously during model runs.

No Fusion considers all collection vectors as separate entities and evaluates them individually against time and user-need requirements. Mixed Fusion works on the concept that fusion should occur within given time constraints instead of immediate individual evaluations. In Mixed Fusion, the decision to fuse or forward data uses a user-defined variable, *Fuse_Limit*, as the cutoff time prior to a collection *Due_Date.* When an entity waiting for fusion hits the *Fuse_Limit* then fusion is terminated and all currently collected information is fused and forwarded. This methodology attempts to stay within the end user's time constraints by gathering, fusing, and sending forward a fused product before the end-user's *Due_Date* deadline. The final fusion method applied, Strict Fusion

holds requests until all collection is completed, or until an entity reached its *Quit_Collect_Date*. Recall that the *Quit_Collect_Date* is after the end-user's *Due-Date*. The *Quit_Collect_Date* is the time after which collection is no longer valuable. The data collected up to the *Quit_Collect_Date* point is fused, and the information disseminated to the end-users. These three fusion methodologies measure the value of fusion for each of the following scenarios.

1) A baseline case of one year of standard, peace-time intelligence collection

2) A year long heightened intelligence environment, possibly a wartime scenario or a perceived threat scenario

3) The decision to purchase an upgraded imagery satellite, or twenty-four more Unmanned Arial Vehicles (UAVs), based on a years worth of Baseline simulation.

4.2 *Overcoming Initialization Bias*

Once the model was created, validation and verification was performed as outlined in Chapter 3, one more test had to be done before scenario modeling could occur. Initialization bias had to be removed from the model. Initialization bias occurs when a process starts from an empty and idle state and generally biases the model statistics by making processes appear to react more quickly at first because they are not initially occupied. As time passes the process may begin to slow down as the model begins to introduce more entities into the system. The objective is to find out when the system begins to level out by examining a selected system performance value. The value of interest for this model is the number of Entities currently in processes and is referred to as Work In Progress (WIP). Arena conveniently calculates WIP measures for all entity types in a model. WIP measured for each *Entity.Type* (*Rank 1-5* and *Opportunity*

_*Collect*) were used to determine when initialization bias was mitigated in ten long term, 3650 day replications. Several key results were noted from these ten initialization bias test runs;

1) The longest average time to overcome system bias was approximately 365 days. As displayed in Figure 4-1. After one year of steep incline of initialization bias fades and the system levels out. Some further increase occurs for the next six months and then the system seems to level out.

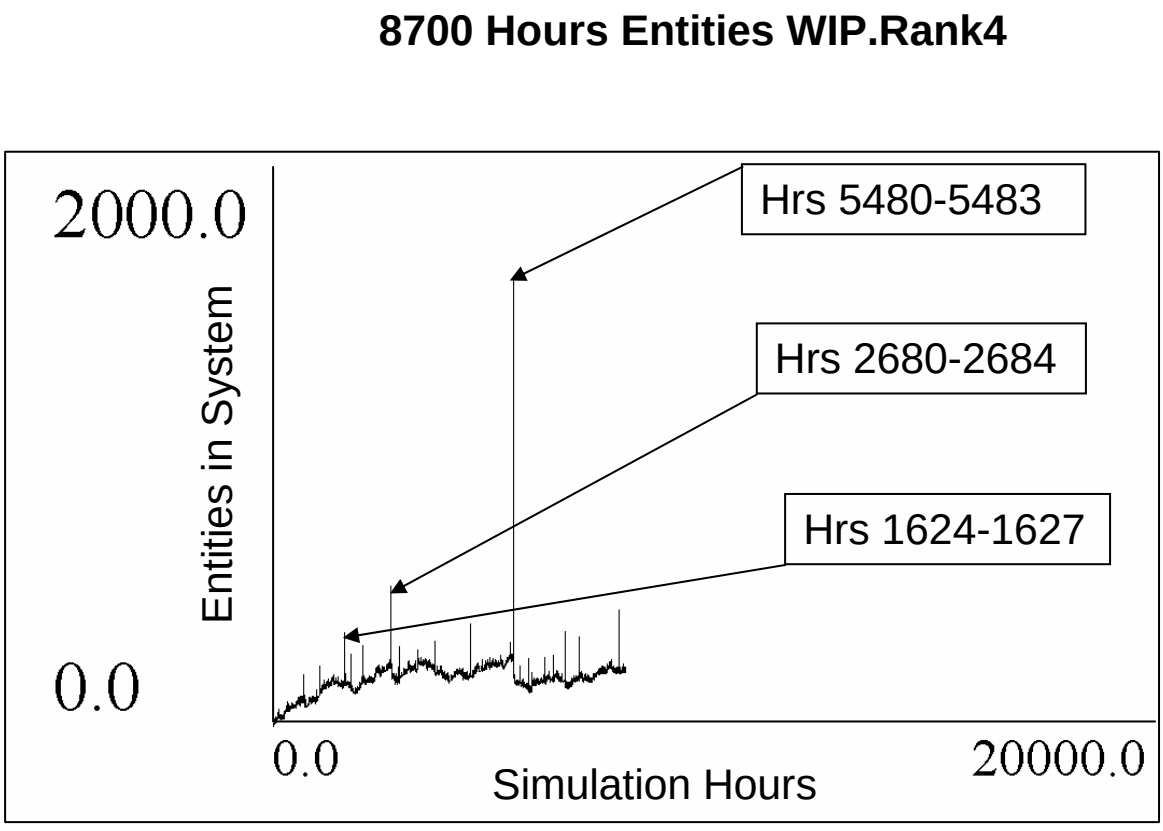

Figure 4-1 Initialization Bias

2) It is intuitive, but should be mentioned, that the WIP calculations for the different entity types are not independent, but rather very highly correlated. The correlation occurs due to the queuing priority that higher ranked entities enjoy. The time a Rank 3 entity

spends in queues throughout the system is directly related to the number of Rank1 and Rank2 entities that are currently in the system. This correlation causes increasing WIP trends and the sharp spiking behavior displayed by lower ranking entities. The sharp spikes noted in Figure 4-1 are the results surges in new arrivals and/or a sudden dump into the system of Type 2, Time Dependent RFIs, from the Time Management sub-process. In Figure 4-1 the spikes appear as single point masses, however, further investigation showed that each spike did have some mass associated with it as shown in Figure 4-2.

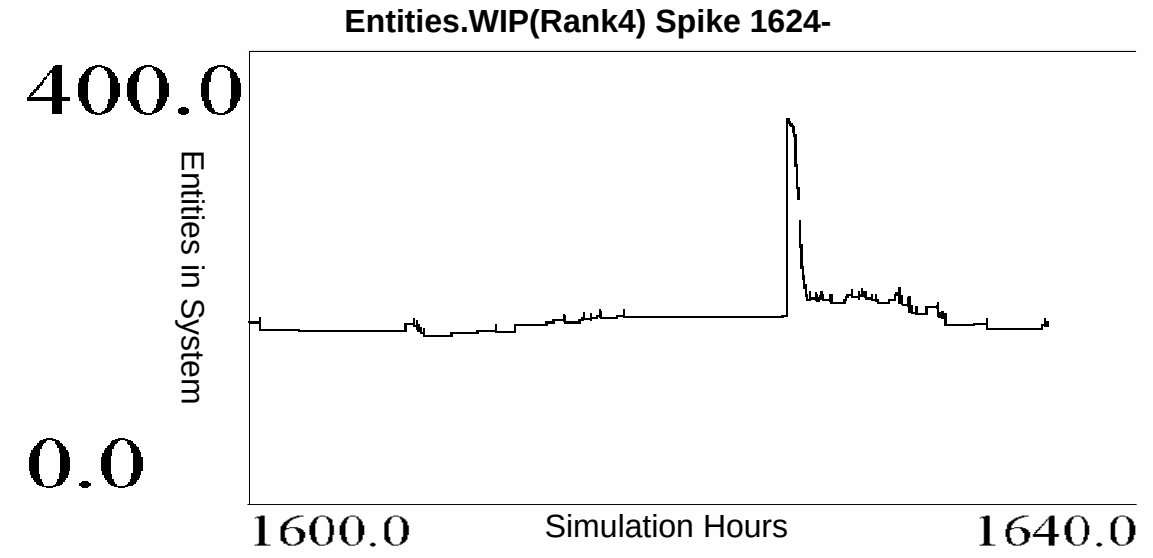

Figure 4-2 WIP Spiking Details

3) WIP estimates are also skewed because of a Standing RFI and a Time Dependent RFI bias. WIP counts all entities in the system, whether in a process queue/delay or in a hold queue. With Time Dependent RFIs and Standing RFI's much waiting, or hold time occurs until the right time to collect. This waiting is not a backlog, but rather a planned hold. These necessary holds tended to skew the simple Arena WIP statistic making work in progress appear high than resource utilization rates would indicate.

4) Use of a one year warm-up period places an appropriate number of initial entities in queues for Mixed Fusion and Strict Fusion, but will have little effect on No Fusion.

The bottom line is that 365 simulation days were selected be run and thrown out before any statistics collection occurs. This means that to study one year of a baseline scenario the model must simulate two years, collecting statistics only on the second year. Every replication run accounts for this warm-up period.

4.3 *Statistical Measures Defined*

This section will discuss the three MOE's; quantity, timeliness and user-needs satisfaction, used to compare the algorithms and architectures in this model.

4.3.1 Measuring Quantity

The Quantity MOE assumes that more is better because the more collection vectors an intelligence architecture is able to process, the more chance there is that an intelligence gap will be filled by those collection vectors. This model assumes that the fusion function will accommodate greater information flows without any slow downs or other adverse effects. The obvious common denominator for measuring quantity is the final number of collection vectors processed at the conclusion of each simulation run.

Counting collection vectors at the conclusion of each simulation run seems innocuously simple at first, but recall that each RFI or *Opportunity_Collect* has a randomly generated user-needs vector. This random vector determines how many resource requests (i.e. collection vectors) are formulated. The No Fusion algorithm is the most straightforward as it already counts every collection vector separately. This means if an RFI has a user-needs vector of (1,0,0,3,2,4), the No Fusion algorithm will receive four different collection vectors associated with the four non-zero User Needs. No

Fusion will evaluate each of the four collection vectors against the original user-needs vector (1,0,0,3,2,4) to see how much satisfaction each report fulfilled (0-100%). This straight forward count of all collection vectors is simple for the No Fusion case, four needs equals four collection vectors. A problem arose however, when we looked at the Mixed Fusion and Strict Fusion algorithms. Both of these algorithms mapped the four collection vectors into a single maximum satisfaction vector. This single maximum satisfaction vector is the collection vector for each of these fusion methodologies. The solution applied to this issue was to count the number of collection vectors fused together and then when statistics were processed duplicates of the maximum satisfaction vector were made so that all collection vectors were accounted for.

4.3.2 Measuring User Satisfaction

Total user satisfaction is a combination of meeting user-needs within set time constraints. We will look at the issues involved in creating user-needs and in terminating fusion or collection due to time constraints. No weighting scheme is assigned to timeliness or user-needs satisfaction. The model user must determine what is more important, on time reports or higher satisfaction reports, these two measures will only provide comparative probabilistic values, not conclusive answers.

4.3.2.1 Measuring Timeliness Satisfaction

Timeliness was discussed in detail in sections 3.5.5, 3.6, and 3.7. The attributes which define timeliness are *Due_Date* and *Quit_Collect_Date*. Timeliness is a user-defined constraint, which bounds total satisfaction.

4.3.2.2 Measuring User-Needs Satisfaction

There are a number of ways to measure satisfaction. Two methods are presented here; a discretized satisfaction ratio, and a continuous satisfaction ratio. In the discretized satisfaction ratio, originally discussed in Chapter 3, the six knowledge areas of a user-needs vector are compared against all applicable collection vectors to see how many knowledge areas were satisfied. This means that if five out of six vectors, or 5/6 of our user needs are satisfied the end-user satisfaction level is 83.5%. Because there are only six knowledge areas in the knowledge matrix the maximum satisfaction fidelity of this discretized methodology is 16.5% (i.e. 1/6). This methodology becomes worse as fewer non-zero user-needs are present in the user-needs vector. For example, if there are only two non-zero user needs in a user-needs vector we can only attain a satisfaction fidelity levels of 0%, 50%, or 100% (i.e. 0/2, 1/2, 2/2). Realizing the problem introduced by this discretized satisfaction ratio, a continuous satisfaction ratio was developed. The continuous satisfaction ratio takes all of the partially and totally satisfied user-needs areas sums them, and then divides by the number of user-needs. Let us say that our user-need vector is (0,3,0,0,3,0) and our maximum satisfaction vector after fusion is (1.00,2.96,0.00,2.50,1.83,0.00). In our example, this would lead to 2.96/2 + 1.83/2 = 0.798, the discretized satisfaction ratio for this example would be 0%. This means that the user has about eighty percent of their request vice nothing. Due to this discovery, the continuous method of assessing user-need satisfaction will be the measurement standard.

Figure 4-3 further illustrates this point.

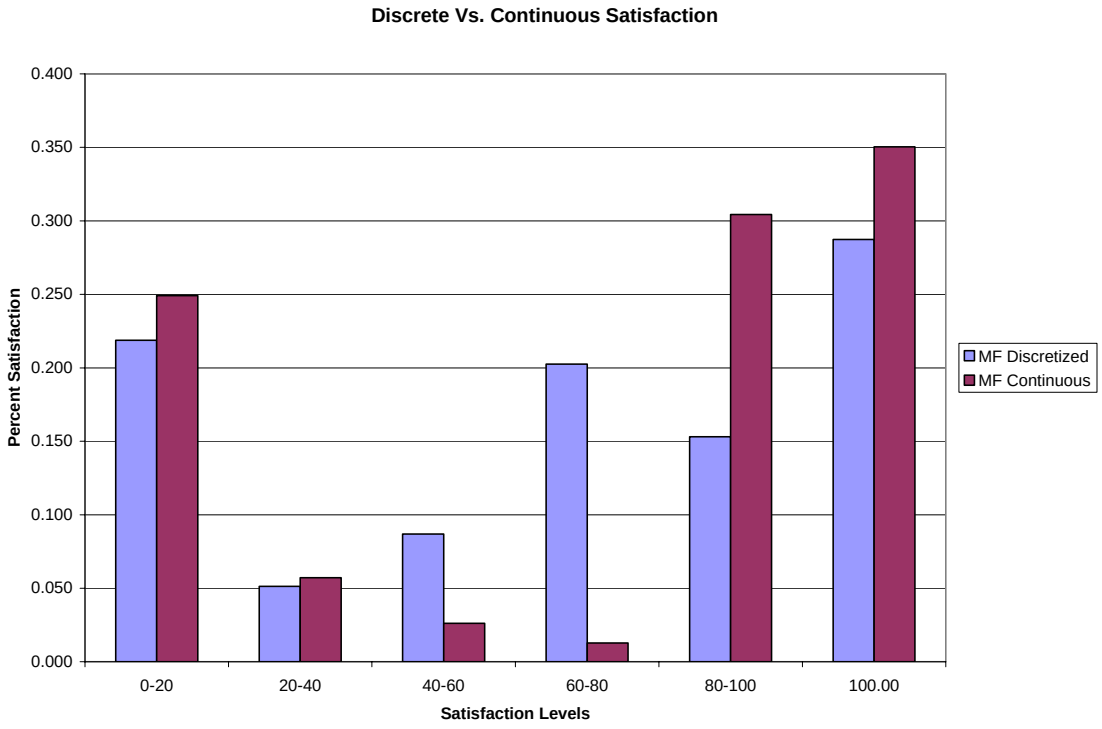

Figure 4-3 Discrete Vs. Continuous Satisfaction Ratios Measured

4.4 *Creating a Baseline Scenario*

The baseline scenario created assumes that no extraordinary events such as a war or terrorist attack have occurred in the period of interest. Intelligence requests and collections are operating within acceptable parameters. Appendices B, C, D, and E list all baseline scenario parameters. Refer to these appendices for any questions regarding entity arrival rates, user-needs distributions, collection resource probability distributions, resource capacities, or resource schedules.

4.4.1 Baseline Results

All results are based on a sample size of n = 50 replications. Each replication was two years long, however, the first year of statistics collection was removed to compensate for initialization bias. Each of the Measures of Effectiveness are analyzed and the results compared between the three fusion algorithms.

Due to the notional nature of the collection inputs and an overestimation of system capabilities by the author, total satisfaction results are low for all of the examples used. However because the data is notional the accuracy of the results is not truly relevant. These same system capabilities are used across all scenarios and architectures. The true relevance of these examples is the fact that differences between scenarios and architectures can be compared and analyzed. To obtain true, accurate results parameter changes could be made to reflect empirical data

4.4.1.1 Quantity

Figure 4-4 illustrates why quantity is not a good MOE to use when making comparisons between fusion algorithms. The intervals were too close for our paired-t-test to pick up any significant statistical differences between the three algorithms. In addition, as discussed in section 3.9, Validation and Verification, the quantity leader in Figure 4-4 is the Strict Fusion algorithm. This is a counter intuitive result because the No Fusion algorithm must process collection vectors quicker than the other two algorithms. The reason for this counter intuitive result lies in the fact that the mixed and strict fusion algorithms begin the post truncation point with entities already in queue due to our warm-up period. This causes the quantity comparison to be biased in their favor, because they have more entities to count than the No Fusion algorithm.

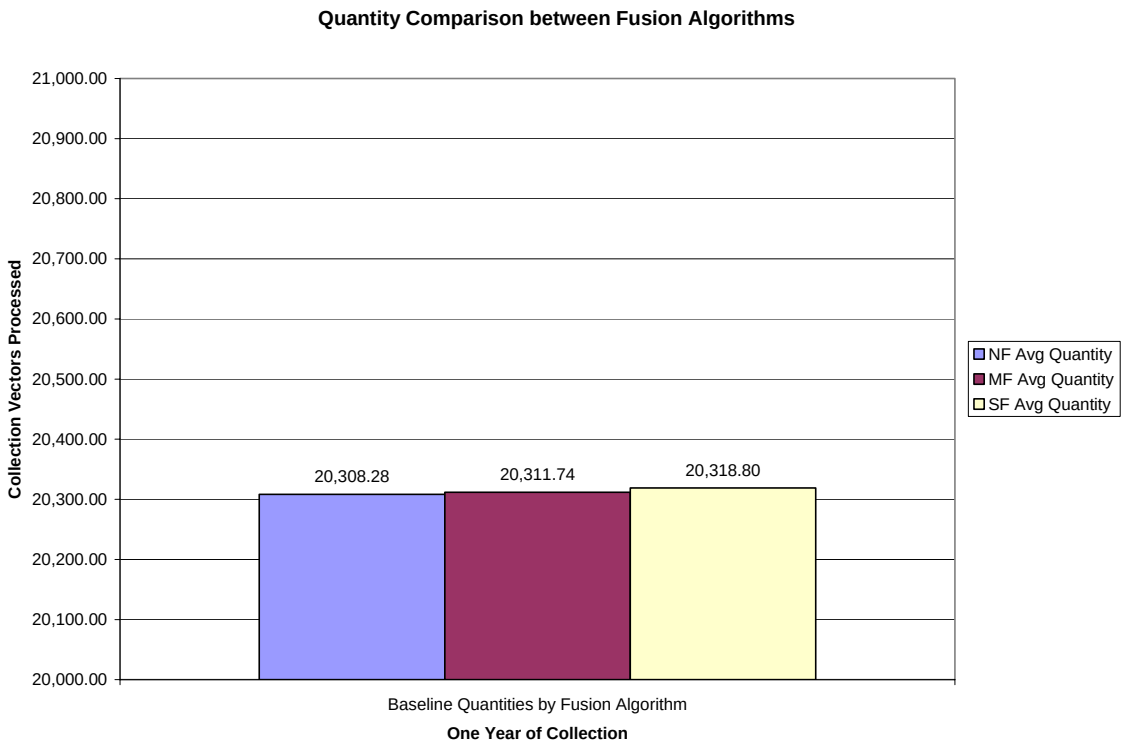

Figure 4-4 Baseline Quantity Measured

4.4.1.2 Baseline Total Satisfaction

Figure 4-5 illustrates that the Mixed Fusion algorithm provides the highest total satisfaction percentage at 9.3%. Total satisfaction is a discrete measure in this model. When all user-needs are met before a *Due_Date* then an end-user is totally satisfied according to the rules applied in this model. Figure 4-5 is a simple quantity count of the number of collection vectors that entered the total satisfaction bin for each algorithm during statistics collection. As we will see the mixed fusion algorithm consistently performs better than the other two algorithms when total user satisfaction is measured.

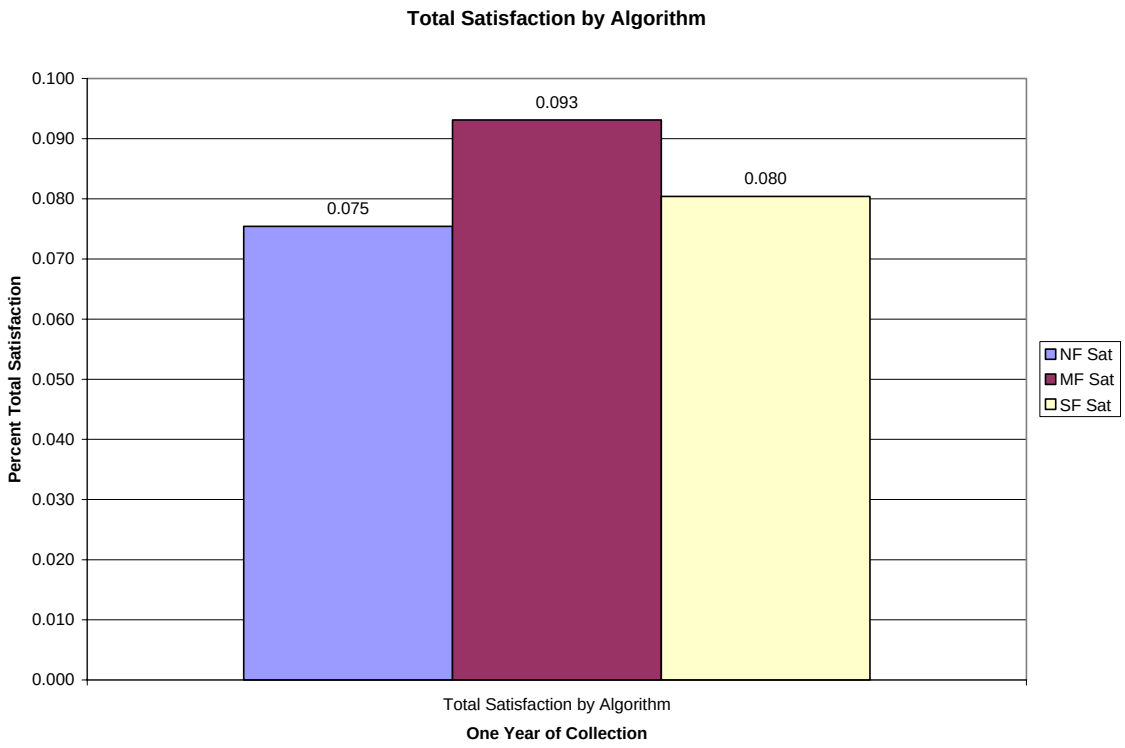

Figure 4-5 Baseline Total User Satisfaction

4.4.1.2.1 Timeliness

Figure 4-6, which shows No Fusion as the timeliest algorithm, validates the model's performance because it makes intuitive sense. No Fusion is always the timeliest algorithm because it has none of the delays built into it that the mixed and strict fusion algorithms possess. More importantly however, note that the mixed fusion algorithm performs almost as well as the no fusion algorithm. A paired t-test performed on the no fusion percent timely and the mixed fusion percent timely found that there was statistical difference between the two points.

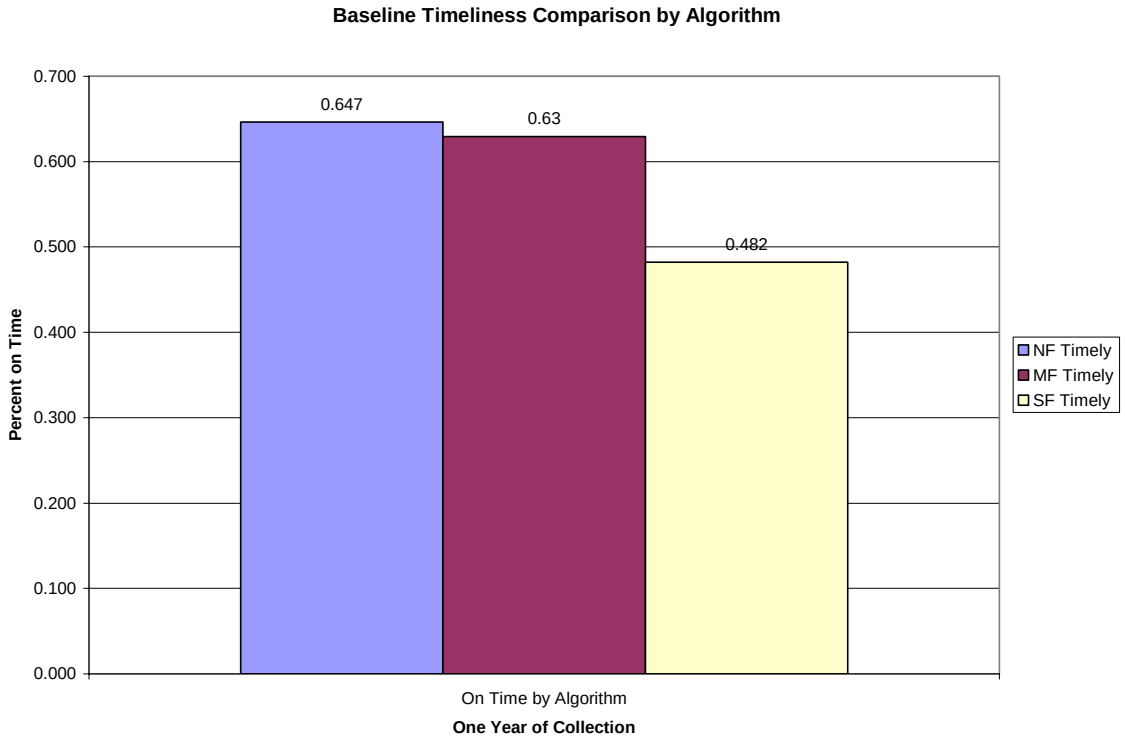

Figure 4-6 Baseline Timeliness Comparison by Algorithm

4.4.1.2.2 User-Needs

Figure 4-7 compares the percent of user-needs satisfaction, using the continuous satisfaction ratio method. Strict fusion obviously provides more user-needs satisfaction with over 75% of the collection vectors being satisfied at the 80% or better satisfaction level. Despite this high level of user-need satisfaction, the strict fusion algorithm is not timely enough meet a higher total satisfaction percentage. The mixed fusion algorithm with approximately 65% of its collection vectors meeting the 80% or better satisfaction level fairs better in a total satisfaction scenario.

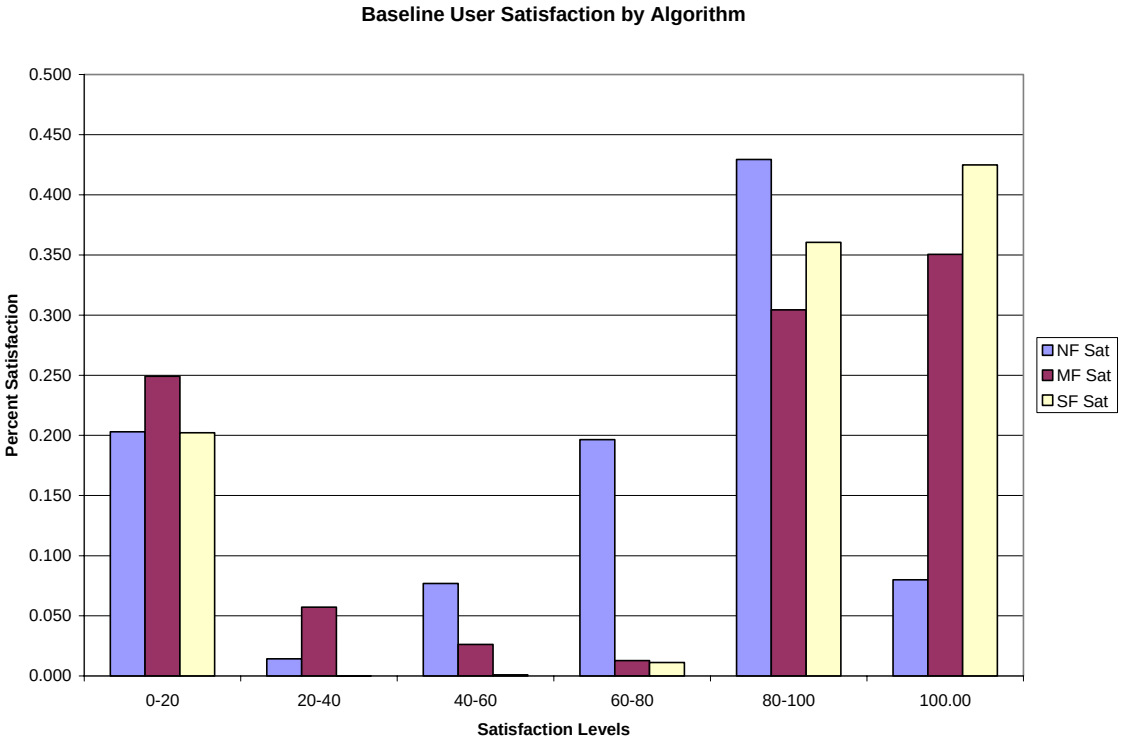

Figure 4-7 Baseline User-Needs Satisfaction by Algorithm

4.4.2 Baseline Inferences

The baseline scenario will be the yardstick against which other architectures and intelligence scenarios are measured. After running and analyzing this baseline scenario the following observations were made:

1) Quantity is not a good measure of effectiveness between fusion algorithms because of the inherent inequality between the base number of entities.

2) Timeliness percentages work to compare both architectures and algorithms.

3) Total satisfaction percentages, and user-needs satisfaction levels appear to be good measures to use for comparing both architectures and algorithms as well.

4.5 *Effect of a Heightened Perceived Threat Environment*

A heightened perceived threat environment is simulated by increasing the arrival rates of Rank1, Rank2, and *Opportunity_Collect* entities. The assumption made here is that more end-users will perceive their needs as urgent or critical, and if there is some sort of threat present then *Opportunity_Collects* will increase because there will be more potential to discover information of intelligence value.

Rank1 arrival rate was increased to Poisson(λ=4). Rank2 arrival rate was increased to Poisson(λ=3), and the *Opportunity_Collect* arrival rate was increased to Poisson(λ=2). No other parameters were changed within the model. Fifty model simulation runs were performed in the same manner as the baseline scenario.

4.5.1 Heightened Perceived Threat (WAR) Results

Intelligence appears to operate more efficiently in the wartime scenario. This seems a bit odd at first glance because the system is more saturated than it was for the base line scenario. What this means is that the system had a considerable amount of slack in the baseline scenario and that some high quality resources were not being efficiently tasked in the baseline scenario as well. Further analysis shows that the system operates more efficiently as well because more high priority requests (Rank 1 & Rank2) are in process. This deluge of high priority requests will speed up some of the model processing, and force the selection of higher quality resources. Often higher quality resources are requested during wartime and due to this assumption, we may be experiencing higher user-need satisfaction levels. This wartime combination results in improved timeliness and total satisfaction percentages. As before the mixed fusion

algorithm continues to perform the best in the area of total satisfaction, therefore the mixed fusion algorithm will be the algorithm of choice to compare the performance of this heightened perceived threat scenario against the baseline scenario.

4.5.2 WAR Quantity

With the increased arrival rates applied to this model, it makes sense that the quantity of collection vectors is significantly increased. Figure 4-8 shows that the number of collection vectors processed in the mixed fusion war scenario is almost double that of the collection vectors processed in the mixed fusion baseline scenario.

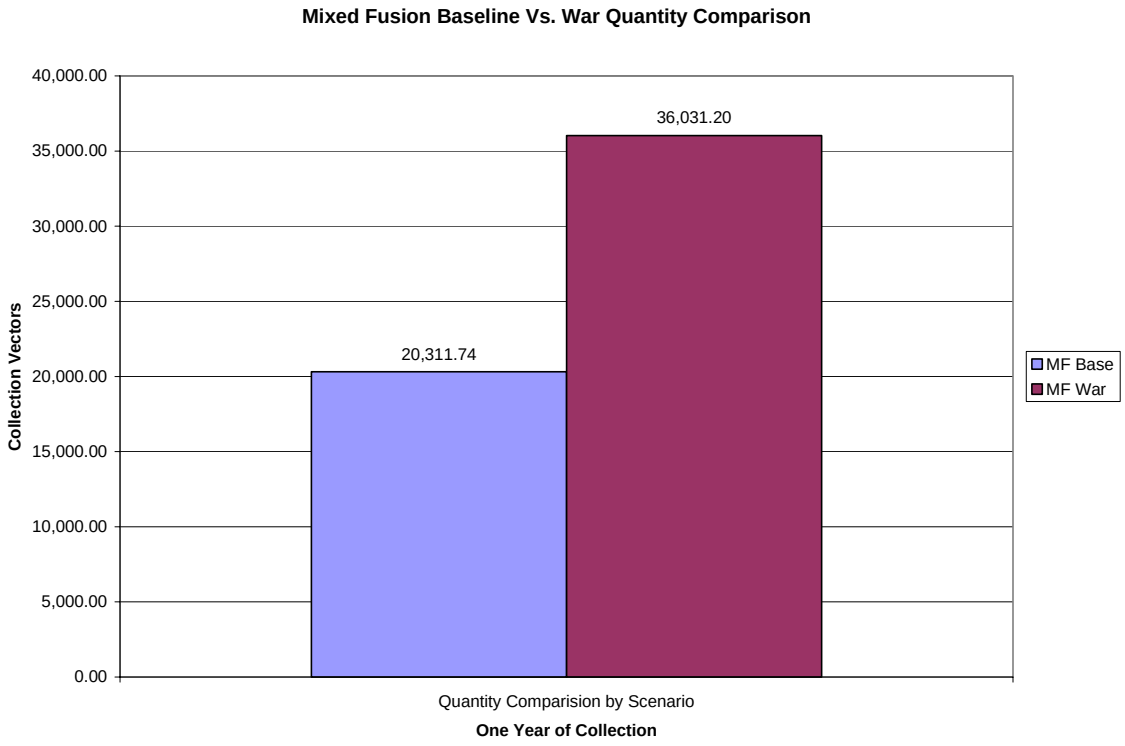

Figure 4-8 Mixed Fusion Baseline Vs. War Quantity

4.5.3 WAR User Satisfaction

Mixed fusion continues to be the total satisfaction leader among the three algorithms in the wartime scenario shown in Figure 4-9. It remains the total satisfaction leader for the same reasons that it was the leader in the baseline scenario. It is timelier than strict fusion and has a higher percentage of satisfaction than no fusion.

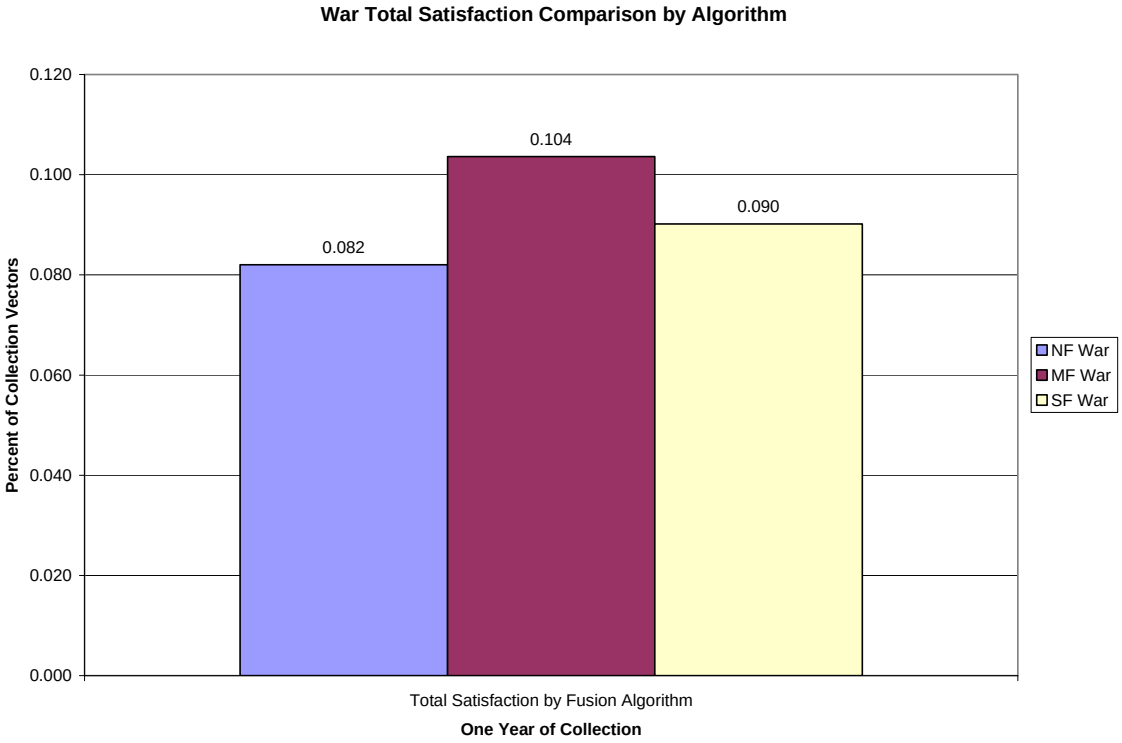

Figure 4-9 War Total Satisfaction Algorithm Comparison

4.5.3.1 WAR Timeliness

The war scenario timeliness bar charts presented the same information as the baseline scenario comparison between algorithms. The same stair step pattern was noted with No fusion as the timeliest (73% timely), followed by mixed fusion (70.8% timely), and finally strict fusion with (53.5% timely). All algorithms had a significantly higher

ratio of timely reports in the war scenario compared to the baseline scenario, as illustrated for the mixed fusion case in Figure 4-10.

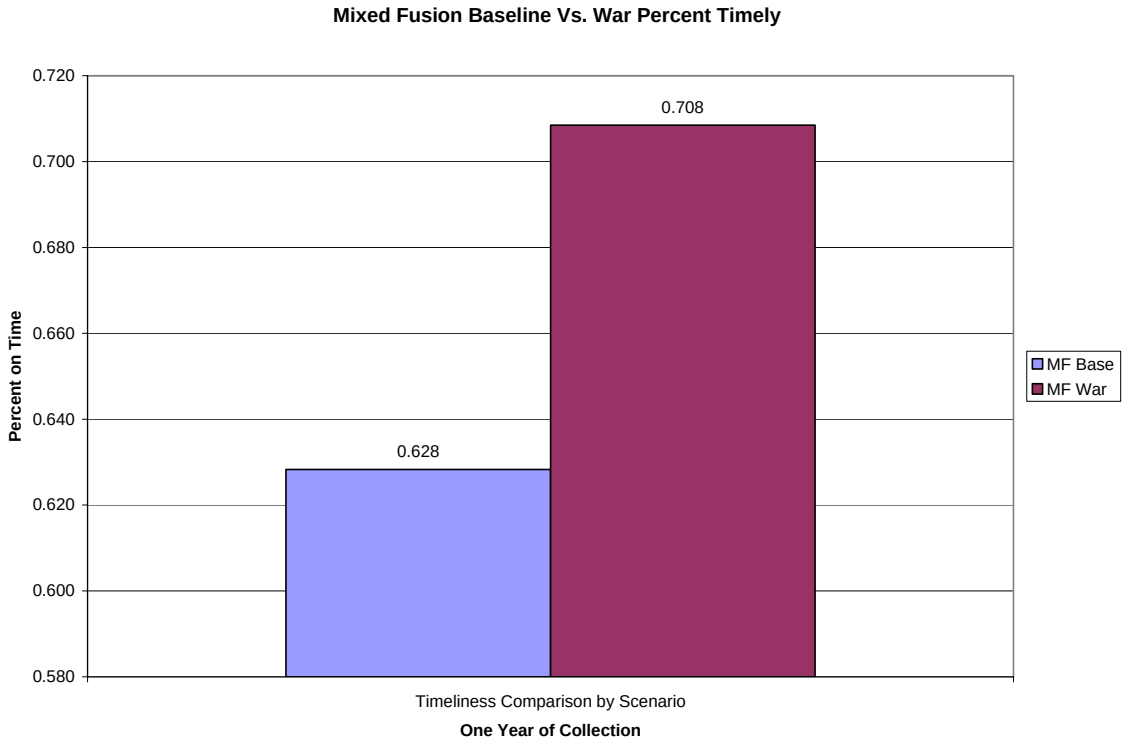

Figure 4-10 Mixed Fusion Base Vs. War Timeliness Comparison

4.5.3.2.1 WAR User-Needs

User needs, like timeliness, plotted similarly to the baseline scenario so once again the more interesting plot is the comparison of the mixed fusion war scenario to the mixed fusion baseline scenario. Figure 4-11 shows clearly that user-needs are more often satisfied at the 80% or higher level in the war scenario than in the baseline scenario. This result was not expected. Extra time was spent attempting to discover the reason why the system would perform in this manner. The most likely reason was already mentioned,

the higher priority requests and collections using a greater percentage of the higher fidelity assets.

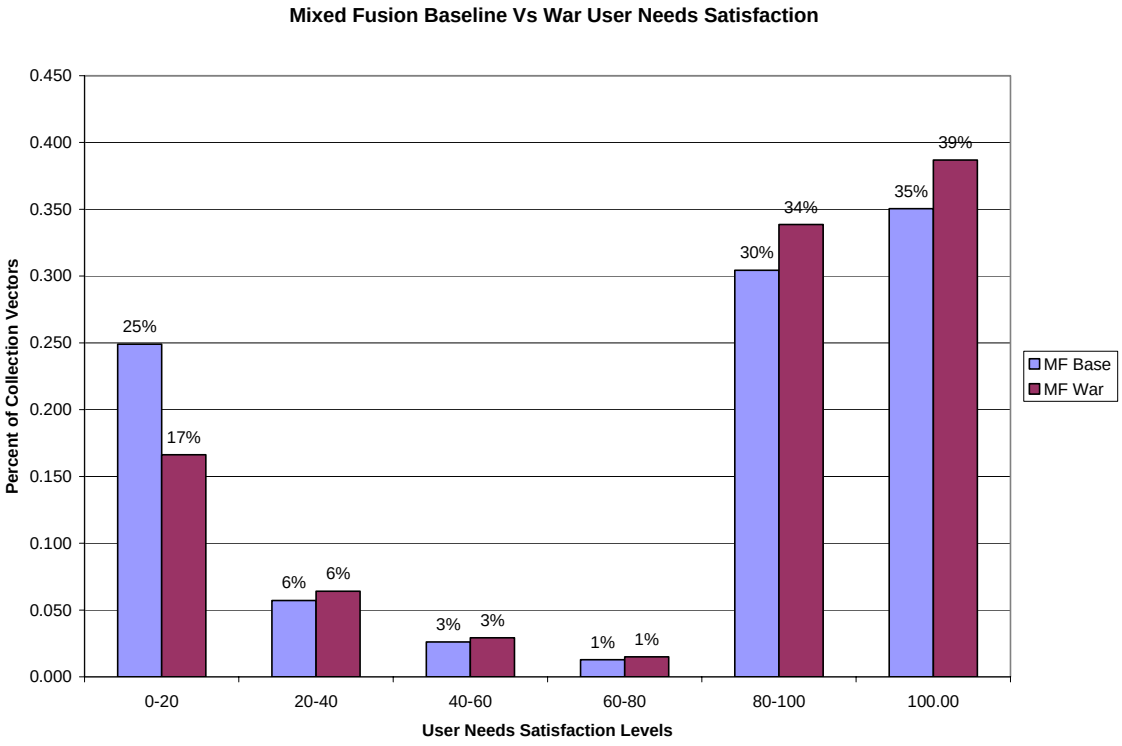

Figure 4-11 Mixed Fusion Baseline Vs. War User Needs Satisfaction

4.5.2 Heightened Perceived Threat Inferences

Curiously, this model does seem to emulate the real world in this wartime scenario. It always seems that in times of crisis more is able to be done than was done in peacetime. This model captures that phenomenon. More entities were processed with a higher percentage of them on time, and with a higher level of user needs satisfaction. All mixed fusion comparisons between the wartime and peacetime scenario were verified using a paired t-test at a 90% confidence interval, and all tests showed that the statistics

compared possessed significant statistical differences. Refer to Appendix E for spreadsheets detailing the paired t-test calculations.

4.6 *Comparing Intelligence Architectures*

Two different architectures are defined, analyzed, and the model MOEs compared in two separate model architecture constructs. The two intelligence architectures evaluated are the purchase of an upgrade to our imagery satellite (IMINT1), or the purchase of 24 more Unmanned Arial Vehicles (UAV), IMINT2.

The imagery satellite upgrade was modeled by modifying IMINT1 in three ways:

1) *Location_Collect* distribution changed from TRIA(2,4,5) to TRIA(3,4,5);

2) *Identity_Collect* distribution changed from TRIA(0,2,4) to TRIA(1,3,4);

3) Capacity schedule for IMINT1 upgraded to 500 entity capacity vice 250.

To model the purchase of 24 new UAV's the capacity of the IMINT2 schedule was adjusted to 30 maximum, a 500% increase, which means that the availabilities for each time scheduled time period were raised 500% as well (i.e instead of 2 available at 1200 there are now 10 available).

4.6.1 Comparing Architectures Results

While the actual results from this architecture comparison are based on strictly notional data they can still give us some insights about our notional system. The results seem to indicate that purchasing the satellite upgrade is a slightly better decision than purchasing a greater number of UAVs. We will investigate these notional results further in each section to see why this might be.

4.6.1.1 Quantity

The notional results from this comparison were particularly surprising. It appeared at first from Figure 4-12 that buying either of the new capabilities would lower the quantity of collected reports. Further analysis done with Figure 4-13 and by performing a paired t-test showed that among the three configurations no significant statistical difference could be detected at a 90% confidence level. As Figure 4-13 shows the variance in the system makes it impossible to tell which model process more collection vectors on average. Recall that CRN's were used to reduce the variance due to different random number draws. Both architectures modeled had the exact same RFI and Opportunity arrival rates, as well as the exact same user-needs requirements and

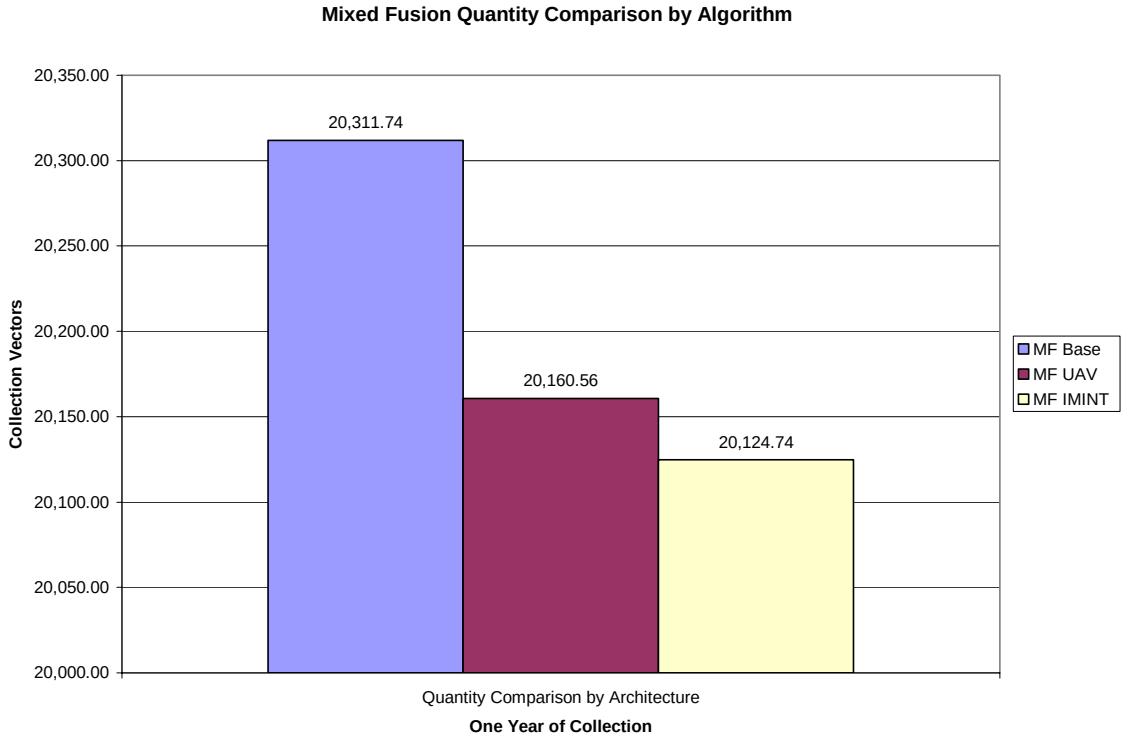

Figure 4-12 Mixed Fusion Quantity Architecture Comparison

administrative attributes. The random numbers will remain in synchronization until resources are allocated in the collection submodel. The variance is reduced because the differences in the modeled systems are due to the actual architecture changes made not because of differing random number streams. The scatter plot in Figure 4-13 shows the confidence intervals of the three different architectures overlap so we cannot determine any significant differences in quantity between the three algorithms.

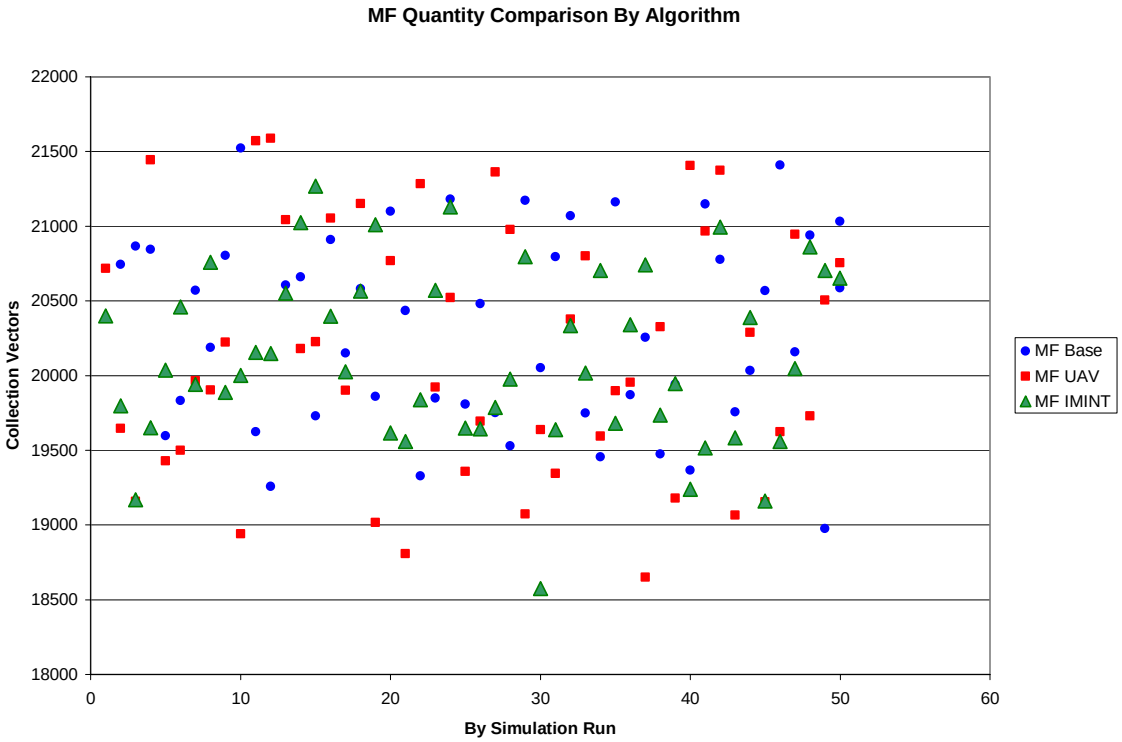

Figure 4-13 Scatter-Plot of MF Quantity Architecture Comparison

4.6.1.2 User Satisfaction

The IMINT1 upgrade choice in Figure 4-14 appears to provide the best total satisfaction percentage, although it is by no means a large improvement over either other architecture. The paired t-tests performed did show that statistical differences were

present. The mean total satisfaction displayed by the IMINT1 option was clearly higher than the means of the other two architectures.

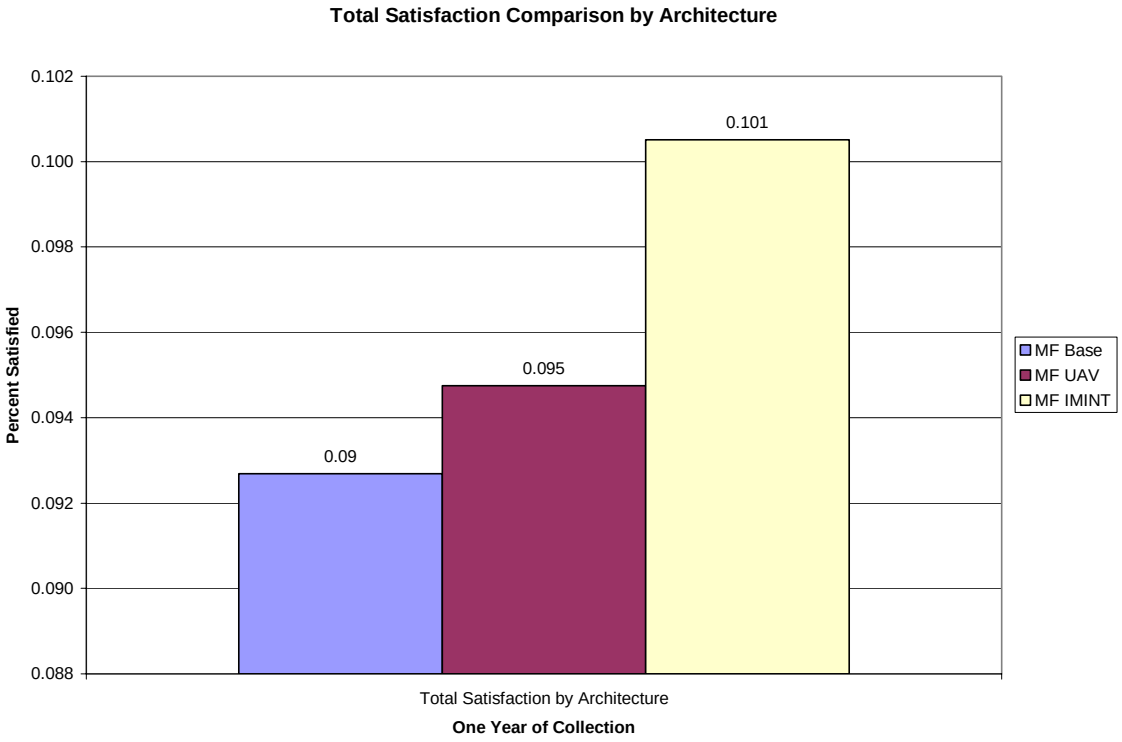

Figure 4-14 Total Satisfaction Architecture Comparison

4.6.1.2.1 Timeliness

Figure 4-15 shows that all three architectures are timely with about 63% of their collection vectors. There is no significant statistical difference between the three architectures at the 90% confidence level.

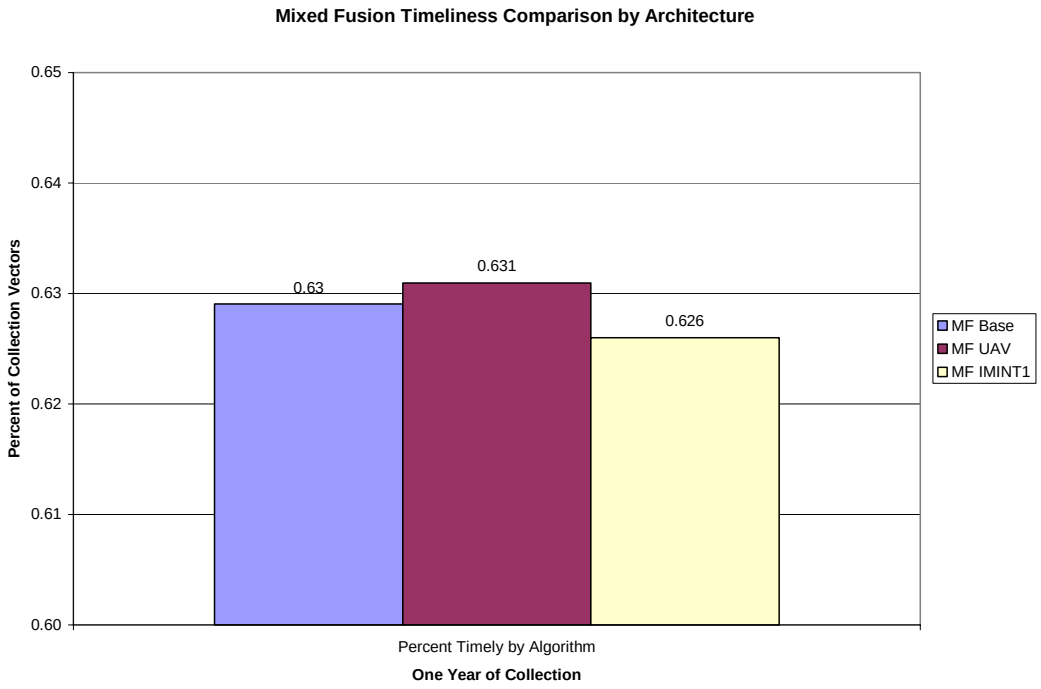

Figure 4-15 Timeliness Comparison between Architectures

4.6.1.2.2 User-Needs

Figure 4-16 shows why the IMINT1 option appears to have a higher total satisfaction percentage than the other two architectures. User-needs have a slightly higher chance of being totally satisfied with the IMINT1 architecture.

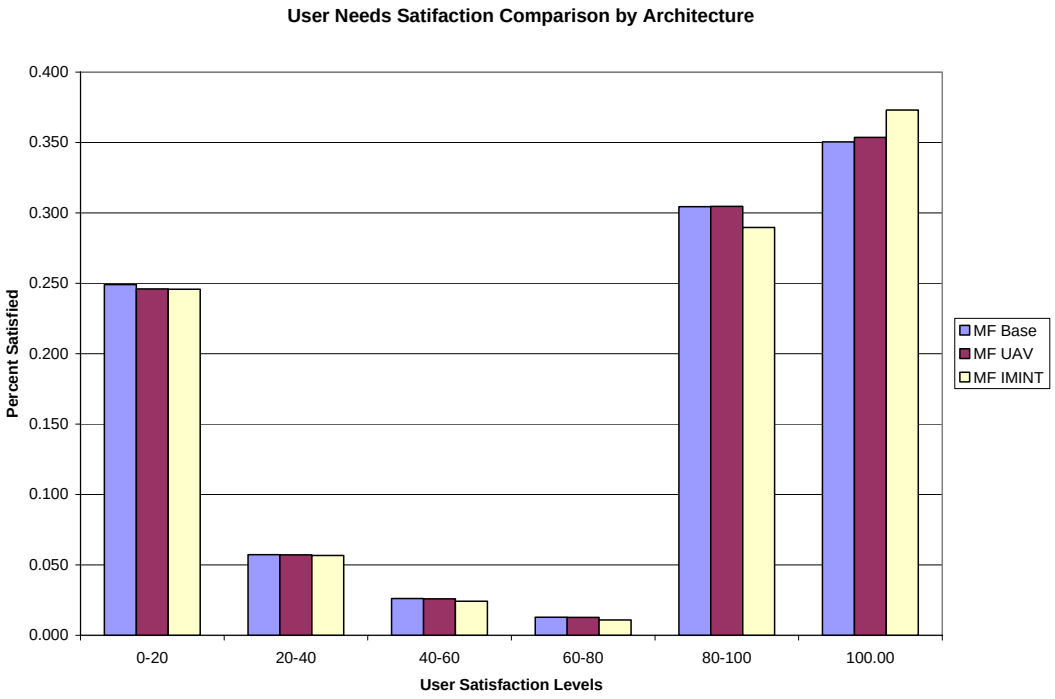

Figure 4-16 User Needs Satisfaction Comparison by Architecture

4.6.2 Comparing Architecture Inferences

The decision to purchase the improved satellite architecture seems to be the correct answers because it provides a better probability of satisfying user-needs. There is too much variance in the system and the means are simply too close to make any sound decisions based on timeliness or quantity at this point. As a final check to see which architecture performs better in its specialty, the Location_Need satisfaction statistics were compiled and plotted in Figure 4-17. The comparison of the two architectures is close, but the IMINT architecture still appears more capable, but only slightly so. The question for the decision maker now is, do the marginal gains accrued over the Baseline scenario warrant purchasing the IMINT1 upgrade?

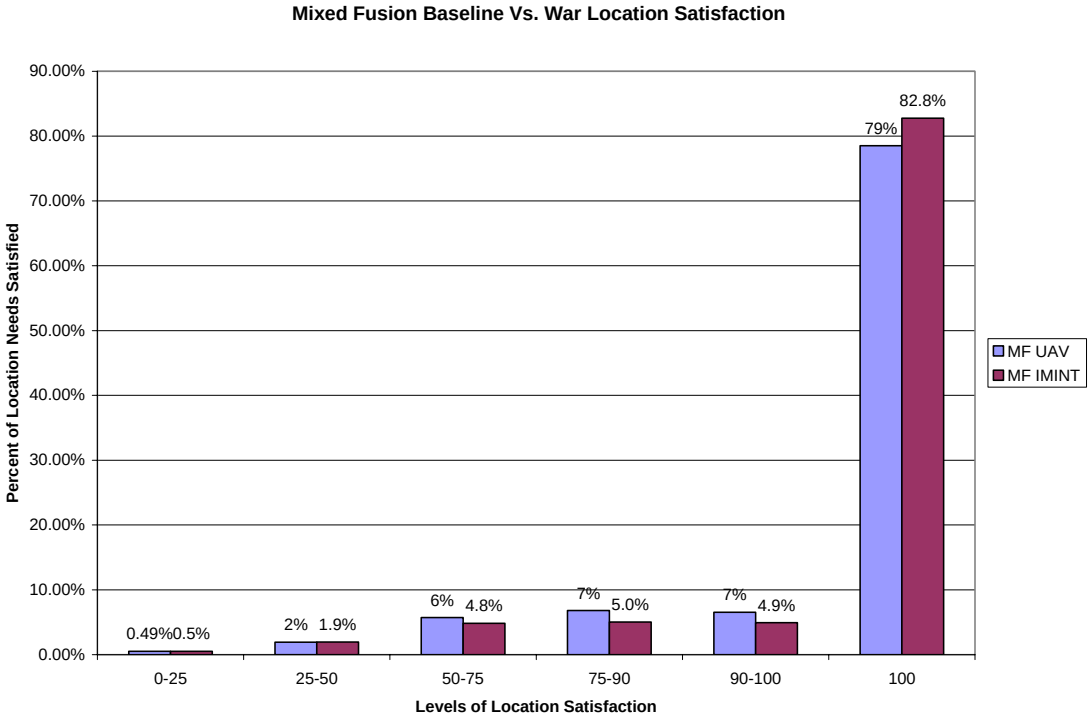

Figure 4-17 Location Satisfaction Comparison by Architecture

4.7 *Analysis Summary*

The data used in this section was purely notional. The output could have been meticulously arranged to tell us whatever the author wanted to prove by modifying the input variables. This must be admitted up front, however this was not the way in which this experiment was conducted. The inputs were purely notional, not designed to model reality, nor designed to inflate model results. Input parameters were all established prior to running the model, and were an attempt at a non-biased selection. The output which admittedly places Total Satisfaction at a fairly abysmal 10% illustrates the fact that the collection probabilities applied to this model do not satisfy user-needs. If this was in fact the case this model would highlight the fact that a new architecture is needed. This does not however invalidate the model as a useful tool rather it shows what the model can do.

While the numbers are low they do show that the model can be used to determine differences between architectures and scenarios. Insight can be gained into the fictional system that was established. In this fictional scenario the decision to purchase the proposed IMINT1 upgrade is most likely not worth the cost. If insight into a fictional system can be established with this model, then with the proper inputs some insights into real world systems can be gained as well.

5. Conclusions

This chapter reviews model capabilities, strengths and weaknesses. Suggestions for future model improvements are provided, as well as some possible future research areas that might benefit the study and advancement of intelligence management.

5.1 *Model Capabilities*

This model is an end-to-end, multi-INT, model of the Intelligence Cycle. It provides the end-user tremendous flexibility to compare and evaluate any intelligence architecture in any scenario. The knowledge matrix concept provides much of this flexibility and gives this model the ability to quantify user-needs satisfaction, which we have established as a separate measure compared to sensor quality. Given accurate probabilities and resource capabilities, this model will provide architectural improvement insights as we illustrated in Chapter four.

This model will let the end-user know what architecture is most likely to collect more information, what architecture will return information in the timeliest manner, and what architecture is most likely to satisfy end-user information needs. These insights should allow an end-user to understand the capabilities of the systems, which they are purchasing, and not only what capabilities they are gaining, but also how those capabilities will mesh with systems already in place. This model will be able to estimate whether or not this new system will fill the intelligence gap(s) that the end-user wishes to alleviate. It evaluates architectures at a macro-level taking a broad look at intelligence across agency bounds and across the intelligence spectrum from strategic-level intelligence down to tactical-level intelligence.

5.2 *Model Strengths*

Because this is an unclassified work, information flow, not accurate modeling parameters was the primary thesis focus. The true strength of this model lies in its realistic process flows and flexibility. Much of the modeling flexibility originates from the knowledge matrix methodology.

Knowledge matrices were created to describe the current state of knowledge concerning a target. Using knowledge matrices as a vehicle to communicate user-needs was an unexpected innovation. This concept shift from quality to satisfaction seems to work well, and may have some real world application in the form of user-input data request shells, which will ensure that information requests ask explicit, answerable questions. Two other often-overlooked RFI issues are incorporated in this model, the time-dependent nature of some RFI's, and the recycling of standing RFIs. This model identifies and manages time dependent RFI's and keeps standing RFIs cycling through the system until a simulated *Quit_Collect* data. Both of these mechanics illustrate appropriate simulation of actual issues.

Not only are RFI procedures modeled well with knowledge matrices, but opportunity collection entities, which are often overlooked in other models, are an integral part of this model. Opportunity collection accounts for resources utilized in an untasked search collection capacity. This in turn leads to another model strength, its consideration of multiple generic intelligence platforms.

Additionally this model can simulate any number of intelligence resources with an endless combination of resource parameters. This inherent resource modeling

flexibility means that this model can simulate any intelligence architecture the end-user might postulate.

Keeping with the flexibility theme the model can also evaluate different pair-wise comparison type fusion algorithms. The mixed fusion and strict fusion scenarios presented here are extremely simple, but the flexible model flow will allow more complex fusion algorithms to be modeled.

5.3 *Model Weaknesses with Suggested Improvements*

While this model is a powerful and flexible tool, it can still be improved. This model could have excellent application to analyze or compare still more architectural changes and scenarios if the following list of improvements were implemented:

a. Resource capabilities are strictly notional. Empirical data is required if this model and its insights are to be used for real world operations. Model output can only be as good as the input.

b. Create a Task Process Post Use (TPPU) vs. Task Process Evaluate Disseminate (TPED) architecture in the Exploitation Delay subprocess. This would allow for modeling of both automated and human resources used in the exploitation process. It would also depict more accurately the number of resources, and the amount of time needed to process intelligence data.

c. Implement more complex fusion algorithms. The current algorithms do not capture any of the synergy effects experienced in intelligence information fusion. To tie in with the TPPU vs. TPED improvement, some value may be added to

fusion in the Exploitation and Processing submodel. This should be further investigated, and added to the model.

d. This model assumes that information flow occurs without any transitional errors or Equivocality issues. In other words, this model assumes all information will either add to our satisfaction or be disregarded. It does not consider erroneous information or misinformation. It may be worthwhile to include this in the model. Also in the realm of adding realism, some sensors may have periodic breakdowns or maintenance schedules, which should be incorporated into the model.

e. Finally, this model views time as its primary constraint. Entities are fused either when all collection vectors requested are received or when a preset time limit occurs. It may improve model performance to have data forwarded as soon as the requested level of satisfaction is attained vice waiting for another situation to occur. This will also mean terminating further collection by other sensors throughout the model as no more information is needed to satisfy the RFI.

5.4 *Future Research*

There are two areas of future research espoused by this work. The first is a method for quantifying knowledge, specifically in an intelligence collection and needs-satisfaction context. Some probability models begin to touch upon knowledge quantification, but some of the basic questions such as boundedness, scale, and the possibly additive or exponential nature of knowledge has not been thoroughly investigated. There are still many unanswered questions as we noted in section 2.10, Quantifying and Modeling Intelligence and Knowledge.

Given that knowledge is quantifiable, the second area for future research recognized by this thesis work is the need for a method to capture intelligence fusion synergies. First, the fact that such synergies do exist must be validated. Then if this synergy does exist, how is it created and how can it be modeled?

Appendix A: Acronym Listing

AFRL	Air Force Research Laboratory
ASD	Assistant Secretary of Defense
AWACS	Airborne Warning and Control System
C3I	Command, Control, Communications, and Intelligence
CEM	Combat Effectiveness Model
CI	Counter Intelligence
COA	Course of Action
COSAGE	Combat Sample Generator
CRN	Common Random Numbers
DISC(*,*,*,*)	Discrete Probability Distribution (Value1, Probability1,Val2, Prob)
DoD	Department of Defense
DSC	Decision Support Center
EADSIM	Extended Air Defense Simulation
EEI	Elements of Essential Information
EXPO(*,*)	Exponential Distribution (Mean, CRN Stream)
FBI	Federal Bureau of Investigation
HUMINT	Human Intelligence
IC	Intelligence Community
IID	Independent and Identically Distributed
IMINT	Imagery Intelligence
INT	Intelligence, usually refers to a specific intelligence discipline
IPB	Intelligence Preparation of the Battlespace
ISR	Intelligence, Surveillance, & Reconnaissance
JCOAT	Joint C4ISR Operations Analysis Tool
JICM	Joint Integrated Contingency Model
MAD	Magnetic Anomaly Detector
MASINT	Measures and Signatures Intelligence
MOE	Measure of Effectiveness
NSSO	National Security Space Organization
RADINT	Radar Intelligence
RF	Radio Frequency
RFI	Request For Information
RSR	Resource Selection Rule
SBR	Space Based Radar
SIGINT	Signals Intelligence
SNB	Smallest Number Busy
SSG	Senior Steering Group
TNOW	Current Simulation Clock Time

TPED	Task, Process, Evaluate, Disseminate
TPED	Task, Process, Exploit, Disseminate
TPPU	Task, Process, Post, Use
TRIA(*,*,*,*)	Triangular Distribution (Min,Mode,Max, CRN Stream)
VCJCS	Vice Chairman of the Joint Chiefs of Staff

Appendix B: User-Defined Model Attributes

	Attribute Name	Nodes Assigned	Possible Values	Description	Further Explanation
1	Priority	Priority_Assign1	1	Denotes Request Significance - Rank is not a Military	Used in the Sort Equation Note: An Opportunity Collect can be any Priority
	Priority	Priority_Assign2	2	Social or Political equivalent	
	Priority	Priority_Assign3	3	relates only to RFI Need Urgency	
	Priority	Priority_Assign4	4		
	Priority	Priority_Assign5	5		
	Priority	Opportunity Matrix	1 to 5	DISC(0.22, 1,0.35,2,0.7,3,0.9,4,1.0,5,33)	
2	RFI_Type	Priority_Assign1	1 to 3	DISC(0.5, 1, 0.95, 2,1.0,3,6)	1 = A Time Critical RFI 2 = A Time Dependent RFI 3 = A Fill at Will RFI
	RFI_Type	Priority_Assign2	1 to 3	DISC(0.4, 1, 0.85, 2,1.0,3,7)	
	RFI_Type	Priority_Assign3	1 to 3	DISC(0.1,1, 0.6, 2,1.0,3,8)	
	RFI_Type	Priority_Assign4	1 to 3	DISC(0.01,1, 0.6, 2,1.0,9)	
	RFI_Type	Priority_Assign5	1 to 3	DISC(0.01, 1, 0.6, 2,1.0,3,493)	
	RFI_Type	Opportunity Matrix	1 or 2	DISC(0.2,1,1.0,2,25)	
3	Date_Due	Crit_Due_Date	0 to 14 hrs	Distributed ~TRIA (2,6,12), Hours, Used for RFI_Type=1	Time (in Hours or Days)
	Date_Due	Other_Due_Date	0 to Infinity	TNOW+DaysToBaseTime(EXPO(7))	by which collection must occur for information to be useful
	Date_Due	Opportunity Matrix	0 to Infinity	TNOW + TRIA(2,6,12), Hours	
	Date_Due	Update_Due_Date	0 to Infinity	TNOW+Reoccurance	
4	Abroad	Matrix_Creation	0 or 1	Disc(0.5,0,1,1,13)	Event of Interest in U.S. = 0
	Abroad	Opportunity Matrix	0 or 1	Disc(0.5,0,1,1,26)	EOI not in U.S. = 1
5	Emit_RF	Matrix_Creation	0 or 1	Disc(0.5,0,1,1,15)	Target uses no RF Comm = 0
	Emit_RF	Opportunity Matrix	0 or 1	Disc(0.5,0,1,1,28)	Target uses RF Comm = 1

	Attribute Name	Nodes Assigned	Possible Values	Description	Further Explanation
6	Detectable Emissions	Matrix_Creation	0 or 1	Disc(0.5,0,1,1,14)	Emits Other = 1
	Detectable Emissions	Opportunity Matrix	0 or 1	Disc(0.5,0,1,1,27)	No other Emissions = 0
7	Sort_Rule	Matrix_Creation	1 to Infinity	Due_Date*Priority	Used to determine Queue
	Sort_Rule	Opportunity_Matrix	1 to Infinity	Due_Date*Priority	order
8	RFI_Create_Date	Matrix_Creation	0 to Infinity	Entity.CreateTime	Used in RFI Loop Logic
	RFI_Create_Date	Opportunity_Matrix	0 to Infinity	TNOW	
9	Continue_Collect_After Due	Matrix_Creation	0 or 1	DISC(0.8, 0, 1.0, 1,18)	Information vital must be
	Continue_Collect_After Due	Opportunity_Matrix	0 or 1	DISC(0.8, 0, 1.0, 1,31)	collected if possible
10	Quit_Collect	Matrix_Creation	0 to Infinity	Due_Date+DaysToBaseTime(EXPO(30,29))	Point at which all collection
	Quit_Collect	Opportunity_Matrix	0 to Infinity	Due_Date+DaysToBaseTime(EXPO(30,32))	must cease
11	Standing_or_ Adhoc	Matrix_Creation	0,6,12,24,48,	DISC(0.6, 0, 0.61, 6, 0.66, 12, 0.74, 24, 0.80, 48, 0.9,	Administrative attribute
	Standing_or_ Adhoc	Matrix_Creation	168, 720, 4368	168,0.95,720,1.0,4368,19)	used to determine RFI
	Standing_or_ Adhoc	Opportunity_Matrix	0	Opportunity Entities are never Standing RFIs	Looping Logic
12	Library_Search	Matrix_Creation	0 or 1	TRIA(0,0.8,1,17)	Search of multiple databases
	Library_Search	Opportunity_Matrix	0	Opportunity Entities collect raw data, not archived data	

	Attribute Name	Nodes Assigned	Possible Values	Description	Further Explanation
13	Location_Need	Matrix_Creation	0 to 5	DISC(0.1, 0, 0.25, 1,0.45,2,0.88,3,0.98,4,1.0,5,20)	User's estimated satisfaction requirement
	Location_Need	OpportunityMatrix	0 to 5	DISC(0.1, 0, 0.25, 1,0.45,2,0.88,3,0.98,4,1.0,5,34)	
14	Activity_State_Need	Matrix_Creation	0 to 5	DISC(0.1, 0, 0.2, 1,0.5,2,0.8,3,0.98,4,1.0,5,22)	User's estimated satisfaction requirement
	Activity_State_Need	OpportunityMatrix	0 to 5	DISC(0.1, 0, 0.2, 1,0.5,2,0.8,3,0.98,4,1.0,5,36)	
15	Track_Need	Matrix_Creation	0 to 5	DISC(0.6,0,0.95,4,1.0,5,16)	User's estimated satisfaction
	Track_Need	Opportunity_Matrix	0 to 5	DISC(0.6,0,0.95,4,1.0,5,30)	requirement
16	Identity_Need	Matrix_Creation	0 to 5	DISC(0.18, 0, 0.3, 1,0.5,2,0.8,3,0.98,4,1.0,5,21)	User's estimated satisfaction
	Identity_Need	Opportunity_Matrix	0 to 5	DISC(0.18, 0, 0.3, 1,0.5,2,0.8,3,0.98,4,1.0,5,35)	requirement
17	Intent_Need	Matrix_Creation	0 to 5	DISC(0.3, 0, 0.55, 1,0.8,2,0.9,3,0.98,4,1.0,5,24)	User's estimated satisfaction
	Intent_Need	Opportunity_Matrix	0 to 5	DISC(0.3, 0, 0.55, 1,0.8,2,0.9,3,0.98,4,1.0,5,38)	requirement
18	Capability_Need	Matrix_Creation	0 to 5	DISC(0.3, 0, 0.45, 1,0.7,2,0.92,3,0.98,4,1.0,5,23)	User's estimated satisfaction
	Capability_Need	Opportunity_Matrix	0 to 5	DISC(0.3, 0, 0.45, 1,0.7,2,0.92,3,0.98,4,1.0,5,37)	requirement
19	Num_Needs	Num_Needs1-7	0 to 7	Num_Needs+1	Number of non-zero
	Num_Needs	NumNum_Needs1 Lib1-7	0 to 7	Num_Needs+1	user-needs
20	Location_Collect	All Collect Submodels	0 to 5	Refer to Appendix C Resource Collection Distributions	Modified in Arena Equations

	Attribute Name	Nodes Assigned	Possible Values	Description	Further Explanation
	Location_Collect	No_Report_ Nodes	0	Past Due_Date, no Resource Assigned	Located in Multiple Nodes
21	Activity_State_Collect	All Collect Submodels	0 to 5	Refer to Appendix C Resource Collection Distributions	Modified in Arena Equations
	Activity_State_Collect	No_Report_ Nodes	0	Past Due_Date, no Resource Assigned	Located in Multiple Nodes
22	Track_Collect	All Collect Submodels	0 to 5	Refer to Appendix C Resource Collection Distributions	Modified in Arena Equations
	Track_Collect	No_Report_ Nodes	0	Past Due_Date, no Resource Assigned	Located in Multiple Nodes
23	Identity_Collect	All Collect Submodels	0 to 5	Refer to Appendix C Resource Collection Distributions	Modified in Arena Equations
	Identity_Collect	No_Report_ Nodes	0	Past Due_Date, no Resource Assigned	Located in Multiple Nodes
24	Intent_Collect	All Collect Submodels	0 to 5	Refer to Appendix C Resource Collection Distributions	Modified in Arena Equations
	Intent_Collect	No_Report_ Nodes	0	Past Due_Date, no Resource Assigned	Located in Multiple Nodes
25	Capability_Collect	All Collect Submodels	0 to 5	Refer to Appendix C Resource Collection Distributions	Modified in Arena Equations
	Capability_Collect	No_Report_ Nodes	0	Past Due_Date, no Resource Assigned	Located in Multiple Nodes
26	Looping_RFI_Count	Standing_Collects	1 to Infinity	Looping_RFI_Count+1	Number of times an RFI
					is re-accomplished
27	Final_Location	Final_Fusion	0 to 5	Location_Fuse (var)	Post Fusion Result
	Final_Location	MF_Final_Fusion	0 to 5	MF_Location_Fuse	

	Attribute Name	Nodes Assigned	Possible Values	Description	Further Explanation
	Final_Location	SFQ Final_Fusion		SFQ_Location_Fuse	
28	Final_Activity_State	Final_Fusion	0 to 5	Activity_State_Fuse(var)	Post Fusion Result
	Final_Activity_State	MF_Final_Fusion	0 to 5	MF_Activity_State_Fuse	
	Final_Activity_State	SFQ Final_Fusion	0 to 5	SFQ_Activity_State_Fuse	
29	Final_Track	Final_Fusion	0 to 5	Track_Fuse(var)	Post Fusion Result
	Final_Track	MF_Final_Fusion	0 to 5	MF_Track_Fuse	
	Final_Track	SFQ Final_Fusion	0 to 5	SFQ_Track_Fuse	
30	Final_Identity	Final_Fusion	0 to 5	Identity_Fuse(var)	Post Fusion Result
	Final_Identity	MF_Final_Fusion	0 to 5	MF_Identity_Fuse	
	Final_Identity	SFQ Final_Fusion	0 to 5	SFQ_Identity_Fuse	
31	Final_Intent	Final_Fusion	0 to 5	Intent_Fuse(var)	Post Fusion Result
	Final_Intent	MF_Final_Fusion	0 to 5	MF_Intent_Fuse	
	Final_Intent	SFQ Final_Fusion	0 to 5	SFQ_Intent_Fuse	
32	Final_Capability	Final_Fusion	0 to 5	Capability_Fuse(var)	Post Fusion Result
	Final_Capability	MF_Final_Fusion	0 to 5	MF_Capability_Fuse	
	Final_Capability	SFQ Final_Fusion	0 to 5	SFQ_Capability_Fuse	
33	Num_Needs_Sat	Count_Sat	0	0	Used for Discrete Satisfaction Measure
	Num_Needs_Sat	Loc_Sat	0 to 1	Num_Needs_Sat +1	
	Num_Needs_Sat	Act_Sat	0 to 2	Num_Needs_Sat +1	
	Num_Needs_Sat	Track_Sat	0 to 3	Num_Needs_Sat +1	
	Num_Needs_Sat	Id_Sat	0 to 4	Num_Needs_Sat +1	
	Num_Needs_Sat	Intent_Sat	0 to 5	Num_Needs_Sat +1	
	Num_Needs_Sat	Capability_Sat	0 to 6	Num_Needs_Sat +1	
34	%Location_Sat	NF_Loc_is_Zero	0.0001	Substitute for 0, so division can be accomplished	Used for Continuous Satisfaction Measure
	%Location_Sat	NF_Loc_Sat	1	Mn(1,Final_Location/Location_Need)	
	%Location_Sat	NF_%Loc_Sat	0 to .99	MX(Final_Location/Location_Need, Location_Collect/Location_Need)	

	Attribute Name	Nodes Assigned	Possible Values	Description	Further Explanation
35	%Activity_State_Sat	NF_Act_is_Zero	0.0001	Substitute for 0, so division can be accomplished	Used for Continuous Satisfaction Measure
	%Activity_State_Sat	NF_Act_Sat	1	mn(1,Final_Activity_State/Activity_State_Need)	
	%Activity_State_Sat	NF_%Act_Sat	0 to .99	mx(Final_Activity_State/Activity_State_Need,Act_ State_Collect/Activity_State_Need)	
36	%Track_Sat	NF_Track_is_Zero	0.0001	Substitute for 0, so division can be accomplished	Used for Continuous Satisfaction Measure
	%Track_Sat	NF_Track_Sat	1	mn(1,Final_Track/Track_Need)	
	%Track_Sat	NF_%Track_Sat	0 to .99	mx(Final_Track/Track_Need, Track_Collect/Track_Need)	
37	%Identity_Sat	NF_Id_is_Zero	0.0001	Substitute for 0, so division can be accomplished	Used for Continuous Satisfaction Measure
	%Identity_Sat	NF_Id_Sat	1	mn(1,Final_Identity/Identity_Need)	
	%Identity_Sat	NF_%Id_Sat	0 to .99	mx(Final_Identity/Identity_Need, Identity_Collect /Identity_Need)	
38	%Intent_Sat	NF_Intent_is_Zero	0.0001	Substitute for 0, so division can be accomplished	Used for Continuous Satisfaction Measure
	%Intent_Sat	NF_Intent_Sat	1	mn(1,Final_Intent/Intent_Need)	
	%Intent_Sat	NF_%Intent_Sat	0 to .99	mx(Final_Intent/Intent_Need, Intent_Collect/Intent_Need)	
39	%Capability_Sat	NF_Capability_is_ Zero	0.0001	Substitute for 0, so division can be accomplished	Used for Continuous Satisfaction Measure
	%Capability_Sat	NF_Capability_is_ Zero	1	mn(1,Final_Capability/Capability_Need)	
	%Capability_Sat	NF_%Capability_ Sat	0 to .99	mx(Final_Capability/Capability_Need, Capability_Collect/Capability_Need)	
40	%_Num_Needs_Sat*	NF_Percent_Need s_ Sat W Lib	0 to 100%	Num_Needs_Sat/(Num_Needs-1)	Used for Continuous Satisfaction Measure
	%_Num_Needs_Sat*	NF_Percent_Need s_ Sat No Lib	0 to 100%	Num_Needs_Sat/Num_Needs	
41	Reoccurrences	Creation_Matrix	0 to Infinity	Due_Date-RFI_Creation_Time	RFI Periodicity

	Attribute Name	Nodes Assigned	Possible Values	Description	Further Explanation
42	SF_Fuse	SF_Fuse_Quits	1 to 6	SFQ_Fuse_for_Set	Establish SF batch size
43	MF_Fuse	MF_Fuse_To Num_Needs	1 to 6	Num_Needs	Establish MF batch size
		MF_Singlton_Rpts	1	Only one RFI available or User-need requested	
		FF_MF_Fuse_Set	1 to 6	MF_Fuse_for_Set (var)	
44	MF_Search	MF_Searcher	1 to Infinity	Entity.SerialNumber	Used because Areana required a user-defined attribute for execution
45	MF_Loop	MF_Loop_Ct	1 to Infinity	Counter for number of RFI's taken out of MF_Q	
46	SFQ_Search	SF_Quit_ Out_Check		Entity.SerialNumber	Used because Areana required a user-defined attribute for execution
47	SFQ_Fuse	SF_Fuse_To Num_Needs	1 to 6	Num_Needs	
48	SFQ_Loop	SF_Loop_Ct	1 to 6	SFQ_Loop+1	Counter for number of RFIs with same Entity.SerialNumber in First_Fuse Queue
	SFQ_Loop	SF_Loop_Ct2	1 to 6	SFQ_Loop+1 (clear the Wait_For All_Clear hold node)	
49	Sat_Sum*	MF_Sum_Need _Sats WLib	0 to 100%	(%Activity_State_Sat+%Location_Sat+%Intent_Sat +%Identity_Sat+%Track_Sat+%Capability_Sat)/(N um_Needs-1)	Sum of Partial Satisfaction Levels

	Attribute Name	Nodes Assigned	Possible Values	Description	Further Explanation
	Sat_Sum*	MF_Sum_Need _Sats WOLib	0 to 100%	(%Activity_State_Sat+%Location_Sat+%Intent_Sat +%Identity_Sat+%Track_Sat+%Capability_Sat)/Num_Needs	

* Note: Any attribute name followed by a * means that this attribute is used in each of the fusion algorithms (NF, MF, and SF) Only one case is presented in this table, but this attribute is actually computed 3 times the only difference in the naming convention is that NF, MF or SF proceeds its Node Assigned name. All calculations are the same.

Appendix B Continued – User Defined Variables

	Attribute Name	Nodes Assigned	Possible Values	Description
1	Location_Fuse	Fusion	0 to 5	MX(Location_Fuse,Location_Collect)
2	Activity_State_Fuse	Fusion	0 to 5	MX(Activity_State_Fuse,Activity_State_Collect)
3	Track_Fuse	Fusion	0 to 5	MX(Track_Fuse,Track_Collect)
4	Identity_Fuse	Fusion	0 to 5	MX(Identity_Fuse,Identity_Collect)
5	Intent_Fuse	Fusion	0 to 5	MX(Intent_Fuse,Intent_Collect)
6	Capability_Fuse	Fusion	0 to 5	MX(Capability_Fuse,Capability_Collect)
7	Post_Fuse	Post_Fusion	1 to 6	Counts a batch of RFI's through fusion
8	No_Fuse_Limit	Assigned in	User-Defined	If this many or fewer hours are left to Due_Date
	No_Fuse_Limit	Arena Equations	0 to Infinity	then the report is sent both to the fusion cell
	No_Fuse_Limit			as well as to the end user.
9	MF_Strict_Search	MF_Q_Ent_Ct	MF_Search	Used as a Variable so all queued entities
	MF_Strict_Search	Clear_Set_Var	0	Clear variable for next iteration
10	MF_Fuse_For_Set	MF_Q_Ent_Ct	0 to MF_Loop	Counter for number of entities
	MF_Fuse_For_Set	MF_Loop_Ct	MF_Loop	taken out of MF_Q
	MF_Fuse_For_Set	MF_Loop_Ct_Out	MF_Loop+1	Final out count
	MF_Fuse_For_Set	Clear_Set_Var	Clear_Set_Var	Clear variable for next iteration
11	MF_Location_Fuse	MF_Fusion	0 to 5	MX(MF_Location_Fuse,Location_Collect)
	MF_Location_Fuse	MF ClearingHouse	0	Clear variable for next iteration

	Attribute Name	Nodes Assigned	Possible Values	Description
12	MF_Activity_State_Fuse	MF_Fusion	0 to 5	MX(MF_Activity_State_Fuse,Activity_State_Collect)
	MF_Activity_State_Fuse	MF ClearingHouse	0	Clear variable for next iteration
13	MF_Track_Fuse	MF_Fusion	0 to 5	MX(MF_Track_Fuse,Track_Collect)
	MF_Track_Fuse	MF ClearingHouse	0	Clear variable for next iteration
14	MF_Identity_Fuse	MF_Fusion	0 to 5	MX(MF_Identity_Fuse,Identity_Collect)
	MF_Identity_Fuse	MF ClearingHouse	0	Clear variable for next iteration
15	MF_Intent_Fuse	MF_Fusion	0 to 5	MX(MF_Intent_Fuse,Intent_Collect)
	MF_Intent_Fuse	MF ClearingHouse	0	Clear variable for next iteration
16	MF_Capability_Fuse	MF_Fusion	0 to 5	MX(MF_Capability_Fuse,Capability_Collect)
	MF_Capability_Fuse	MF ClearingHouse	0	Clear variable for next iteration
17	MF_Post_Fuse	MF_Post_Fusion	0 to 6	MF_Post_Fuse+1
	MF_Post_Fuse	MF ClearingHouse	0	Clear variable for next iteration
18	SFQ_Strict_Search	SF_Quit_ Out_Check	0 To 6	SFQ_Search
	SFQ_Strict_Search	SF_Loop_Ct_Out	1 to 6	SFQ_Fuse_For_Set
	SFQ_Strict_Search	SF_Clear_Vars	0	Clear variable for next iteration
19	SFQ_Fuse_For_Set	SF_Quit_ Out_Check	0	Set counter variable
	SFQ_Fuse_For_Set	SF_Loop_Ct	1 to 6	SFQ_Loop (Search First_Fuse)
	SFQ_Fuse_For_Set	SF_Loop_Ct2	1 to 6	SFQ_Loop (Search Wait_For All_Clear node)
	SFQ_Fuse_For_Set	SF_Loop_Ct_Out	1 to 7	SFQ_Loop+1
	SFQ_Fuse_For_Set	SF_Clear_Vars	0	Clear variable for next iteration

	Attribute Name	Nodes Assigned	Possible Values	Description
20	SFQ_Location_Fuse	SF_Quit_Fusion	0 To 5	MX(SFQ_Location_Fuse,Location_Collect)
	SFQ_Location_Fuse	SFQ ClearingHouse	0	Clear variable for next iteration
21	SFQ_Activity_State_Fuse	SF_Quit_Fusion	0 To 5	MX(SFQ_Activity_State_Fuse,Activity_State_Collect)
	SFQ_Activity_State_Fuse	SFQ ClearingHouse	0	Clear variable for next iteration
22	SFQ_Track_Fuse	SF_Quit_Fusion	0 To 5	MX(SFQ_Track_Fuse,Track_Collect)
	SFQ_Track_Fuse	SFQ ClearingHouse	0	Clear variable for next iteration
23	SFQ_Identity_Fuse	SF_Quit_Fusion	0 To 5	MX(SFQ_Identity_Fuse,Identity_Collect)
	SFQ_Identity_Fuse	SFQ ClearingHouse	0	Clear variable for next iteration
24	SFQ_Intent_Fuse	SF_Quit_Fusion	0 To 5	MX(SFQ_Intent_Fuse,Intent_Collect)
	SFQ_Intent_Fuse	SFQ ClearingHouse	0	Clear variable for next iteration
25	SFQ_Capability_Fuse	SF_Quit_Fusion	0 To 5	MX(SFQ_Capability_Fuse,Capability_Collect)
	SFQ_Capability_Fuse	SFQ ClearingHouse	0	Clear variable for next iteration
26	SFQ_Post_Fuse	SFQ_Post_Fusion	1 To 6	SFQ_Post_Fuse+1
	SFQ_Post_Fuse	SFQ ClearingHouse	0	Clear variable for next iteration

Appendix C: Resource Parameter Tables

Each of the tables that follow in this section characterize the probability of a resource collecting information in each area of a knowledge matrix, given that there is something to collect. The triangular distribution is used for all of the resources in this model. The parameters RF, Detect Emissions, and Abroad are the three sample parameters used to determine what collection resource can best fulfill a user-need.

Table 5-1 IMINT Parameters

IMINT 1	Sat LvL	Loc	Activity/ State	Track	Id	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF ?	Find Emis	Abroad
	5	Max						TRIA(1,6, 8)	Satellite	250	N	N	B
	4	Mode											
	3		Max		Max		Max						
	2		Mode		Mode	Max							
	1			Max/ Mode		Mode	Mode						
	0	Min	Min	Min	Min	Min	Min						

IMINT 2	Sat LvL	Loc	Activity/ State	Track	Id	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF ?	Find Emis	Abroad
	5	Max	Max					TRIA(1,2, 8)	UAV	6	N	N	Y
	4		Mode		Max		Max						
	3	Mode		Max		Max	Mode						
	2			Mode	Mode	Mode							
	1												
	0	Min	Min	Min	Min	Min	Min						

N = No, Y=Yes, B=Both, Min = Minimum, Mode= most likely collect, Max= Maximum

Table 5-2 SIGINT Parameters

	Sat LvL	Loc	Activity/ State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	Abroad
SIGINT1	5		Max		Max	Max	Max	EXP(2)hr	RJ	30	Y	N	Y
	4			Max									
	3	Max			Mode								
	2	Mode	Mode	Mode		Mode	Mode						
	1												
	0	Min	Min	Min	Min	Min	Min						

	Sat LvL	Loc	Activity/ State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	Abroad
SIGINT2	5		Max		Max	Max	Max	EXP(2)hr	Army	20	Y	N	Y
	4			Max									
	3	Max			Mode								
	2	Mode	Mode	Mode		Mode	Mode						
	1												
	0	Min	Min	Min	Min	Min	Min						

	Sat LvL	Loc	Activity/ State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	Abroad
SIGINT3	5		Max		Max	Max	Max	EXP(2)hr	Other	200	Y	N	B
	4			Max									
	3	Max			Mode								
	2	Mode	Mode	Mode		Mode	Mode						
	1												
	0	Min	Min	Min	Min	Min	Min						

Table 5-3 RADINT Parameters

RADINT1	Sat LvL	Loc	Activity/ State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	broad
	5			Max				TRIA(.05, 0.10, 0.25)	SBR	200	N	N	B
	4			Mode									
	3	None				None							
	2		Max	Min	Max		Max						
	1		Mode		Mode		Mode						
	0		Min		Min		Min						

RADINT2	Sat LvL	Loc	Activity/ State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	Abroad
	5			Max	Max			TRIA(0.05, 0.1, 0.25)	AWACS	200	N	N	Only
	4			Mode									
	3	Max	Max	Min	Mode								
	2	Mode											
	1		Mode			Max/Mode	Max/Mode						
	0	Min	Min		Min	Min	Min						

Table 5-4 MASINT Parameters

MASINT1	Sat LvL	Loc	Activity/State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	Abroad
	5	Max						TRIA(0,0.1,) 0.25)	Other	100	N	Y	All
	4		Max										
	3	Mode	Mode	None	Max								
	2					Max	Max						
	1				Mode	Mode	Mode						
	0	Min	Min		Min	Min	Min						

MASINT2	Sat LvL	Loc	Activity/State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	Abroad
	5		Max					TRIA(0,3,6)	Other	500	N	Y	All
	4												
	3	Max	Mode	None	Max								
	2	Mode				Max	Max						
	1				Mode	Mode	Mode						
	0	Min	Min		Min	Min	Min						

Table 5-5 OSINT Parameters

OSINT1	Sat LvL	Loc	Activity/State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	Abroad
	5							Exp(24)hr	FBIS	800	N	N	Y
	4					Max							
	3		Max	None			Max						
	2	Max				Mode							
	1	Mode	Mode		Max/Mode		Mode						
	0	Min	Min		Min	Min	Min						

OSINT2	Sat LvL	Loc	Activity/State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	Abroad
	5							EXP(6)hr	CNN	800	N	N	Al
	4		Max										
	3	Max		None	Max	Max	Max						
	2		Mode		Mode								
	1	Mode				Mode	Mode						
	0	Min	Min		Min	Min	Min						

Table 5-6 Counter -Intelligence Parameters

	Sat LvL	Loc	Activity/State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	Abroad
CI1	5		Max					EXP(48)hr	DHS	Capacity Dependent on Human Resources	N	N	N
	4					Max	Max						
	3	Max	Mode		Max								
	2	Mode		None									
	1				Mode	Mode	Mode						
	0	Min	Min		Min	Min	Min						

	Sat LvL	Loc	Activity/State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	Abroad
CI2	5		Max				Max	EXP(48)hr	FBI	Capacity Dependent on Human Resources	N	N	N
	4					Max							
	3	Max	Mode		Max								
	2	Mode		None									
	1				Mode	Mode	Mode						
	0	Min	Min		Min	Min	Min						

Table 5-7 HUMINT Parameters

HUMINT1	Sat LvL	Loc	Activity/State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	Abroad
	5	Max	Max		Max	Max	Max	EXP(72)hr	CIA	Capacity Dependent on Human Resources	N	N	Y
	4												
	3												
	2	Mode	Mode	None	Mode	Mode	Mode						
	1												
	0	Min	Min		Min	Min	Min						

HUMINT2	Sat LvL	Loc	Activity/State	Track	Identity	Intent	Capability	Expected Time Delay	System	Entity Capacity	RF?	Find Emis	Abroad
	5	Max	Max		Max	Max	Max	EXP(48)hr	Indigenous Populance	Capacity Dependent on Human Resources	N	N	Y
	4												
	3												
	2	Mode	Mode	None	Mode	Mode	Mode						
	1												
	0	Min	Min		Min	Min	Min						

Appendix D: Resource Schedules

The following charts illustrate how seven of the resources within this model are established using the Arena Resource scheduling capability. The Y-axis illustrates the capacity level for each entity (i.e. how many targets it can collect against at any one point in time). The X-axis illustrates one twenty-four hour period. Each resource in this model is based on a 24 hours = one day reoccurring schedule. Other schedule formats could be established if the end-user so desires.

The seven resources modeled are:

IMINT1
IMINT2
SIGINT2
SIGINT3
RADINT1
RADINT2
MASINT2

The other eight resources in this model all use a fixed capacity strategy. Fixed capcity means that each of these resources has preset maximum collection level that never changes over time.

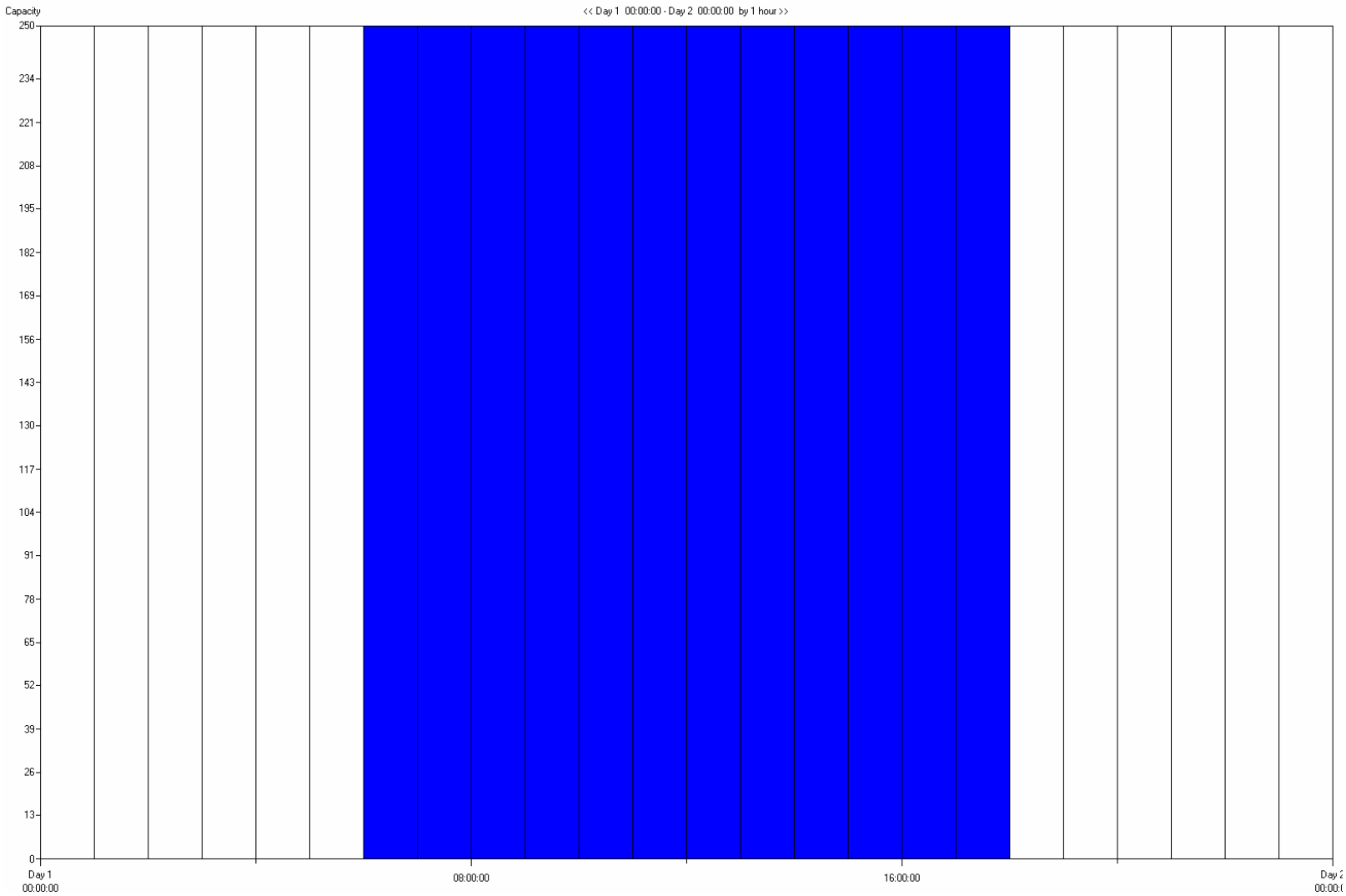

Figure D-1 IMINT1 Resource Schedule

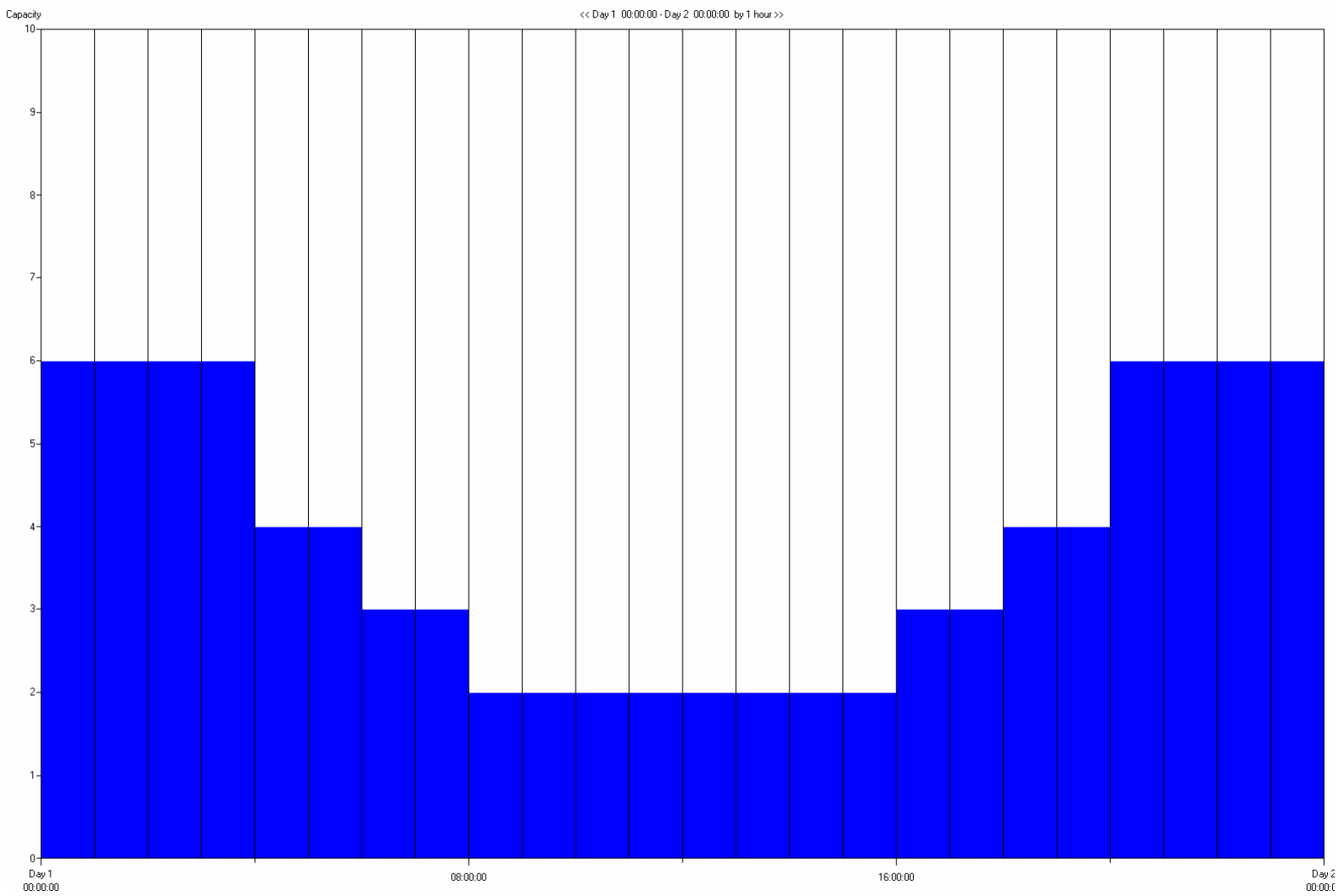

Figure D-2 IMINT2 Resource Schedule

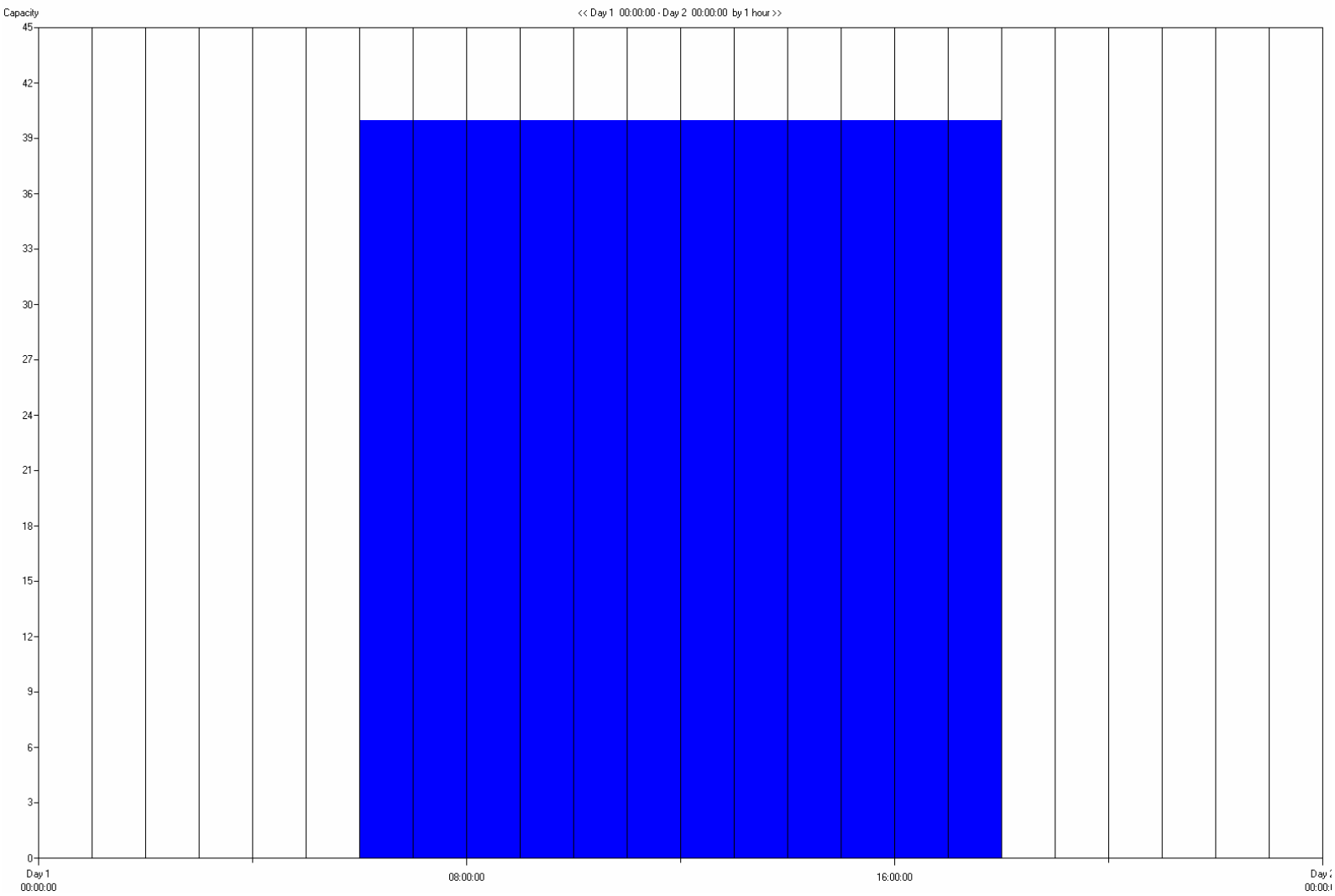

Figure D-3 SIGINT1 Resource Sched

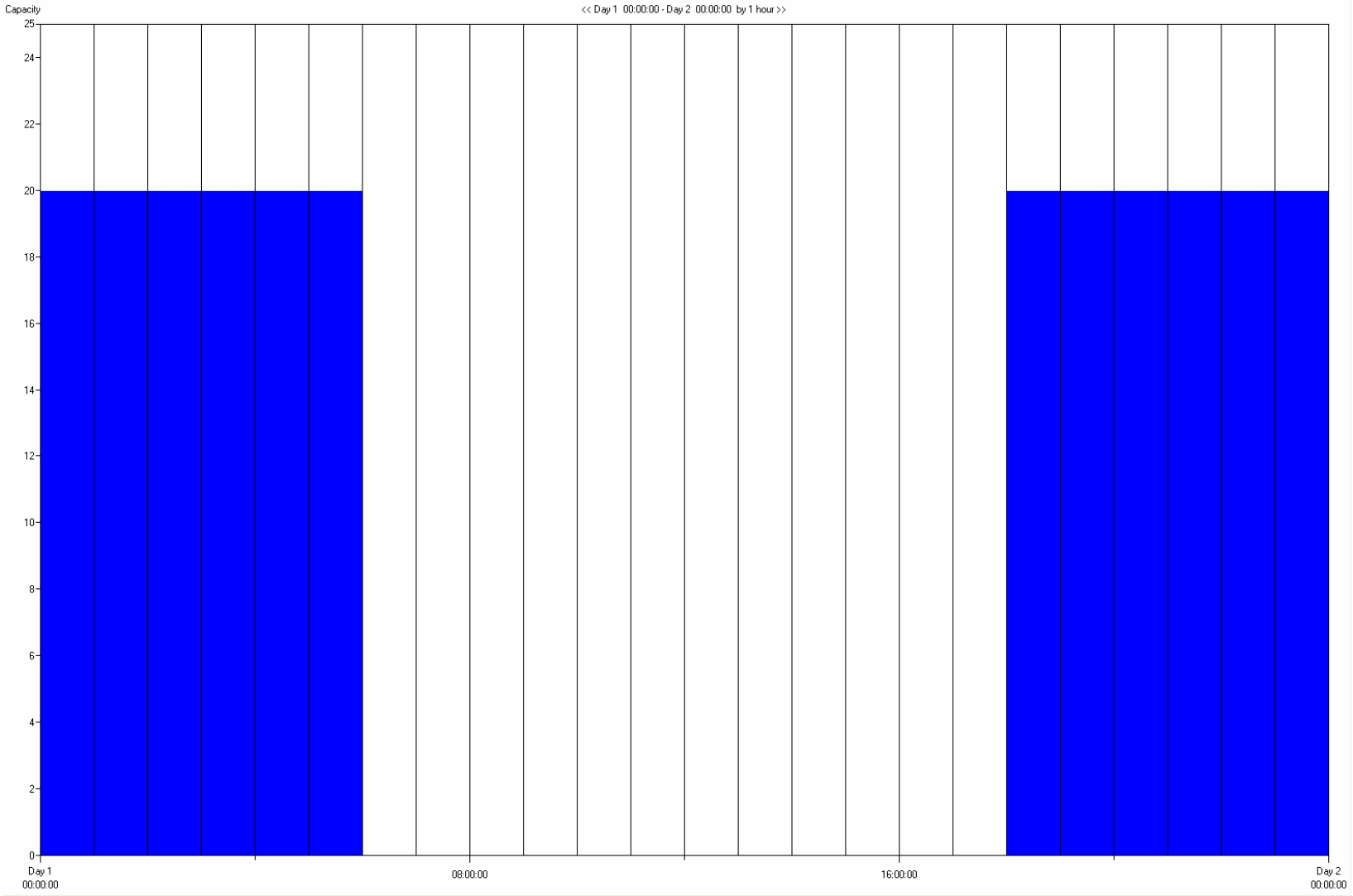

Figure D-4 SIGINT2 Resource Schedule

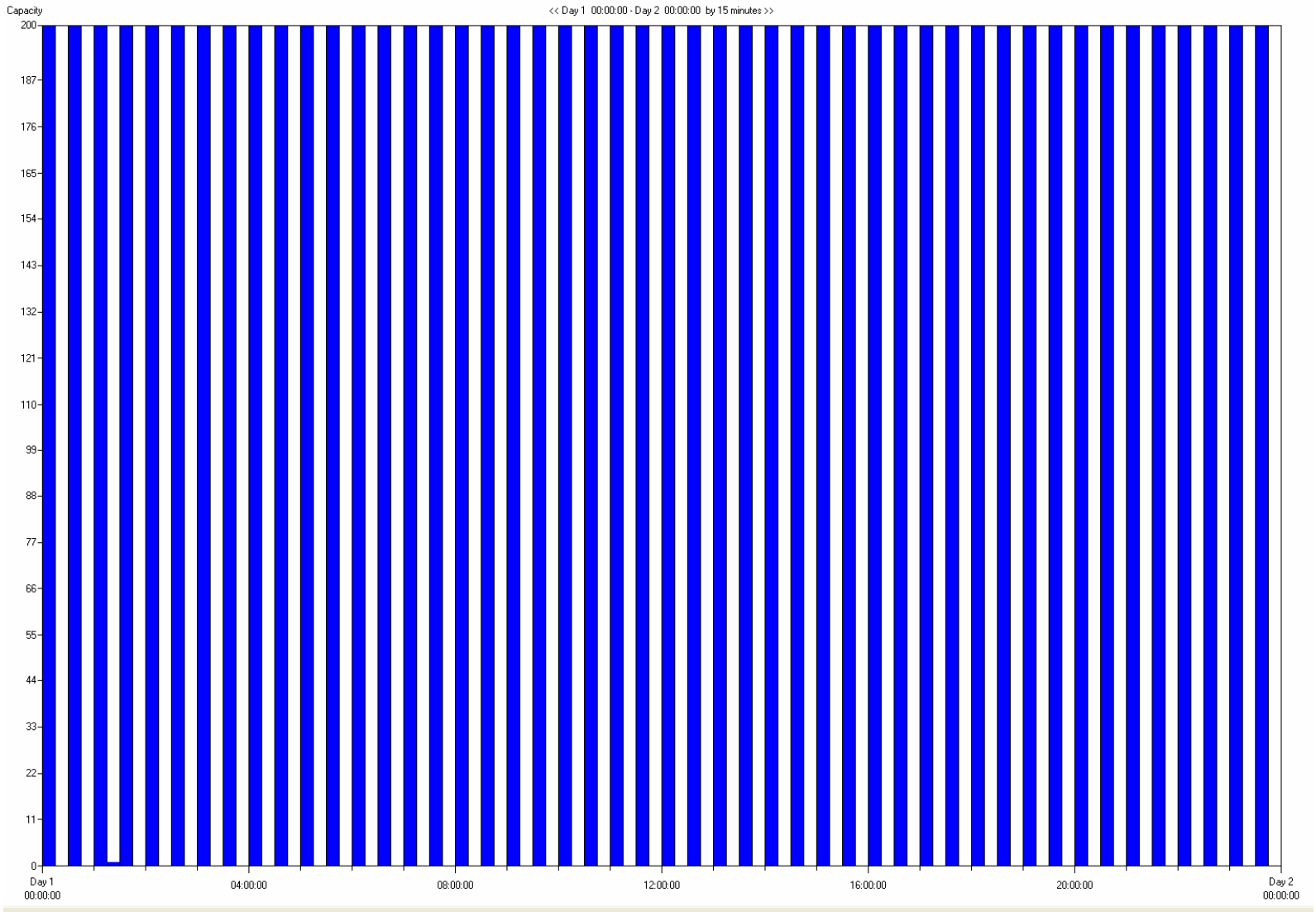

Figure D-5 RADINT1 Resource Schedule

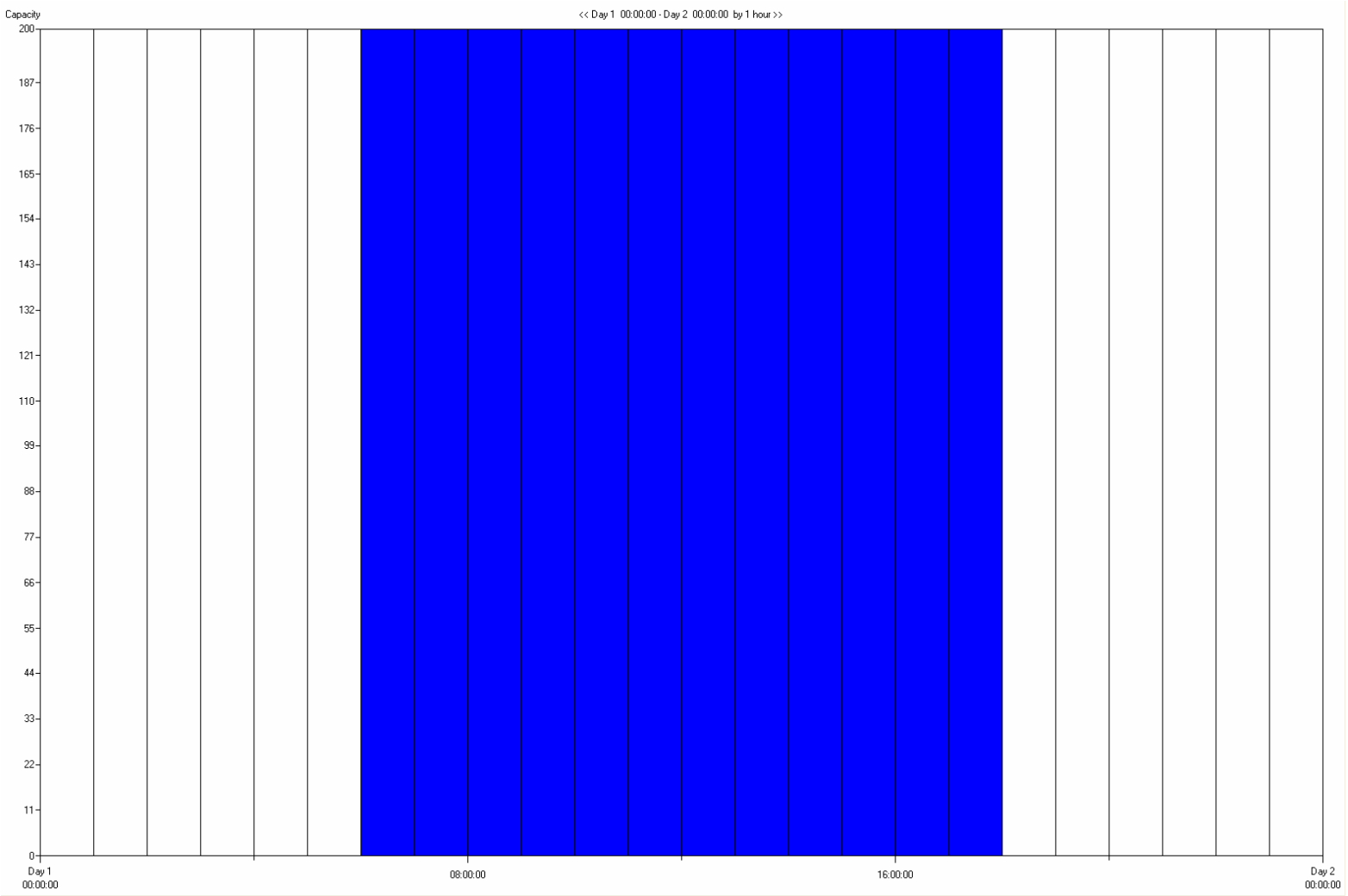

Figure D-6 RADINT2 Resource Schedule

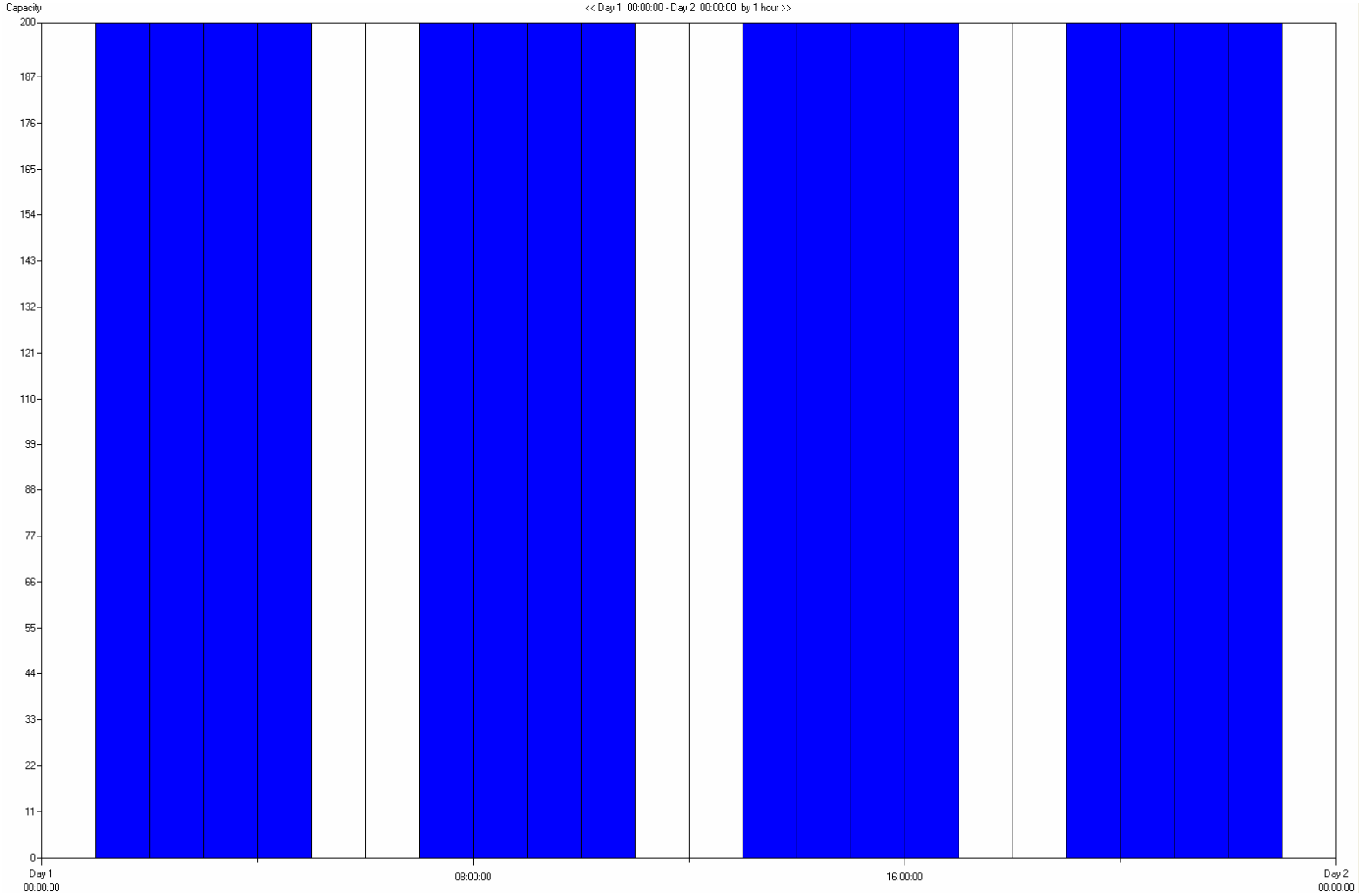

Figure D-7 MASINT2 Resource Schedule

Appendix E: Paired T-Tests

For each of the following paired t-test spreadsheets the statistic measured is the difference between the mean values of the two algorithms or architectures in question. In each comparison, fifty replications were run using a common random number variance reduction technique.

Base MF %Timely Vs. UAV MF %Timely				
Run	Base MF % Timely	UAV MF %Timely	Difference	Mean Squared Dif
1	0.606480132	0.600791659	0.01	0.0001
2	0.600655611	0.633138203	-0.03	0.0009
3	0.611971055	0.649825147	-0.04	0.0013
4	0.619639259	0.592426786	0.03	0.0009
5	0.654403511	0.665191209	-0.01	0.0001
6	0.643138046	0.631641026	0.01	0.0002
7	0.641405794	0.651207052	-0.01	0.0001
8	0.637457896	0.645749598	-0.01	0.0000
9	0.610045662	0.625988924	-0.02	0.0002
10	0.587623118	0.665135677	-0.08	0.0057
11	0.6342234	0.591766724	0.04	0.0020
12	0.670959032	0.584973829	0.09	0.0077
13	0.6215967	0.599325159	0.02	0.0006
14	0.615972894	0.632836117	-0.02	0.0002
15	0.638538341	0.614327384	0.02	0.0007
16	0.614108082	0.607248029	0.01	0.0001
17	0.626916778	0.63682042	-0.01	0.0001
18	0.626372559	0.605190998	0.02	0.0005
19	0.652550481	0.677709418	-0.03	0.0005
20	0.599450263	0.605421542	-0.01	0.0000
21	0.621305539	0.659612931	-0.04	0.0013
22	0.664907652	0.594390152	0.07	0.0053
23	0.636322418	0.647209396	-0.01	0.0001
24	0.608941976	0.623282331	-0.01	0.0002
25	0.639626451	0.658125839	-0.02	0.0003
26	0.635631286	0.652010561	-0.02	0.0002
27	0.646752063	0.587070499	0.06	0.0038
28	0.663218803	0.608828296	0.05	0.0032
29	0.608350652	0.675055049	-0.07	0.0042
30	0.633339982	0.627730536	0.01	0.0001
31	0.603096749	0.66945823	-0.07	0.0041
32	0.616563835	0.630747338	-0.01	0.0001
33	0.637518987	0.611623882	0.03	0.0008
34	0.646296963	0.629937736	0.02	0.0003
35	0.602778565	0.647469722	-0.04	0.0018
36	0.646638486	0.644267388	0.00	0.0000
37	0.640272498	0.680285239	-0.04	0.0014
38	0.65593551	0.626949378	0.03	0.0010
39	0.634690603	0.667031649	-0.03	0.0009
40	0.661710037	0.600327026	0.06	0.0040
41	0.608870396	0.595888778	0.01	0.0002
42	0.610030805	0.603770937	0.01	0.0001
43	0.655127037	0.673152567	-0.02	0.0003
44	0.638147057	0.618413919	0.02	0.0005
45	0.60642746	0.662925454	-0.06	0.0030
46	0.602485052	0.663371382	-0.06	0.0035
47	0.62279875	0.602835864	0.02	0.0005
48	0.599808978	0.633603649	-0.03	0.0010
49	0.673412385	0.624762276	0.05	0.0026
50	0.612381722	0.611052226	0.00	0.0000
	Qbar =	0.00	Variance Qbar =	0.0000
	90% Confidence interval	-0.01	to	0.0047
			0.95 T-statistic =	1.2990

Base MF %Timley Vs. IMINT MF % Timely				
Run	Base MF % Timely	IMINT MF %Timely	Difference	Mean Squared Dif
1	0.606480132	0.608608265	0.00	0.0000
2	0.600655611	0.623396303	-0.02	0.0006
3	0.611971055	0.652146695	-0.04	0.0018
4	0.619639259	0.648007735	-0.03	0.0009
5	0.654403511	0.63575564	0.02	0.0003
6	0.643138046	0.608191994	0.03	0.0011
7	0.641405794	0.64446896	0.00	0.0000
8	0.637457896	0.611070964	0.03	0.0006
9	0.610045662	0.639412682	-0.03	0.0010
10	0.587623118	0.623918804	-0.04	0.0014
11	0.6342234	0.619616987	0.01	0.0002
12	0.670959032	0.634455033	0.04	0.0012
13	0.6215967	0.616387699	0.01	0.0000
14	0.615972894	0.592722949	0.02	0.0005
15	0.638538341	0.578380666	0.06	0.0034
16	0.614108082	0.632820161	-0.02	0.0004
17	0.626916778	0.629563046	0.00	0.0000
18	0.626372559	0.624118648	0.00	0.0000
19	0.652550481	0.611470728	0.04	0.0015
20	0.599450263	0.630658646	-0.03	0.0011
21	0.621305539	0.632189785	-0.01	0.0002
22	0.664907652	0.627432201	0.04	0.0013
23	0.636322418	0.615691231	0.02	0.0004
24	0.608941976	0.598532892	0.01	0.0001
25	0.639626451	0.63825131	0.00	0.0000
26	0.635631286	0.647253475	-0.01	0.0002
27	0.646752063	0.63465912	0.01	0.0001
28	0.663218803	0.632495745	0.03	0.0008
29	0.608350652	0.624416983	-0.02	0.0003
30	0.633339982	0.664279946	-0.03	0.0011
31	0.603096749	0.633363886	-0.03	0.0010
32	0.616563835	0.625424146	-0.01	0.0001
33	0.637518987	0.621521707	0.02	0.0002
34	0.646296963	0.594832166	0.05	0.0025
35	0.602778565	0.650035565	-0.05	0.0024
36	0.646638486	0.635644051	0.01	0.0001
37	0.640272498	0.619111904	0.02	0.0004
38	0.65593551	0.647192947	0.01	0.0000
39	0.634690603	0.631191097	0.00	0.0000
40	0.661710037	0.662438923	0.00	0.0000
41	0.608870396	0.64272173	-0.03	0.0013
42	0.610030805	0.599428435	0.01	0.0001
43	0.655127037	0.650717459	0.00	0.0000
44	0.638147057	0.611732392	0.03	0.0006
45	0.60642746	0.65113245	-0.04	0.0022
46	0.602485052	0.653732106	-0.05	0.0028
47	0.62279875	0.614805208	0.01	0.0000
48	0.599808978	0.613507813	-0.01	0.0002
49	0.673412385	0.609815003	0.06	0.0038
50	0.612381722	0.609868294	0.00	0.0000
	Qbar =	0.0018	Variance Qbar =	0.0000
	90% Confidence interval	-0.0034	to	0.0069

UAV MF % Timely Vs. IMINT MF % Timely				
Run	UAV MF %Timely	IMINT MF %Timely	Difference	Mean Squared Dif
1	0.600791659	0.608608265	-0.01	0.0001
2	0.633138203	0.623396303	0.01	0.0001
3	0.649825147	0.652146695	0.00	0.0000
4	0.592426786	0.648007735	-0.06	0.0033
5	0.665191209	0.63575564	0.03	0.0008
6	0.631641026	0.608191994	0.02	0.0005
7	0.651207052	0.64446896	0.01	0.0000
8	0.645749598	0.611070964	0.03	0.0011
9	0.625988924	0.639412682	-0.01	0.0002
10	0.665135677	0.623918804	0.04	0.0016
11	0.591766724	0.619616987	-0.03	0.0009
12	0.584973829	0.634455033	-0.05	0.0026
13	0.599325159	0.616387699	-0.02	0.0004
14	0.632836117	0.592722949	0.04	0.0015
15	0.614327384	0.578380666	0.04	0.0012
16	0.607248029	0.632820161	-0.03	0.0007
17	0.63682042	0.629563046	0.01	0.0000
18	0.605190998	0.624118648	-0.02	0.0004
19	0.677709418	0.611470728	0.07	0.0042
20	0.605421542	0.630658646	-0.03	0.0007
21	0.659612931	0.632189785	0.03	0.0007
22	0.594390152	0.627432201	-0.03	0.0012
23	0.647209396	0.615691231	0.03	0.0009
24	0.623282331	0.598532892	0.02	0.0005
25	0.658125839	0.63825131	0.02	0.0003
26	0.652010561	0.647253475	0.00	0.0000
27	0.587070499	0.63465912	-0.05	0.0024
28	0.608828296	0.632495745	-0.02	0.0006
29	0.675055049	0.624416983	0.05	0.0024
30	0.627730536	0.664279946	-0.04	0.0015
31	0.66945823	0.633363886	0.04	0.0012
32	0.630747338	0.625424146	0.01	0.0000
33	0.611623882	0.621521707	-0.01	0.0001
34	0.629937736	0.594832166	0.04	0.0011
35	0.647469722	0.650035565	0.00	0.0000
36	0.644267388	0.635644051	0.01	0.0000
37	0.680285239	0.619111904	0.06	0.0035
38	0.626949378	0.647192947	-0.02	0.0005
39	0.667031649	0.631191097	0.04	0.0012
40	0.600327026	0.662438923	-0.06	0.0041
41	0.595888778	0.64272173	-0.05	0.0024
42	0.603770937	0.599428435	0.00	0.0000
43	0.673152567	0.650717459	0.02	0.0004
44	0.618413919	0.611732392	0.01	0.0000
45	0.662925454	0.65113245	0.01	0.0001
46	0.663371382	0.653732106	0.01	0.0001
47	0.602835864	0.614805208	-0.01	0.0002
48	0.633603649	0.613507813	0.02	0.0003
49	0.624762276	0.609815003	0.01	0.0002
50	0.611052226	0.609868294	0.00	0.0000
	Qbar =	0.0038	Variance Qbar =	0.0000
	90% Confidence interval	-0.0019	to	0.0094
			0.95 T-statistic =	1.2990

Base MF Quantity Vs. UAV MF Quantity				
Run	Base MF Quanity	UAV MF Quantity	Difference	Mean Squared Dif
1	20586	20716	-130.00	79,062.19
2	20744	19645	1,099.00	898,362.75
3	20867	19159	1,708.00	2,423,688.51
4	20846	21444	-598.00	561,270.67
5	19598	19429	169.00	317.55
6	19834	19500	334.00	33,423.15
7	20572	19966	606.00	206,861.23
8	20188	19904	284.00	17,641.15
9	20805	20224	581.00	184,745.23
10	21524	18942	2,582.00	5,908,885.87
11	19624	21571	-1,947.00	4,402,359.31
12	19259	21589	-2,330.00	6,156,254.19
13	20605	21042	-437.00	345,955.71
14	20660	20179	481.00	108,781.23
15	19731	20227	-496.00	418,841.95
16	20910	21054	-144.00	87,131.23
17	20151	19902	249.00	9,568.75
18	20582	21152	-570.00	520,100.59
19	19859	19017	842.00	477,232.27
20	21101	20769	332.00	32,695.87
21	20436	18808	1,628.00	2,180,997.31
22	19329	21284	-1,955.00	4,435,994.19
23	19850	19924	-74.00	50,706.03
24	21181	20522	659.00	257,881.15
25	19810	19358	452.00	90,492.67
26	20482	19696	786.00	402,996.43
27	19751	21362	-1,611.00	3,105,278.35
28	19529	20978	-1,449.00	2,560,576.03
29	21172	19074	2,098.00	3,790,108.11
30	20054	19639	415.00	69,600.99
31	20796	19344	1,452.00	1,692,132.67
32	21070	20379	691.00	291,405.63
33	19750	20802	-1,052.00	1,447,642.11
34	19457	19594	-137.00	83,047.71
35	21162	19899	1,263.00	1,236,143.71
36	19872	19956	-84.00	55,309.63
37	20257	18651	1,606.00	2,116,501.23
38	19476	20327	-851.00	1,004,364.75
39	19942	19179	763.00	374,323.71
40	19368	21405	-2,037.00	4,788,131.71
41	21149	20967	182.00	949.87
42	20776	21374	-598.00	561,270.67
43	19758	19067	691.00	291,405.63
44	20033	20289	-256.00	165,795.55
45	20568	19156	1,412.00	1,589,667.07
46	21408	19624	1,784.00	2,666,101.15
47	20159	20946	-787.00	880,181.71
48	20940	19730	1,210.00	1,121,099.79
49	18975	20507	-1,532.00	2,833,094.91
50	21031	20756	275.00	15,331.39
	Qbar =	151.18	Variance Qbar =	25,727.23
	90% Confidence interval	-57.18	to	359.54
			0.90 T-statistic =	1.299

Base MF Quantity Vs.IMINT MF Quantity				
Run	Base MF Quanity	IMINT MF Quantity	Difference	Mean Squared Dif
1	20586	20399	187.00	0.00
2	20744	19798	946.00	576,081.00
3	20867	19169	1,698.00	2,283,121.00
4	20846	19651	1,195.00	1,016,064.00
5	19598	20036	-438.00	390,625.00
6	19834	20459	-625.00	659,344.00
7	20572	19942	630.00	196,249.00
8	20188	20757	-569.00	571,536.00
9	20805	19887	918.00	534,361.00
10	21524	20001	1,523.00	1,784,896.00
11	19624	20156	-532.00	516,961.00
12	19259	20148	-889.00	1,157,776.00
13	20605	20552	53.00	17,956.00
14	20660	21025	-365.00	304,704.00
15	19731	21268	-1,537.00	2,972,176.00
16	20910	20396	514.00	106,929.00
17	20151	20025	126.00	3,721.00
18	20582	20565	17.00	28,900.00
19	19859	21010	-1,151.00	1,790,244.00
20	21101	19616	1,485.00	1,684,804.00
21	20436	19559	877.00	476,100.00
22	19329	19838	-509.00	484,416.00
23	19850	20572	-722.00	826,281.00
24	21181	21130	51.00	18,496.00
25	19810	19649	161.00	676.00
26	20482	19643	839.00	425,104.00
27	19751	19787	-36.00	49,729.00
28	19529	19978	-449.00	404,496.00
29	21172	20797	375.00	35,344.00
30	20054	18575	1,479.00	1,669,264.00
31	20796	19638	1,158.00	942,841.00
32	21070	20335	735.00	300,304.00
33	19750	20017	-267.00	206,116.00
34	19457	20705	-1,248.00	2,059,225.00
35	21162	19682	1,480.00	1,671,849.00
36	19872	20340	-468.00	429,025.00
37	20257	20741	-484.00	450,241.00
38	19476	19736	-260.00	199,809.00
39	19942	19948	-6.00	37,249.00
40	19368	19238	130.00	3,249.00
41	21149	19517	1,632.00	2,088,025.00
42	20776	20995	-219.00	164,836.00
43	19758	19583	175.00	144.00
44	20033	20388	-355.00	293,764.00
45	20568	19162	1,406.00	1,485,961.00
46	21408	19560	1,848.00	2,758,921.00
47	20159	20047	112.00	5,625.00
48	20940	20862	78.00	11,881.00
49	18975	20703	-1,728.00	3,667,225.00
50	21031	20652	379.00	36,864.00
	Qbar =	187.00	Variance Qbar =	15,428.37
	90% Confidence interval	25.65	to	348.35
			0.90 T-statistic =	1.299

UAV MF Quantity Vs. IMINT MF Quantity				
Run	UAV MF Quantity	IMINT MF Quantity	Difference	Mean Squared Dif
1	20716	20399	317.00	79,062.19
2	19645	19798	-153.00	35,652.99
3	19159	19169	-10.00	2,099.47
4	21444	19651	1,793.00	3,087,681.55
5	19429	20036	-607.00	413,217.55
6	19500	20459	-959.00	989,666.83
7	19966	19942	24.00	139.71
8	19904	20757	-853.00	790,000.99
9	20224	19887	337.00	90,709.39
10	18942	20001	-1,059.00	1,198,630.83
11	21571	20156	1,415.00	1,902,137.47
12	21589	20148	1,441.00	1,974,530.83
13	21042	20552	490.00	206,279.47
14	20179	21025	-846.00	777,606.51
15	20227	21268	-1,041.00	1,159,541.31
16	21054	20396	658.00	387,107.95
17	19902	20025	-123.00	25,223.79
18	21152	20565	587.00	303,799.39
19	19017	21010	-1,993.00	4,116,110.59
20	20769	19616	1,153.00	1,248,091.15
21	18808	19559	-751.00	619,085.71
22	21284	19838	1,446.00	1,988,607.63
23	19924	20572	-648.00	467,609.79
24	20522	21130	-608.00	414,504.19
25	19358	19649	-291.00	106,811.31
26	19696	19643	53.00	295.15
27	21362	19787	1,575.00	2,369,075.07
28	20978	19978	1,000.00	929,643.07
29	19074	20797	-1,723.00	3,093,447.79
30	19639	18575	1,064.00	1,057,154.11
31	19344	19638	-294.00	108,781.23
32	20379	20335	44.00	66.91
33	20802	20017	785.00	561,270.67
34	19594	20705	-1,111.00	1,315,196.11
35	19899	19682	217.00	32,826.19
36	19956	20340	-384.00	176,248.83
37	18651	20741	-2,090.00	4,519,110.67
38	20327	19736	591.00	308,224.83
39	19179	19948	-769.00	647,735.23
40	21405	19238	2,167.00	4,541,928.19
41	20967	19517	1,450.00	1,999,905.07
42	21374	20995	379.00	117,772.51
43	19067	19583	-516.00	304,505.31
44	20289	20388	-99.00	18,176.43
45	19156	19162	-6.00	1,748.91
46	19624	19560	64.00	794.11
47	20946	20047	899.00	745,079.71
48	19730	20862	-1,132.00	1,363,803.55
49	20507	20703	-196.00	53,740.51
50	20756	20652	104.00	4,648.51
	Qbar =	35.82	Variance Qbar =	19,042.89
	90% Confidence interval	-143.44	to	215.08
			0.90 T-statistic =	1.299

Base MF %TotSat Vs.UAV MF %TotSat				
Run	Base MF %TotSat	UAV MF %TotSat	Difference	Mean Squared Dif
1	0.090789857	0.092923344	-0.0021	0.0000
2	0.089471654	0.094476966	-0.0050	0.0000
3	0.090046485	0.10350227	-0.0135	0.0001
4	0.086731267	0.09112106	-0.0044	0.0000
5	0.093121747	0.09501261	-0.0019	0.0000
6	0.089644046	0.093948718	-0.0043	0.0000
7	0.096441766	0.10027046	-0.0038	0.0000
8	0.096740638	0.092745177	0.0040	0.0000
9	0.086709925	0.092958861	-0.0062	0.0000
10	0.09333767	0.101837187	-0.0085	0.0000
11	0.099571953	0.085253349	0.0143	0.0003
12	0.096785918	0.092130252	0.0047	0.0000
13	0.091773841	0.093717327	-0.0019	0.0000
14	0.091239109	0.093959066	-0.0027	0.0000
15	0.088844965	0.082266278	0.0066	0.0001
16	0.08986131	0.093093949	-0.0032	0.0000
17	0.096124262	0.089388001	0.0067	0.0001
18	0.084491303	0.085003782	-0.0005	0.0000
19	0.089229065	0.093968554	-0.0047	0.0000
20	0.094166153	0.096634407	-0.0025	0.0000
21	0.092728518	0.101924713	-0.0092	0.0001
22	0.099901702	0.092839692	0.0071	0.0001
23	0.102871537	0.095814094	0.0071	0.0001
24	0.089891884	0.095214891	-0.0053	0.0000
25	0.094649167	0.096394256	-0.0017	0.0000
26	0.097012011	0.102457352	-0.0054	0.0000
27	0.09604577	0.09446681	0.0016	0.0000
28	0.095857443	0.093335876	0.0025	0.0000
29	0.088182505	0.105431477	-0.0172	0.0002
30	0.088361424	0.090635979	-0.0023	0.0000
31	0.088622812	0.096774194	-0.0082	0.0000
32	0.089795918	0.086363413	0.0034	0.0000
33	0.100151899	0.09277954	0.0074	0.0001
34	0.093693786	0.103603144	-0.0099	0.0001
35	0.093044136	0.094979647	-0.0019	0.0000
36	0.09666868	0.091200641	0.0055	0.0000
37	0.097349065	0.107876253	-0.0105	0.0001
38	0.096888478	0.095242781	0.0016	0.0000
39	0.09156554	0.095886125	-0.0043	0.0000
40	0.100733168	0.088203691	0.0125	0.0002
41	0.083360915	0.095960319	-0.0126	0.0001
42	0.091596072	0.088659119	0.0029	0.0000
43	0.102338293	0.111658887	-0.0093	0.0001
44	0.090001498	0.095864754	-0.0059	0.0000
45	0.097870478	0.091877219	0.0060	0.0001
46	0.091227578	0.086985324	0.0042	0.0000
47	0.097574284	0.088179127	0.0094	0.0001
48	0.09226361	0.10562595	-0.0134	0.0001
49	0.096916996	0.094358024	0.0026	0.0000
50	0.087204603	0.09259973	-0.0054	0.0000
	Qbar =	0.00	Variance Qbar =	0.0000
	90% Confidence interval	-0.0028	to	-0.0003
			0.90 T-statistic =	1.2990

Base MF %TotSat Vs. IMINT MF %TotSat				
Run	Base MF %TotSat	IMINT MF %TotSat	Difference	Mean Squared Dif
1	0.090789857	0.104024707	-0.01323	0.00003
2	0.089471654	0.103141731	-0.01367	0.00003
3	0.090046485	0.100683395	-0.01064	0.00001
4	0.086731267	0.106152359	-0.01942	0.00014
5	0.093121747	0.100069874	-0.00695	0.00000
6	0.089644046	0.099907131	-0.01026	0.00001
7	0.096441766	0.103149132	-0.00671	0.00000
8	0.096740638	0.101893337	-0.00515	0.00001
9	0.086709925	0.095288379	-0.00858	0.00000
10	0.09333767	0.102894855	-0.00956	0.00000
11	0.099571953	0.102748561	-0.00318	0.00002
12	0.096785918	0.090877506	0.00591	0.00019
13	0.091773841	0.099503698	-0.00773	0.00000
14	0.091239109	0.098644471	-0.00741	0.00000
15	0.088844965	0.095824713	-0.00698	0.00000
16	0.08986131	0.100313787	-0.01045	0.00001
17	0.096124262	0.096629213	-0.00050	0.00005
18	0.084491303	0.098857282	-0.01437	0.00004
19	0.089229065	0.091861019	-0.00263	0.00003
20	0.094166153	0.107157423	-0.01299	0.00003
21	0.092728518	0.107009561	-0.01428	0.00004
22	0.099901702	0.100211715	-0.00031	0.00006
23	0.102871537	0.09955279	0.00332	0.00012
24	0.089891884	0.094320871	-0.00443	0.00001
25	0.094649167	0.100870273	-0.00622	0.00000
26	0.097012011	0.106144683	-0.00913	0.00000
27	0.09604577	0.104108758	-0.00806	0.00000
28	0.095857443	0.101061167	-0.00520	0.00001
29	0.088182505	0.092465259	-0.00428	0.00001
30	0.088361424	0.109932705	-0.02157	0.00019
31	0.088622812	0.104185762	-0.01556	0.00006
32	0.089795918	0.097762478	-0.00797	0.00000
33	0.100151899	0.103711845	-0.00356	0.00002
34	0.093693786	0.095242695	-0.00155	0.00004
35	0.093044136	0.09241947	0.00062	0.00007
36	0.09666868	0.098820059	-0.00215	0.00003
37	0.097349065	0.101152307	-0.00380	0.00002
38	0.096888478	0.101388326	-0.00450	0.00001
39	0.09156554	0.10647684	-0.01491	0.00005
40	0.100733168	0.105884188	-0.00515	0.00001
41	0.083360915	0.100220321	-0.01686	0.00008
42	0.091596072	0.098547273	-0.00695	0.00000
43	0.102338293	0.109125262	-0.00679	0.00000
44	0.090001498	0.098832647	-0.00883	0.00000
45	0.097870478	0.106304144	-0.00843	0.00000
46	0.091227578	0.105981595	-0.01475	0.00005
47	0.097574284	0.107796678	-0.01022	0.00001
48	0.09226361	0.099750743	-0.00749	0.00000
49	0.096916996	0.102159107	-0.00524	0.00001
50	0.087204603	0.097520821	-0.01032	0.00001
	Qbar =	-0.01	Variance Qbar =	0.00000
	90% Confidence interval	-0.01	to	-0.00677
			0.90 T-statistic =	1.29900

UAV MF %TotSat Vs IMINT MF%TotSat				
Run	UAV MF %TotSat	IMINT MF %TotSat	Difference	Mean Squared Dif
1	0.092923344	0.104024707	-0.01110	0.00002
2	0.094476966	0.103141731	-0.00866	0.00001
3	0.10350227	0.100683395	0.00282	0.00008
4	0.09112106	0.106152359	-0.01503	0.00008
5	0.09501261	0.100069874	-0.00506	0.00000
6	0.093948718	0.099907131	-0.00596	0.00000
7	0.10027046	0.103149132	-0.00288	0.00001
8	0.092745177	0.101893337	-0.00915	0.00001
9	0.092958861	0.095288379	-0.00233	0.00002
10	0.101837187	0.102894855	-0.00106	0.00003
11	0.085253349	0.102748561	-0.01750	0.00013
12	0.092130252	0.090877506	0.00125	0.00006
13	0.093717327	0.099503698	-0.00579	0.00000
14	0.093959066	0.098644471	-0.00469	0.00000
15	0.082266278	0.095824713	-0.01356	0.00005
16	0.093093949	0.100313787	-0.00722	0.00000
17	0.089388001	0.096629213	-0.00724	0.00000
18	0.085003782	0.098857282	-0.01385	0.00006
19	0.093968554	0.091861019	0.00211	0.00007
20	0.096634407	0.107157423	-0.01052	0.00002
21	0.101924713	0.107009561	-0.00508	0.00000
22	0.092839692	0.100211715	-0.00737	0.00000
23	0.095814094	0.09955279	-0.00374	0.00001
24	0.095214891	0.094320871	0.00089	0.00005
25	0.096394256	0.100870273	-0.00448	0.00000
26	0.102457352	0.106144683	-0.00369	0.00001
27	0.09446681	0.104108758	-0.00964	0.00001
28	0.093335876	0.101061167	-0.00773	0.00000
29	0.105431477	0.092465259	0.01297	0.00037
30	0.090635979	0.109932705	-0.01930	0.00017
31	0.096774194	0.104185762	-0.00741	0.00000
32	0.086363413	0.097762478	-0.01140	0.00003
33	0.09277954	0.103711845	-0.01093	0.00002
34	0.103603144	0.095242695	0.00836	0.00021
35	0.094979647	0.09241947	0.00256	0.00008
36	0.091200641	0.098820059	-0.00762	0.00000
37	0.107876253	0.101152307	0.00672	0.00017
38	0.095242781	0.101388326	-0.00615	0.00000
39	0.095886125	0.10647684	-0.01059	0.00002
40	0.088203691	0.105884188	-0.01768	0.00013
41	0.095960319	0.100220321	-0.00426	0.00000
42	0.088659119	0.098547273	-0.00989	0.00001
43	0.111658887	0.109125262	0.00253	0.00008
44	0.095864754	0.098832647	-0.00297	0.00001
45	0.091877219	0.106304144	-0.01443	0.00007
46	0.086985324	0.105981595	-0.01900	0.00016
47	0.088179127	0.107796678	-0.01962	0.00018
48	0.10562595	0.099750743	0.00588	0.00015
49	0.094358024	0.102159107	-0.00780	0.00000
50	0.09259973	0.097520821	-0.00492	0.00000
	Qbar =	-0.01	Variance Qbar =	0.00000
	90% Confidence interval	-0.01	to	-0.00489
			0.90 T-statistic =	1.299

Bibliography

Berkowitz, Bruce D. and Allen E Goodman. *Strategic Intelligence for American National Security*. New Jersey: Princeton University Press, 1989.

Brown, Thomas A. and Shuford, Emir H. *Quantifying Uncertainty Into Numerical Probabilities for The Reporting of Intelligence*. Santa Monica CA: RAND Report For DARPA. R-1185-ARPA, July 1973.

Clemen, Robert T. and Reilly Terence, *Making Hard Decisions with DecisionTools*. Pacific Grove, CA: DUXBURY 2001.

Davis, Jack. "Combating Mind Set," *Studies in Intelligence*, Vol 36, No 5, 1992, 35-36.

Department of Defense. *Doctrine for Intelligence Support to Joint Operations*. Joint Publication 2-0. Washington: GPO, 9 March 2000.

Department of Defense. *Joint Tactics, Techniques, and Procedures for Joint Intelligence Preparation of the Battlespace.* Joint Publication 2-01.3. Washington: GPO, 24 March 2000.

Gonzales, Daniel and Louis R. Moore III. "*Measuring the Value of High Level Fusion*". RAND Corporation

"*Intelligence, Surveillance, and Reconnaissance Processing, Exploitation, and Dissemination System (ISR-PEDS) Study, Phase II Final Report*," Air Force Studies and Analysis Agency

Keegan, John. *Intelligence in War*. New York: Alfred A Knopf, 2003.

Kelton, David, W. and others, *Simulation with Arena, Third Edition.* Boston: McGraw-Hill, 2004.

Kent Sherman, "*Words of Estimative Probability."* Center for the Study of Intelligence, Studies in Intelligence Fall 1964, Washington D.C. CIA.

Kiethley, Hans. National Security Space Architecture. *Multi-INT Fusion Performance Model.* Presentation to the Joint C4ISR Decision Support Center (DSC) FY2000 Study Task 3 (DSC 00-2), 2000.

Kraiman, James B. *MULTI-SENSOR FUSION EFFECTS ON THE CHARACTERISATIONS AND OPTIMIZATION OF TPED ARCHITECTURE PERFORMANCE.* AFRL-IF-RS-TR-2001-74, 13441-4505, Dynamic Technologies Inc. May 2001, 2-3

Law, Averill and David W. Kelton. *Simulation Modeling and Analysis, Third edition*. Boston, ; McGraw-Hill, 2000.

Pawling, Carl R. *Modeling and Simulation of the Military Intelligence Process.* MS thesis, AFIT/GOR/ENS/09-04. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 2004.

Perry, Walter. "Modeling Knowledge in Combat Models." *Military Operations Research*, Vol8, N1, 2003.

Shafer, Glenn. *A Mathematical Theory of Evidence*, New Jersey: Princeton University Press, 1976.

Schneedweiss, Winfrid G. *Boolean Functions with Engineering Applications and Computer Programs.* Berlin: Spinger-Verlag, 1989.

Szayna, Thomas S. et al. *The Emergence of Peer Competitors A Framework for Analysis.* Contract DASW01-96-C-0004. Santa Monica CA: RAND Arroyo Center, 2001.

Tiwana, Amrit, *The Knowledge Management Toolkit*. New Jersey: Prentice Hall PTR, 2000.

Wackerly, Dennis, D. and others, *Mathematical Statistics with Applications*, California: DUXBURY, 2002.

Waltz, Edward. *Knowledge Management in the Intelligence Enterprise*. Massachusetts: Artech House, 2003.

Vita

Captain Kevin J. Whaley graduated from Chariho Regional High School in Woodriver Junction, Rhode Island. He enlisted in the Air Force as a Russian Cryptologic Linguist in August 1989. He graduated from the Defense Language Institute's Basic Russian course in September 1990, and from the Basic Cryptologic Technician course at Goodfellow AFB in March 1991. His first duty assignment was to RAF Chicksands, UK from March 1991 – May 1994. He returned to DLI to attend the Intermediate Russian language course from May 1994 – March 1995. He then served as a certified section and supervisor at the Medina Regional SIGINT Operations Center (MRSOC) from March 1995 – January 1997. At this time Staff Sergeant Whaley was awarded an education and commissioning opportunity through the Airman Scholarship Commissioning Program (ASCP). He attended the University of Texas, at San Antonio where he graduated Magna Cum Laude, with a Bachelor of Business Administration in May 1999.

His first commissioned assignment was the Goodfellow AFB for the Basic Officer Intelligence Course, and the Combat Targeting Course (CTC), September 1999-June 2000. Then Lieutenant Whaley was stationed in Aviano AB, Italy from June 200-August 2003. While at Aviano he worked as the sole intelligence officer for the 603d Air Control Squadron (ACS) and as the 31st Fighter Wing's Chief Targeteer. He deployed once to Bosnia to support Operation Joint Forge in August 2000, and twice to Kuwait, Ali Al Salem AB, and Al Jaber AB in 2001 and 2002 respectively. Upon graduation he will be assigned to the National Reconnaissance Office.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
21-03-2005	Master's Thesis	Jul 2004 – Mar 2005

4. TITLE AND SUBTITLE

A Knowledge Matrix Modeling of the Intelligence Cycle

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Whaley, Kevin J, Captain USAF

5d. PROJECT NUMBER
2004-119

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/05-18

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Security Space Office (NSSO)
Concepts and Analysis - Mr. Shaw
P.O. Box 222310
Chantilly, VA 20153-2310

10. SPONSOR/MONITOR'S ACRONYM(S)

NSSO

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

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

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This effort models information flow through the United States Intelligence Community's Intelligence Cycle using a knowledge matrix methodology. The knowledge matrix methodology takes explicit data from multiple sources and fuses that data to measure a current level of knowledge about a target, or situation. Knowledge matrices are used to develop a measure of user-needs satisfaction. User-needs satisfaction compares requested levels of knowledge to a probability of collecting that knowledge within a designated timeframe. This effort expands the work done by Captain Carl Pawling in his March 2004 thesis, Modeling and Simulation of the Military Intelligence Process, by modeling intelligence as an opportunistic, multi-source, multi-entity system of systems. The value of intelligence fusion is compared, and analyzed between three different algorithms; no fusion, a mixed forward and fuse strategy, and strict fusion strategy. These fusion algorithms are then applied to competing intelligence collection architectures in varying intelligence activity scenarios to determine which architectures will most improve the probability of satisfactory collection. Satisfactory collection is measured in terms of quantity, timeliness, and user-need satisfaction of completed intelligence reports.

15. SUBJECT TERMS
INTELLIGENCE, SIMULATION, MODELING

16. SECURITY CLASSIFICATION OF: UNCLASSIFIED			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			**DR. J.O. Miller, AFIT/ENS**
U	**U**	**U**	**UU**	174	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565, ext 4326