Air Force Institute of Technology

AFIT Scholar

3-2008

Simulation of National Intelligence Process with Fusion

Joseph S. Lupa

Follow this and additional works at: https://scholar.afit.edu/etd

Part of the Operational Research Commons

Recommended Citation

SIMULATION OF NATIONAL INTELLIGENCE PROCESS WITH FUSION

THESIS

Joseph Lupa, Captain, USAF

AFIT/GOR/ENS/08-13

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

SIMULATION OF NATIONAL INTELLIGENCE PROCESS WITH FUSION

THESIS
Joseph Lupa
Captain, USAF

AFIT/GOR/ENS/08-13

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/08-13

SIMULATION OF NATIONAL INTELLIGENCE PROCESS WITH FUSION

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

Joseph Lupa, B.A.

Captain, USAF

March 2008

AFIT/GOR/ENS/08-13

SIMULATION OF NATIONAL INTELLIGENCE PROCESS WITH FUSION

Joseph Lupa, B.A.

Captain, USAF

Approved:

_____	_____
John O. Miller, Ph.D.	Date
Chairman	
_____	_____
Kenneth W. Bauer, Ph.D.	Date
Member	

AFIT/GOR/ENS/08-13

Abstract

This work is a follow on effort of two previous Master's theses. The first was *Modeling and Simulation of the Military Intelligence Process* by Captain Carl Pawling in 2004. The other was *A Knowledge Matrix Modeling of the Intelligence Cycle* by Captain Kevin Whaley in 2005. Both of these were done to facilitate the study and analysis of the intelligence process for the National Security Space Organization (NSSO). Here, modifications are made the Pawling model to include tasking multiple intelligence sources for data collection to fulfill Requests for Information (RFIs) and fusing the collected data into one new piece of intelligence. One fusion method is the one suggested by Whaley, which simply takes the best intelligence collected, while the other method captures the synergy of intelligence fusion. Both methods are compared to each other and to the baseline model where no fusion takes place.

Dedication

I dedicate this thesis to my wife and children for keeping my life multi-dimensional. Without them, my life would be very boring. I also thank those who supported me in my time of need.

Acknowledgements

I would like to thank the members of my thesis committee. Their constant assistance was highly appreciated. I would also like to thank the members of the LaTeX gang for their assistance with formatting code for the tables and figures. Special thanks goes to Capt Ryan Kappedal for his expertise in the intelligence process and for helping me get through AFIT every step of the way.

Joseph Lupa

Table of Contents

List of Figures

List of Tables

SIMULATION OF NATIONAL INTELLIGENCE PROCESS WITH FUSION

1. Introduction

1.1 Background

Information fusion is a relatively new field of study. Much research has been done in the last few decades, and the methods and terminology are still developing. The recent spike in interest in this field is mainly due to the amount of information overflow experienced today. Technological developments such as new types of sensory equipment and increased data storage capabilities along with increased processing speed and accessability drive the need to fuse and exploit information for use in the Global War on Terrorism.

Joint Publication 2.0 [5] defines fusion as "the process of collecting and examining information from all available sources and intelligence disciplines to derive as complete an assessment as possible of detected activity." Fusion theory uses techniques and tools to take information from individual sources and fuse them together to provide a more accurate and robust description of something. In doing so, we exploit the synergy achieved by attaining more knowledge about the subject than could be possible by using each source individually. To illustrate this, consider the story of Operation Goldregen in World War II. The US Army received signals intelligence (SIGINT) that the Luftwaffe was going to launch Operation Goldregen, but they did not know what the operation entailed. Elsewhere at Army headquarters was some human intelligence (HUMINT) from a prisoner of war who was a former Luftwaffe clerk that described the details of the operation that included an attack by large numbers of low-flying aircraft. If the information had been fused, the com-

mander could have been warned ahead of time and perhaps employed some sort of defense [15].

1.2   Research Objective

The US intelligence process was first modelled by Pawling [13] in his AFIT Master's thesis in 2004. Using Arena, he created a model that simulated the flow of a Request for Information (RFI) through the entire intelligence process. This model only included sourcing each RFI to a single intelligence discipline and determining if the information collected from that single source met the requirement or not, thus it lacked fusion of information from multiple sources. He suggests expanding his model to incorporate multiple collection efforts from different intelligence sources and a representation of the fusion of that data.

In his 2005 thesis, Whaley [16] created a simulation model of the intelligence process that incorporated a basic method of representing fusion of intelligence from multiple sources. The technique uses a knowledge matrix that quantifies the levels of information about different aspects of a given subject. It then takes these matrices from different sources and fuses them in a way that creates a resulting matrix comprised of the maximum value of each element of the input matrices. This method does not capture the synergy that comes from a fusion process. He suggests building on his research to try and model this fusion synergy and measure its effects on the intelligence process.

It has been suggested that the model created by Whaley contains too much detail in his representation, making it too complex to evaluate the overall process. So this work will build on Pawling's model by adding collection of intelligence from multiple sources and comparing different fusion methodologies to the baseline level of no fusion. The first methodology will be the one used by Whaley where given knowledge levels A and B, the fused knowledge level $C = max(A, B)$. The other

methodology will be the one first introduced by Keithly [8] where $C = 1 - (1 - A)(1 - B)$.

A third methodology will be discussed although not implemented in this study. Neural Networks provide a very flexible and adaptive approach to fusing the quantified information in the knowledge matrices. A Neural Network requires initial data sets of inputs with their associated outputs in order to train the network to recognize patterns and establish a weighting scheme for the nodes in the network. Actual intelligence data is classified and difficult to obtain, so only notional data is used in this study. Because the input data is notional, the output data would have to be deterministically computed by the user. In this sense, the relation of the outputs to the inputs would already be known; therefore, it would be pointless to use a Neural Network to discover the relationship.

1.3 Overview

Chapter 2 contains a brief summary of the intelligence process and the different intelligence disciplines. It will reference relevant literature on how to quantify data as knowledge, different methods of information fusion, and previous work done to model the intelligence process and information fusion. Chapter 3 explains the methodology used to develop the develop the model in Arena. Chapter 4 presents the simulation output with analysis and results of the research. Chapter 5 presents lessons learned with conclusions and recommendations for future research.

2. Literature Review

2.1 Introduction

This section begins with a description of the national intelligence process. It will detail the steps involved in the intelligence cycle with emphasis on the Department of Defense's (DOD) view of information fusion. Further discussion of types of fusion, fusion techniques, and fusion theory will follow. This will include examples of work that has been done in the field of information fusion to include neural network fusion strategies. The knowledge matrix approach to quantifying intelligence will be a key part of this. Finally a discussion of why a simulation approach to evaluating fusion is needed.

2.2 The Intelligence Cycle

Intelligence operations are essential throughout the range of military operations. The purpose of intelligence is to provide commanders and decision makers with the information they need to accomplish missions and ensure national security. The DOD developed a joint doctrine to dictate terminology and outline the process of gathering and producing intelligence that is accurate, timely, and relevant. The process is a cycle of six categories that are interrelated and overlapping. The cycle, seen in Figure 2.1, consists of Planning and Direction, Collection, Processing and Exploitation, Analysis and Production, Dissemination and Integration, and Evaluation and Feedback [5].

The Planning and Direction phase is where the cycle starts. Intelligence needs are identified by planners based on possible threats or from essential information commanders must know to accomplish missions. These needs are transferred into Priority Information Requests (PIRs). PIRs in turn provide a basis for Intelligence Operations and drive the Intelligence Process. Intelligence agencies review PIRs

Figure 2.1   The Intelligence Cycle [5]

and source them out to different collection agencies. An information request can be tasked to more than one collection agency either because the request cannot be fulfilled by a single agency or to generate redundancy for improved accuracy.

The next step is the collection of the actual data needed to satisfy the requests for information generated in the Planning and Direction phase. Collection agencies task their assets to collect data at specified times and/or places about specific targets. The sources of the data are categorized into seven different Intelligence Disciplines which can be further broken down into subcategories as shown in Figure 2.2. The result of the collection process is simply raw data. It is generally not useful at the time of collection either because it is not intelligible yet or because it is only one part of an overall big picture that is trying to be generated.

The raw data is transformed into usable information in the Processing and Exploitation phase. Sometimes this phase runs concurrently with the collection phase as is the case with some SIGINT systems where the data is automatically

INTELLIGENCE DISCIPLINES, SUBCATEGORIES, AND SOURCES

GEOINT -- Geospatial Intelligence
- -- Imagery
- -- IMINT - Imagery Intelligence
- -- Geospatial Information

HUMINT -- Human Intelligence
- -- Debriefings   -- Source Operations
- -- Interrogation Operations   -- Document and Media Exploitation

SIGINT -- Signals Intelligence
- -- COMINT - Communications Intelligence
- -- ELINT - Electronic Intelligence
    - ** Technical ELINT
    - ** Operational ELINT
- -- FISINT - Foreign Instrumentation Signals Intelligence

MASINT -- Measurement and Signature Intelligence
- -- Electromagnetic Data   -- Radio Frequency Data
- -- Geophysical Data   -- Radar Data
- -- Materials Data   -- Nuclear Radiation Data

OSINT -- Open-Source Intelligence
- -- Academia   -- Media Broadcasts
- -- Interagency   -- Internet
- -- Newspapers/Periodicals

TECHINT -- Technical Intelligence

CI -- Counterintelligence

Figure 2.2   Intelligence Disciplines [5]

processed upon collection. HUMINT teams can take considerably more time to process data acquired from a debriefing or an interrogation. Once the data has been transformed into information, it can be made immediately available to a commander or further developed in the next phase by an intelligence analyst.

In the Analysis and Production phase, the information generated in the Processing and Exploitation phase is transformed into its final product, actual intelligence. Here the information from one or more sources is integrated, evaluated and interpreted to generate a final product that will satisfy one or more of the initial PIRs. The integration is now more formally known as fusion. Fusion exploits the synergy of combining intelligence from several sources. It is used to maximize the strengths and minimize the weaknesses of the different intelligence disciplines. A fused intelligence product from multiple sources provides the highest level of understanding about the target. Fusion is also used as a safeguard against enemy deception efforts by cross-

checking information from single-source intelligence reports. A further discussion of fusion will come later.

The Dissemination and Integration phase consists of delivering the intelligence report to the person or agency requesting the information and then the information being integrated into the decision making process. The means of delivery is classified as either push or pull. Pull is more common because it can save time and resources as the end user can access the data as they need it. The push method usually occurs in the event that the intelligence is urgent as in the case with a warning to the theater that could affect operations.

The Evaluation and Feedback phase occurs concurrently with all the other phases. It is a process of Total Quality Management and seeks to ensure that the entire process is operating at a satisfactory level. Qualitative measures are taken throughout the process to evaluate such attributes as timeliness, accuracy, usability, completeness, relevance, objectiveness, and availability. If any of these attributes fall below an acceptable level, it is an indication of some problem with the intelligence cycle.

2.3   Fusion

The theory behind information fusion is that when information from multiple sources is fused, the result is more robust knowledge about the subject of the information such that the decision made from that knowledge is in some way qualitatively or quantitatively better than it would have been if the information from any of the individual sources was used [4]. So the reason for fusion is to gain a more accurate description of the battlespace or whatever target information is being acquired on. Gathering intelligence from multiple sources and/or disciplines and fusing them together generates a more complete picture than any one source alone. Fused information is at least as good as information from any one source [12].

Fusion can be classified in several different ways [12]. It can be classified based on the relationship among the sources, the abstraction level, or the input/output of the fusion process. When classified on relationships, fusion is referred to as complementary if independent sources provide different pieces of information that are not redundant. Then the pieces can be put together to form a broader picture. Redundant fusion occurs when the independent sources provide the same information. Cooperative fusion happens when the information is fused to provide a better description of the scene than could be provided by the sources individually. This classification is an example of the synergy that can result from information fusion.

Different abstraction level classifications include Low-Level, Medium-Level, High-Level, and Multilevel fusion. Low-Level fusion is also called signal or measurement level fusion and is simply combining the information inputs into one output that is more accurate than any of the individual inputs. Here the information is just in the form of raw data. Medium-Level fusion is also called feature or attribute level fusion. This is when information inputs are used to try to predict or estimate some new piece of information not readily observable using the available sources. High-Level fusion is also known as symbol or decision level fusion. It combines symbolic representations or decisions to make a more confident or higher level decision. Finally Multilevel fusion is just what it sounds like - when more than one of the aforementioned abstraction levels are applied in the fusion process.

The third way of classifying fusion is similar to the abstraction level method in that it considers the abstraction level of inputs and outputs to the process. It begins with the lowest level, Data In-Data Out, and progresses through increasing levels of input/output including Data In-Feature Out, Feature In-Feature Out, Feature In-Decision Out, and finally Decision In-Decision Out. These input/output classifications correspond to the abstraction level classifications but attempt to reduce ambiguity when the inputs are of a different abstraction level than the outputs.

The Joint Directors of Laboratories (JDL) further classified fusion by defining fusion levels. This was initially done in 1985, and the JDL Data Fusion Model has been modified since then [8]. It begins with Level 0 fusion in which an observation or piece of collected intelligence is organized and normalized. In this study, Level 0 fusion is associated with representing intelligence data with a knowledge matrix that will be discussed in detail later. Level 1 fusion deals with refining and correlating the data which leads to information about position, track, and identity of objects. Level 2 fusion is referred to as aggregation. Relationships among objects are examined to determine which objects are associated with one another. Level 3 fusion is the interpretation of the objects capability and prediction of the objects intent. Level 4 fusion is like a feedback loop that assesses the fusion process controls the continuous improvement and refinement of entire process.

The fusion modelled in this thesis will be both complementary and redundant in terms of sources. It will be medium level in terms of abstraction. Inputs and outputs will be knowledge matrices, so it will be Data In-Data Out. Under the JDL Data Fusion Model, it will deal with levels 0 through 3.

2.4   Fusion Methods

There have been dozens of methods and techniques developed to perform information fusion. This section will highlight some of the more popular methods. One popular technique is the use of a Kalman filter [12]. It is used to predict a discrete-time state vector at time $k+1$ based on information collected at time k. The prediction equation used with the Kalman filter approach is

$$\mathbf{X}_{k+1} = \mathbf{F}\mathbf{X}_k + \mathbf{J}\mathbf{w}_k \tag{2.1}$$

where

$\mathbf{X}_k$ = the state matrix,

F = system or transition matrix,

J = input matrix, and

$\mathbf{w}_k$ = disturbance input vector or noise [10].

The type of fusion that usually takes place here is low level fusion of redundant data. Using this method requires a priori knowledge of the state space being observed. The model also assumes that the system noise can be modelled as zero mean Gaussian noise. The state estimate provided by a Kalman filter is statistically optimal in that it minimizes the mean squared error between the estimated and actual observed velocity and position states of the target.

Another popular method is the use of Bayesian inference. It is based on Bayes' rule for calculating the probability of Y given that event X has occurred:

$$Pr(Y|X) = \frac{Pr(X|Y)Pr(Y)}{Pr(X)} \tag{2.2}$$

Bayesian inference is commonly used in decision level fusion. It relies on basic probability theory to determine the belief that an event will occur in terms of conditional probability. As with the Kalman filter, some a priori knowledge is required. In 2.2, the probabilities $Pr(X)$ and $Pr(X|Y)$ must be known or estimated. If they are estimated, then the output of this type of fusion will only be as good as the estimate of the inputs in terms of quality. This is an example of the garbage-in, garbage-out principle, and it can sometimes be accounted for by applying the Bayesian filter iteratively throughout a series of time periods where the resulting probabilities from the previous time period are used as the input probabilities for the current time period [10].

Dempster-Shafer Evidential Theory is another probability based method that is a generalized form of Bayesian inference. It is more flexible than Bayesian inference for two reasons: 1) the prior probabilities do not have to be known. They are only assigned when the information is available, and 2) different types of information can

be included with different levels of detail. It works by starting with the assumption that all possible states are known and are enumerated. For example a target is of a certain type. Next information is collected on that target from multiple sources or disciplines and each individual source of information is used to compute a mass distribution function $m(H)$ or more simply put, a belief that a certain hypothesis is true. These mass functions are then combined using Dempster's rules of combination and a decision is made to choose the hypothesis with the highest amount of supporting evidence [10]. Another benefit of Dempster-Shafer fusion over Bayesian fusion is that the former includes a method for representing uncertainty when the probabilities cannot be determined. While this does make the Dempster-Shafer method more flexible, there is a tradeoff for accuracy.

A fusion technique that has applications across most of the other methods is fuzzy logic [10]. Despite the name, fuzzy logic is a well-defined and applicable method with precise outputs. Fuzzy logic is best applied when the boundaries between the sets of values are are not definitely defined or fuzzy. An example would be the difference between warm and hot. There is no set temperature to distinguish the two, and different people might distinguish the two sets at different boundaries. Fuzzy systems use membership functions that define those boundaries and transform both exact and fuzzy inputs from multiple sources into fuzzy sets. Once in the sets, production rules are implemented to evaluate all of the inputs simultaneously. The logical output of this process is then defuzzified to produce a well-defined output value. Fuzzy logic is useful in military applications to define the battlespace and classify possible targets.

A powerful and popular method of information fusion is with a Neural Network. Neural networks get their name from the similarities to the neurons in the brain. Humans brains have the ability to learn, adapt, and parallel process large amounts of data to characterize information. This is the key feature to neural networks. Using learning sets of inputs and output, a neural network can be trained to

recognize patterns and classify targets. Neural networks have been used to perform complementary fusion for automatic target recognition in military applications [12]. Neural networks are a set of inter-connected processing nodes called neurons that are configured in layers. The first layer takes the inputs where the nodes simultaneously process the input data and send their output as input to the next layer of nodes. The logical process usually consists of a series of weighting of the input data to generate an output that characterizes the target. To do this, the network must first be trained using a predetermined data set in which the classification of the target for each set of inputs is known.

2.5   The Knowledge Matrix

All of the fusion methods above contain logical and mathematical algorithms based on either continuous or discrete quantifiable data, so to use these methods requires that the inputs to the processes be quantified numerically. The question then becomes how is intelligence information quantified? A well-accepted and useful method was developed by Keithley [8] for the Multi-INT Fusion study for the Decision Support Center (DSC). He proposed a 6x6 knowledge matrix made up of six quality levels for each of six different types of knowledge. The knowledge types are location, track, identification, activity, capability, and intent. The quality levels range from 0(low) to 5(high) as shown in Figure 2.3.

A knowledge matrix can be used to quantify the quality of some piece of information about an entity. Each cell of the matrix is filled in with the likelihood that the data in this particular piece of information achieves at least that quality level for that type of knowledge. Likelihoods can be referred to as probabilities, so the entries of each cell have to take on a value between 0 and 1. The level of quality decreases as you go down a column in the matrix, thus the probability of the intelligence source attaining that level increases. Figure 2.4 shows an example of a knowledge matrix generated from some information gathered from an intelligence

| Quality | Knowledge Type |
	Loc	Track	ID	Act	Cap	Intent
Highest 5	5 m	Vectors and patterns	Specify object and parent	Precise actions	All elements	All long- and short-term objectives
High 4	10 m	Vectors	Specify object	Many specific actions	Many details	Major objectives
Medium 3	20 m	General speed and direction	Classify (e.g., wheeled, tracked vehicle)	Identifiable actions	Some details	Primary objectives
Medium-low 2	100 m	Toward or away	Distinguish (e.g., vehicle, structure)	Single identifiable action	General information	General objectives
Low 1	1 km	Stationary or not	Discriminate	Unidentifiable action	Minimal information	Single objectives
Lowest 0	10 km	Detect	Detect	Detect	Detect	Detect

Figure 2.3   Knowledge Matrix level descriptions [14]

| Quality Level | Type of Knowledge |
	Location	Track	Identity	Activity	Capability	Intent
5	0.0	0.0	0.0	0.0	0.0	0.0
4	0.3	0.0	0.0	0.0	0.0	0.2
3	0.8	0.0	0.0	0.0	0.0	0.6
2	0.9	0.7	0.9	0.0	0.0	0.9
1	0.95	0.9	0.95	0.0	0.9	0.9
0	0.99	0.95	0.98	0.0	0.92	0.9

Figure 2.4   Sample Knowledge Matrix [11]

source. If the desired quality was a 90% level, then this matrix would demonstrate that the desired quality was achieved at level 2 for location, 1 for track, 2 for identity, 1 for capability, 2 for intent, and no knowledge for activity. The way to interpret these results is to refer back to the level descriptions. It can be said that the location of the entity is know within 100m, it is moving, it's identity can be categorized, minimal capability information is available, and it's general objectives are known.

The 90% level was arbitrary to illustrate how to deduce the quality of a piece of information from its knowledge matrix. Typically in simulations, the quality for

a particular knowledge type is generated by sampling from a uniform distribution from 0 to 1. The uniform data sample is assigned to a variable. If that variable takes the value of say 0.7, then the sample matrix in Figure 2.4 would indicate that location quality level 3, track quality level 2, identity level 2, capability level 1, and intent level 2 were achieved.

2.6   Why Simulation?

Modeling and simulation is used to study the performance of the intelligence cycle because it is not practical to use real classified data, and alternate process architectures may not be feasibly applied in the real world. So as with many other problems and issues faced by the military, a discrete event simulation model is used to gain insight into how fusion affects the intelligence process [6]. The intelligence cycle was modelled by Pawling [13] to examine the performance of each portion of the intelligence process and compare two different theories of distribution: Task, Process, Exploit, Disseminate (TPED) and Task, Process, Post, Use (TPPU). He developed a high level simulation model that compared several Measures of Effectiveness (MOEs) including an aggregate measure termed information needs satisfaction. This measure of overall performance simply calculates the ratio of satisfiability of needs that were met at a given level of satisfaction requested. One thing lacking in the model is the representation of information fusion. Pawling states that a simplification was made in this area for the purpose of a higher, less detailed, simulation model. He suggests that future research in this are might include integrating the fusion process into the model.

Whaley [16] furthered the efforts of Pawling by adding multi-INT fusion into the process. He uses an adaptation of the knowledge matrix concept developed by Keithley [8] to represent the level of satisfaction the customer gets from a given piece of information, so it becomes more of a satisfaction matrix than a knowledge matrix. The type of fusion he uses is a simplified method where the resulting fused

satisfaction matrix is made up of the max value of each element of each of the input matrices. Otherwise stated, if F represents the resultant fused matrix, and I^n represent the input matrices, $F_{i,j} = max(I_{i,j}^1, I_{i,j}^2, ..., I_{i,j}^n)$ for all n input matrices. While this is a useful method for capturing statistics about the performance of different architectures, it fails to capture the synergy of intelligence or information fusion as Whaley states in his suggestions for future research.

Fusion methods have been tested in engineering simulations for the development of multi-sensor systems. Bossé *et al.* [2] developed a simulation testbed for the design of data fusion systems for military applications. This model supports the implementation of different fusion methods and algorithms to fuse the multi-sensor data and collect MOPs to determine which methods could be actual candidates for use in the development a real-world fusion architecture.

Another example of a fusion simulation modeling is by FOI, the Swedish Defence Research Agency. In 2002, FOI decided to build a simulation laboratory specifically for information fusion research [7]. Again the target audience is the military. The intent is to create, evaluate, and demonstrate new technologies for information fusion in military applications. They cited a need for a more flexible simulation framework that would allow the implementation and testing of different fusion methods.

The Knowledge Matrix concept was further developed in a RAND study for the United States Army [14]. The study took different pieces of information from either like or different intelligence disciplines and represented the level of knowledge from each piece of information in a knowledge matrix. The knowledge matrices were generated in a deterministic simulation model. Then they used an equation that they suggested was an extension of the Dempster-Shafer theory of evidence to combine the knowledge matrices into one knowledge matrix with knowledge probabilities greater than any of the input matrices. The equation can also be derived from probability theory by computing the joint probability of two independent events.

First observation (G)

Quality	Loc*	Track	ID	Act	Cap	Intent
5	0.118	0.00	0.0	0.00	0.00	0.00
4	0.393	0.00	0.30	0.00	0.00	0.90
3	0.865	0.00	0.80	0.00	0.00	0.90
2	1.0	0.70	0.85	0.00	0.00	0.90
1	1.0	0.90	0.90	0.00	0.90	0.90
0	1.0	0.95	1.0	1.0	0.92	0.90

Second observation (H)

Quality	Loc*	Track	ID	Act	Cap	Intent
5	0.001	0.00	0.70	0.30	0.00	0.00
4	0.005	0.00	0.80	0.40	0.00	0.00
3	0.020	0.00	0.90	0.50	0.70	0.00
2	0.393	0.70	0.95	0.60	0.80	0.30
1	1.00	0.80	0.99	0.70	0.90	0.40
0	1.00	0.90	1.00	0.80	0.92	0.50

Combined observation (K)

Quality	Loc*	Track	ID	Act	Cap	Intent
5	.121	0.00	0.70	0.30	0.00	0.00
4	.403	0.00	0.86	0.40	0.00	0.90
3	.873	0.00	0.98	0.50	0.70	0.90
2	1.00	0.91	0.99	0.60	0.80	0.93
1	1.00	0.98	1.00	0.70	0.99	0.94
0	1.00	1.00	1.00	1.00	0.99	0.95

Where $K_{i,j} = 1 - (1 - G_{i,j})(1 - H_{i,j})$

Figure 2.5   Fusing two input matrices into a combined matrix [14]

Their technique was to build a combined matrix, K, from observation matrices, G and H using the formula $K_{i,j} = 1 - (1 - G_{i,j})(1 - H_{i,j})$. Figure 2.5 illustrates how this equation is applied to the knowledge matrix. Note that each entry in the combined matrix is at least as good as the corresponding entries in either of the input matrices. This method captures the synergy that occurs due to information fusion. This also ensures that applying fusion will not decrease the level of knowledge. The equation can also be expanded to include more than two knowledge matrices while still adhering to the computation of the joint probability of the independent events.

3. Methodology

This chapter begins with a brief outline of the functionality of the original model developed by Pawling followed by a description of the changes made to implement data collection from multiple sources and the fusion of the data collected. The problem of input data is of importance, since real data was not available for this study, so a discussion of how the notional data was derived will be included. There will also be a section describing how the results of the simulation were calculated and collected.

3.1   Original Model

Pawling [13] created a model in Arena of each of the steps in the intelligence cycle. Each phase is organized into a submodel, and routing among these submodels is accomplished with a submodel referred to as the communications module. It begins with the Planning and Direction submodel. Here there are five different *users* modeled that each generate Requests for Information (RFIs). Each RFI gets assigned quality required, time required and priority level attributes. Each user creates standing RFIs that go directly to the Collection submodel and additional RFIs that get routed to a library search submodel. Upon reaching the Library Search submodel the RFI's timeliness is checked. If the timeliness has expired, defined by the simulation time being past the time required, TimeR, attribute value, the RFI is sent directly to the Evaluation submodel without any further processing. Otherwise, the RFI will undergo a delay and get assigned a quality achieved, QualA, attribute. Quality achieved is compared to quality required, QualR, to determine if the RFI must go the the Collection submodel. Before leaving the Planning submodel, each RFI is assigned binary attributes that will be used in the Communications submodel to route the RFIs to their required submodels. Not all RFIs will go through every part of the intelligence cycle because one of the intended purposes of the thesis was

to compare Task, Process, Exploit, Disseminate (TPED) and Task, Process, Post, Use (TPPU) methodologies.

The Communications submodel contains the logic that routes the RFIs to the submodels they are required to pass through. The logic is too complicated for standard Arena modules, so it is accomplished in VBA. Each of these submodels has the same underlying construct. First the timeliness of the RFI is checked. This is done by comparing the TimeR attribute to the difference between the current simulation time and the time the RFI was created. If the RFI is no longer timely, it is sent directly to the Evaluation submodel without any further processing. If it is timely, it will undergo a delay associated with the process modeled, and the QualA attribute is updated to represent the increase in quality associated with that process. Three of the submodels, Processing, Exploitation, and Analysis, also have an additional fusion branch built in. Each fusion branch consists of a timeliness check, a delay, and a quality update. Since fusion was not modeled as part of this study, the quality achieved remains unchanged after an RFI passes through this branch.

The last stop for every RFI in the model is the Evaluation submodel. Here all of the statistics are collected and measures such as quality, timeliness, and user satisfaction are evaluated. The actual disposal of each RFI takes place back in the Planning submodel rather than in the Evaluation submodel. This is done to facilitate the implementation of feedback as part of the intelligence cycle. Currently the model is not programmed to do anything with the feedback feature.

Also, some of the process times were represented with an exponential distribution. This makes it possible, although unlikely, to have unusually large processing times. These were replaced with triangular distributions that had similar means as the exponentials.For a more detailed explanation of Pawling's model, the reader is referred to his 2004 thesis.

3.2 Model Modifications

The first thing done was to eliminate some of the redundant VBA code. There were six VBA blocks in the Communications submodel. Each block had the same code in it with the exception of the number of if-then statements that would search the *next station* attributes for a value of one in order to route the RFI to that next station. The first block had seven, then the second block would strip off the first if-then statement leaving six. The third block would have five and so on. The code in the first block was kept and saved as a subroutine. Then the only code contained in each block was one line that called the subroutine. Additionally, just before an RFI would leave a station such as the Processing station, the attribute that indicated processing was required was changed from one to zero facilitating the use of a single subroutine rather than including a separate block of code that would not check for the processing attribute.

The next thing was to take out some portions of the model. The original model used a single attribute for quality required, QualR, and another single attribute for quality achieved, QualA. The QualA attribute was updated at each step of the intelligence process. Since the new method is to use a knowledge matrix, the individual attributes were deleted. The blocks that would update the QualA attribute were removed from each submodel. For the purposes of this study, the value of the knowledge matrix will retain the same values as assigned in the Collection submodel. The knowledge matrix will only change if multiple knowledge matrices are fused together in the Analysis submodel. The representations of fusion in the Processing and Exploitation submodels were removed. Additionally, many of the expressions that were functions of QualA or QualR had to be modified or removed because they were no longer being used.

3.2.1   Planning and Direction

The first major change to the Pawling model was the removal of the library search portion. As it was, each user modeled had two *create* nodes. One was for standing RFIs and the other for additional RFIs. It was assumed that standing RFIs could not be satisfied with information in the library and were routed directly the planning stage. Additional RFIs would go to a simulated library search. A discrete distribution would return an indicator of whether there was information in the library to satisfy the request; if so, the value of QualA was set. This value was checked against the QualR value. If it was good enough, then the RFI would skip the collection stage and proceed through the appropriate portions of the model. The number and frequency of RFIs being generated was changed. Instead of each user generating 20 RFIs at approximately 24-hour intervals, each user generates one RFI at approximately one-hour intervals.

With the use of the knowledge matrix, it is not sufficient to check a single quality value to determine if a requirement was met. In fact, to compare the different fusion methods, actual quality levels are recorded as statistics rather than counting the number that met the requirement versus the number that did not. Additionally, each RFI needs to go through the Collection submodel to simulate tasking different resources for intelligence and fusing the intelligence in the Analysis submodel. Substituting a library search for collection was therefore eliminated; however, the idea of a library was not removed from the model entirely. Four intelligence sources are represented in the Collection submodel, and one of them is OSINT or Open Source Intelligence. This source acts as a substitute for the library search portion of the model.

Since additional RFIs were sent directly to a library search, they were removed from the model as well. Only standing RFIs remain, and they are given nearly the same attributes as before with a few additions. The quality required was changed to a Uniform(0,1) distribution. This represents the level of confidence the user wishes to

3-4
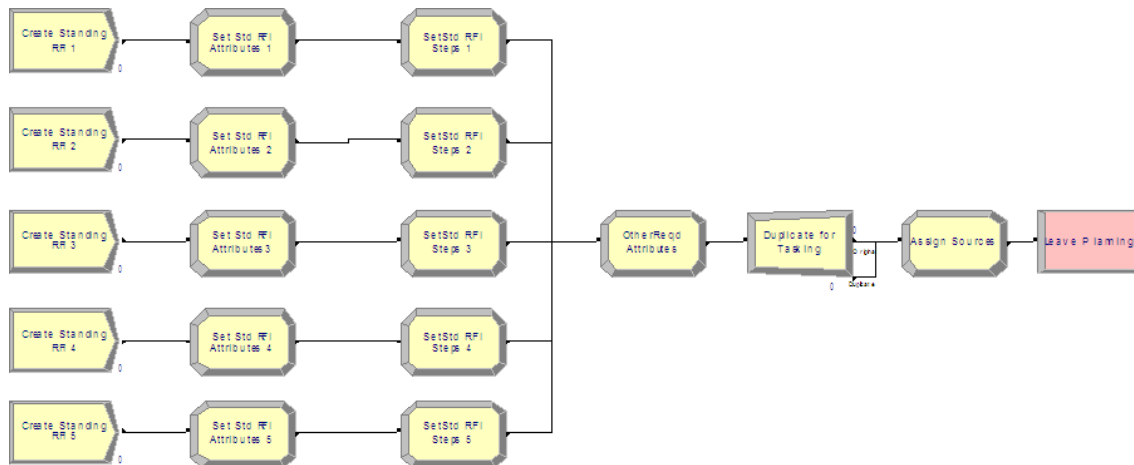
Figure 3.1   Planning and Direction Submodel

have about a particular type of knowledge represented by a column in the knowledge matrix, so there are actually six quality required attributes assigned to each RFI, QualR1-QualR6. Later, the quality level at which that confidence level is met gets checked and recorded into the replication statistics. Also, rather than assigning an information source attribute here, the number of sources to be used is set. Next the attributes for routing the RFI to the next station are set, and then the RFI gets duplicated. The number of sources dictates the number of duplicates. For example, if three sources are required, two duplicates are made. Each duplicated RFI has the same serial number as the original that was cloned. The result is a set of RFIs representing a single RFI. This is shown in Figure 3.1. The source to be tasked by each RFI is set just before leaving the Planning submodel.

3.2.2   Collection

Upon entering the Collection submodel (Figure 3.2), the 36 attributes representing the knowledge matrix are initialized. These attributes take the form "kmcr" where c and r both range in value from one to six. The first value, c, is associated with the types of knowledge in the matrix. For example one refers to location, two refers to track, three refers to identity and so forth. The second value, r, is associated

with the quality levels. Although the actual quality levels range from zero to five, they are represented with a range from one to six for ease of use in VBA code.
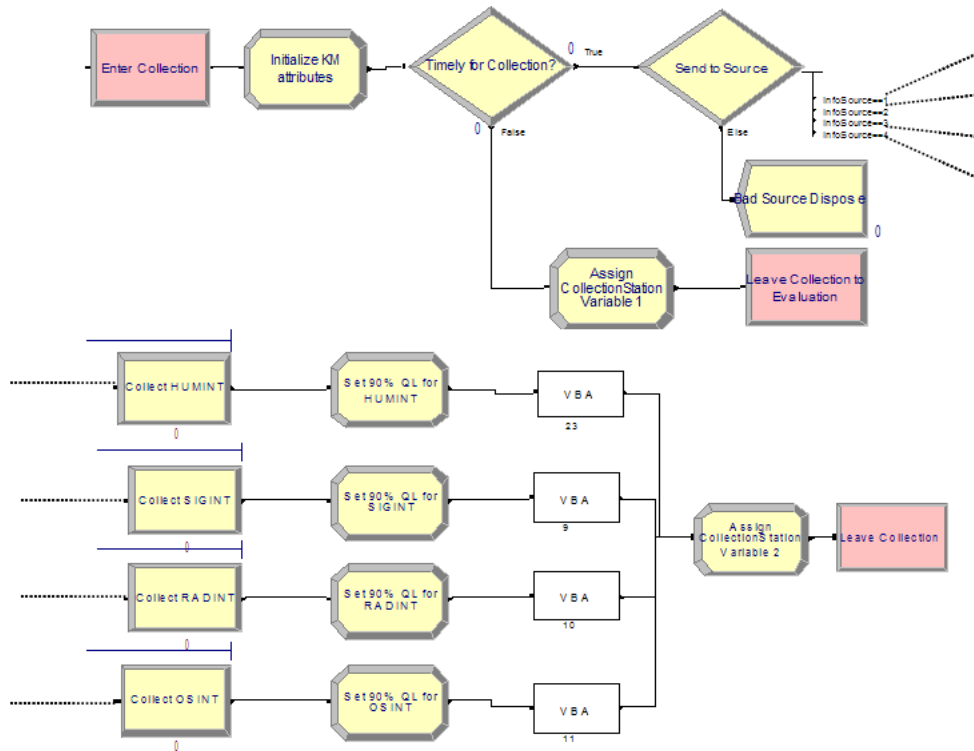
Figure 3.2   Collection Submodel

The timeliness of the RFI is checked before the simulated collection. If the time required has already passed, the RFI is routed directly to the Fusion submodel via the Analysis submodel. In the Pawling model, untimely RFIs were sent directly to the Evaluation submodel. Since the RFIs were duplicated in the Planning submodel, they must pass through the Fusion submodel to be batched back into a single RFI before being sent to Evaluation.

After the timeliness check, a *decision* node directs the RFIs to the appropriate source for collection where they pass through a *seize-delay-release* node to simulate the time and manpower needed to obtain the intelligence data. Next an attribute is set from a triangular distribution to represent the quality level each intelligence source will obtain for each type of intelligence with 90% confidence. These values are

3-6

used as a seed in the following VBA blocks to populate the knowledge matrix. This process is discussed in more detail in Section 3.3. The VBA code used to accomplish this can be found in the appendix. This code contains a call to a function called NormProb. This function was taken from the internet [1], and it calculates the value of the standard normal cumulative distribution function. Just before exiting this submodel, attributes used to route the entity through the Communications submodel to the next submodel are set.

3.2.3 Processing and Exploitation
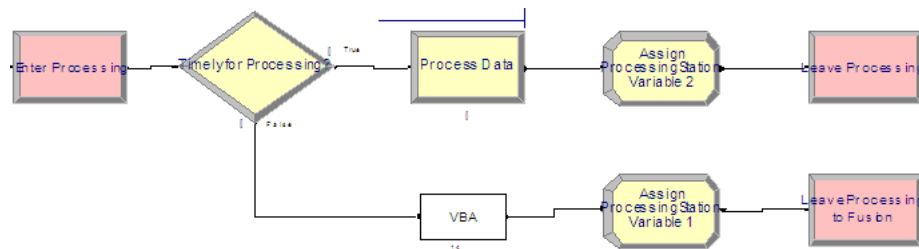
Figure 3.3   Processing Submodel

In the Pawling model, a resource would be seized as an RFI would enter either of these submodels. Following that, the timeliness was checked. If the RFI was not timely, the resource was released without any simulated time passing. If the RFI was timely, it would enter a *delay* node followed by a *release* node to free the resource. The following caveat was included:

> Although untimely items grab the resource, that resource is released af-
> ter a 0 time delay, effectively not using the resource. This should not
> affect time weighted statistics, but would affect discrete statistics such
> as number of times a resource was used.

The purpose for this could not be found, so these submodels were changed to perform the timeliness check first. If the RFI was deemed timely, it entered a single *seize-delay-release* node, assigned appropriate routing attributes, then sent to the Communications submodel. Untimely entities were sent to the Analysis submodel

3-7

to undergo fusion after having their knowledge matrix attributes reset to zero. This is done to replace the previous method of sending the RFI directly to Evaluation. The idea is that since the RFI was not properly processed and exploited due to its tardiness, the information collected (represented by the knowledge matrix) would be useless and therefore not considered in the fusion process. The Processing submodel is shown in Figure 3.3. The Exploitation submodel is identical to the Processing submodel.

3.2.4   Analysis
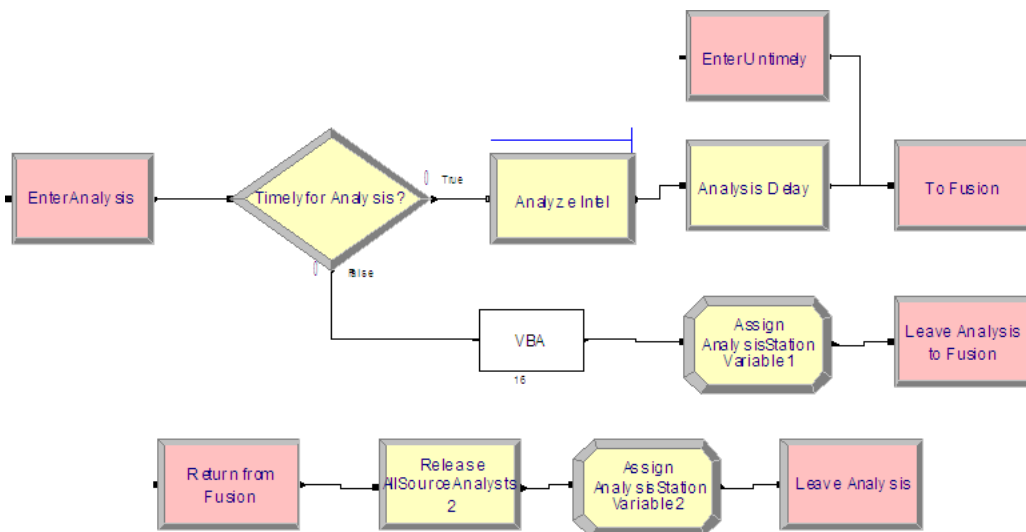
Figure 3.4   Analysis Submodel

The Analysis submodel is shown in Figure 3.4. The timeliness comes first, and untimely RFIs are sent directly to the Fusion submodel without being analyzed and therefore the knowledge matrix gets reset to zero as before. Prior to the trial runs, timely RFIs woulde seize a resource and undergo a delay before being sent to the Fusion submodel. Then after returning from the Fusion submodel, the resource would be released. This caused a bottleneck in the system because some RFIs would get stuck in the Fusion queue waiting for their duplicated counterparts to undergo fusion. The problem was that the duplicates could never get to the Fusion submodel

because there were no free analyst resources available to process them. This was remedied by implementing a single *seize-delay-release* node before sending the RFIs to the Fusion submodels.

Note that there is an additional station at the top of the figure to accommodate the untimely RFIs. The station labeled "To Fusion" is the only station manually altered to direct entities to the different Fusion submodels being compared. The Fusion submodels will be discussed in section 3.2.7. After returning from the Fusion submodel, the routing attributes are set, and the RFI is sent to the Communications submodel.

3.2.5 Production, Dissemination, and Integration

Each of these processes are modeled exactly the same way Processing is modeled. They were also changed from the original to perform the timeliness check first, then seize the resource if the RFI is timely. The only difference from the Processing submodel is that untimely RFIs are now sent directly to the Evaluation submodel as they will have already undergone fusion. Since these submodels look the same as the Processing submodel, additional figures are not included. As with the Pawling model, the Integration submodel is not implemented in this study. By doing this, it is assumed that the user is able to use the information as soon as they receive it. Timeliness is then determined by when the user receives the information. If the assumption is changed to say that the user will have to integrate the data in some fashion before they are able to use it, then the Integration submodel would have to be implemented and the timeliness definition be altered to when the user is actually able to use the data.

3.2.6 Evaluation

The Evaluation submodel collects all of the statistics used to evaluate the system. This is similar to the evaluation process of the intelligence cycle where
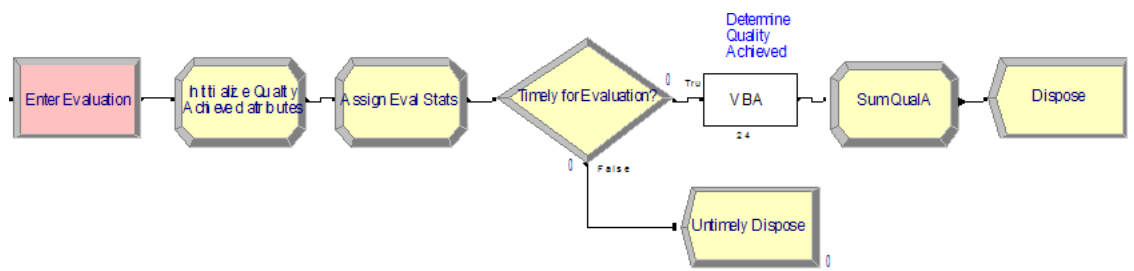
Figure 3.5   Evaluation Submodel

statistics are collected, analyzed, and used to continuously improve the process. When an RFI enters the Evaluation submodel, attributes to hold the quality achieved for each type of intelligence are initialized; then several variables are adjusted. The variables are listed in Table 3.1. Note that each variable is actually an array of five elements, one for each priority level, so data can be analyzed by priority level. They are not broken down by user because each user is identical in this notional study. If future research will include modeling different user request distributions, it may be necessary to break down the statistics by user as well.

Table 3.1   Evaluation Variables

Variable	Computation
S_Max_WaitTime_Priority(Priority_User)	MX(S_Max_WaitTime_Priority(Priority_User), Entity.WaitTime)
S_Tot_WaitTime_Priority(Priority_User)	S_Tot_WaitTime_Priority(Priority_User) + Entity.WaitTime
S_Num_Out_Priority(Priority_User)	S_Num_Out_Priority(Priority_User)+1
S_Num_Timely_Priority(Priority_User)	Timely*(S_Num_Timely_Priority(Priority_User) + 1) + (1 - Timely)* S_Num_Timely_Priority(Priority_User)

After the variables are calculated, untimely RFIs can be disposed. Timely RFIs enter a VBA block where the quality achieved for each type of intelligence is determined by comparing the quality required to the collected values in the knowledge matrix. For example, say the quality required for location, which was set in the Planning and Direction submodel, is 0.7. This means that the user requires that

3-10

the intelligence is accurate at a 70% level of confidence. Now assume the intelligence collected is as shown in Figure 3.6. The VBA code would determine that the best quality level that achieved that level of confidence was level 2.

QL	Prob
5	0.16
4	0.38
3	0.77
2	0.97
1	0.99
0	0.99

Figure 3.6   Determining the Quality of an Observation

3.2.7   Fusion Submodels
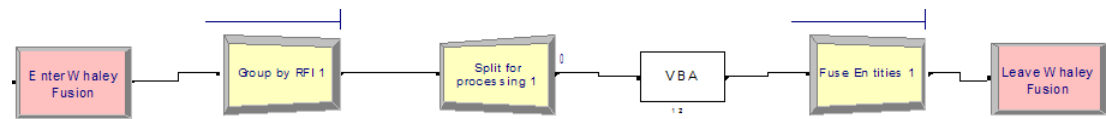
Figure 3.7   Fusion Submodel

There are three different Fusion submodels. The first is called Whaley Fusion and refers to the fusion method used in Whaley's 2005 thesis [16]. It is shown in Figure 3.7. When an RFI enters the submodel, it enters a batching queue. It waits here until all of the other RFIs with the same serial number reach the queue. The number of RFIs to wait for is contained in the attribute NumSources. This is done as a temporary batch, so the RFIs will retain their attributes once split again in the next node. The temporary grouping and splitting is done to ensure that the next RFI or RFIs that enter the VBA block are grouped by serial number set. The VBA code (located in the appendix) examines the values for the knowledge matrix, determines the maximum for each entry, then assigns the fused value to the knowledge matrix attributes of the last RFI of the set. The pseudocode for how this is accomplished is shown in Figure 3.8.

3-11

```
Get number of RFIs (NumEntities) from NumSources attribute
If NumEntities is greater than 1
   Get the serial number of the RFI
   If serial number not equal to current serial number (new RFI set)
      set index equal to 1
      set the current serial number to the serial number
      save each KM attribute to a temporary array
   Else
      increment the index
      compare each KM attribute to corresponding array value
      save the larger of the two to the temporary array
      If index equals NumEntities (last RFI of the set)
         Save array values to KM attributes of this RFI
      End If
   End If
End If
```

Figure 3.8   Pseudocode for Whaley Fusion

Next the set of RFIs go to a permanent *batch* node where only the attribute values of the last RFI are kept in the batched RFI. This models the fusing of the intelligence data. After fusion is complete, the RFI is returned to the Analysis submodel.

The Keithley Fusion submodel performs the same way as the Whaley Fusion submodel. The only difference is in the VBA code. Here all of the knowledge matrix attributes for a set of RFIs are stored in a temporary dynamically dimensioned array. This is done because all of the values must be used at the same time to perform the fusion calculation. In the Whaley model, only two values needed to be compared at a time, and the greater of the two was retained. The calculated values are stored in the attributes of the last RFI of the set to be kept in the final batched RFI representing an RFI with fused intelligence data. The actual VBA code is found in the appendix. The pseudocode for how this is accomplished is shown in Figure 3.9. Note that if there is only one RFI in the set, nothing happens because there is nothing to fuse.

3-12

```
Get number of RFIs (NumEntities) from NumSources attribute
If NumEntities is greater than 1
   Get the serial number of the RFI
   If serial number not equal to current serial number (new RFI set)
      set index equal to 1
      set the current serial number to the serial number
      declare a 6 X 6 X NumEntities array
      fill the first 6 X 6 "layer" with the KM values
   Else If this is not the start a new set of RFIs
      increment index
      fill the next 6 X 6 layer with the KM values
      If index equals NumEntities (last RFI of the set)
         compute Keithley formula using values in the layers for each element
         save formula result to KM attributes of this RFI
      End If
   End If
End If
```

Figure 3.9   Pseudocode for Keithley Fusion

The No Fusion submodel does nothing. Since no fusion takes place, the RFIs are not batched either. This results in more RFIs exiting the system than were created, and more statistics will be collected during these simulation runs than in the fusion runs. The submodel remains in the model in case future users wish to change this feature.

3.3   Obtaining Notional Data

Actual intelligence data is typically classified. Additionally, if this thesis were to crossover into a classified realm, then converting the actual intelligence reports into knowledge matrices for use as input into the model would be very time consuming making it impossible to complete this research in time. So the knowledge matrix data must be derived rather than collected. It would be unreasonable to just assign random numbers to the different quality levels within the matrix. Also, since different

sources of intelligence can obtain information in some areas better than others, an intelligent way of obtaining data is required.

3.3.1 Representing differences among resources

In this model, RFIs can be tasked out to more than one intelligence discipline for information collection. For the simplification of the model, only four possible sources are used. The model can be easily adapted to include more, but the initial study only incorporates Signals Intelligence (SIGINT), Human Intelligence (HUMINT), Radar Intelligence (RADINT), and Open Source Intelligence (OSINT). It is reasonable to assume that one intelligence discipline may be able to obtain better quality information about an object's location, intent, or capability than another discipline might. The representation of this fact is accomplished by choosing the initial value of any column of a knowledge matrix according to a specified distribution. Whaley [16] assigned triangular distributions to each column of the knowledge matrix to characterize the probability of each intelligence resource collecting information at a given quality level. Tables 3.2 - 3.5 show the parameter values used in the definition of the triangle distributions among the different areas of the knowledge matrix for each intelligence discipline. There is an exception in the area of location where the parameter values used are not the quality levels but the actual distances associated with the quality levels. Some of the distributions were modified slightly from Whaley's values.

Table 3.2   HUMINT parameters

Quality	Location	Track	Identity	Activity	Capability	Intent
5	Max		Max	Max	Max	Max
4						
3						
2	Mode	None	Mode	Mode	Mode	Mode
1						
0	Min		Min	Min	Min	Min

3-14

Table 3.3   RADINT parameters

Quality	Location	Track	Identity	Activity	Capability	Intent
5		Max				
4		Mode				
3	None					None
2		Min	Max	Max	Max	
1			Mode	Mode	Mode	
0			Min	Min	Min	

Table 3.4   SIGINT parameters

Quality	Location	Track	Identity	Activity	Capability	Intent
5			Max	Max	Max	Max
4		Max				
3	Max		Mode			
2	Mode	Mode		Mode	Mode	Mode
1						
0	Min	Min	Min	Min	Min	Min

Table 3.5   OSINT parameters

Quality	Location	Track	Identity	Activity	Capability	Intent
5						
4						Max
3		None		Max	Max	
2	Max					Mode
1	Mode		Max/Mode		Mode	
0	Min		Min	Min	Min	Min

3.3.2 Spreading the probabilities

Once an initial probability is found using the triangle distributions, the remainder of the probabilities must be filled in to each column. This is done differently for the location column than for the other columns because there are actual values assigned to the quality levels for location. For discussion, the value obtained from the triangular distribution for any given column of the knowledge matrix will be referred to with a capital X. Location being different from the rest of the areas will be discussed first. X is generated for a SIGINT observation from a triangular distribution using a min of 5 meters, a mode of 20 meters and a max of 100 meters.

Now for example let X be 80 meters. This simulates the event that a sensor detects an object and reports that the location of that object is known within 80 meters. In the case of a real sensor, there would be some error associated with that number, and the data collected from the sensor would actually be that the location of the entity is known within 80 meters within a 90% confidence level based on the Target Location Error (TLE) of the sensor. A 90% confidence level was used as a baseline throughout the study for the generation of all knowledge matrices. So a probability of .90 would be associated with a quality of 80 meters, but 80 meters is not one of the defined quality levels.

The standard normal distribution is used to determine the rest of the probabilities for the quality levels as was done by Keithley [8]. Five percent of the area under the upper tail of the standard normal distribution corresponds to a Z value of 1.65. This means that 90% of the area under the standard normal curve is between $Z = -1.65$ and $Z = 1.65$ as shown in Figure 3.10.

The standard normal variable is computed using $Z = (X - \mu)/\sigma$. For this case, $\mu = 0$ because it is taken to be the center of the circle with a radius of $X = 80$ meters, the TLE given for this particular observation. The standard deviation will be different for each observation depending on the TLE for that observation. The standard deviation, σ, can now be computed as $\sigma = 80/1.65 = 48.48$. The

90% of area
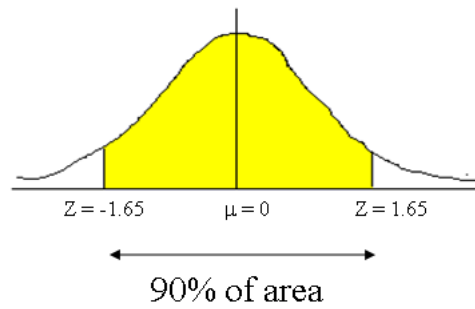
Figure 3.10   Area Under Standard Normal Curve

standardized variable, Z, can then be computed for each of the distances associated with the quality levels using $Z = x/48.48$, where lowercase x is substituted with the distance associated with each quality level. Then the probabilities are calculated by finding the area under the standard normal curve from $-Z$ to Z for each level as shown in Table 3.6. Probabilities such as .9999 are truncated to .99 rather than rounded up to 1.

Table 3.6   Spreading Location Probabilities

Quality	Location	$Z = x/48.48$	Probability
5	5	.103	.08
4	10	.206	.16
3	20	.413	.32
2	100	2.06	.96
1	1000	20.6	.99
0	10000	206.3	.99

For the rest of the columns, the number generated from the triangular distributions shown before will be in the range of the quality levels, but they will not be taken discretely. That means a number such as 3.7 could be returned. Just as before, this number is taken to be the 90% confidence level, so the Z variable associated with 3.7 will now be 1.65. Since there is no scaling among quality levels as there is with distance, each quality level is distributed along the Z axis with equal space in between as shown in Figure 3.11.

| QL	Prob
5	0.76
4	0.87
3	0.94
2	0.97
1	0.99
0	0.99
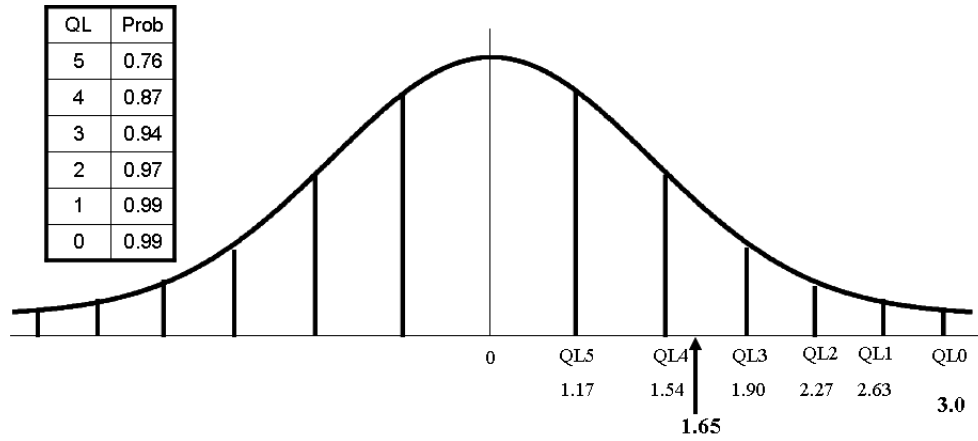
Figure 3.11   Spread of Quality Levels

The lowest quality level, 0, is assigned a value of $Z = 3$ to be used as an upper bound on the probabilities because it corresponds to a 0.99 confidence level. The ordered pairs (0,3) and (3.7,1.65) can now be taken as points on a line and the equation of that line can be determined as a function of quality level. This function is used to compute the Z value for each quality level, and the area under the standard normal curve is computed as before. If the function value is negative, then 0 probability is assigned to that quality level.

3.4   Using Neural Networks

At the onset of this study, the intention was to represent information fusion with a more sophisticated approach such as a Neural Network. The idea was to take some sample data, determine the appropriate number of nodes, layers, and weights to apply that would yield the appropriate outputs using Matlab or some other computing tool. Then armed with that knowledge, program the known Nerual Network formulation into a VBA block in one of the Fusion submodels.

The first attempt was with a basic version of a Neural Network known as the Adaline which comes from *ada*ptive *line*ar element [3]. The adaline is a single node network that learns to recognize linear patterns by minimizing the errors between

3-18

the network's output and the desired output. An image of the adaline is shown in
Figure 3.12.
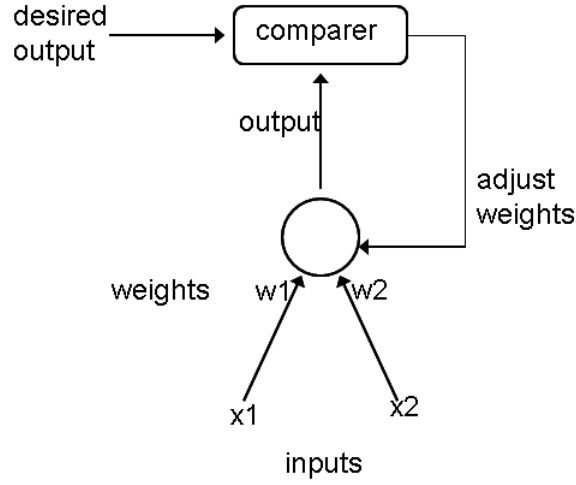
Figure 3.12   Adaline Neural Network Model

The output is computed by taking the weighted sum of the inputs as shown
here where $\mathbf{x}$ represents the input vector and $\mathbf{w}$ represents the vector of weights.

$$y = \mathbf{xw} \qquad (3.1)$$

The error (E) is computed by subtracting the actual output from the desired output.
Then a learning law called the delta rule is applied to adjust the weights. The new
weight vector, $\mathbf{w}$, is computed using the following equation:

$$\mathbf{w} = \mathbf{w}_{old} + \frac{\beta E \mathbf{x}}{|\mathbf{x}|^2} \qquad (3.2)$$

where β is a learning constant between 0 and 1.

This process was tested using Matlab. A 6 X 6 matrix of random numbers was
generated, referred to as *desired*. Then a matrix called *ob1* was created by dividing
the *desired* matrix by 2, and *ob2* was created by dividing the *desired* matrix by 4.
The *desired* matrix can now be computed using the expression $(1)ob1 + (2)ob2$. Thus

the weights would be $w_1 = 1$ and $w_2 = 2$. The inputs, *ob1* and *ob2*, were run through the adaline program and the resulting weights were not 1 and 2 as expected. The final weights after training the network were $w_1 = 5/3$ and $w_2 = 2/3$. This result is also correct because

$$(5/3)\frac{x}{2} + (2/3)\frac{x}{4} = x. \tag{3.3}$$

Once the weights are known, they can be programmed into a VBA block in the Arena simulation, and fusion can be represented via a Neural Network. The problem lies in the absence of training data. With no actual data to compare the output of intelligence fusion to, an adaline can not be properly trained. If one were to compute sample outputs from the notional input data to use for training, the adaline would be pointless because the relationship would have already been known by virtue of the computation to get the sample outputs. Until there is actual data to train a Neural Network, this approach will not be beneficial.

4. Results and Analysis

This chapter begins with a discussion of the verification and validation of the model. Then the method for determining the truncation point for statistics collection is explained. The initial set of runs yielded some unexpected results that indicated there was a problem with the model that was not evident in the trial runs used to validate the model. A solution to the problem was implemented and described in Section 4.3.2. The results of the simulation are shown next followed by a brief summary.

4.1   Validation and Verification

Validation is ensuring that the correct model was built. This is done by making sure that the model correctly describes the system being modeled. The original model was validated by Subject Matter Experts (SMEs) who approved of Pawling's model in its ability to accurately represent the intelligence process. This study did not change the basic structure of the original model. The most significant change was the removal of the library search portion. This was necessary because the object of this study was to compare the quality of intelligence data when it is fused versus not being fused. For this reason, performing an initial library search and exiting the process if a certain quality threshold was met is not applicable. However, a library, or Open Source, was included as one of the intelligence sources in the Collection submodel. Other minimal changes that did not affect the overriding structure of the Pawling model included only seizing resources in a submodel after the timeliness of the RFI had been checked and removing some of the fusion legs that were not being implemented from some of the submodels.

Verification is making sure the model was built correctly. There were several things done to ensure the accuracy of the model. The original model was built to compare two approaches to implement the intelligence process. They are Task,

0.99	0.99	0	0.99	0.99	0.99
0.99	0.98	0	0.98	0.99	0.6
0.21	0.92	0	0.89	0.94	0
0.04	0.75	0	0.63	0.82	0
0.02	0.41	0	0.15	0.57	0
0.01	0	0	0	0.18	0

0.99	0.99	0.99	0.99	0.99	0.99		0.9999	0.9999	0.99	0.9999	0.9999	0.9999
0.99	0.93	0.99	0.97	0.11	0.98		0.9999	0.9986	0.99	0.9994	0.9911	0.992
0.31	0.44	0.96	0.81	0	0.91		0.4549	0.9552	0.96	0.9791	0.94	0.91
0.06	0	0.88	0.37	0	0.71		0.0976	0.75	0.88	0.7669	0.82	0.71
0.03	0	0.72	0	0	0.33		0.0494	0.41	0.72	0.15	0.57	0.33
0.02	0	0.45	0	0	0		0.0298	0	0.45	0	0.18	0

Figure 4.1   Sample Knowledge Matrix Outputs

Process, Exploit, Disseminate (TPED) and Task, Process, Post, Use (TPPU). For this reason, some of the RFIs would skip some of the submodels in the simulation. Since this study took a different approach, each RFI was made to pass through every submodel. Then the animation was turned on, and a minimal amount of RFIs were introduced into the system. Their progress was monitored as they traversed the model. This was done several times with different attribute values and distributions to make sure all paths in the model were correctly followed when necessary. In doing so, a redundancy was found in the form of a timeliness check. RFIs were being checked for timeliness every time they left the Communications submodel as well as upon entering the next submodel. The timeliness check in the Communications submodel was removed.

Another portion of the model that had to be verified is the VBA code used in the Fusion submodels. This was done by adding additional code that would output the knowledge matrices to an Excel spreadsheet for viewing. One RFI was introduced to the system, and a single duplicate was created. After each RFI passed through the Collection submodel, the knowledge matrices were outputted to Excel. After the RFIs were fused, the fused knowledge matrix was outputted to Excel. An example of this output is shown in Figure 4.1. On the left are the two input matrices, and the right matrix is the result of fusion using the Keithley method. The values appear 'upside down' because of the way the loop was set up in the code that outputted the

data to the spreadsheet. Doing this helped discover the need to make the variables CurSerNum and index global variables rather than local as they were being reset every time a new RFI entered the VBA block.

4.2 Initial Transient

When performing a steady state simulation, the initial output depends heavily on the initial condition of the system. If the queues are initialized in an empty state, it will take the system some time to ramp up and reach a steady state where the output is no longer influenced by the initial condition. It is at this time that statistics collection should begin; otherwise, the statistics will be biased on the initial condition.

In this model, each submodel represents a process that has a queue. Each queue starts in an empty state. The amount of time spent waiting in each queue and going through each process affects the timeliness of the RFIs. The percent timely statistic can give a good indication of when all of the queues have reached a steady state. The simulation was set up to run using the Whaley fusion submodel for ten replications of 365 days each. At the beginning of each simulated day, the percent timely statistic was computed by dividing the number of timely RFIs exiting the system by the total number of RFIs exiting the system. The daily ratios were outputted to Excel, then copied into Matlab for analysis. The observations were averaged over the ten reps. The moving average of the means of the observations with a window size of ten observations is shown in Figure 4.2. The graph indicates that an appropriate truncation point is at about 160 days.
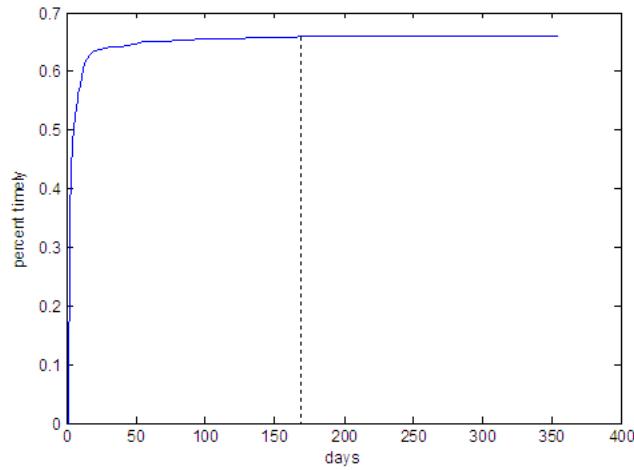
Figure 4.2   Moving Average Plot

4.3   Results

It was expected that both methods of representing fusion would yield higher quality levels than the method of no fusion. It is also expected that the Keithley method will yield higher quality levels then the Whaley method. This can be proven without simulation as shown in Figure 4.3, but the question to answer is how much better will it actually be? The Keithley method takes more memory space and computational time to implement than the Whaley method. The Whaley method only requires a 6 X 6 X 1 dimensional array, but the third dimension of the array using the Keithley method can be as large as there are information sources used to collect the intelligence. In this study it is a maximum of four, so the difference is inconsequential. Also, the Keithley method took 143.33 minutes to run compared to 143.07 for the Whaley method. This is a small difference, but again the maximum number of intelligence sources was four. This difference will increase as the number of sources increases.

$$
\begin{aligned}
\textbf{Prove: } 1 - (1-A)(1-B) &\geq \max(A,B) \\
\textbf{Proof: } \text{Assume } \max(A,B) &= A \\
1 - (1-A)(1-B) &= 1 - (1 - A - B + AB) \\
&= A + B - AB \\
\text{Since } A \in [0,1], AB &\leq B \\
\Rightarrow B - AB &\geq 0 \\
\Rightarrow A + (B - AB) &\geq A = \max(A,B)
\end{aligned}
$$

Figure 4.3   Proof of Expected Result

4.3.1   Initial Results

The output of the first run seemed to yield reasonable results. The quality levels, shown in Figure 4.4, were all at expected levels. The fusion methods outperformed the non-fusion method, and the Keithley method outperformed the Whaley method on average. Further exploration into the results showed a flaw in the system. Each priority level is equally likely to occur, and when the quality levels of the different areas of intelligence were broken down by priority level, priority four RFIs had higher quality levels with the no-fusion method than with either fusion method. Additionally, no priority five RFIs were making it through the system as timely. This disparity was most pronounced in the identification area where the quality level for for priority four RFIs with no fusion exceeded all other quality levels for identification (see Figure 4.5).

A step-by-step examination of the system was conducted to find the cause of this apparent error. It was discovered that HUMINT and SIGINT yield the highest quality levels for identification. They are also the two most likely forms of intelligence to be processed because they are processed at higher rates than RADINT or OSINT. This means that when no fusion is taking place, more HUMINT and SIGINT tasked RFIs exit the system as timely yielding increased quality averages. When fusion
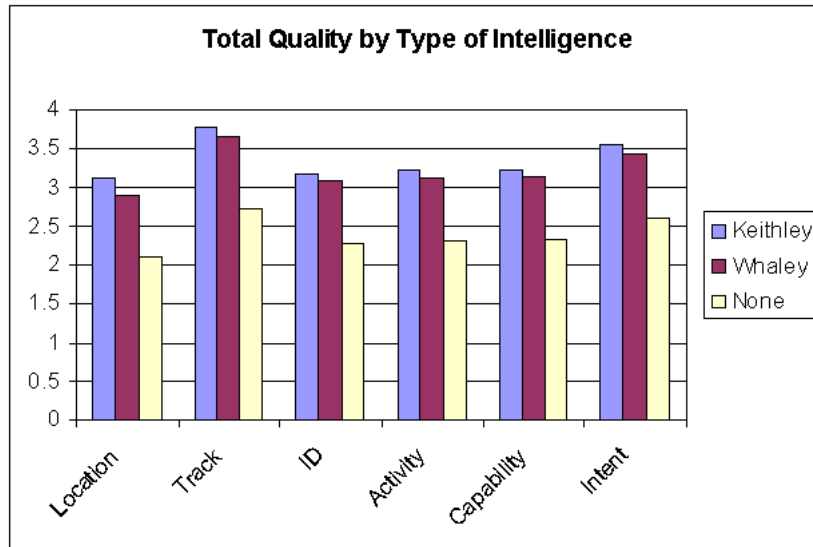
Figure 4.4   Average Quality Levels from Initial Run
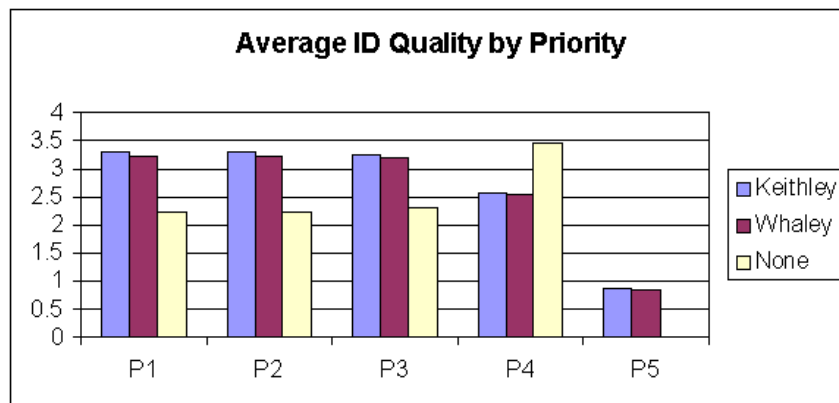
Figure 4.5   Identification Quality by Priority from Initial Run

takes place, the RADINT and OSINT tasked RFIs go to the batch queue in the Fusion submodel to wait for their duplicate RFIs from other sources. Statistics on the batch queue show that for each run, there is an average of 7580 RFIs in the queue waiting for an average of 250 hours. This does not accurately depict the real world system. Once RFIs are past their timely requirement, they should stop waiting and continue on through the system.

4.3.2 Reneging

The solution to this problem was to add a way to simulate reneging in the model. The RFIs are processed in a lowest value first order depending on the Priority_User attribute, so priority five RFIs must wait until the resource is free to be processed. This only happens when all other priority one, two, three, and four RFIs have been processed. When no fusion takes place, the resources are so overloaded with RFIs, no priority five RFIs ever get processed. They remain in the process queue for the entire simulation. An additional submodel was added to solve this problem (Figure 4.6). There is a *separate* node in the Planning and Direction submodel that duplicated the RFIs. The number of duplicates was one less than the number of sources, NumSources, required to fulfill the RFI. After duplication, both the original and duplicates are sent through the model, so the number of RFIs with the same serial number is equal to NumSources.

This was modified to create a number of duplicates equal to NumSources. Then the duplicates are sent through the model while the original is sent to the Renege submodel. Upon entering the Reneging submodel, the RFI is delayed for a number of hours equal to the time required, TimeR, attribute. Then the serial number of the RFI is saved to a variable, and the RFI enters a *search* node. It is important that the value is saved to a variable and not an attribute; otherwise, the search condition will not work. Once in the *search* node, the specified queue is searched to see if an RFI exists in that queue with the serial number attribute equal to the serial number
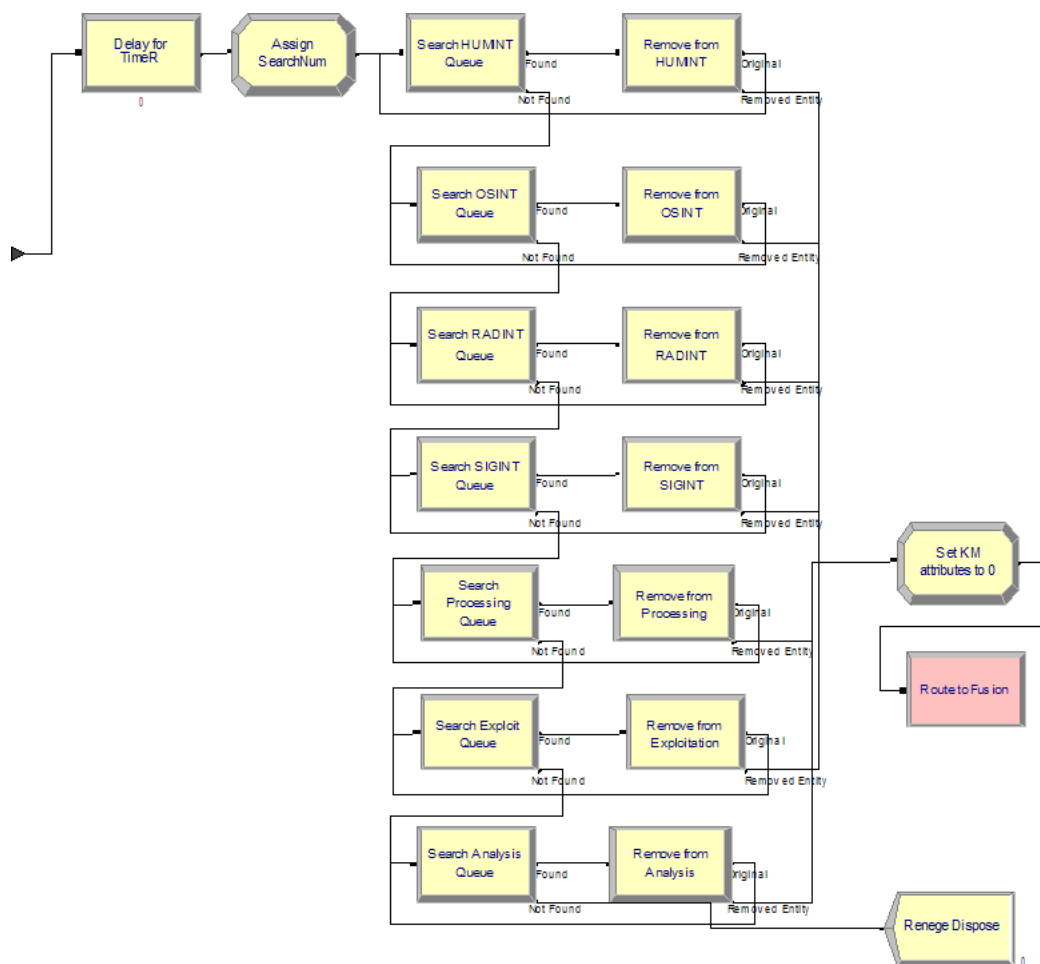
Figure 4.6   Reneging Submodel

variable just saved. If a match is found, the RFI is removed from the queue. The KM attributes of the removed RFI are set to zero because the RFI is no longer timely. Then the RFI is sent to the Enter Untimely station in the Analysis submodel where it will proceed to one of the Fusion submodels. If the search is unsuccessful, the next queue is searched until all queues have been searched for untimely RFIs. If any of the searches is successful, the original RFI is looped back to search that queue again because there could be more than one RFI with the same serial number in the queue.

One more reneging branch was added at the end of the Analysis submodel. Since the RFIs have been fused at this point, another duplicate is made. The duplicate is sent to a *delay* node for an amount of time equal to TimeR - Entity.CreateTime, or the amount of time left before the RFI is no longer timely. Then the production queue is searched for untimely RFIs the same was as before. The dissemination and integration queues are not searched because they are processed differently and do not cause a priority back-up as the other queues do.

4.3.3   Final Results

After adding the Reneging submodel, the model was rerun for each fusion method. The final quality results are shown in Figure 4.7. The values in Figure 4.7 are the averages of the mean quality level from each run. The 95% confidence intervals around the means is small, on the order of 0.01, indicating small variance in the data. Two sample t-tests were performed on the quality level results using Excel. The t-test results (found in the appendix) show that the differences in quality levels for all six types of intelligence are significant at $\alpha = 0.05$ for the Keithley vs None tests, the Whaley vs None tests, and the Keithley vs Whaley tests. So it can be said that representing fusion with the Keithley method yields statistically higher quality levels at $\alpha = 0.05$ than using the Whaley method. Both fusion methods yield statistically higher quality levels at $\alpha = 0.05$ than using no fusion at all.
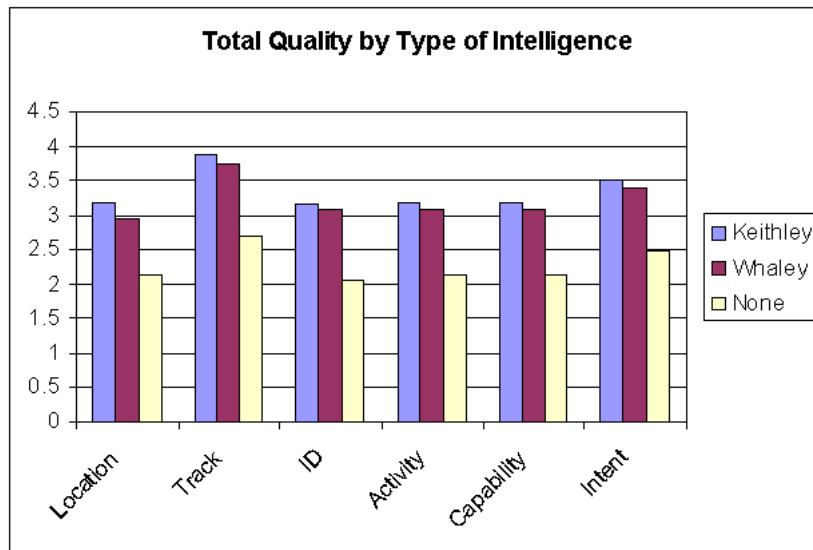
Figure 4.7   Average Quality Levels by Knowledge Area

One of the questions posed previously was how much better the Keithley method would be compared to the Whaley method. Table 4.1 reveals that the difference between the two is very small. Both fusion methods produce results that are about one whole quality level better than using no fusion. The difference between the Keithley and Whaley methods do not produce significant quality increases when the levels are considered discretely rather than continuously. This means that the quality levels achieved for each type of intelligence for the individual RFIs will likely not be different for the Keithley or Whaley methods. However, the probability level associated with that quality level will be higher for the Keithley method.

Table 4.1   Mean Quality Difference Between Methods

Knowledge Type	K vs W	W vs N	K vs N
Location	0.236	0.796	1.032
Track	0.128	1.058	1.186
Identity	0.075	1.019	1.094
Activity	0.093	0.958	1.051
Capability	0.089	0.949	1.039
Intent	0.120	0.904	1.024
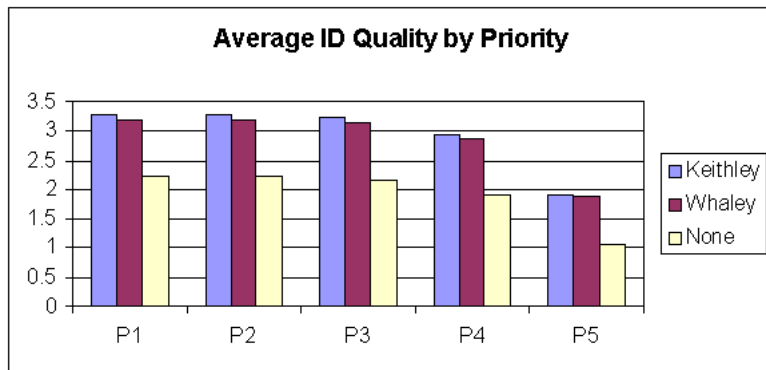
Figure 4.8   Identification Quality by Priority After Reneging

The Whaley method can be thought of as a lower bound on the effects of fusion. The best quality from each piece of intelligence is kept, and no synergy takes place. The Keithley method suggests an upper bound on the effects of fusion because it is assumed that each piece of intelligence is independent. So it is somewhat surprising that the difference between the two methods was not greater. One hypothesis is that because the Quality Required(QualR) was modeled as a Uniform(0,1) number, a low QualR is just as likely as a high QualR. Low values for QualR will result in few differences in the Quality Attained for each method. Another hypothesis is that when the RFIs are duplicated for collection, the number of sources is 1, 2, 3, or 4. Since each value is equally likely, 25% of the RFIs do not get tasked to more than one source, and no fusion takes place. This will bias the quality statistics used to compare the methods.

The problem with the escalated priority four quality levels and the absence of priority five RFIs was corrected with the added reneging logic in the model. Figure 4.8 shows the same chart as before with the quality levels for identification knowledge broken out by priority level. Figure 4.8 also illustrates the apparent decline in quality level for the lower priority RFIs. This is the same result that would likely be expected in a real world process.

4-11
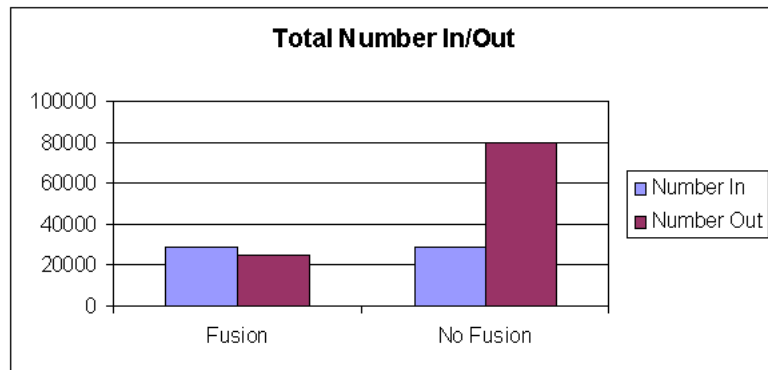
Figure 4.9   Number of RFIs Entering and Exiting System

Other statistics of interest were identical for both the Keithley and Whaley fusion methods. This is because they go through exactly the same processes with the same distributions. The only difference between the two is the way the quality levels are computed in the VBA code and the length of time it takes for that VBA code to execute. VBA processing time does not affect simulation statistics. The remainder of the statistics are shown as fusion versus non-fusion because the data is the same for both fusion methods.

The number of RFIs entering and exiting the model was recorded and is shown in Figure 4.9. The number in is approximately the same for each. The difference of course is with the number out. Note that the number out displayed in this figure does not include the entities used in the Reneging portion of the model. In the no fusion method, each RFI is duplicated anywhere from one to four times. The number of duplicates it equally likely to be one, two, three, or four. The expected value of this distribution is 2.5. So one would expect to see about 2.5 times as many RFIs exiting the system as there are entering the system in the no fusion case because the RFIs do not get fused back together. This makes it interesting to note that the number out for no fusion was actually 2.7 times as many as the number in. There are approximately equal numbers in for each priority level because each priority level is equally likely to be assigned. The number out decreases by priority level.
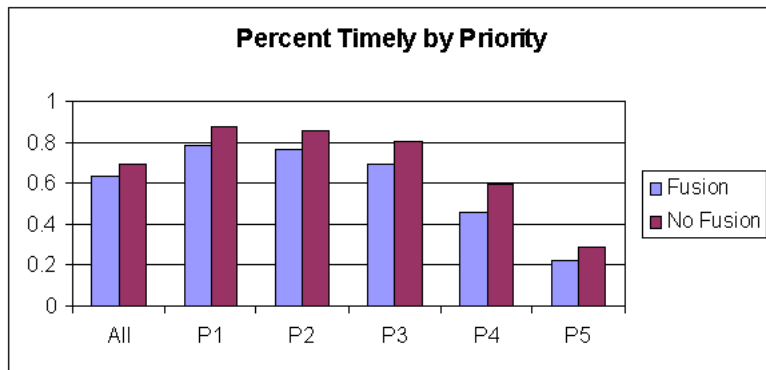
Figure 4.10   Percent of timely RFIs Exiting System

The percent of timely RFIs is shown in Figure 4.10. For the fusion methods, the overall percent of timely RFIs was about 63%. This is consistent with the steady state value seen in Section 4.2. As expected, the percent of timely RFIs decreases as the priority level gets lower. The percent of timely RFIs is higher when no fusion takes place not because there are more RFIs in the no fusion system, but because the RFIs do not experience a fusion delay as they wait in the batch queue at the beginning of the Fusion submodels.

Table 4.2   Number of Reneging RFIs by Priority

	Fusion	No Fusion
Priority 1	37.3	63.4
Priority 2	73.8	118.75
Priority 3	289.25	385
Priority 4	3309.8	4404.4
Priority 5	13045	16503

The average number of RFIs removed from queues from reneging is shown in Table 4.2. There are very few priority one RFIs that get sent on through the model unprocessed. There are a considerable number of priority five RFIs that go unprocessed. Any RFIs that are removed from a process queue have their KM attributes set to zero because in the model, they no longer add value to the user if they have not been properly processed. This explains why the average quality level decreases as the priority decreases.

4.4   Summary

The model was initially validated using several trial runs with a very small number of RFIs. The animation was turned on, and the parameters were adjusted to force the RFIs into each portion of the model. The model appeared to be working properly, and a truncation point was determined. The backlog in the Fusion submodel that affected the quality levels of lower priority RFIs was discovered only after the model was run for an extended period. The Reneging submodel was added, and the problem was corrected. The statistics show that quality levels are increased when fusion occurs. The Keithley method for modeling fusion yields slightly higher quality levels than the Whaley method but the difference is negligible. The percent of timely RFIs increases when no fusion takes place, but increased quantity does not imply increased quality.

5. Conclusions and Future Research

The research in this thesis is a follow-on effort to the research done by Pawling[13] and Whaley[16]. Pawling developed the first model with a high level of abstraction for the National Security Space Architect (NSSA) to be able to do quick turn studies of the intelligence process. This model lacked fusion. Whaley then produced another more detailed model that included a basic method to represent fusion, but the model was very complicated and did not use a very robust fusion method. This work took the model created by Pawling, stripped away some extraneous elements, and added two methods for representing fusion. One method was the one proposed by Whaley with minor modifications, and the other was proposed by Keithley[8] and further investigated by RAND[14]. The results of the study are summarized below with a discussion of some of the lessons learned in producing the model. This is followed by suggestions for future research.

5.1 Conclusions

Adding fusion to the intelligence process model increases the quality levels of the intelligence data produced. This was first shown in Whaley's thesis, but the fusion method used there failed to capture the synergy of information fusion. This was due to two issues. One was the formula used to compute the quality levels of the fused information. The other was the lack of use of a complete knowledge matrix. Whaley only used a portion of the knowledge matrix, a six-dimensional vector of the quality levels that met or exceeded the quality required. This model implemented a complete 6 X 6 knowledge matrix to represent the quality of the information collected for each RFI.

Another important feature that was implemented in this study was the addition of entity reneging. This was not present in either of the above studies, so it was possible for lower priority RFIs to get permanently stuck in process queues. Reneging

models the situation where someone gets in a line for a service and waits for a period of time before giving up and leaving the line. Without the reneging feature, RFIs would get stuck in queues and never make it to the fusion portion of the model. This would cause a backup in the fusion queue of RFIs waiting for their counterparts that were stuck in other queues. This had an adverse affect on the statistics collected at the end of the simulation.

The final result showed that no fusion produced many more timely RFIs, but they were of lower quality than the fused RFIs. The Whaley method increased the quality of the fused RFIs to the maximum quality level of each of the RFIs being fused, but failed to capture the synergistic effect of information fusion. The Keithley method is based on fundamental probability theory. It uses the formula for computing the joint probability of independent events. It produces the highest quality levels because it captures the fusion synergy.

5.1.1 Lessons Learned

The most important takeaway from this research is in regard to model validation. After building the model, it was thoroughly checked to make sure that every logical element was working as designed. Small numbers of entities were sent through every portion of the model with their movement controlled by adjusting parameters to constant values that would direct them through the portion of the model being checked. Once everything check out, the model was run for an extended period of time. Here is was determined that the model was in fact not valid. A bottleneck occurred in the Fusion submodel, particularly for low priority RFIs. This was corrected by adding reneging to the model, but the lesson learned is to validate and then revalidate.

Another important lesson is how to work with VBA in Arena. This is mainly concerned with using VBA variables within the Arena construct. First, when processing values from more than one entity, all variables concerned must be declared as

global variables at the beginning of the VBA code rather than inside the subroutine executed when the entity enters a VBA block. Every time a new entity enters the VBA block, the variables declared within that blocks subroutine are reset, and their values are empty. Another important detail is that variables that hold counting numbers should be declared as single or double precision values rather than integer. When large simulations are run, the values of the variables exceed the limits of integer variables.

5.2 Future Research

The first intent of this research was to use a more sophisticated representation of fusion in the intelligence process such as a neural network. Literature was consulted and a basic neural network was developed to represent the fusion of two knowledge matrices into one knowledge matrix. Neural networks require initial data in order to be properly trained. The relationships between the inputs and outputs must be established, so the weights of the nodes in the network can be properly set. Since there is no data available to train the neural network, this method could not be implemented. If data was made available, and that data could be quantified in the form of a knowledge matrix, then it is suggested that commercial neural network software be used to train a network. Then take the settings from that network and program them into the VBA code that performs fusion in the model.

This model can be used by any agency looking to study the intelligence collection process. Future research might include adding more detail to any portion of the model or reintroducing the library search portion of the Pawling model. Further investigation of the hypotheses stated in section 4.3.3 is recommended to determine if the difference between the two methods of fusion might actually be greater than the difference found in this study. Another important suggestion would be to obtain actual data to determine the appropriate distributions throughout the model. Since

5-3

actual data may be classified or difficult to obtain, perhaps a subject matter expert could offer suggestions or estimates of actual distributions.

Bibliography

1. Berg, T. V. D. "Calculating the Cumulative Normal Distribution," *SITMO*, http://www.sitmo.com/doc/Calculating_the_Cumulative_Normal_Distribution, Jan 2008.

2. Bosse, E., J. Roy, and S. Paradis. "Modeling and Simulation in Support of the Design of a Data Fusion System". *Information Fusion 1: 77-87*, 2000.

3. Caudill, M. and C. Butler. *Understanding Neural Networks: Computer Explorations*, 37-39, The MIT Press, Cambridge, MA, 1992.

4. Dasarathy, B. V. "Elucidative Fusion Systems - An Exposition". *Information Fusion 1: 5-15*, 2000.

5. Department of Defense. *Joint Intelligence.* Joint Publication 2-0. Washington: GPO, June 2007.

6. Hill, R. R., J. O. Miller, and G. A. McIntyre. "Applications of Discrete Event Simulation Modeling to Military Problems". *Proceedings of the 2001 Winter Simulation Conference: 780-788*, 2001.

7. Horling, P., V. Mojtahed, P. Svensson, and B. Spearing. "Adapting a Commercial Simulation Framework to the Needs of Information Fusion Research". *Procedings of the Fifth International Conference on Information Fusion: 220-227*, July 2002.

8. Keithley, H. *Multi-INT Fusion Performance Study.* Joint C4ISR Decision Spport Center, DCS-00-02, 2000.

9. Kelton, W. D., R. P. Sadowski, and D. T. Sturrock. *Simulation with Arena*, McGraw-Hill, 2007.

10. Klein, L. A. *Sensor and Data Fusion.* Washington: SPIE-The International Society for Optical Engineering, 2004.

11. Moore III, L. R., and D. Gonzales. *Measuring the Value of High Level Fusion.* Santa Monica CA: RAND Report, June 2004.

12. Nakamura, E.F., A. F. Loureiro, and A. C. Frery. "Information Fusion for Wireless Sensor Networks: Methods, Models, and Classifications". *ACM Computing Surveys*, 39(3):Article 9, August 2007.

13. Pawling, C. 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, Wright-Patterson AFB OH, March 2004.

14. Pernin, C. G., L. R. Moore, and K. Comanor. *The Knowledge Matrix Approach to Intelligence Fusion.* Santa Monica CA: RAND Report, 2007.

15. RAND Corporation. *Notes on Strategic Air Intelligence in World War II*, 1949.

16. Whaley, K. J. *A Knowledge Matrix Modeling of the Intelligence Cycle.* MS thesis, AFIT/GOR/ENS/05-18. School of Engineering and Management, Air Force Institute of Technology, Wright-Patterson AFB OH, March 2005.

Appendix A. Two Sample t-tests

The following Appendix contains the two sample t tests performed to compare the average quality levels of the fusion methods against each other and against no fusion. The tests were done in Excel using the output statistics from the Arena model. All tests are significant at $\alpha = 0.05$.

Average Activity Quality Levels

t-Test: Two-Sample Assuming Equal Variances
Whaley vs Keithley

	Variable 1	Variable 2
Mean	3.08196	3.17498
Variance	8.53499E-05	7.39185E-05
Observations	20	20
Pooled Variance	7.96342E-05	
Hypothesized Mean Difference	0	
df	38	
t Stat	-32.96298218	
P(T<=t) one-tail	7.28582E-30	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	1.45716E-29	
t Critical two-tail	2.024394147	

t-Test: Two-Sample Assuming Equal Variances
Whaley vs None

	Variable 1	Variable 2
Mean	3.08196	2.12428
Variance	8.53499E-05	4.32859E-05
Observations	20	20
Pooled Variance	6.43179E-05	
Hypothesized Mean Difference	0	
df	38	
t Stat	377.6195823	
P(T<=t) one-tail	7.83132E-70	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	1.56626E-69	
t Critical two-tail	2.024394147	

t-Test: Two-Sample Assuming Equal Variances
Keithley vs None

	Variable 1	Variable 2
Mean	3.17498	2.12428
Variance	7.39185E-05	4.32859E-05
Observations	20	20
Pooled Variance	5.86022E-05	
Hypothesized Mean Difference	0	
df	38	
t Stat	434.0319831	
P(T<=t) one-tail	3.95004E-72	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	7.90007E-72	
t Critical two-tail	2.024394147	

Average Capability Quality Levels

t-Test: Two-Sample Assuming Equal Variances
Whaley vs Keithley

	Variable 1	Variable 2
Mean	3.093085	3.18307
Variance	0.000163092	0.000146118
Observations	20	20
Pooled Variance	0.000154605	
Hypothesized Mean Difference	0	
df	38	
t Stat	-22.88539675	
P(T<=t) one-tail	3.95624E-24	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	7.91249E-24	
t Critical two-tail	2.024394147	

t-Test: Two-Sample Assuming Equal Variances
Whaley vs None

	Variable 1	Variable 2
Mean	3.093085	2.14374
Variance	0.000163092	3.14457E-05
Observations	20	20
Pooled Variance	9.72688E-05	
Hypothesized Mean Difference	0	
df	38	
t Stat	304.3948786	
P(T<=t) one-tail	2.81924E-66	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	5.63848E-66	
t Critical two-tail	2.024394147	

t-Test: Two-Sample Assuming Equal Variances
Keithley vs None

	Variable 1	Variable 2
Mean	3.18307	2.14374
Variance	0.000146118	3.14457E-05
Observations	20	20
Pooled Variance	8.87818E-05	
Hypothesized Mean Difference	0	
df	38	
t Stat	348.8119759	
P(T<=t) one-tail	1.59612E-68	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	3.19223E-68	
t Critical two-tail	2.024394147	

Average Identity Quality Levels

t-Test: Two-Sample Assuming Equal Variances
Whaley vs Keithley

	Variable 1	Variable 2
Mean	3.085655	3.16056
Variance	0.000172685	0.000146849
Observations	20	20
Pooled Variance	0.000159767	
Hypothesized Mean Difference	0	
df	38	
t Stat	-18.73991308	
P(T<=t) one-tail	4.28786E-21	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	8.57572E-21	
t Critical two-tail	2.024394147	

t-Test: Two-Sample Assuming Equal Variances
Whaley vs None

	Variable 1	Variable 2
Mean	3.085655	2.06651
Variance	0.000172685	5.89757E-05
Observations	20	20
Pooled Variance	0.00011583	
Hypothesized Mean Difference	0	
df	38	
t Stat	299.4505258	
P(T<=t) one-tail	5.25156E-66	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	1.05031E-65	
t Critical two-tail	2.024394147	

t-Test: Two-Sample Assuming Equal Variances
Keithley vs None

	Variable 1	Variable 2
Mean	3.16056	2.06651
Variance	0.000146849	5.89757E-05
Observations	20	20
Pooled Variance	0.000102912	
Hypothesized Mean Difference	0	
df	38	
t Stat	341.0386543	
P(T<=t) one-tail	3.75738E-68	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	7.51475E-68	
t Critical two-tail	2.024394147	

Average Intent Quality Levels

t-Test: Two-Sample Assuming Equal Variances
Whaley vs Keithley

	Variable 1	Variable 2
Mean	3.37561	3.49585
Variance	0.000160981	0.000160262
Observations	20	20
Pooled Variance	0.000160621	
Hypothesized Mean Difference	0	
df	38	
t Stat	-30.0018095	
P(T<=t) one-tail	2.28962E-28	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	4.57923E-28	
t Critical two-tail	2.024394147	

t-Test: Two-Sample Assuming Equal Variances
Whaley vs None

	Variable 1	Variable 2
Mean	3.37561	2.471715
Variance	0.000160981	6.48824E-05
Observations	20	20
Pooled Variance	0.000112932	
Hypothesized Mean Difference	0	
df	38	
t Stat	268.9738794	
P(T<=t) one-tail	3.09629E-64	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	6.19257E-64	
t Critical two-tail	2.024394147	

t-Test: Two-Sample Assuming Equal Variances
Keithley vs None

	Variable 1	Variable 2
Mean	3.49585	2.471715
Variance	0.000160262	6.48824E-05
Observations	20	20
Pooled Variance	0.000112572	
Hypothesized Mean Difference	0	
df	38	
t Stat	305.2404203	
P(T<=t) one-tail	2.5373E-66	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	5.0746E-66	
t Critical two-tail	2.024394147	

Average Location Quality Levels

t-Test: Two-Sample Assuming Equal Variances
Whaley vs Keithley

	Variable 1	Variable 2
Mean	2.93646	3.17209
Variance	0.000142913	0.000132755
Observations	20	20
Pooled Variance	0.000137834	
Hypothesized Mean Difference	0	
df	38	
t Stat	-63.4676842	
P(T<=t) one-tail	1.78446E-40	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	3.56893E-40	
t Critical two-tail	2.024394147	

t-Test: Two-Sample Assuming Equal Variances
Whaley vs None

	Variable 1	Variable 2
Mean	2.93646	2.14039
Variance	0.000142913	4.55609E-05
Observations	20	20
Pooled Variance	9.4237E-05	
Hypothesized Mean Difference	0	
df	38	
t Stat	259.3226982	
P(T<=t) one-tail	1.24041E-63	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	2.48082E-63	
t Critical two-tail	2.024394147	

t-Test: Two-Sample Assuming Equal Variances
Keithley vs None

	Variable 1	Variable 2
Mean	3.17209	2.14039
Variance	0.000132755	4.55609E-05
Observations	20	20
Pooled Variance	8.91578E-05	
Hypothesized Mean Difference	0	
df	38	
t Stat	345.5204721	
P(T<=t) one-tail	2.28816E-68	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	4.57631E-68	
t Critical two-tail	2.024394147	

Average Track Quality Levels

t-Test: Two-Sample Assuming Equal Variances
Whaley vs Keithley

	Variable 1	Variable 2
Mean	3.740735	3.86865
Variance	0.000115091	0.000115996
Observations	20	20
Pooled Variance	0.000115544	
Hypothesized Mean Difference	0	
df	38	
t Stat	-37.63124545	
P(T<=t) one-tail	5.4974E-32	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	1.09948E-31	
t Critical two-tail	2.024394147	

t-Test: Two-Sample Assuming Equal Variances
Whaley vs None

	Variable 1	Variable 2
Mean	3.740735	2.6823
Variance	0.000115091	3.27495E-05
Observations	20	20
Pooled Variance	7.39201E-05	
Hypothesized Mean Difference	0	
df	38	
t Stat	389.2985519	
P(T<=t) one-tail	2.46199E-70	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	4.92399E-70	
t Critical two-tail	2.024394147	

t-Test: Two-Sample Assuming Equal Variances
Keithley vs None

	Variable 1	Variable 2
Mean	3.86865	2.6823
Variance	0.000115996	3.27495E-05
Observations	20	20
Pooled Variance	7.43729E-05	
Hypothesized Mean Difference	0	
df	38	
t Stat	435.0162616	
P(T<=t) one-tail	3.62431E-72	
t Critical one-tail	1.685954461	
P(T<=t) two-tail	7.24863E-72	
t Critical two-tail	2.024394147	

Appendix B. VBA Code

This appendix contains the VBA code used to implement the logic required in the model that is not possible with standard Arena modules. Most of the VBA blocks in the model call the subroutines near the end of the code. The function at the very end is credited to M.A. (Thijs) van den Berg [1].

```
Option Base 1
Dim oExcelApp As Excel.Application
Dim oWorkbook As Excel.Workbook
Dim oWorksheet As Excel.Worksheet
Dim nextrow As Integer
Dim nextcol As Integer
Dim tempkm() As Single
Dim CurSerNum As Single
Dim index As Integer
Dim tempfusedkm(6, 6) As Single


'Private Sub ModelLogic_RunBeginReplication()
'
'nextrow = 1
'nextcol = nextcol + 1
'oExcelApp.Visible = True
'End Sub
'
'Private Sub ModelLogic_RunBeginSimulation()
'Set oExcelApp = CreateObject("Excel.Application")
'oExcelApp.SheetsInNewWorkbook = 1
'Set oWorkbook = oExcelApp.Workbooks.Add
'Set oWorksheet = oWorkbook.Activesheet
'nextcol = 0
'End Sub
'
'Private Sub ModelLogic_RunEndSimulation()
"oExcelApp.Visible = True
'CurSerNum = 0
'End Sub


Private Sub VBA_Block_9_Fire()
GenerateKM
End Sub

Private Sub VBA_Block_10_Fire()
GenerateKM
End Sub

Private Sub VBA_Block_11_Fire()
GenerateKM
End Sub
```

```vba
Private Sub VBA_Block_12_Fire()

Dim m As Model
Dim s As SIMAN
Set m = ThisDocument.Model
Set s = m.SIMAN
Dim j As Integer

Dim NumEntities As Integer
Dim KM As Single
Dim SerNum As Single
Dim txtkm As String

NumEntities = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("NumSources"))
If NumEntities > 1 Then
   SerNum = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("SerNum"))
   If SerNum <> CurSerNum Then
     index = 1
     CurSerNum = SerNum
     For col = 1 To 6
     For QL = 1 To 6
       txtkm = "km" & col & QL
       tempfusedkm(col, QL) = s.EntityAttribute(s.ActiveEntity,
s.SymbolNumber(txtkm))
     Next
     Next
   Else
     index = index + 1
     For col = 1 To 6
     For QL = 1 To 6
       txtkm = "km" & col & QL
       KM = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber(txtkm))
       If tempfusedkm(col, QL) < KM Then tempfusedkm(col, QL) = KM
       If index = NumEntities Then
          s.EntityAttribute(s.ActiveEntity, s.SymbolNumber(txtkm)) =  \&
          tempfusedkm(col,QL)
       End If
     Next
     Next
   End If
End If

End Sub
```

```vba
Private Sub VBA_Block_13_Fire()

Dim m As Model
Dim s As SIMAN
Set m = ThisDocument.Model
Set s = m.SIMAN
Dim j As Integer

Dim NumEntities As Integer
Dim KM As Single
Dim SerNum As Single
Dim product As Single
Dim fusedkm As Single
Dim txtkm As String

NumEntities = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("NumSources"))
If NumEntities > 1 Then
    SerNum = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("SerNum"))
    If SerNum <> CurSerNum Then
        index = 1
        CurSerNum = SerNum
        ReDim tempkm(6, 6, NumEntities)
        For col = 1 To 6
        For QL = 1 To 6
            txtkm = "km" & col & QL
            KM = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber(txtkm))
            tempkm(col, QL, index) = KM
        Next
        Next
    Else
        index = index + 1
        For col = 1 To 6
        For QL = 1 To 6
            txtkm = "km" & col & QL
            KM = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber(txtkm))
            tempkm(col, QL, index) = KM
        Next
        Next
        If index = NumEntities Then
            For col = 1 To 6
            For QL = 1 To 6
                product = 1
                For j = 1 To index
                    product = product * (1 - tempkm(col, QL, j))
                Next
                fusedkm = 1 - product
```

```vba
        txtkm = "km" & col & QL
        s.EntityAttribute(s.ActiveEntity, s.SymbolNumber(txtkm)) = fusedkm
      Next
      Next
    End If
  End If
End If
End Sub

Private Sub VBA_Block_14_Fire()
DeleteKM
End Sub

Private Sub VBA_Block_15_Fire()
DeleteKM
End Sub

Private Sub VBA_Block_16_Fire()
DeleteKM
End Sub

Private Sub VBA_Block_17_Fire()
RouteEntities
End Sub

Private Sub VBA_Block_18_Fire()
RouteEntities
End Sub

Private Sub VBA_Block_19_Fire()
RouteEntities
End Sub

Private Sub VBA_Block_20_Fire()
RouteEntities
End Sub

Private Sub VBA_Block_21_Fire()
RouteEntities
End Sub

Private Sub VBA_Block_22_Fire()
RouteEntities
End Sub
```

```
Private Sub VBA_Block_23_Fire()
GenerateKM
End Sub


Private Sub VBA_Block_24_Fire()

Dim m As Model
Dim s As SIMAN
Set m = ThisDocument.Model
Set s = m.SIMAN

For col = 1 To 6
   txtQR = "QualR" & col
   QR = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber(txtQR))
   For QL = 1 To 6
      txtkm = "km" & col & QL
      testkm = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber(txtkm))
      If testkm > QR Then QA = QL - 1
   Next
   txtQA = "QualA" & col
   s.EntityAttribute(s.ActiveEntity, s.SymbolNumber(txtQA)) = QA
Next
End Sub



Public Sub GenerateKM()

Dim m As Model
Dim s As SIMAN
Set m = ThisDocument.Model
Set s = m.SIMAN

Dim Z(6, 6) As Double
Dim KM(6, 6) As Double
LocArray = Array(10000, 1000, 100, 20, 10, 5)
Dim txtkm As String
Dim Area(6) As Double

' Location
Area(1) = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("L90"))
If Area(1) = 0 Then
   For QL = 1 To 6
      KM(1, QL) = 0
   Next
Else
   LSigma = Area(1) / 1.65
```

B-6

```
      For QL = 1 To 6
         Z(1, QL) = LocArray(QL) / LSigma
         KM(1, QL) = Round(2 * NormProb(Z(1, QL)) - 1, 2)
         If KM(1, QL) = 1 Then KM(1, QL) = 0.99
      Next
   End If

   ' Other areas
   Area(2) = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("T90"))
   Area(3) = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("ID90"))
   Area(4) = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("A90"))
   Area(5) = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("C90"))
   Area(6) = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("I90"))

   For col = 2 To 6
      If Area(col) = 0 Then Area(col) = 0.1
      For QL = 6 To 2 Step -1
         Z(col, QL) = 3 - (1.35 / Area(col)) * (QL - 1)
         If Z(col, QL) < 0 Then Z(col, QL) = 0
         KM(col, QL) = Round(2 * NormProb(Z(col, QL)) - 1, 2)
      Next
      If KM(col, 2) = 0 Then KM(col, 1) = 0 Else KM(col, 1) = 0.99
   Next
   For col = 1 To 6
   For QL = 1 To 6
      txtkm = "km" & col & QL
      s.EntityAttribute(s.ActiveEntity, s.SymbolNumber(txtkm)) = KM(col, QL)
   Next
   Next

   End Sub


   Public Sub RouteEntities()

   ' Determine NextStation after leaving Planning

   Dim m As Model
   Dim s As SIMAN
   Set m = ThisDocument.Model
   Set s = m.SIMAN

   'variable to hold next station
   Dim Next_Station As Double
```

```vb
'Dim variables to hold station variables locally for easier access
Dim Stn_Planning As Double
Dim Stn_Library As Double
Dim Stn_Collection As Double
Dim Stn_Processing As Double
Dim Stn_Exploitation As Double
Dim Stn_Analysis As Double
Dim Stn_Production As Double
Dim Stn_Dissemination As Double
Dim Stn_Integration As Double
Dim Stn_Evaluation As Double

'Dim variables to get entity processing steps
Dim Need_Collect As Double
Dim Need_Process  As Double
Dim Need_Exploit As Double
Dim Need_Analyze As Double
Dim Need_Produce As Double
Dim Need_Disseminate As Double
Dim Need_Integrate As Double

'Assign local variables values of stations
Stn_Planning = s.VariableArrayValue(s.SymbolNumber("PlanningStation"))
Stn_Library = s.VariableArrayValue(s.SymbolNumber("LibraryStation"))
Stn_Collection = s.VariableArrayValue(s.SymbolNumber("CollectionStation"))
Stn_Processing = s.VariableArrayValue(s.SymbolNumber("ProcessingStation"))
Stn_Exploitation = s.VariableArrayValue(s.SymbolNumber("ExploitationStation"))
Stn_Analysis = s.VariableArrayValue(s.SymbolNumber("AnalysisStation"))
Stn_Production = s.VariableArrayValue(s.SymbolNumber("ProductionStation"))
Stn_Dissemination = s.VariableArrayValue(s.SymbolNumber("DisseminationStation"))
Stn_Integration = s.VariableArrayValue(s.SymbolNumber("IntegrationStation"))
Stn_Evaluation = s.VariableArrayValue(s.SymbolNumber("EvaluationStation"))

'Assign local variables values of stations required for processing
Need_Collect = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("Collect"))
Need_Process = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("Process"))
Need_Exploit = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("Exploit"))
Need_Analyze = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("Analyze"))
Need_Produce = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("Produce"))
Need_Disseminate = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("Disseminate"))
Need_Integrate = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("Integrate"))
```

```
'Determine where to go next
If Need_Collect = 1 Then
    Next_Station = Stn_Collection
ElseIf Need_Process = 1 Then
    Next_Station = Stn_Processing
ElseIf Need_Exploit = 1 Then
    Next_Station = Stn_Exploitation
ElseIf Need_Analyze = 1 Then
    Next_Station = Stn_Analysis
ElseIf Need_Produce = 1 Then
    Next_Station = Stn_Production
ElseIf Need_Disseminate = 1 Then
    Next_Station = Stn_Dissemination
ElseIf Need_Integrate = 1 Then
    Next_Station = Stn_Integration
Else
    Next_Station = Stn_Evaluation
End If

'Assign Entity.NextStation to Next_Station
s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("NextStation")) = Next_Station

End Sub


Public Sub DeleteKM()

Dim m As Model
Dim s As SIMAN
Set m = ThisDocument.Model
Set s = m.SIMAN

For QL = 1 To 6
For col = 1 To 6

    txtkm = "km" & QL & col
    s.EntityAttribute(s.ActiveEntity, s.SymbolNumber(txtkm)) = "0"

Next
Next

End Sub
```

```
Public Function NormProb(x As Double) As Double

    Dim t As Double
    Const b1 = 0.31938153
    Const b2 = -0.356563782
    Const b3 = 1.781477937
    Const b4 = -1.821255978
    Const b5 = 1.330274429
    Const p = 0.2316419
    Const c = 0.39894228

    If x >= 0 Then
        t = 1# / (1# + p * x)
        NormProb = (1# - c * Exp(-x * x / 2#) * t * _
        (t * (t * (t * (t * b5 + b4) + b3) + b2) + b1))
    Else
        t = 1# / (1# - p * x)
        NormProb = (c * Exp(-x * x / 2#) * t * _
        (t * (t * (t * (t * b5 + b4) + b3) + b2) + b1))
    End If
End Function
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
04–03–2008	Master's Thesis	Jul 2007 – Mar 2008

4. TITLE AND SUBTITLE

SIMULATION OF NATIONAL INTELLIGENCE PROCESS WITH FUSION

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Lupa Jr, Joseph Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

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

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/08-13

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

National Security Space Organization (NSSO)
Attn: Mr. James Kindle
P.O. Box 222310
Chantilly, VA 20153-2310
James.Kindle.CTR@osd.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

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

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This work is a follow on effort of two previous Master's theses. The first was *Modeling and Simulation of the Military Intelligence Process* by Captain Carl Pawling in 2004. The other was *A Knowledge Matrix Modeling of the Intelligence Cycle* by Captain Kevin Whaley in 2005. Both of these were done to facilitate the study and analysis of the intelligence process for the National Security Space Organization (NSSO). Here, modifications are made the Pawling model to include tasking multiple intelligence sources for data collection to fulfill Requests for Information (RFIs) and fusing the collected data into one new piece of intelligence. One fusion method is the one suggested by Whaley, which simply takes the best intelligence collected, while the other method captures the synergy of intelligence fusion. Both methods are compared to each other and to the baseline model where no fusion takes place.

15. SUBJECT TERMS

Simulation, Modeling, Intelligence Process, Information Fusion, Knowledge Matrix

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. J.O. Miller, (ENS)
U	U	U	UU	87	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-6565, ext 4326; e-mail: John.Miller@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18