MODELING AND SIMULATION OF THE MILITARY INTELLIGENCE PROCESS

THESIS

Carl R. Pawling, Captain, USAF

AFIT/GOR/ENS/04-09

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

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

AFIT/GOR/ENS/04-09

MODELING AND SIMULATION OF THE MILITARY INTELLIGENCE PROCESS

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

Carl R. Pawling, B.S.

Captain, USAF

March 2004

AFIT/GOR/ENS/04-09

MODELING AND SIMULATION OF THE MILITARY INTELLIGENCE PROCESS

Carl R. Pawling, B.S.

Captain, USAF

Approved:

\\signed\\	4 MAR 2004
Dr. John O. Miller Thesis Advisor	Date
\\signed\\	4 MAR 2004
Maj. Stephen P. Chambal, Ph.D. Committee Member	Date

AFIT/GOR/ENS/04-09

Abstract

There is concern within U.S space and intelligence organizations that the current Tasking, Processing, Exploitation, and Dissemination processes may be insufficient to support current and future Intelligence, Surveillance, and Reconnaissance systems. As part of a larger intelligence process, more detailed analysis becomes critical to determine what portions need to be improved. This analysis can be accomplished by simulation, which is appropriate due to the complexity of the process and the ability to compare variations in the process. We construct a high level model of a generalized military intelligence process based in part on the Intelligence Cycle outlined in Joint Publications. Using the Arena® process oriented simulation software, our modular simulation can be used for quick turn studies on changes to the process, specifically with respect to classical measures such as quality, quantity, and timeliness. A sample study using the basic framework of the intelligence process with statistical analysis is also conducted.

Acknowledgements

I greatly appreciate the guidance and support of my advisor, Dr. J. O. Miller. I also thank my reader Maj Stephen Chambal for his enthusiasm and efforts to assist whenever possible. Additional thanks go to my sponsors at the National Security Space Architect for their support in model development and testing. Their expertise and active involvement helped to ensure that this research remained relevant to the problem at hand. Special thanks go to my classmates, friends, and faculty who helped me to stay focused on the problem at hand. Above all, I thank my Lord and Savior Jesus Christ for perseverence and strength to continue.

Carl R. Pawling

Table of Contents

List of Figures

List of Tables

MODELING AND SIMULATION
OF THE MILITARY INTELLIGENCE PROCESS

1. Introduction

1.1 Background

The importance of assessing the information flow of the military intelligence process has been brought to light in recent years. Although, the need for this assessment is not new, it becomes more critical as both capability and demand increases. According to the National Security Strategy, "We must transform our intelligence capabilities and build new ones to keep pace with" terrorist and other threats [White House, 2002:30]. Military use of the intelligence process is vital both in and of itself and as a part of information superiority which is an enabler of military power [JP 1, 2000:IV-8,9,10]. The intelligence process begins when a need for information or intelligence is identified and encompasses how these information needs are met. As such, it includes all of the satellites, aircraft, communications, and other systems used to gather and transmit data as well as the people, organizations, and resources involved in turning raw data into useful information. The intelligence process can take on many forms, two of which are Task, Process, Exploit, Disseminate (TPED) and Task, Process, Post, Use (TPPU).

To illustrate the TPED and TPPU processes, take for example a person with a standard 35mm camera, lets call him Bob. The need for information about a particular place has arisen. Once this need is realized and it is determined that a picture can satisfy the need, Bob is directed or *tasked* to go take a picture of this place. Once Bob has taken the picture and returned, the picture is still of no use. A series of steps must be take to put the picture into a usable form, i.e, the

picture must be developed or *processed*. Now that the image is in a usable form, the remainder of the intelligence process can be carried out in several ways. For the TPED approach, an analyst would take additional steps to *exploit* the picture, e.g., mark on it to indicate important aspects and add notes to describe what those aspects are. Once this is complete, the picture with the markings and additional information would be sent or *disseminated* to the person or organization that needed the picture. Alternatively, the TPPU approach, would bypass the exploitation up front and send or *post* the picture to a web page. Then the person or organization that needed the picture could retrieve it and *use* it without the overhead of exploitation. Although this is a simplified example, it illustrates how TPED and TPPU can differ. In reality, the intelligence process is more complex and is dependent on several systems, organizations, and user requirements.

1.2 Problem Statement

The National Security Space Architect (NSSA) has indicated that the TPED and/or TPPU processes may be insufficient to support current and future ISR systems. If this is the case, more detailed analysis of the intelligence process becomes critical to determine what portions need to be improved. This analysis could be accomplished several ways such as detailed statistical analysis, an analytical queueing network, or simulation. Simulation is an appropriate tool due to the complexity of the process and the ability to compare multiple ways of implementing the process. In order to use simulation, an appropriate end-to-end model of the process is required, and NSSA has indicated that such a model for comparing various forms of the intelligence process does not exist. NSSA has identified the need for an appropriate end-to-end model of the intelligence process to provide quick turn analysis of the impacts of changes to the systems on various mission areas such as Intelligence, Surveillance, and Reconnaissance (ISR)[NSSA, 2003].

1.3 Research Objective

The objective is to develop an Arena® model of a generalized national or military intelligence process. The model needs to be detailed enough to examine the various interactions and resources and their impact on fulfilling various missions, yet flexible enough to allow simulation of TPED, TPPU, or other hybrid implementations of the intelligence process. Moreover, the implementation of the model in Arena should allow quick turn analysis. Part of this is the development of appropriate measures of effectiveness and performance. One important measure is information needs satisfaction. Information needs satisfaction encompasses other classical intelligence measures such as quality, quantity, and timeliness. It should also encompass the correct generation of intelligence requirements that are passed through the remainder of the intelligence process which is beyond the scope of this model.

1.4 Research Focus

This study will focus on providing a basic framework for national or military intelligence process analysis. Determining the appropriate level of detail for modeling various portions of the intelligence process is critical. Rather than focus on minute details in all areas, the focus will be on a top level model with additional detail where needed. Understanding previous models and simulations of the intelligence process will provide a background for model development and abstraction of the real world process. Comparing prior work with the intended application of the framework will reveal areas where improvement and new development is needed. This focus should enable development of the flexibility to model multiple implementations of the intelligence process and allow quick turn analysis to be completed.

1.5 Methodology

In developing a model, it is necessary to ensure that the model is structured for specific measures of interest. Those measures should provide the required information for the intended uses of the model. Four measures of interest for analyzing the intelligence process are quality, quantity, timeliness, and information needs satisfaction (QQTI). The first three measures, quality, quantity, and timeliness (QQT), have historically been used when assessing various aspects of the intelligence process. As such previous models may be designed to provide those quantitative measures for a specific implementation of the intelligence process. Where applicable some of the concepts in these models may be used in model development. More importantly, examination of open source and unclassified descriptions of the intelligence process will be the primary source for model development. In addition to basic descriptions, factors that influence the process and consequently the QQTI measures will be taken into account based on subject matter expert (SME) discussions. The use of Arena to model the military intelligence process in a modularized design should allow the fidelity of individual portions of the model to be easily increased and allow simulation studies to be easily accomplished. Further detail will be given in Chapter 3.

1.6 Assumptions and Limitations

The nature of developing a model dictates that some assumptions be built in from the beginning. One of those assumptions is the level of abstraction from the real world. Based on the intended uses of the model, a moderate level of detail should be sufficient for this study. That is, comparing the impact of changes to various portions of the intelligence process for quick look studies does not require the detail of engineering level models. Furthermore, the moderate level of detail extends to all portions of the intelligence process model including an embedded communications model. By maintaining the focus of building a top level framework for simulation

studies, only open source and unclassified information will be required to develop the structure of the model. Additional hypothetical data can be used to populate the model for testing the implementation in Arena, but a well defined scenario and review by SMEs will be needed for verification and validation (V&V). Since the model is only an abstraction of the real world, it can be used to assess the impact of altering a portion of the process and provide insight into how that change might affect the process in real life without actually altering the process.

1.7 Preview

This thesis contains five chapters. This chapter, Introduction, contains background information and development of the research focus. The second chapter, Literature Review, discusses the intelligence process in more detail and examines previous work in modeling and assessing the intelligence process. The third chapter, Methodology, discusses the Arena model development of a generalized intelligence process. The fourth chapter, Results and Analysis, presents the results of simulation runs and statistical analysis of the simulation output. The fifth chapter, Conclusions, presents insights and conclusions based on the research and recommendations for further study.

2. Literature Review

2.1 Introduction

In order to simulate the military intelligence process, a clear understanding of the process is needed. Examining published descriptions in addition to prior models and simulations of the intelligence process will provide information needed to gain that understanding. This chapter will examine the intelligence process as described in Joint Pub 2-0, *Doctrine for Intelligence Support to Joint Operations* [JP 2-0, 2000] and how Tasking, Processing, Exploitation & Dissemination (TPED) and Tasking, Posting, Processing and Using (TPPU) fit into that process. The understanding gained from this examination will help to ensure that a model of the process accurately represents the documented process. Furthermore, this chapter will explore other models and simulations with various properties and levels of abstraction that have been used to assess portions of the intelligence process. Examination of such prior work allows greater understanding of how others have abstracted the real process for their purposes. However, documentation of the process and prior work will not be sufficient in themselves and discussions with Subject Matter Experts (SME) will also be required for model development. The measures of interest for assessing the process are also key for developing a model and as such will be described. Finally, this chapter will contain a detailed summary of how each of the items tie together for the development of a new simulation model.

2.2 Basic Process Description

2.2.1 The Intelligence Cycle

The intelligence process is described in Joint Publication 2-0, *Doctrine for Intelligence Support to Joint Operations*, as the intelligence cycle. The intelligence cycle has six phases, Planning and Direction, Collection, Processing and Exploitation,

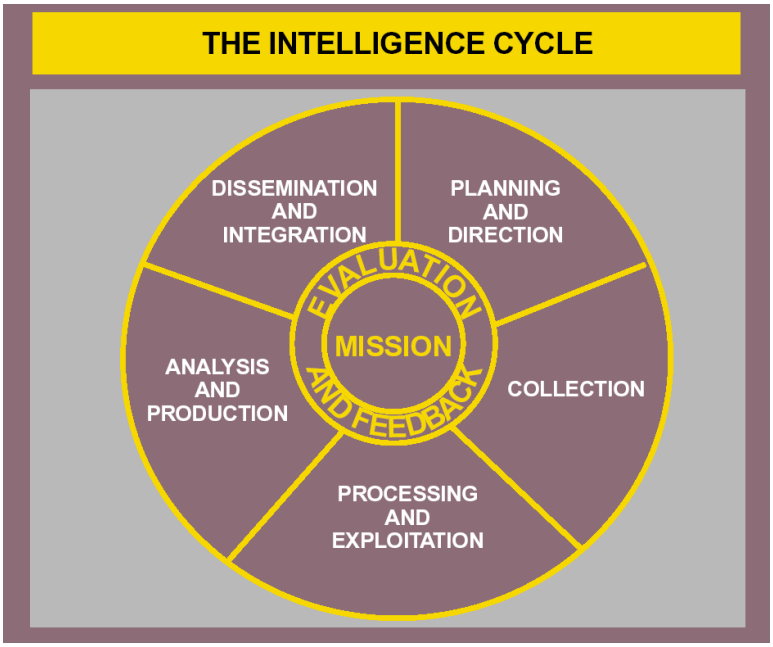

Figure 2.1 The Intelligence Cycle[JP 2-0, 2000:II-1]

Analysis and Production, Dissemination and Integration, and Mission Evaluation and Feedback (see Figure 2.1) [JP 2-0, 2000:II-1].

The first phase, Planning and Direction, involves planning for future contingencies in theaters and determining what resources might be required for those contingencies. A small number of Priority Information Requests (PIRs) are created for critical information needs and are ranked in order of importance to the commander [JP 2-0, 2000:II-3]. Intelligence staff examine the PIRs of the commander and other units to determine specific information that can be used to answer the PIRs, resolve resource conflicts, and remove redundant requests [JP 2-0, 2000:II-4]. Existing information and previously scheduled information gathering is used when possible, but the remaining information requirements are turned into requests for information (RFIs) which can lead to either production or collection requirements [JP 2-0, 2000:II-4]. From those collection requirements, a collection plan is developed to prioritize the requirements and direct them to to the organizations best suited to meet the requirements [JP 2-0, 2000:II-5].

The second phase, Collection, carries out the collection plan. Simply put, collection is the gathering of information. In this phase organizations or agencies which operate collection assets such as satellites or surveillance equipment would task those assets to gather information at specified times and places. The means and methods of collection are highly dependant on the source of the information which are generally categorized into various intelligence disciplines. Six overarching disciplines are Imagery Intelligence (IMINT), Human Intelligence (HUMINT), Signals Intelligence (SIGINT), Measurement and Signature Intelligence (MASINT), Open-Source Intelligence (OSINT), Technical Intelligence (TECHINT), and Counterintelligence (CI) [JP 2-0, 2000:II-3]. These disciplines can be broken down further based on the methods and technologies used for collection. Each source of information generates different kinds of data which, when collected, often cannot be immediately used and must therefore go through some processing.

The third phase, Processing and Exploitation, takes the raw data gathered during the collection phase and transforms it so that it can be used for analysis and production [JP 2-0, 2000:II-7]. The amount and type of work involved depends on the type of intelligence that has been gathered. It could involve image storage and conversion, film development, document translation, or report generation [JP 2-01, 1996:III-26-29]. In some cases, the differences between processing and exploitation are clear. For example imagery processing might be conversion of data from satellites into images usable by a person while imagery exploitation would involve marking up the image to indicate important features or related information.

The fourth phase, Analysis and Production, uses processed and/or exploited information to generate intelligence products to meet PIRs and RFIs [JP 2-0, 2000:II-8]. The intelligence products are usually categorized by their primary use: indications and warning, current intelligence, general military intelligence, target intelligence, scientific and technical intelligence, and counterintelligence [JP 2-0, 2000:II-

10-12]. These categories may overlap and are not confined to any particular source of intelligence.

The fifth phase, Dissemination and Integration, involves sending intelligence products to the user and the user using those intelligence products[JP 2-0, 2000:II-12]. This phase of the intelligence cycle is heavily dependent on communications systems due to the likelihood of significant geographic separation between an intelligence production center and the user. The dissemination of information can be either "pushed" to the used to answer a request or "pulled" by the user from databases and other centralized sources of information[JP 2-01, 1996:III-41]. The "pull" method has the potential to save time due to the involvement of fewer people and organizations, but it is dependent on the required information being available. For the information to be available, it must have been previously collected and an appropriate amount of processing been accomplished. Additionally, the user that is pulling the information must have access to the databases or repositories that contain the information. Once the user has the intelligence product, integration or use of the information may still require some intelligence resources if it is incomplete or needs additional explanation.

The last phase, Mission Evaluation and Feedback, is integral to all of the other phases and is appropriately placed in the center of Figure 2.1. It is not conducted independently but must be accomplished throughout each phase to ensure that the process is working as expected. Qualitative attributes that are used to evaluate the quality of intelligence are timeliness, accuracy, usability, completeness, relevance, objectiveness, and availability [JP 2-0, 2000:II-14]. Of these, timeliness can be easily quantified by comparing the time of delivery with the time of need. Timely intelligence is delivered to the user before the time that it is needed. An example of calculating timeliness focused on collection is given in Figure 2.2. The other attributes may be difficult to quantify, but they are nonetheless vital when evaluating

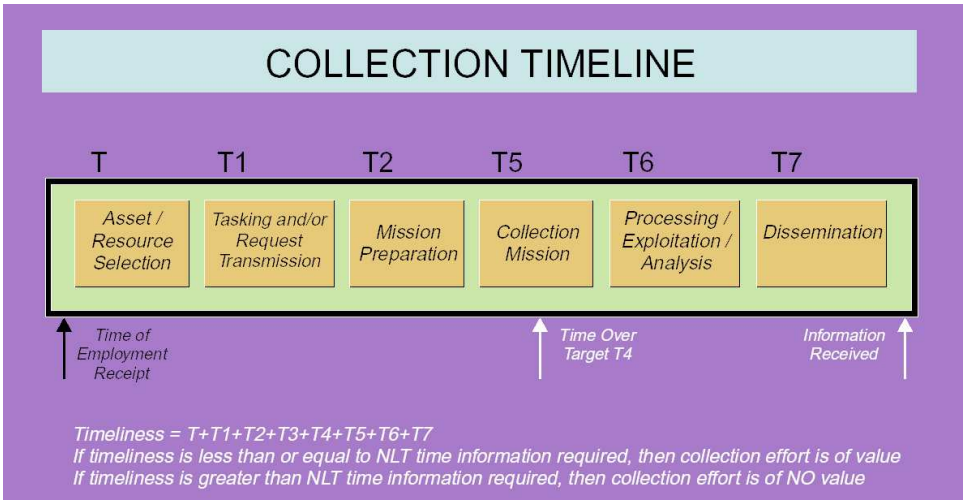

Figure 2.2 Example of how to determine timeliness [JP 2-01, 1996:III-19].

the overall quality of intelligence. When any of these attributes are lacking, they indicate problems within the intelligence cycle.

2.2.2 Variation of Process Implementation (TPED vs. TPPU)

The intelligence cycle described in Section 2.2.1 provides a good general description of the intelligence process, but does not capture much of the variation that occurs when the intelligence process is implemented. Two primary implementations of the intelligence process are Task, Process, Exploit, Disseminate (TPED) and Task, Process, Post, Use (TPPU). Although not the only possibilities, they are two current philosophies of how to accomplish the intelligence process.

The first two parts of both TPED and TPPU are tasking and processing. This should be the case for any implementation of the intelligence process. Tasking of collection assets is a necessary component since information must be collected in order to use it. Furthermore, processing cannot be omitted since the raw data that is collected must be put into a usable format. The remainder of the process varies in how much additional effort is put into adding relevant information to the collected data, when the data is sent to the user, and how the user acquires the

information. The TPED process essentially follows the intelligence cycle from phase to phase beginning with the tasking of collection resources. The important aspects of the TPED process are that processing and exploitation are completed to generate an intelligence product that is then disseminated to the user. The TPPU process differs in that data is posted to centralized libraries or databases at several places throughout the intelligence process. Data can be immediately posted after collection, processing, and/or exploitation. One benefit of this method is that the user can then access the raw or processed data much sooner than waiting for a finished intelligence product.

2.3 Assessment Measures

One of the critical pieces of information that is needed when developing a model is the question to be answered or the problem to be explored. In order to answer the question, one must have a means of collecting data. This is where specific measures come into play. Measures that are implemented in a simulation model should be quantitative in nature and provide sufficient information to answer the question or gain insight into the problem. When assessing the intelligence process, three measures are commonly used: quality, quantity, and timeliness (QQT) of information. Another measure, information needs satisfaction, is less common but as an aggregate measure allows more insight into the overall process. Each of these measures and some of their uses are outlined in the following paragraphs.

Quality of information is often a qualitative measure, but in some cases can be quantitative. A quantitative example is image resolution. However, the image resolution alone does not capture all aspects of quality, e.g., an image of something that is obscured by clouds has low quality even if it has high resolution. Furthermore, other sources of information do not necessarily have such objectively quantifiable attributes. Some of the attributes of quality intelligence are timeliness, accuracy, usability, completeness, relevance, objectiveness, and availability [JP 2-0, 2000:II-14].

However, timeliness is often separated from quality as a separate measure. Specific types of information may also have other attributes that contribute to overall quality. In general, quality of information is important, since wrong or bad information can result in unacceptable loss of life, especially in times of conflict. Qualitative attributes can be put into categories such as "excellent," "good," or "poor" which have a clear order. Accordingly, quality can also be represented with ordinal numbers. For example, one could rate the quality of information on a scale from one to five, with one being the highest quality. More detailed methods of rating quality could ordinally rank each of several attributes and aggregate those ratings into an overall quality rating. Another more detailed approach to examining quality is using a Knowledge Matrix.

Quantity of information is usually a quantitative measure. Just as the number of products produced by a factory is important, so is the number of information requests fulfilled through the intelligence process. The quantity of information that a collection platform, intelligence analyst, or communication system can handle is vital in determining the number of information requests that the overall system can process. The number of requests that can be answered is as important as the quality of information. Unanswered requests may leave a commander and troops in a vulnerable situation. In general, a large amount of information that is poor is of little value and may actually be worse than a small amount of good information.

Timeliness of responses to information requests is also a quantitative measure. As previously discussed in Section 2.2.1, timeliness is used to determine if requests are met on time. As a quantitative measure, it can be tracked as a total time from request as in Figure 2.2 or as a difference between the time required and the time completed. Late responses may be as detrimental as unanswered requests. The later the response, the less useful it is likely to be.

Although quality, quantity, and timeliness of information are important to asses the various aspects of the intelligence process, a single aggregated measure can

sometimes be more useful. NSSA has indicated that information needs satisfaction (INS) is one such measure [NSSA Meeting, 2003]. Although it can be viewed as the process of meeting information needs, as a measure it can describe either the proportion of needs that were met or the degree to which needs were met. When specifically related to QQT, INS would simply be the proportion of intelligence requests that meet both of the quality and timeliness requirements. Alternatively, INS could be rated on a scale with items that meet both of the quality and timeliness requirements at the top and decreasing scores based on how poor or how late the response to a request was received. This would allow percentages based on various levels of satisfaction.

2.4 Previous Work

Many models and simulations have been used to assess various parts of the intelligence process in the past. In each case, the designers of the model selected a level of detail as well as one or more implementations or architectures of the intelligence process such as TPED or TPPU. Models have ranged from low detail and one architecture to high detail and many architectures. Each position in the level of detail versus number of architectures space provides important insight into different problems. Examining models from each of the four corners of this space will provide vital insight into how a model should be developed for the problem at hand. The first corner explored is that containing models that are highly detailed and model only a single architecture. The second corner explored is that containing models that have a low amount of detail and model only a single architecture. The Intelligence, Surveillance, Reconnaissance - Tasking, Processing, Exploitation, Dissemination (ISR-TPED) model is an example of this and is described in Section 2.4.2. The third corner explored is that containing models that are highly detailed and model multiple architectures. The Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) Space and Missile

Table 2.1 Prior models in the level of detail versus number of architecture space.

	Amount of Detail	
# Arch	Low	High
One	ISR-TPED	ISR Platform Models
Several	QUICM	COSMOS

Operations Simulator (COSMOS) formerly known as Intelligence, Surveillance and Reconnaissance Simulator (ISRSIM) is an example of this and is described in Section 2.4.3. The last corner is that containing models that have a low amount of detail and model multiple architectures. The Quick ISR Concept of Operations (CONOPS) Modeler (QUICM) is an example of this and is described in Section 2.4.4. In Table 2.1 some of these prior simulations are given along with where they are in the detail vs number of architecture space. The advantages and disadvantages of each corner of the space are discussed with each of the models.

2.4.1 ISR Platform Models

Many low level (i.e, highly detailed) models exist for many specific ISR platforms. These allow detailed analysis of those specific systems and their individual performance but provide little to no detail on overall impact to the Intelligence process.

2.4.2 ISR-TPED

Some higher level aggregate models exist, but are usually designed to examine a specific architecture of the intelligence process. An example of this sort of simulation is ISR-TPED. It provides a way to quickly analyze the TPED architecture, but lacks in its ability to explore other architectures such as TPPU.

ISR-TPED is an analytical simulation which has advantages over a discrete event simulation (DES). The primary advantage is runtime. An analytic simulation significantly reduces the time to examine the system since it is evaluated analytically

and can essentially be done in one pass. However, this advantage carries a significant disadvantage with it, primarily that the type of probability distributions are extremely limited due to the complexity of computation.

ISR-TPED has another significant disadvantage, its rigid structure. The structure strictly follows the Tasking, Processing, Exploitation, Dissemination path. That rigid structure reduces the complexity and allows the analytic solution to be found easier. However, the rigidity brings into question its ability to model the real world.

Another detractor for the current study of ISR-TPED is that it relies on a moderately detailed ground picture for the scenario. The ground picture allows one target per grid-space with associated terrain/clutter type. Sensor performance must be calculated by another simulation called ISR Performance Evaluation Tool (ISR-PET), which is automatically executed if the selected combination is not found in a pre-built database. An additional problem with ISR-PET is that it currently only includes synthetic aperture radar (SAR) performance models. Although it may be possible to approximate other sensors or collection assets with the SAR models, the benefit of the detailed model is lost. Additionally, evaluating the comparative performance of various architectures does not necessarily require detailed sensor models, as long as they can be appropriately approximated.

The benefits of a tool in the same category as ISR-TPED is that one can assess the specified architecture and find areas for more detailed study or process improvement.

2.4.3 COSMOS

At another extreme are models that are very detailed and allow comparisons of multiple architectures. An example of this is the COSMOS simulation environment. COSMOS provides engineering level models for sensors and other elements. This allows for accurate representation of current systems. However, it also makes it difficult to evaluate suggested systems when that level of detail is unknown. COSMOS

allows the intelligence architecture to be modified so that comparative analysis can be done. This sort of detailed analysis can be useful if a specific architecture is known or "highly suspected" to provide an improvement over current operations. However, to assess an architecture that has unknown impact, this sort of detailed model carries with it a large amount of overhead that can unduly increase the cost of model development, significantly increase the required run-time, and produce an inordinate amount of unneeded information. These together make COSMOS an inappropriate tool for analysis in quick turn studies where such detail is unnecessary.

2.4.4 QUICM

The final category of models or simulations are the most appropriate for the desired purpose of comparative analysis of various intelligence process architectures. QUICM provides much of the required functionality. First of all, it does not include detailed sensor models, but does include a variety of information sources. This is useful since it allows for the interaction of several types of information sources throughout the intelligence process. It also avoids the unnecessary (for our purposes) overhead of the detailed scenarios of target locations and terrain models.

An additional benefit of QUICM is that it allows for the comparison of TPED with TPXX type (i.e., TPPU) architectures. Unfortunately, it does not currently allow for hybrid architectures, i.e., all information follows the TPED path or it follows the TPXX path.

2.5 Summary

The six phases of the intelligence cycle (see Figure 2.1) provide a high level model of how intelligence operations are conducted. From the generation of information requirements through dissemination to the user. More importantly, it allows us to compare TPED and TPPU. TPED closely follows the intelligence cycle whereas

TPPU allows the user to retrieve the raw data or processed information from a centralized source prior to the generation of an intelligence product. Once the process has been clearly defined, measuring its performance becomes possible.

Intelligence operations have traditionally used quality, quantity, and timeliness to measure the performance of the intelligence process. Each of these elements are vital, but a high level aggregate measure can be more useful when comparing overall performance. One such measure is information needs satisfaction which can be simply represented as a proportion of needs that have been completely met, e.g., both on time and of sufficient quality. Use of these measures allows for assessment of the actual process and of a model of the process.

Modeling and simulation is commonly used to assess the performance of portions of the intelligence process. In order to model the intelligence process, some level of abstraction has to be selected along with the particular implementation of the process. The models described in Section 2.4 range from high fidelity to low fidelity and range from a single element of the architecture to multiple implementations of the intelligence process. For quick look studies a low fidelity or high level model that describes multiple implementations is ideal. The QUICM model comes closest to the desired product, but is still lacking in that it only models two implementations of the intelligence process. The ability to model hybrid implementations allows for a better representation of reality and analysis of transitional stages between complete TPED and TPPU implementations. For example, such a transitional stage might be altering the process for one type of information or for a specific user and examining the impact of the overall system. Rather than modify one of the existing models, a new model tailored to the intelligence cycle would be preferred.

3. Methodology

3.1 Introduction

The primary purpose of this study is to develop a new simulation model of the intelligence process for use in quick turn studies. The top level focus and the need for relatively short runtime precludes the inclusion of too much detail in the model. As such, a high level description of the process is initially more beneficial to model development than detailed descriptions that require additional work to aggregate. The process description in Section 2.2 provides a basis for a conceptual model. An appropriate simulation environment for model development must also be selected with the intended audience in mind. This chapter outlines the selection of the Arena simulation environment for model implementation. A detailed description of the model as implemented is given in Section 3.3. Since the measures used to assess the process performance are as important as the model itself, the implementation of these measures is given in Section 3.4. Some of the steps taken to validate and verify the model are discussed in Section 3.5. Furthermore, some of the data that may be required for many potential simulation studies is discussed in Section 3.6 followed by a brief description of a sample study setup in Section 3.7.

3.2 Selection of Simulation Environment

When developing a new simulation model, many options are available for a development environment. At a minimum, a simulation programming language is desirable. However, the use of a visual simulation environment designed for process modeling aids in the implementation of a conceptual process model. A visual environment also aids in teaching others how a model works, which is beneficial when the developer of the model may not be the only user of it. In many cases, a visual development environment may not be feasible due to the scale or detail required for

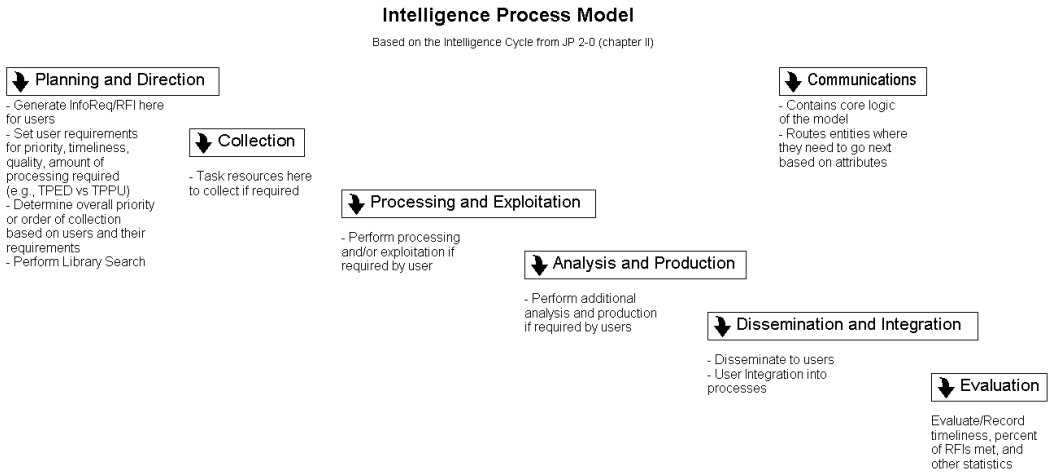

Figure 3.1 Top level of the Intelligence Process Model

the model. However, this study intends to develop only a top level model which makes a visual development environment ideal. Although several process modeling environments exist and would be adequate for the purposes of this study, Arena was selected because it was availaible, appropriate, credible, and widely used.

3.3 Intelligence Process Model (IPM) Description

The conceptual model derived from the process description in Section 2.2 was implemented in Arena from a top-down perspective. The top level of the model shown in Figure 3.1 is composed of seven submodels, six of which are taken directly from the Intelligence Cycle given in Figure 2.1. The last submodel is a Communications submodel which ties all of the other submodels together. This modular approach allows any submodel to be easily modified or replaced and keeps the various portions of the process distinct. Each of the submodels is described in the following sections.

3.3.1 Planning and Direction Submodel

The *Planning and Direction* submodel is modeled as the beginning of the intelligence process and corresponds directly with the process of the same name from the Intelligence Cycle (see Figure 2.1). The purpose of this submodel is to generate user information requirements and prioritize them globally. As an additional part of the planning process, users perform a library search to determine if existing information may meet their needs. The details of the three main portions of this submodel are given below.

Prior to examining the various portions of this submodel, it is important to note that a single entity type, *RFI,* is used throughout the simulation model. These entities are related to real world requests for information, but are not exactly the same. One difference is that *RFI*s in the model are only gathered from a single information source, whereas in the real world information from multiple sources may be required. The primary reason for this difference is that allowing a single *RFI* to have multiple sources of information is beyond the scope of this study. If multiple source capability is needed, it could be added during future study. Additionally, the issue of how various high level questions are decomposed into smaller questions that can be addressed from a single source must be addressed. Another difference is that an *RFI* in the model more closely resembles a tracking sheet that would theoretically follow a real information request through the process. The rationale for choosing this abstraction is that assessment of the intelligence process at a top level does not require actual information, only the status of requests. More specifically, only the knowledge of required information quality and the knowledge of actual information quality received is needed for assessment of meeting quality requirements. As such, each *RFI* is assigned various attributes that hold information about the request requirements and actual information quality that are determined elsewhere in the model. This method also aids in the generation of statistics to evaluate the measures of performance.

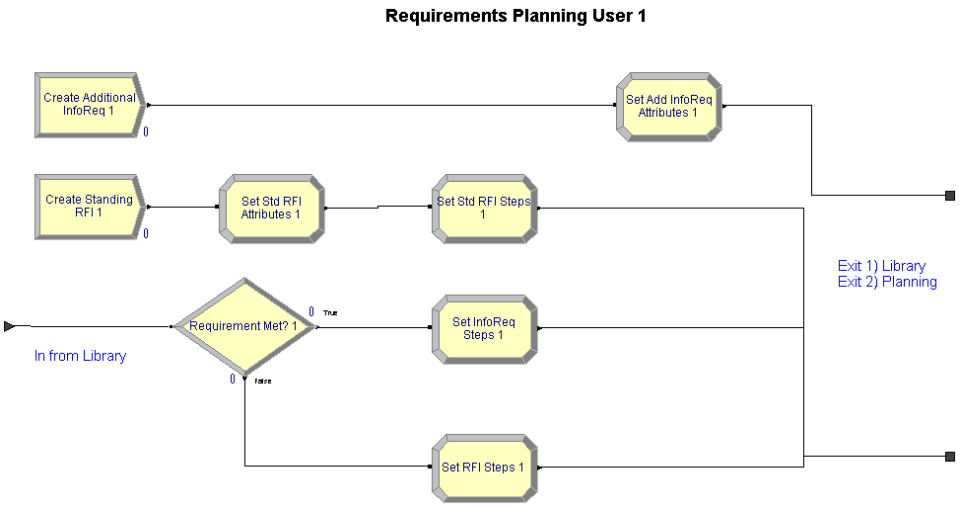

Figure 3.2 User Requirements Planning submodel of the Planning and Direction submodel

3.3.1.1 User Requirements Planning

The *Planning and Direction* submodel contains five user requirements planning submodels. One of these submodels is given in Figure 3.2, but all are identical with the exception of numbering. The *RFI* entities are generated within these submodels. Each user has two *RFI* generators, one for standing requirements that are assumed to require collection without a library search and another for additional requirements that are expected to undergo a library search before a collection determination is made.

Once the entities are created, the attributes of the requirements they represent are assigned to them. Specifically, each is assigned a value for the following attributes: *User*, *Standard*, *InfoSource*, *TimeR*, *QualR*, and *Priority_User*. The first two of these attributes are assigned values used to track where the requirement originated, whereas the last four are assigned values used to describe the requirement itself. A description of the purpose of each of these attributes is given in Table 3.1.

If an *RFI* is from a standing requirement, then it will also be assigned attributes that determine which steps of the intelligence process must be undertaken

Table 3.1 Description of *RFI* attributes that describe requirements.

Attribute	**Description**
User	Indicates which of 5 users generated the requirement
Standard	Indicates if the requirement is standing (1) or additional (0)
InfoSource	Indicates which one of 13 sources are needed to satisfy the requirement
TimeR	Indicates the time from creation that a requirement needs to be filled
QualR	Indicates a required level of information quality from 1 to 5 (5 is best)
Priority_User	Indicates user ranked priority of a requirement from 1 to 5 (1 is highest)

Table 3.2 Description of *RFI* attributes that describe steps needed to fulfill a requirement.

Attribute	**Description**
Collect	Indicates if collection is (1) or is not (0) needed
Process	Indicates if processing is (1) or is not (0) needed
Exploit	Indicates if exploitation is (1) or is not (0) needed
Analyze	Indicates if analysis is (1) or is not (0) needed
Produce	Indicates if production is (1) or is not (0) needed
Disseminate	Indicates if dissemination is (1) or is not (0) needed
Integrate	Indicates if user integration is (1) or is not (0) needed

to fulfill a particular requirement. These attributes, *Collect, Process*, *Exploit*, *Analyze*, *Produce*, *Disseminate*, and *Integrate*, correspond directly to the various portions of the submodels in the IPM (see Table 3.2). Note that there are not attributes that correspond to the Planning, Communications, or Evaluation portions of the model since all entities must travel through these portions of the IPM regardless of other steps of the process that must be completed.

If an *RFI* is from an additional requirement, then it will be sent out of the user planning submodel so that a library search can be performed. When these entities return from a library search, the results of the search are evaluated to determine whether or not the requirement can be met with the information found or if new collection must be accomplished. In order for a requirement to be met, relevant information must be found, the information quality requirement must be met, and the age of the information must not be too old. This binary determination is made by multiplying three corresponding expressions together that evaluate to either 0 or

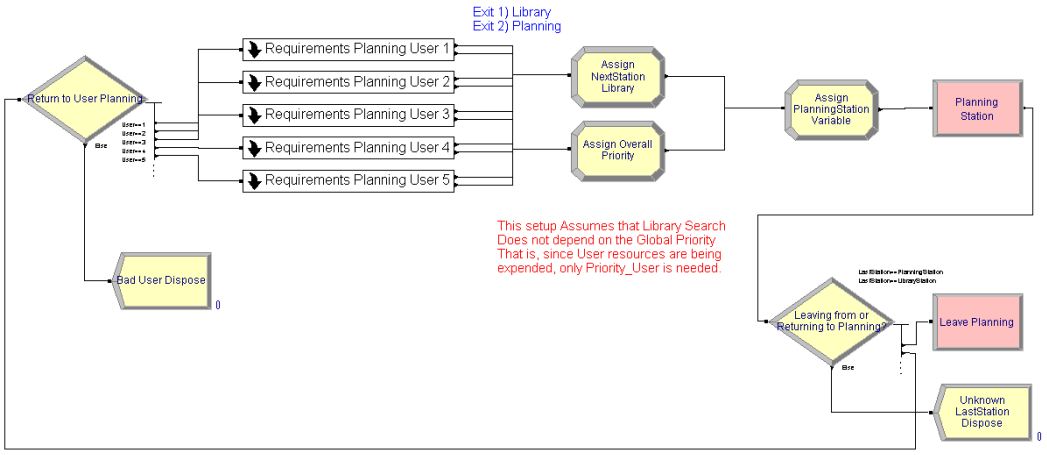

Figure 3.3 Planning portion of the Planning and Direction submodel

1. If all three expressions are met (i.e., equal one), then the *Collect* attribute will be assigned a value of 0. Otherwise, the *Collect* attribute will be assigned a value of 1. The remainder of the attributes that determine what steps of the intelligence process must be accomplished (see Table 3.2) are also set based on the results of the library search, but are left up to the user to determine what proportion of requirements will need each step accomplished.

3.3.1.2 Overall Planning

Once *RFI*s are generated and leave the *Requirements Planning User* submodels, they are tagged to either go to a library search or are assigned a global priority based on how they exited the submodels. For those items that need to have a library search performed, an attribute named *NextStation* is assigned a value that corresponds to the library. The role of the *NextStation* attribute is discussed in more detail in Section 3.3.7. The remainder of the items are assigned a global priority attribute, *Priority_Global*, that is dependent upon the *User* attribute. The method of assigning values to *Priority_Global*, is left up to the simulation user, but in many cases should involve the use of the *Priority_User* attribute. As with the *Priority_User*, the values *Priority_Global* should be such that lower values have a higher priority. In addition

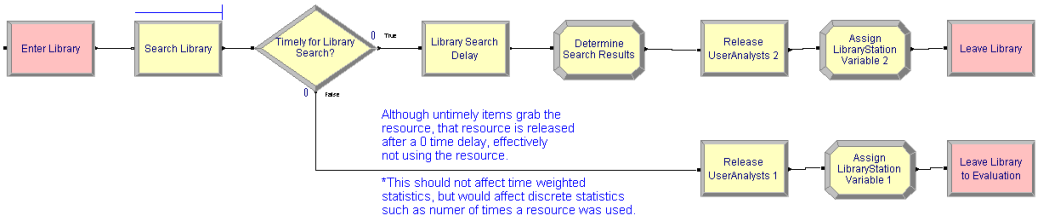

Figure 3.4 Library Search portion of the Planning and Direction submodel

to assigning the overall priority, the *NextStation* attribute is given a value of 0. All entities are also assigned a *LastStation* attribute that corresponds to the the planning portion of the *Planning and Direction* submodel. This portion of the submodel also sends and receives entities from the *Communications* submodel. For those items that arrive from a library search, they are sent back to the user of origin. Two dispose blocks are included in this portion of the model to aid in verification. No entities should ever meet the conditions to be disposed here. In the event that entities are disposed here, it would indicate that the *User* attribute has been assigned an incorrect value or the *LastStation* attribute has a value that the planning portion of the model is not constructed to handle.

3.3.1.3 Library Search

The library search portion of the *Planning and Direction* submodel has the primary purpose of determining the results of users attempting to find existing information that meets the needs of their requirements. When *RFI*s enter the library, they are immediately put in a low value first (LVF) queue based on the *Priority_User* attribute. *RFI*s will wait in this queue until a member of the *UserAnalysts* set that corresponds to the value of the *User* attribute becomes available. This setup was chosen since it is primarily the duty of the user to determine if any existing information will meet their needs prior to requesting new collection. When an item leaves the queue, it is immediately checked to see if it still meets the timeliness

requirement. This is done by means of the following logical expression for *Timely* which evaluates to either 1 (true) or 0 (false):

$$(TNOW - Entity.CreateTime) - TimeR \leq Timely\_Threshold \qquad (3.1)$$

This expression takes the difference between the time an *RFI* has been in process and the time required and compares that difference with the threshold value which has a default value of 0. If the difference is less than the threshold, then there is still time to work on a requirement before it is due. In the event that an item is not timely, then it will be sent to the *Evaluation* submodel for statistics collection and disposal. The resource that was seized when the item left the queue is immediately released before any simulated time passes. It is important to note here that this can affect some statistics. In general, discrete statistics that involve counts, such as the number of times a resource was used, will be affected. However, time-weighted statistics, such as resource utilization, will not be affected since a zero simulated time has elapsed. In spite of the effect of this construction on the statistics, it was implemented to ensure resources were not unrealistically expended on requests that were no longer timely. The simulation user should be aware that this construction for a timeliness check is replicated throughout the IPM.

If an *RFI* passes the timeliness check, it encounters a delay based on the *QualR* and *InfoSource* attributes. As implemented, the simulation user would input the probability distributions as an array of expressions with one dimension corresponding to the levels of required quality and the other dimension corresponding to the information sources list. In general, if a dependency is explicitly noted, then it will be represented in the simulation as an array of expressions. The use of expressions provides the flexibility to model times or other values with probability distributions,

Table 3.3 *RFI* attributes that describe library search results.

Attribute	**Description**
FoundInLibrary	Indicates if item was (1) or was not (0) found in library
QualA	Indicates quality of item found in library (real value 0 to 5, 5 is best)
AgeMetInLibrary	Indicates if item found does (1) or does not (0) meet age requirement

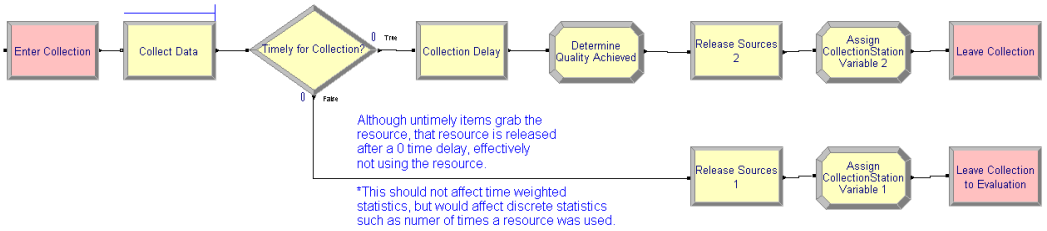

Figure 3.5 *Collection* submodel

mathematical functions, logical expressions, or a combination. As such, additional dependencies can easily be added if necessary.

Once the delay for an *RFI* is complete, the search results for that item are determined. A database or library is not actually searched to determine the search results, rather the results are determined probabilistically. Three attributes, *FoundInLibrary*, *QualA*, and *AgeMetInLibrary* (see Table 3.3), are assigned values that describe the search results. The value assigned to each of these attributes is dependent upon both the *User* and *InfoSource* attributes of the *RFI*. These dependencies allow the emulation of the ability of users to access various databases and libraries. The *FoundInLibrary* and *AgeMetInLibrary* attributes are only used in the user requirements planning portion of the model, whereas, the *QualA* attribute is used and updated throughout the entire model. Prior to leaving the library search for either the *Communications* or *Evaluation* submodels, the resource that was seized will be released, and the *LastStation* attribute will be updated to a value that corresponds to the library portion of the *Planning and Direction* submodel.

3.3.2 Collection Submodel

The *Collection* submodel corresponds directly to the collection phase of the Intelligence Cycle. This purpose of this submodel is to task collection from information sources and determine the quality of information collected. The visual structure of the *Collection* submodel is quite similar to that of the library portion of the *Planning and Direction* submodel, yet the underlying information differs. Entities arrive from the *Communications* submodel and are immediately placed in an LVF queue based on the *Priority_Global* attribute. Entities wait in the queue until an appropriate resource becomes available. When a member of the *Sources* resource set corresponding to a value of the *InfoSource* attribute becomes available, the *RFI* in the queue that has the highest global priority (i.e., lowest value of *Priority_Global*) for a given *InfoSource* value will be removed from the queue. When an entity leaves the queue, it will immediately undergo a timeliness check using Equation (3.1). If the item has failed to meet the timeliness requirement, then the resource that was seized will be released before any simulated time passes. Again, it is important that the simulation user understands that this will affect discrete count-type statistics but not time-weighted statistics involving the resources. The *LastStation* attribute will then be assigned a value corresponding to the *Collection* submodel, and the *RFI* will be sent to the *Evaluation* submodel for statistics collection. On the other hand, if the item is still timely, then the *RFI* will undergo a delay based on the *QualR* and *InfoSource* attributes. The delay corresponds to the time taken to collect information and is determined from an array of expressions (e.g., probability distributions, mathematical functions, or logical expressions) that is indexed on the value of those attributes. Once the delay is complete, a value rating the actual quality of collected information is assigned to the *QualA* attribute. The actual quality is determined from an array of expressions based on the *QualR* and *InfoSource* attributes. Prior to leaving the *Collection* submodel, the seized resource will be released and the

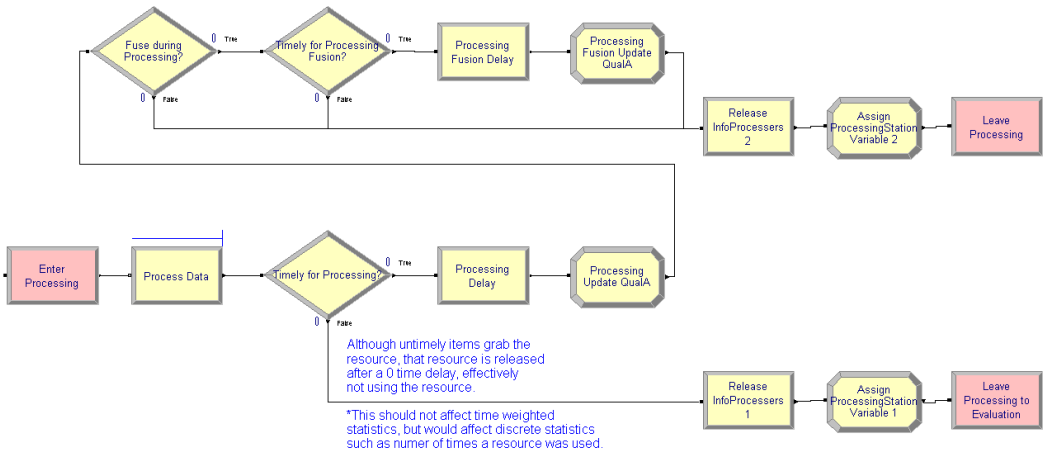

Figure 3.6 Processing submodel

LastStation attribute will then be assigned a value corresponding to the *Collection* submodel. Finally, the *RFI* will be sent back to the *Communications* submodel.

3.3.3 Processing and Exploitation Submodel

The *Processing and Exploitation* submodel corresponds directly to the Processing and Exploitation phase of the Intelligence Cycle. In order to further modularize the model, this submodel has been broken into two portions, one for processing and one for exploitation. Other than being contained in the same submodel, they are independent of each other. This independence allows entities to go through either portion alone or both.

3.3.3.1 Processing

The purpose of the processing portion of the *Processing and Exploitation* submodel is to task processing resources and determine the effects of processing on the quality of information. As entities enter processing, they are placed in an LVF queue based on the *Priority_Global* attribute. The *RFI*s will remain in the queue until a member of the *InfoProcessors* resource set corresponding to the value stored in the *InfoSource* attribute. Once an item is taken out of the queue, a timeliness check is

done using Equation (3.1) to avoid wasting resources on *RFI*s that are not timely. If an *RFI* fails the timeliness check, then the seized resource is released before any simulated time passes. As noted before, this will affect discrete count-type statistics but not time-weighted statistics involving the resources. Just prior to leaving for the *Evaluation* submodel, the *LastStation* attribute is assigned a value corresponding to the processing portion of the model. If an *RFI* passes the timeliness check, it will undergo a delay corresponding to the time needed to process the data associated with that *RFI*. This delay is drawn from an array of expressions that is dependent on *QualARank* and the *InfoSource* attribute.

QualARank is the following expression that ranks the value of the *QualA* attribute on a scale from one to six:

$$QualARank = MN\left(MX\left(ANINT\left(QualA\right), 0\right) + 1, 6\right) \tag{3.2}$$

The values are adjusted to range from one to six rather than zero to five since the first element of arrays in Arena have an index value of one. To clarify some of Equation (3.2), the *MN()* function takes the minimum of a list of values, the *MX()* function takes the maximum of a list of values, and the *ANINT()* function rounds a value to the nearest integer. In some studies it may be more appropriate for the value of *QualA* to be truncated instead of rounded for this ranking. If this is the case, then the simulation user could simply replace the *ANINT()* function with the *AINT()* function which truncates to an integer.

When the delay for processing is complete, the *QualA* attribute is updated. This update allows for the effect of processing on actual information quality to be modeled. The update to *QualA* here is only dependent on the *InfoSource* attribute. The next four blocks in the model add rudimentary support for information fusion.

Some proportion of *RFI*s will be selected at random to undergo fusion. For those that undergo fusion, another timeliness check is conducted using Equation

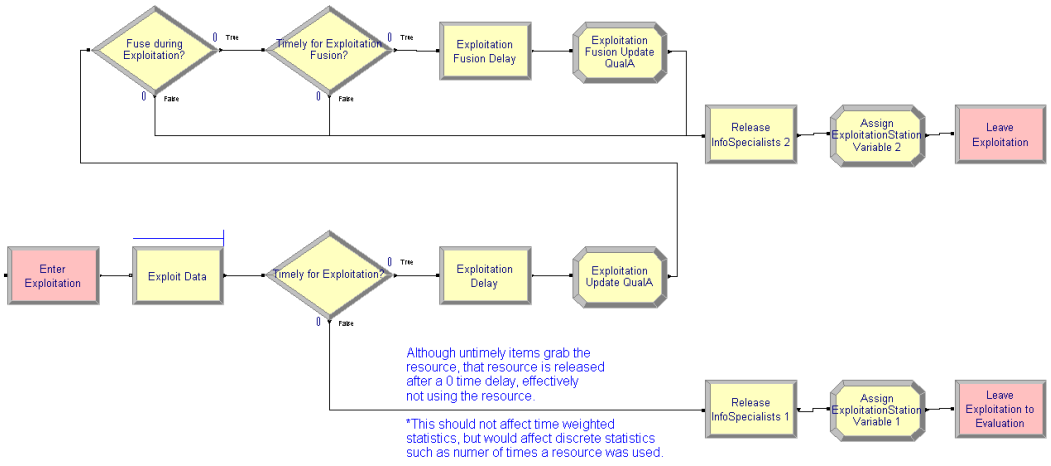

Figure 3.7 Exploitation submodel

(3.1). If the *RFI* passes the timeliness check, then it will undergo a delay that is dependent on *QualARank* in Equation (3.2) and the *InfoSource* attribute. Once the delay is complete, an additional update is made to *QualA* that is dependent on the *InfoSource* attribute. This update to *QualA* is done to reflect how fusion during processing affects actual quality. The information fusion modeled here amounts to additional processing on a given *RFI*. The assumption behind this implementation is that all other information required for fusion is available. This assumption was made since there is no association between individual *RFI*s in the current model. Developing a more robust fusion model would require *RFI*s to have multiple data sources which is beyond the scope of this study. As such, the fusion portion of this model has limited applicability.

Prior to leaving this portion of the model, the resources seized by the *RFI* are released. The *LastStation* attribute is assigned a value that corresponds to the processing portion of the model, and the *RFI* is sent back to the *Communications* submodel.

3.3.3.2 Exploitation

The exploitation portion of the *Processing and Exploitation* submodel is used to task information specialists to perform exploitation of information and determine the effect of exploitation on information quality. This portion of the model visually looks the same as the processing portion of the model, but the underlying information differs. As *RFI*s arrive from the *Communications* submodel they are placed in an LVF queue based on *Priority_Global*. When a member of the *InfoSpecialists* resource set corresponding to the *InfoSource* attribute of an *RFI* in the queue becomes available, the *RFI* will undergo a timeliness check using Equation (3.1). If the timeliness check fails, then the seized resource will be released and no simulation time will have elapsed. As before, this construction affects discrete count statistics but not time-weighted statistics. Prior to leaving for the *Evaluation* submodel, the *LastStation* attribute is assigned a value corresponding to the exploitation portion of the model. If the timeliness check is passed, then the *RFI* will be delayed according to the time needed to exploit information. As implemented, the delay for exploitation is dependent upon *QualR*, *QualARank* (Equation (3.2)), and *InfoSource*. Since this is technically a three-dimensional array of expressions, it could not be implemented as a single array in Arena. In order to overcome this, six expression arrays were created. The first array was indexed on *QualR* and drew values from five other arrays (one for each possible value of *QualR*) indexed on *QualARank* and *InfoSource*. Once the exploitation is complete, the affect on actual quality will be assessed. The value of *QualA* is assigned a value from an expression array indexed on the *InfoSource* attribute.

As with the processing portion of the model, this portion contains a rudimentary fusion model. This fusion model makes the assumption that all other required information is available at the time. Consequently, fusion as implemented has limited applicability. For the proportion of *RFI*s that are randomly selected to undergo fusion, a timeliness check will be conducted. If an item passes the timeliness check,

a delay will be incurred that is based on the *QualARank* expression (Equation (3.2)) and *InfoSource* attribute. The effect of fusion during exploitation on actual quality is then assessed when the *QualA* attribute is updated based on the *InfoSource* attribute.

Prior to leaving the exploitation portion of the model for the *Communications* submodel, the resources used for exploitation are released. Additionally, the *LastStation* attribute is assigned a value corresponding to the exploitation portion of the model.

3.3.4 Analysis and Production Submodel

The *Analysis and Production* submodel corresponds to the Analysis and Production phase of the Intelligence Cycle. As with the *Processing and Exploitation* submodel, this submodel consists of two independent portions. As the name of the submodel suggests, one portion is for analysis and the other for production. For some information sources, it may be the case that analysis and production cannot be distinguished. If so, then the simulation user could set up the simulation to use one of them and not the other and assign appropriate delay and quality update expressions to the part that is used. To accomplish this an expression more complex than a simple proportion will be needed when the *Analyze* and *Produce* attributes are assigned (see Section 3.3.1.1).

3.3.4.1 Analysis

The analysis portion of the model is visually similar to the processing and exploitation portions of the model but the underlying information differs. When *RFI*s enter the analysis portion of the model they enter an LVF queue based on *Priority_Global*. When a member of the *AllSourceAnalysts* resource set becomes available, the *RFI* with the highest priority is removed from the queue. If more than one member of

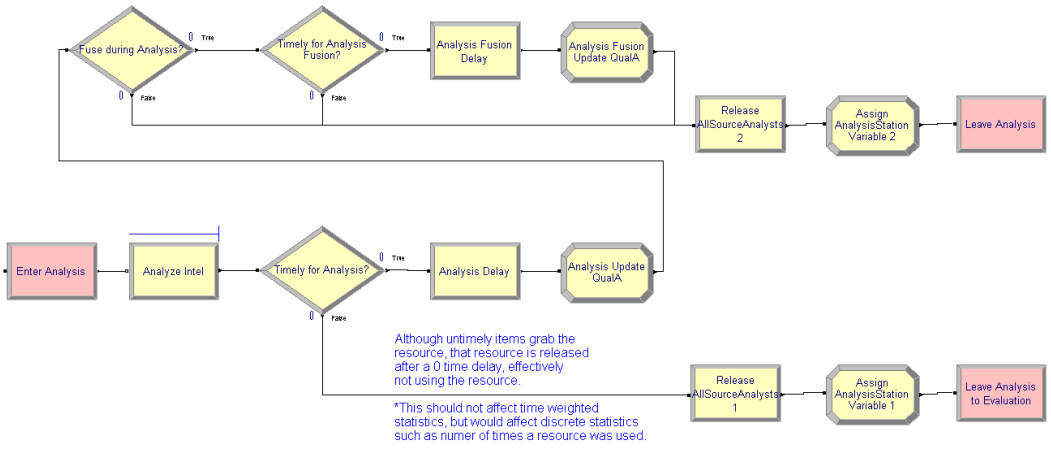

Figure 3.8 Analysis submodel

the *AllSourceAnalysts* set is available, then the one chosen is selected at random. The *AllSourceAnalysts* set is divided into thirteen specialties that may or may not correspond to the *Sources* resource set, depending on the study. The specialty of the analyst used is stored in the *AnalystSpecialty* attribute. Upon exiting the queue, a timeliness check is done using Equation (3.1). If an item is not timely, then the resource is released prior to any simulation time elapsing. As noted before, this affects count statistics but not time-weighted statistics. The *LastStation* attribute is assigned a value corresponding to the analysis portion of the model prior to being sent to the *Evaluation* submodel for statistics collection. The *RFI*s that are still timely incur a delay based on *QualR*, *QualARank* (Equation (3.2)), and *InfoSource*. This three-dimensional expression array is implemented similarly to the three-dimensional array discussed in Section 3.3.3.2. After the delay, the effect of analysis on actual information quality is assessed by updating the *QualA* attribute from an expression array based on the *InfoSource* and *AnalystSpecialty* attributes. As with the processing and exploitation portions of the model, the analysis portion contains a rudimentary capacity for fusion. It contains the same assumption that any additional required information is available for fusion. For the proportion of *RFI*s that undergo fusion, a timeliness check is done using Equation (3.1). For items that are timely, they will undergo a delay based on the *QualARank* expression

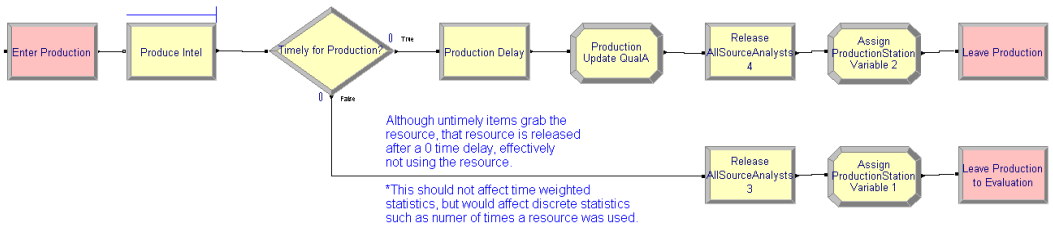

Figure 3.9 Production submodel

(Equation (3.2)) and *InfoSource* attribute. Upon completion of fusion, the *QualA* attribute is updated based on the *InfoSource* attribute to reflect how fusion during analysis affects actual quality. Prior to *RFI*s leaving this portion of the model for the *Communications* submodel, the resources used are released and the *LastStation* attribute is assigned a value corresponding to the analysis portion of the model.

3.3.4.2 Production

The production portion of the model visually resembles the *Collection* submodel rather than the analysis portion of the model since it does not contain any blocks to model fusion. As *RFI*s arrive from the *Communications* submodel, they are placed in an LVF queue based on *Priority_Global*. A member of the *AllSourceAnalysts* resource set is selected at random if more than one is available. The specialty of the selected analyst is stored in the *AnalystSpecialty* attribute of the *RFI*. Note that if the *RFI* has already undergone analysis, this value will be overwritten and the selected specialty is independent of the specialty used for analysis. If the simulation user needs the specialty to be identical, then they could modify the seize block to select a specific member of the *AllSourceAnalysts* set based on the value stored in the *AnalystSpecialty* attribute. Care must be taken to account for those items that have not undergone analysis since the *AnalystSpecialty* attribute will contain the invalid set index value of zero. When an *RFI* leaves the queue, it undergoes a timeliness check using Equation (3.1). For those items that fail the timeliness check, the resource will be released after no simulation time has elapsed. As noted before,

this affects discrete count statistics but not time-weighted statistics. Additionally, the *LastStation* attribute will be assigned a value corresponding to the production portion of the model prior to being sent to the *Evaluation* submodel. Those items that pass the timeliness check will incur a delay for production based on *QualR*, , *QualARank* (Equation (3.2)), and *InfoSource*. This three-dimensional array is implemented similarly to the three-dimensional array discussed in Section 3.3.3.2. Once the production delay is complete, the effect of production on quality is determined by updating *QualA* from an expression array depending on the *InfoSource*, and *AnalystSpecialty* attributes. After the quality is updated, the resource being used is released and the *LastStation* attribute is assigned a value corresponding to the production portion of the model. Finally, the *RFI* is sent back to the *Communications* submodel.

3.3.5 Dissemination and Integration Submodel

The *Dissemination and Integration* submodel corresponds to the Dissemination and Integration phase of the Intelligence Cycle. This portion of the model is intended to reflect more of the user process and as such involves the use of user resources. Similar to other submodels, this submodel is divided into two independent portions, one for dissemination and one for integration. Integration here relates to the user integrating information into their processes not information integration as related to exploitation or analysis.

3.3.5.1 Dissemination

The purpose of the dissemination portion of the model is to reflect the user process of actually acquiring information that has been through some portion of the intelligence process. *RFI*s arrive from the *Communications* submodel and are placed in an LVF queue based on *Priority_User*. Note that this differs from most of the queues to this point, and is based off of user priorities rather than global priorities since it is

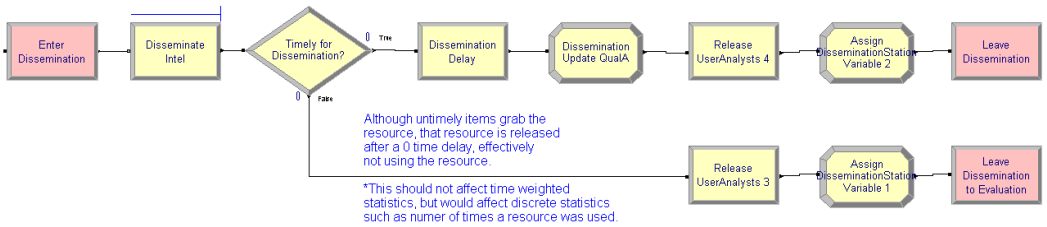

Figure 3.10 Dissemination submodel

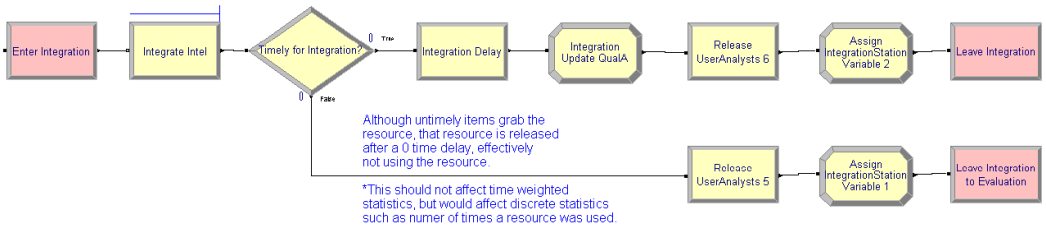

Figure 3.11 Integration submodel

primarily a user process. *RFI*s wait in the queue until a member of the *UserAnalysts* resource set that corresponds to the value of the *User* attribute becomes available. Note that no communications resources are currently used here. When an item is removed from the queue, a timeliness check is performed using Equation (3.1). Those items that are not timely are sent to the *Evaluation* submodel after releasing the seized resource and assigning *LastStation* a value corresponding to the dissemination portion of the model. The resource is released after zero simulation time has passed which, as noted before, affects discrete count statistics but not time-weighted statistics. Those items that are timely will then undergo a delay that depends on the *User* attribute. When dissemination is complete, the effect on quality is determined by updating the *QualA* attribute from an expression array based on *InfoSource*. The seized resource is then released, and the *LastStation* attribute is assigned a value corresponding to the dissemination portion of the model.

3.3.5.2 Integration

The purpose of the integration portion of the model is to reflect the integration of information into user processes. The use or non-use of this portion of the model changes the interpretation of the timeliness measure. If this portion of the model is not used, then timeliness is essentially determined by when a user receives requested information. If this portion of the model is used, then timeliness is determined by when a user is able to use information they have received. As *RFI*s arrive from the *Communications* submodel, they enter an LVF queue based on the *Priority_User* attribute. They will wait in the queue until a member of the *UserAnalysts* resource set corresponding to the *User* attribute becomes available. Upon leaving the queue, the *RFI*s undergo a timeliness check using Equation (3.1). Those items that fail the timeliness check will release the resource before any simulation time elapses. As noted before, this affects count statistics but not time-weighted statistics. The *LastStation* attribute will be assigned a value corresponding to the integration portion of the model prior to sending the untimely *RFI*s to the *Evaluation* submodel. Those items that pass the timeliness check will incur a delay based on the *User* attribute. Upon completion of integration the affect on quality is assessed by updating the *QualA* attribute from an expression array based on the *InfoSource* attribute. Finally, the seized resource is released, the *LastStation* attribute is assigned a value for the integration portion of the model, and the *RFI* is sent back to the *Communications* submodel.

3.3.6 Evaluation Submodel

The purpose of the *Evaluation* submodel is to collect statistics and dispose of entities. If there are no problems with the model, all entities should exit the system through the *Evaluation* submodel. In addition to the default statistics gathered, additional statistics are needed to evaluate the quality, timeliness, and information needs satisfaction (as it relates to meeting both quality and timeliness) measures.

Table 3.4 *Evaluation* submodel expressions assigning variables for statistics.

S_Num_Timely_Priority(Priority_User)	=	Timely*(S_Num_Timely_Priority(Priority_User)+1) + (1-Timely)*S_Num_Timely_Priority(Priority_User)
S_Num_Timely_User(User)	=	Timely*(S_Num_Timely_User(User)+1) + (1-Timely)*S_Num_Timely_User(User)
S_Num_Timely_Standard(Standard+1)	=	Timely*(S_Num_Timely_Standard(Standard+1)+1) + (1-Timely)*S_Num_Timely_Standard(Standard+1)
S_Num_Qual_Priority(Priority_User)	=	QualMet*(S_Num_Qual_Priority(Priority_User)+1) + (1-QualMet)*S_Num_Qual_Priority(Priority_User)
S_Num_Qual_User(User)	=	QualMet*(S_Num_Qual_User(User)+1) + (1-QualMet)*S_Num_Qual_User(User)
S_Num_Qual_Standard(Standard+1)	=	QualMet*(S_Num_Qual_Standard(Standard+1)+1) + (1-QualMet)*S_Num_Qual_Standard(Standard+1)
S_Num_QT_Priority(Priority_User)	=	Timely*QualMet*(S_Num_QT_Priority(Priority_User)+1) + (1-Timely*QualMet)*S_Num_QT_Priority(Priority_User)
S_Num_QT_User(User)	=	Timely*QualMet*(S_Num_QT_User(User)+1) + (1-Timely*QualMet)*S_Num_QT_User(User)
S_Num_QT_Standard(Standard+1)	=	Timely*QualMet*(S_Num_QT_Standard(Standard+1)+1) + (1-Timely*QualMet)*S_Num_QT_Standard(Standard+1)
S_Num_Out_Priority(Priority_User)	=	S_Num_Out_Priority(Priority_User)+1
S_Num_Out_User(User)	=	S_Num_Out_User(User)+1
S_Num_Out_Standard(Standard+1)	=	S_Num_Out_Standard(Standard+1)+1
S_Num_Timely_Src(InfoSource)	=	Timely*(S_Num_Timely_Src(InfoSource)+1) + (1-Timely)*S_Num_Timely_Src(InfoSource)
S_Num_Qual_Src(InfoSource)	=	QualMet*(S_Num_Qual_Src(InfoSource)+1) + (1-QualMet)*S_Num_Qual_Src(InfoSource)
S_Num_QT_Src(InfoSource)	=	Timely*QualMet*(S_Num_QT_Src(InfoSource)+1) + (1-Timely*QualMet)*S_Num_QT_Src(InfoSource)
S_Num_Out_Src(InfoSource)	=	S_Num_Out_Src(InfoSource)+1
S_Tot_WaitTime_Priority(Priority_User)	=	S_Tot_WaitTime_Priority(Priority_User) + Entity.WaitTime
S_Max_WaitTime_Priority(Priority_User)	=	MX(S_Max_WaitTime_Priority(Priority_User), Entity.WaitTime)

The additional data needed is gathered as entities enter the *Evaluation* submodel. The data is stored in variables and is gathered according to the expressions given in Table 3.4.

The expressions make extensive use of the *Timely* and *QualMet* logical expressions which evaluate to 0 or 1 to track many of the counts stored in the variables. The *QualMet* expression is given in the following equation:

$$QualR - QualA \leq QualMet\_Threshold \tag{3.3}$$

This expression compares the difference between the required and actual quality with a simulation user specified threshold value which by default is zero. At the end of a simulation run, these variables are used to generate various statistics.

Another feature that is partially implemented in the *Evaluation* submodel is an initial capability to allow feedback of *RFI*s into the remainder of the system. However, this capability should not be used since the model is not programmed to do anything useful with feedback items. As currently implemented, any entity that

leaves the *Evaluation* submodel will travel through the *Communications* submodel to the planning portion of the model where it will be immediately disposed.

3.3.7 Communications Submodel

The primary purpose of the *Communications* submodel is to route entities between the other submodels as well as provide appropriate communications delays and resources. The communications model is broken into two portions, the first receives entities from the other parts of the model and the second portion sends entities to other portions of the model. The primary tasks of the first portion of the model are to determine where, i.e., which portion of the model, the entity should go to next, task appropriate resources for communications, and send items to the second portion of the model. The primary tasks of the second portion of the model are to update information quality and send the *RFI* to the submodel determined by the first portion of the *Communications* submodel. Prior to discussing each of the paths, it is important to note that only minimal communications delays and resources are modeled. A detailed communications model is not appropriate due to the high level of the overall model.

The first portion of the *Communications* submodel, shown in Figure 3.12, has 10 entry points corresponding to the various portions of the model discussed above. From top to bottom, as entities arrive from the planning portion of the model, a check is done to see if the *NextStation* attribute is set to a value other than zero. This occurs when items need to undergo a library search, and it is set in the planning portion of the model. If the value is greater than zero, then the entities are routed to the second portion of the *Communications* submodel. Otherwise, the *NextStation* attribute is assigned a value using *VBA* block 1. The *VBA* blocks were used in lieu of *Assign* blocks since the required logical expressions exceeded the maximum length of an expression field. Furthermore, the logic of the code in *VBA* is easier to create and understand than equivalent but exceptionally long logical expressions.

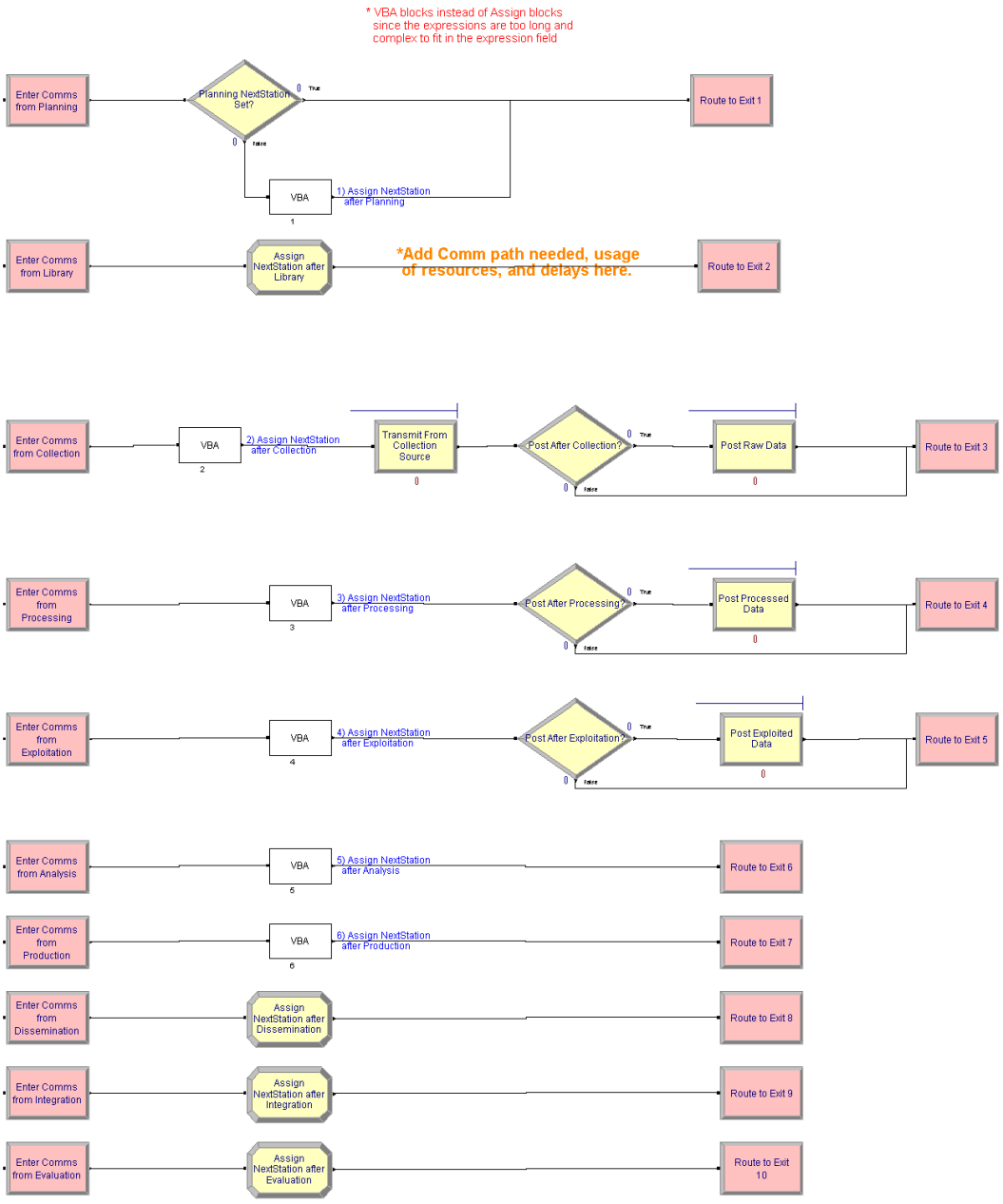

Figure 3.12 First portion of *Communications* submodel

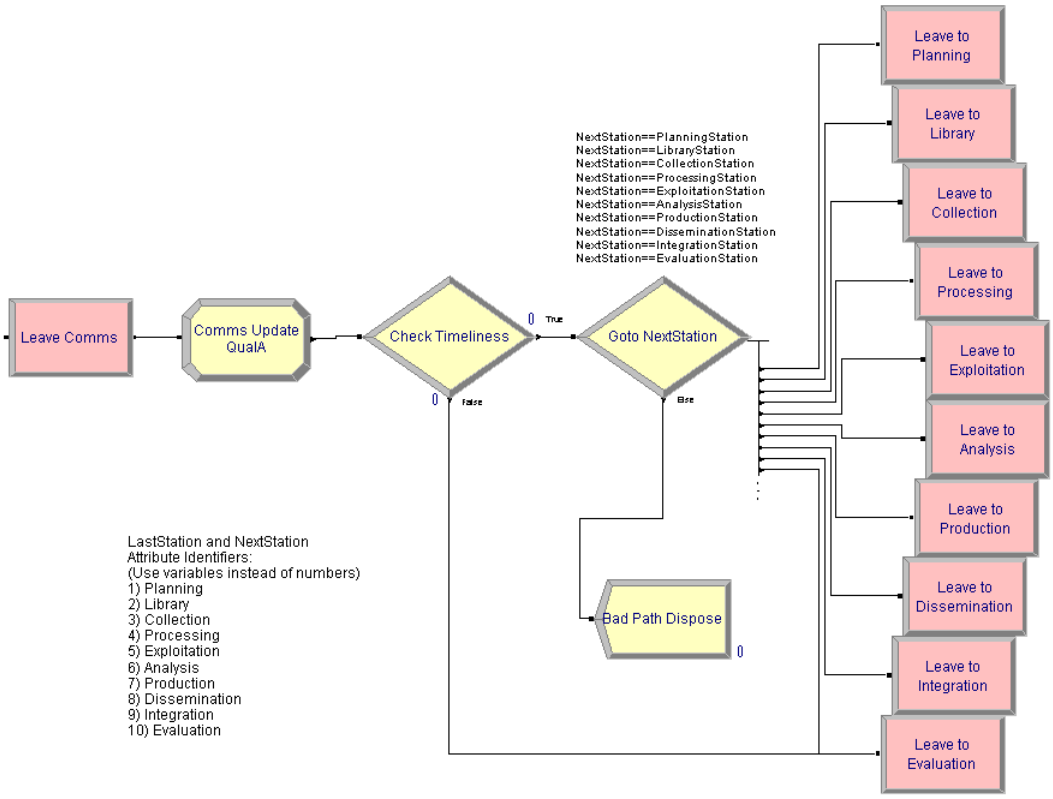

Figure 3.13 Second portion of *Communications* submodel

Algorithm 1 Portion of code for *VBA* block 1.

```
' Determine NextStation after leaving Planning
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
```

The portion of the code that contains the logic for where to go next is given in Algorithm 1. The complete code for all *VBA* blocks is given in Appendix B. The code sequentially examines the values of the attributes given in Table 3.2 and sets the *NextStation* attribute to the value corresponding to the first attribute that has a value of one. This is similar to the *VBA* blocks used for other entry points into the *Communications* submodel. The second entry block receives entities from the library portion of the model. As the *RFI*s arrive, the *NextStation* attribute is assigned a value corresponding to the planning portion of the model. The reason for this is that the planning portion of the model needs to determine what to do with the search results. For more information, see Section 3.3.1. *RFI*s are then sent to the second portion of the *Communications* submodel.

The third entry block receives *RFI*s from the *Collection* submodel and assigns the *NextStation* attribute according to *VBA* block 2. The logic is similar to that depicted in Algorithm 1 except that the value of the *Collect* attribute would not be checked since the *RFI* has already been through collection. The *RFI*s then enter a FIFO queue and wait until a member of the *CommsCollection* resource set corresponding to the *InfoSource* attribute is available. Upon leaving the queue, a

delay is incurred based on the *InfoSource* attribute. Once the delay is complete, the resource is released and the value of the *PostInfo* variable is checked. If the value is 0, then the *RFI* will be routed to the second portion of the *Communications* submodel. If the value is 1, then the *RFI* will enter an LVF queue for posting based on the *Priority_Global* attribute. The *RFI* will wait in the queue until a *CommsPost* resource becomes available. Note that this resource is shared with two other posting queues which have equal priority for using resources. Upon leaving the queue, the entity will undergo a delay based on the *InfoSource* attribute and release the resource when the delay is complete. The *RFI* will then be routed to the second portion of the *Communications* submodel. It is important to note here that this communications modeling is not robust and does not adequately capture the complexity of the communications environment. As such it becomes a prime candidate for further study.

The fourth entry block receives *RFI*s from the processing portion of the model. As the entities arrive, the *LastStation* attribute is assigned a value by *VBA* block 3. The logic is similar to that depicted in Algorithm 1 except that it begins with a check for needing exploitation and does not check if prior steps are needed. The value of the *PostInfo* variable is then checked and routed according to its value. If the value is 0, then the *RFI* is sent to the second portion of the *Communications* submodel. If the value is 1, then the *RFI* will enter an LVF queue based on the *Priority_Global* attribute. It will wait until a *CommsPost* resource becomes available and undergo a delay based on the *InfoSource* attribute. Note that the *CommsPost* resource is shared with two other posting queues. Once the delay is complete, the *RFI* is sent to the second portion of the *Communications* submodel.

The fifth entry block receives *RFI*s from the exploitation portion of the model and assigns the *LastStation* attribute a value with *VBA* block 4. The code for this *VBA* block is similar to Algorithm 1 except that it begins with a check for needing analysis and does not check if prior steps are needed. Depending on the value of the

PostInfo variable, the entity will either be sent directly to the second portion of the *Communications* submodel or will enter an LVF queue based on the *Priority_Global* attribute. If the value is 1, then the entity will enter the queue and wait until a *CommsPost* resource becomes available. When the *RFI* leaves the queue, it will undergo a delay based on the *InfoSource* attribute. Once the delay is complete, the *RFI* is sent to the second portion of the *Communications* submodel.

The next five entry blocks receive *RFI*s from the analysis, production, dissemination, integration, and evaluation portions of the model respectively. No delays are incurred as entities go through these parts of the model. The *LastStation* attribute is assigned a value in each case by examining the attributes for later portions of the model similar to Algorithm 1. The assign blocks contain the logical expression equivalent of the VBA code and are used since the expressions are relatively short. Upon arriving from integration the only option is to be sent to the *Evaluation* submodel. The entities arriving from the *Evaluation* submodel are sent to the planning portion of the model, however as noted in the previous section the planning portion of the model will simply dispose of the items. For all five paths, the entities will be sent to the second portion of the *Communications* submodel.

The second portion of the *Communications* submodel, shown in Figure 3.13, receives *RFI*s from the first portion of the submodel. The first thing to occur is an update to the *QualA* attribute to reflect the effect of communications on the quality of information. The update here is only dependent upon the *InfoSource* attribute and occurs every time an entity goes through the submodel. Depending on the simulation setup, some entities may travel through this quality update as many as 10 times. This may be an item to check if output quality is less than expected. Once the quality has been updated, *RFI*s undergo a timeliness check according to Equation (3.1). Those items that are no longer timely are sent immediately to the *Evaluation* submodel for statistics collection and disposal. Otherwise, the value of the *NextStation* attribute is used to branch to the various portions of the

model. If the *NextStation* attribute contains an unknown value, entities will be immediately disposed. This was implemented for model verification and no entities should be disposed of here. The use of variables containing known values helps to avoid incorrect routing.

3.4 Implementation of Measures

The primary measures of interest, quality, quantity, timeliness, and information needs satisfaction (as it relates to meeting both quality and timeliness) are implemented for a variety of aspects of the model. Upon examination of raw data for any of these measures, it becomes clear that a composite implementation is more useful than a measure such as the average information quality. Furthermore, a statistic such as the raw number of requests that entered and exited the system provides limited insight. As such, the quantity measure was rolled into the implementation of the other measures to create proportions of requests that met quality, timeliness, or both requirements. The use of proportions in this way also makes it easier to compare various alternatives to a baseline system. Separating these proportions into various categories, such as by user or information source, also provides additional insight into the system. The variables that collect various counts and sums according to the expressions in Table 3.4 are used to generate these proportions at the end of a simulation run.

One quality measure, collected by

$$Pct\_QualMet\_Usr\_3 \;=\; S\_Num\_Qual\_User(3)/S\_Num\_Out\_User(3),$$

calculates the proportion of requests by user 3 that met the quality requirements. Note that this is independent of meeting the timeliness requirement. Additional

quality measures are implemented similarly. An example of a timeliness measure,

$$Pct\_Timely\_Src\_01 \;=\; S\_Num\_Timely\_Src(1)/S\_Num\_Out\_Src(1)$$

calculates the proportion of requests needing information from the first information source that meet the timeliness requirement. As with the quality measures, additional timeliness measures are implemented similarly. With respect to meeting both timeliness and quality requirements, an example measure is

$$Pct\_QT\_Pr\_4 \;=\; S\_Num\_QT\_Priority(4)/S\_Num\_Out\_Priority(4)$$

which calculates the proportion of items with $Priority\_User = 4$ that meet both the timeliness and quality requirements. Other proportions for meeting both timeliness and quality requirements are implemented similarly.

In addition to these defined statistics, some measures such as resource utilization, average wait times in queues, and average number of items in the system are implemented by default within Arena so long as the option to collect them is enabled. Because of preliminary results for wait times in queues, additional statistics were gathered to examine the total wait time in system (i.e., combined wait times for all queues) broken down by priority. For example, the statistic

$$S\_Avg\_WaitTime\_Pr\_2 \;=\; S\_Tot\_WaitTime\_Priority(2)/S\_Num\_Out\_Priority(2)$$

collects the average wait time of items with $Priority\_User = 2$ over a simulation run. Similar statistics are implemented for the other priorities and the maximum wait time by priority. The benefit of these statistic is that it helps to provide insight into why wait times for queues might appear longer than expected. Although the total wait times are not broken out by other categories like the proportion statistics,

such statistics could be easily implemented by the simulation user. A complete list of implemented statistics is given in Appendix E.

3.5 Validation and Verification

Early in the model development process, a meeting was held with subject matter experts (SMEs) to ensure that the initial conceptual model was correct. Based on the results of this meeting and further research, a more robust conceptual model was developed. As the conceptual model was being developed, it was also drafted in Arena. By the second major revision of the conceptual and draft Arena model, a description was written and sent to SMEs for review. In addition to the written description, basic flowcharts of potential entity flow throughout the model as well as the core model logic for determining that flow were submitted to SMEs for review (see Appendix C). SME comments on the description and flowcharts were used to update the conceptual and implemented models. Once these updates were complete, a meeting with SMEs was held to conduct a low level walk-through of the model implementation in Arena. This low level review resulted in minor revisions to the implemented model. Overall, involving SMEs early and throughout the process helped to ensure that the right model was being developed. Multiple reviews of the conceptual and implemented models also helped to ensure that the implementation was correct.

The development of a data request sheet for collecting notional or real data presented an additional opportunity to verify the coding of the model. Furthermore, once notional data was constructed for the sample study, inputing data into the model ensured that these areas of the model were correct. With notional data in the model, examination of simple animation and output statistics was used to determine if the model was working as expected. The use of animation and extensive statistics allows any problems to be easily traced to various portions of the model. As an example, some portions of the model were not used for the sample study, yet an

incorrectly set input variable was sending items to that portion of the model causing an unexpected bottleneck in the system. Some of the sample study cases discussed in Section 3.7 were used to stress the system. Reasonable results provide assurance that the model is correct. Simulation results are discussed in Section 4.3.

3.6 Data Requirements

The data requirements for simulation experiments can be overwhelming. Even in a high level model such as this, the amount of information required is extensive. The data required for this model can be divided into the following basic categories: processing (i.e., delay) times, resources, effects on quality, and request properties.

The simulation requires processing and delay times for each portion of the model except the evaluation submodel. Everywhere in the model that a delay is incurred, a resource is required. Furthermore, each time processing of some sort occurs on an *RFI* and update is made to the actualized quality attribute, *QualA*. The request properties are used to determine when and what type of *RFI*s are generated. A complete listing of the required data for a simulation study is given in the form of a data request sheet in Appendix D. The sheet is filled in with the data used for the sample study discussed in the next section. One key aspect to note about the data is that it is often in the form of probability distributions rather than just raw numbers.

3.7 Sample Study Overview

In order to demonstrate how the IPM simulation could be used for studies, a sample study was conducted. Real world data that would be needed to populate the model was not available due to its sensitivity and security classification. As such, notional data was generated for the sample study excluding parts of the model that were not under examination. The notional data was used as a baseline for

comparative study. Prior to conducting any comparisons, replication truncation and termination points were selected. The analysis to determine the replication length is given in Section 4.2.

In addition to performing analysis on the baseline simulation results, seven additional simulations were performed to examine various aspects of the model. The cases were defined by making the following changes to the baseline model:

1. Change *Timely_Threshold* from 0 to 48 hours
2. Change *Timely_Threshold* from 0 to 12 hours
3. Change *QualMet_Threshold* from 0 to 3
4. Change *QualMet_Threshold* from 0 to 1
5. Increase number of Additional Requirements by 50%
6. Increase Exploitation times by 50%
7. Increase Analysis times by 50%

Note that only one change was made for each case and any prior changes were reset to the baseline settings. Details of the eight simulations are given in Section 4.3.

4. Results and Analysis

4.1 Introduction

Since the method of multiple independent replications will be used for conducting comparative studies, determination of appropriate replication length is important. A single long run of the baseline system described by the notional data given in Appendix D was used for determining an appropriate truncation point and replication length. The process of determining the replication truncation and termination points is discussed in Section 4.2. Once the replication length was determined the simulation was run for 25 replications for the baseline system as well as for each of seven additional cases used to stress various parts of the model. The results and analysis of these simulation runs are discussed in Section 4.3.

4.2 Replication Truncation and Termination

Although multiple replications are used for comparison of system designs, truncating initial data for statistics collections allows removal of some estimator bias. Two measures were examined when determining truncation points, entity total time in system (TIS) and work in process (WIP). Ideally, if there were a specific measure that an analyst were interested in, they would use that measure for determining replication truncation and termination points. However, since the sample study examines a variety of measures, the TIS and WIP measures were selected since they are representative of the system in general. The time in system for each entity that exited the system and the number of items being serviced and in queues (i,.e., WIP), were collected in a text file over a period of about 15 simulated years for only one replication. Matlab and Octave were used to import and analyze the data. The TIS data was examined directly while the WIP data was first discretized by taking the time-average WIP per day. It was determined that 120 days of simulation time

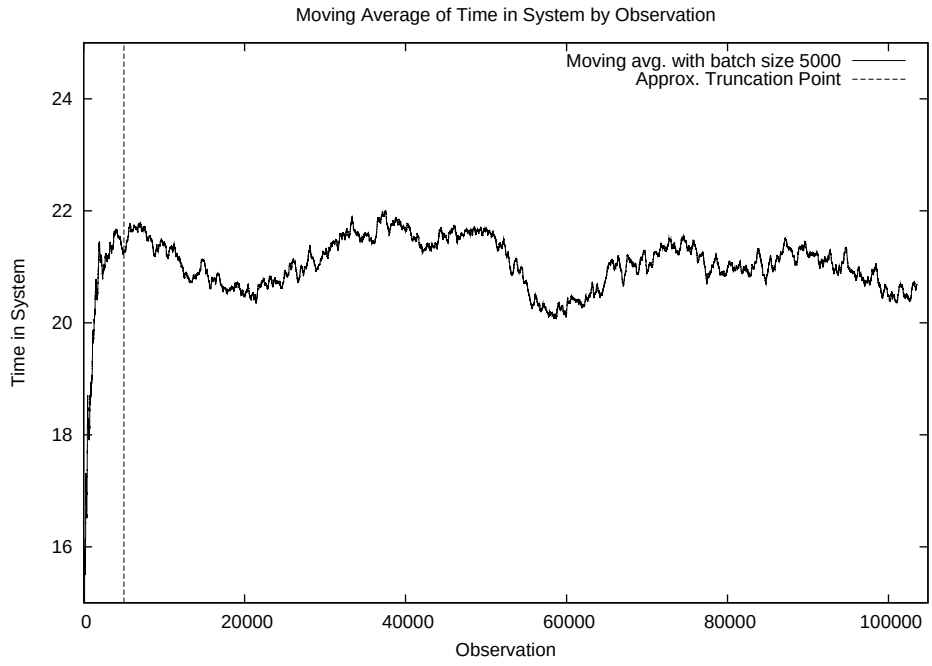

Figure 4.1 Moving average plot of *RFI* time in system.

would be truncated and the simulation would be run for an additional 4 years (1460 days) of simulated time for a total replication length of 1580 days.

4.2.1 Time in System

In the process of determining a truncation point, a moving average plot of TIS was constructed using a modified version of the algorithm given in "The Statistical Analysis of Simulation Results" [Welch, 1983:294]. The moving average plot for TIS shown in Figure 4.1 was constructed using a batch size of 5000 observations. Since multiple replications will be used for conducting the comparative studies, the amount of truncated data would ideally be relatively small. Furthermore, a clear truncation point is difficult to select. As a compromise a truncation time of 30 days was selected, which corresponds to about 5000 observations which is indicated in the figure. This allows some of the initialization bias to be accounted for without throwing away too much data.

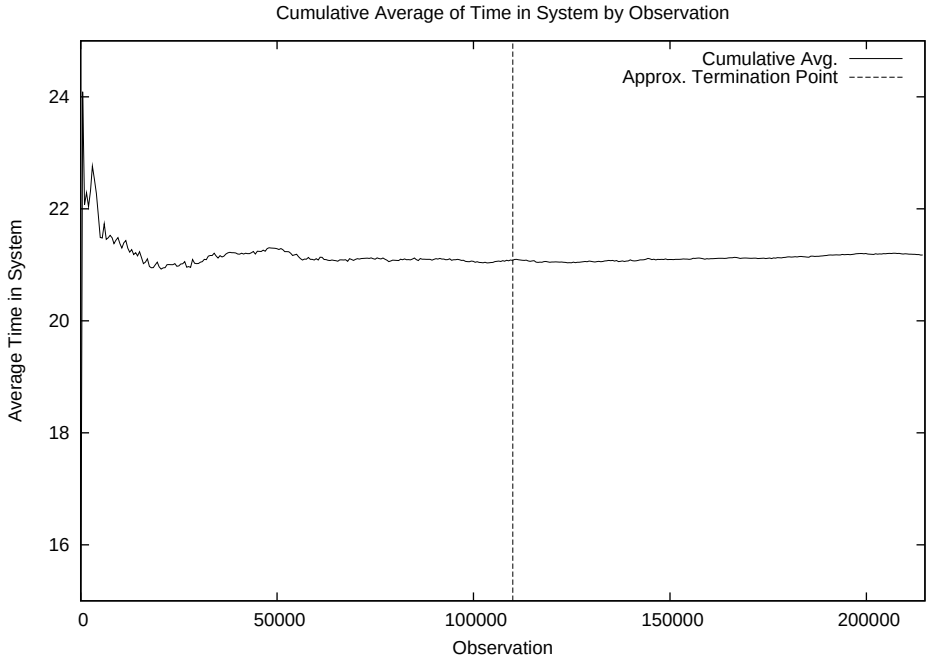

Figure 4.2 Cumulative average plot of *RFI* time in system using truncated data.

To determine replication length, several methods were employed. One method was a cumulative average plot of the TIS. This plot, given in Figure 4.2, was generated using the the previous data with the first 5000 observations truncated. The idea is to select a termination point for replications that appears to be near steady state. The cumulative average appears to stabilize after about 60,000-75,000 observations. A termination point of two years of simulated time beyond truncation was selected which corresponds to about 110,000 observations which is marked in the figure.

4.2.2 Work in Process

After the WIP data was discretized by taking the time-average WIP per day, a moving average plot was constructed in the same manner as before. From Figure 4.3 a truncation point can be selected much easier than for the TIS data. However, previously selected truncation of 30 days of data appears to be insufficient. As such a

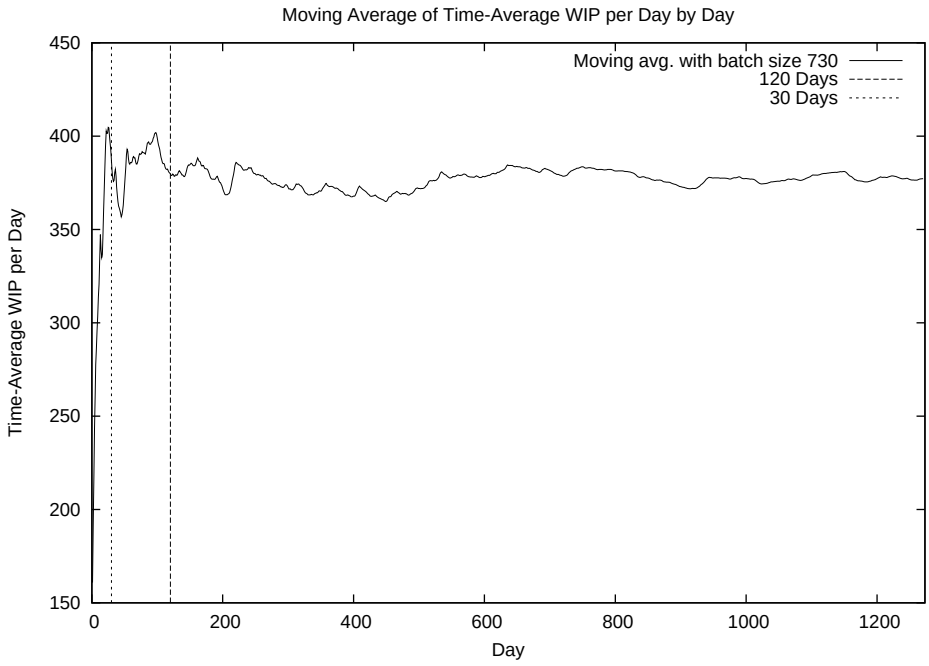

Figure 4.3 Moving Average Plot of *RFI* time-average WIP per day.

truncation point of 120 days was selected with keeping the amount of data discarded relatively small.

After truncating 120 days of the data, a cumulative average plot of the time-average WIP per day was constructed. This chart allows one to visually see when the variation in the data begins to have a reduced affect on the overall mean that will be calculated for each replication. The 2 year replication length that was selected for the TIS data is marked in Figure 4.4. From the plot, the 2 year point may be too short, so a termination point of 4 years was selected since it appears to be near the beginning of the steady state for the cumulative average. Since multiple replications are going to be used, it should not be necessary to select a termination point well into steady state. If multiple replications were not going to be used, then a longer replication length would be required for methods such as batch means.

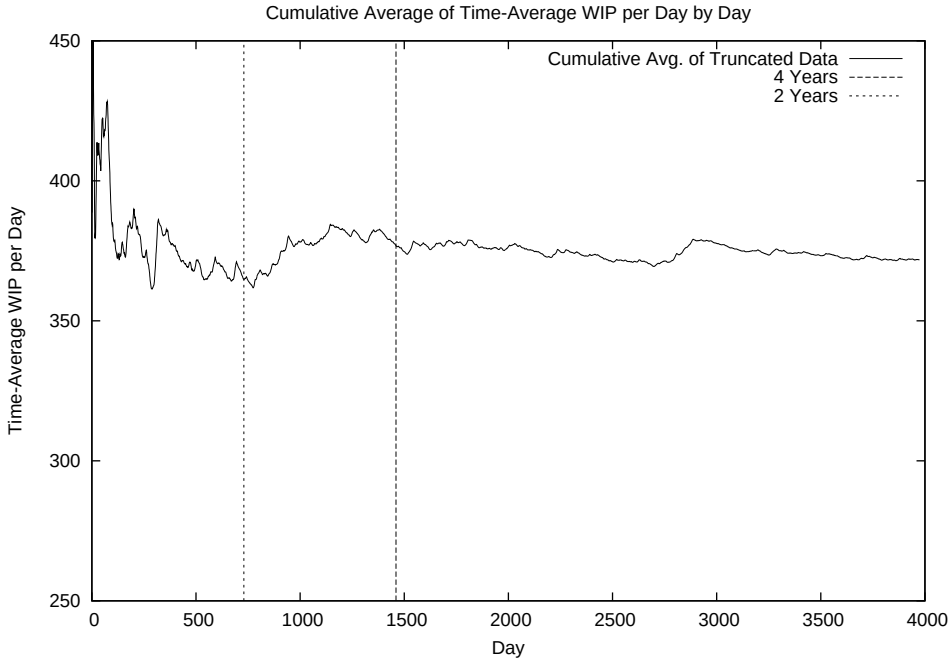

Figure 4.4 Cumulative Average Plot of *RFI* time-average WIP per day using truncated data.

4.3 Simulation Results

The simulation was run for each of the following cases for 25 replications with a replication length of 1580 days and truncation of the first 120 days:

1. Change *Timely_Threshold* from 0 to 48 hours
2. Change *Timely_Threshold* from 0 to 12 hours
3. Change *QualMet_Threshold* from 0 to 3
4. Change *QualMet_Threshold* from 0 to 1
5. Increase number of Additional Requirements by 50%
6. Increase Exploitation times by 50%
7. Increase Analysis times by 50%

The results of each replication were stored in a Microsoft Access database. Arena version 5.00.02, which was used, stores the data in the Access 97 database format.

Several queries were used to import the data into Microsoft Excel and OpenOffice.org Calc for basic analysis and chart generation.

The large quantity and categorical nature of many of the output statistics lend themselves to representation with bar charts. The benefit of the bar charts is that they allow the outputs to be quickly compared for practical significance. Some of these are reported and discussed in the following sections. Additional detailed results including tables of point estimates with associated 95% confidence interval half-widths are reported in Appendix F. One thing to note in general is that the half-widths are relatively small compared to the point estimates. This indicates that a sufficient number of replications was selected to get relatively precise point estimates. However, this does not imply that the point estimates are the true expected values since they do not give an indication of point estimator bias. In many cases, this will allow small differences in point estimates to be statistically significant, but does not indicate any practical significance. As such, practical significance and application will be the focus of discussion.

4.3.1 Baseline System

Since the baseline model contains only notional data, the actual numbers for individual statistics have little meaning in themselves. However, they allow a starting point for comparison both within the various parts of the model as well as with other models. Results for standard process performance measures will be discussed first followed by the additional implemented measures.

One statistic examined was the number of items in the system (NIS) or work in process (WIP). This includes both items waiting in queues and being serviced by resources. Figure 4.5 show the average over all replications and the average of the maximum values observed in each replication. One thing to note about the values is that the average maximum value is more that twice that of the average. However, it does not indicate how often the NIS is near the average maximum. Since there are

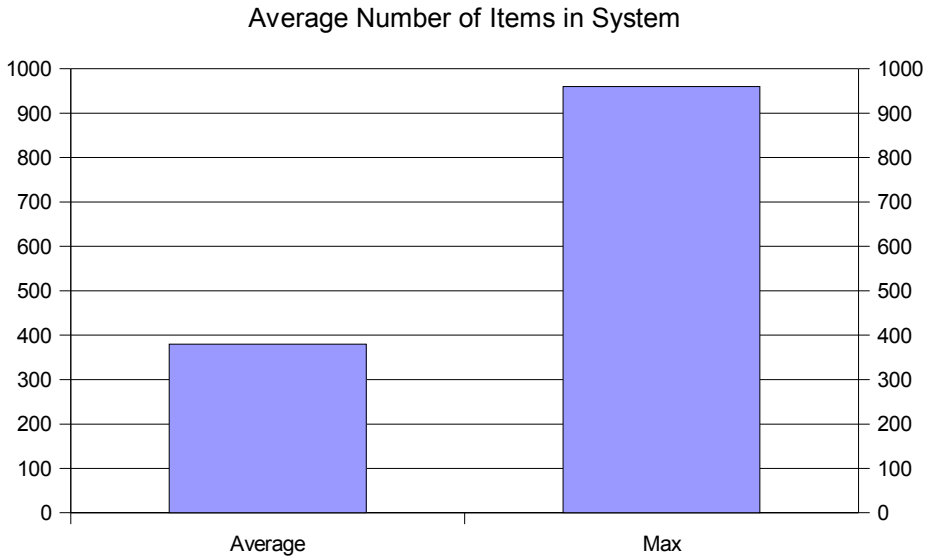

Figure 4.5 Baseline: Average and average maximum number of *RFI*s in system

relatively few resources in the baseline system, the NIS can be attributed primarily to items that are waiting to be serviced. This measure alone gives little insight into the system other than being able to compare overall "efficiency" since it depends on both the rate items enter and leave the system. However, a large NIS value does not necessarily indicate an "inefficient" system, especially considering the scale of a high level model of the intelligence process.

In addition to examining WIP, the average number of items waiting in queues (NQ) provides more insight into where potential bottlenecks may be occurring. For example, in Figure 4.6 the average number of items waiting for collection are more than half the average WIP value. Alternatively, the average NQ for some of the queues is essentially zero which may indicate an over allocation of resources. These NQ values are dependent upon both the arrival of items and the ability of resources to process those items. Moreover, they are aggregate values for multiple types and quantities of resources. As an aggregate measure, it can be easily biased high or low depending on the layout of the underlying arrival and departure rates of items in the queue. The average of the maximum values of NQ for each replication are

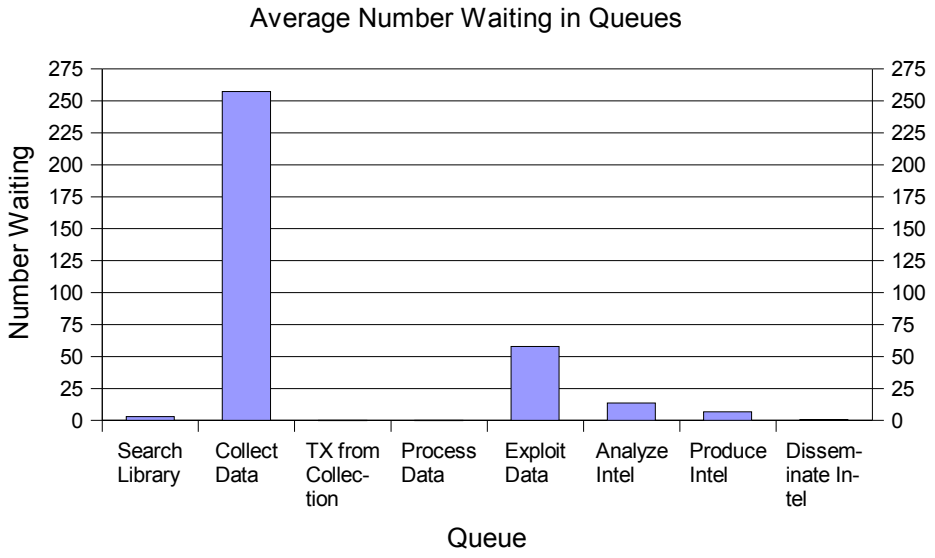

Figure 4.6 Baseline: Average number of *RFI*s waiting in each queue

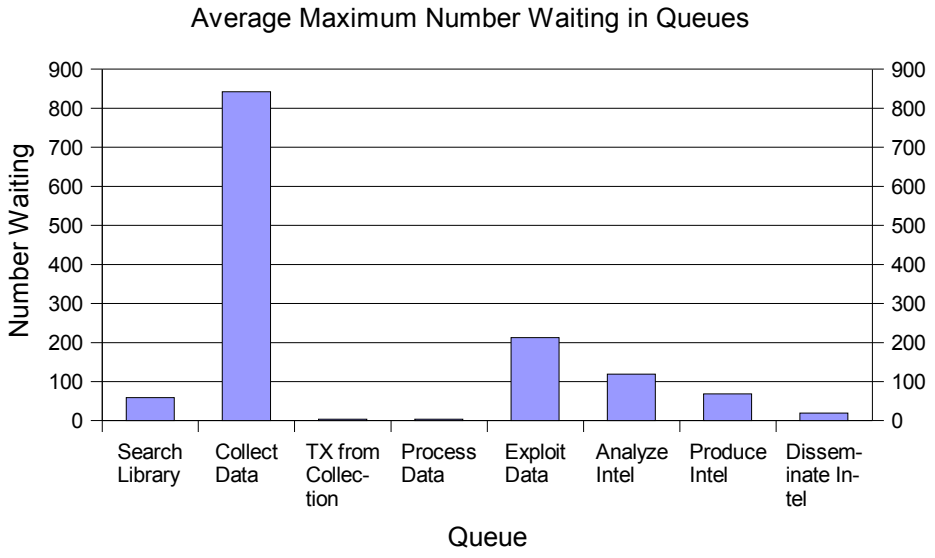

Figure 4.7 Baseline: Average maximum number of *RFI*s waiting in each queue

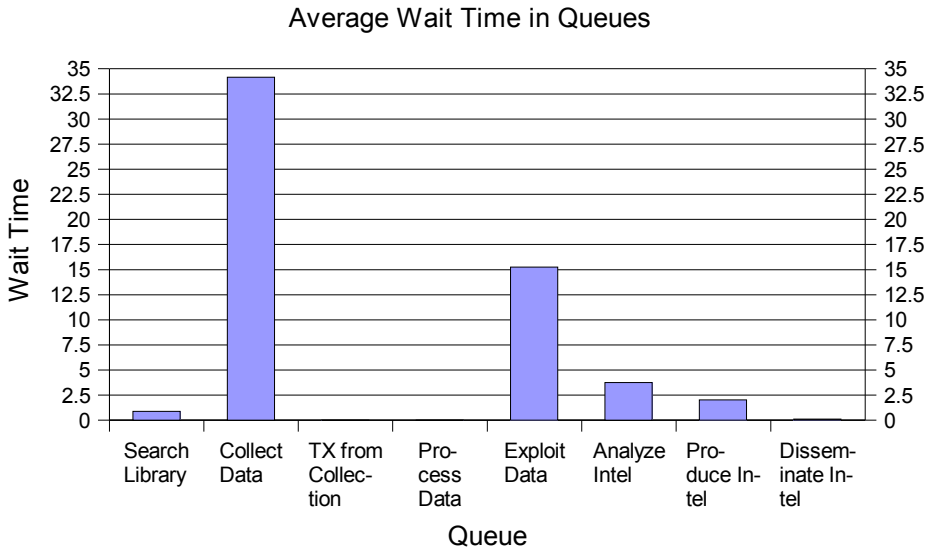

Figure 4.8 Baseline: Average wait time (hours) in each queue

also plotted in Figure 4.7. In general for the baseline system, the maximum values are about twice that of the average values. As with the average maximum WIP this gives no indication of how often the NQ values are near the maximum. It is also important to note that even if the system has large NQ values that it may still be performing well if items are spending a relatively small amount of time in the queue.

Accordingly, the average wait time (WT) in each queue is plotted in Figure 4.8 and the average over the replications of the maximum wait time in each queue in Figure 4.9. It is interesting to note that the relative differences in average WT for the various queues are similar to that of the NQ measures. However, the average maximum WT presents a different picture. Even though the average wait time in the exploitation queue is only about 15 hours, some items spent around 1,700 hours waiting. Based on the input data for the baseline simulation, the longest an item could be in the system and still be timely is 120 hours, so those items were certainly not timely. This emphasizes the point that WT is as critical as service times in a system where timeliness is of great importance. The average maximum WT does not indicate how often items are waiting so long in the queue. As such, the impact of a

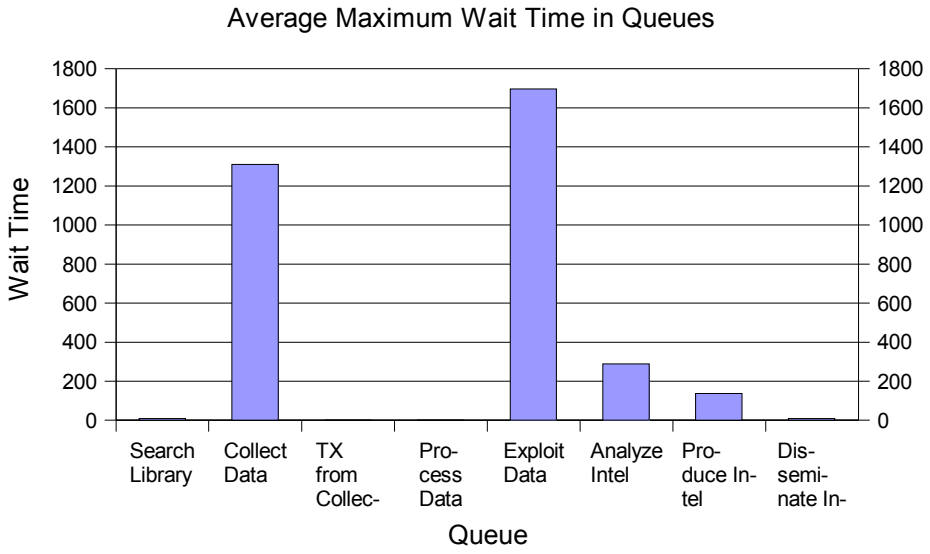

Figure 4.9 Baseline: Average maximum wait time (hours) in each queue

few items that wait extensive times may be minimal. Additional timeliness measures were implemented to help assess such an impact. Alternatively, a different view of WT might provide better insight into the long times.

Partitioning the total WT in system (this includes all queues) by priority provides a clearer picture of what is happening. The *Priority_Global* attribute would be ideal for this decomposition of WT, but its possible values are not constrained as those of the *Priority_User* attribute are. As such, the total WT was partitioned by *Priority_User*. For this study there is no difference since *Priority_Global* is set to equal *Priority_User*. The average total WT by priority is shown in Figure 4.10 and average over the replication maximum WT by priority in Figure 4.11. Given that the queues are generally priority based, the total WT by priority figures appear as one would expect. The highest priority items spend very little time waiting while the low priority items spend a long time waiting. This result in itself aids model verification.

Another standard process model measure is resource utilization. The average resource utilization for the various resources are shown in Figures 4.12 and 4.13.

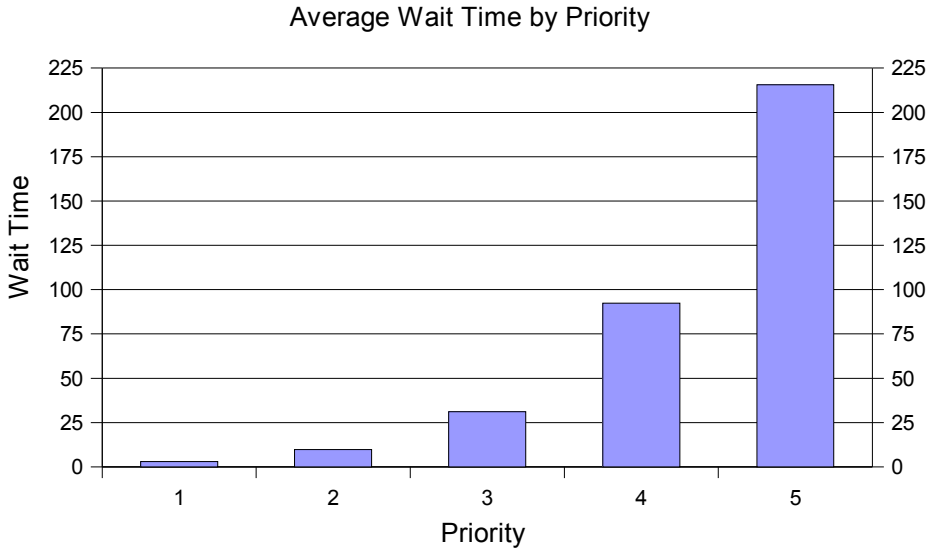

Figure 4.10 Baseline: Average total wait time (hours) by *Priority_User*

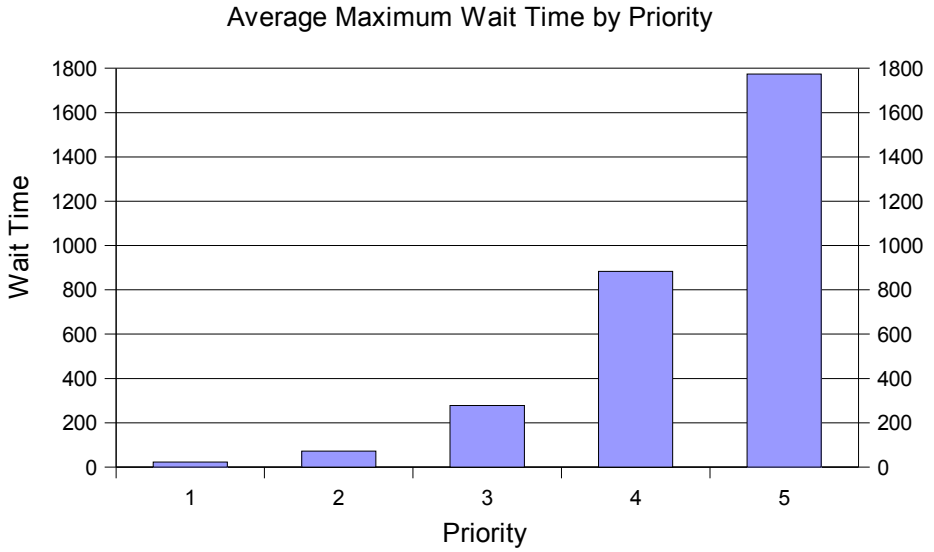

Figure 4.11 Baseline: Average maximum total wait time (hours) by *Priority_User*

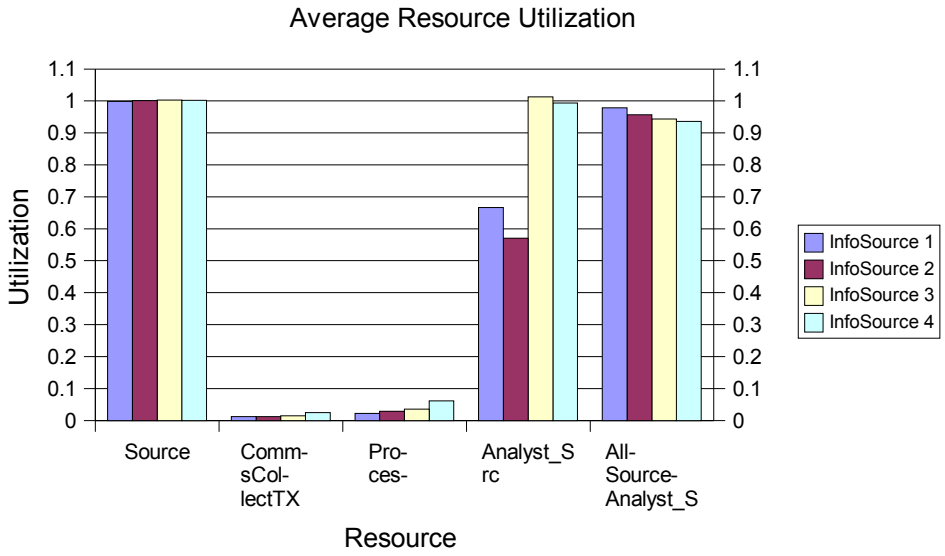

Figure 4.12 Baseline: Average resource utilization

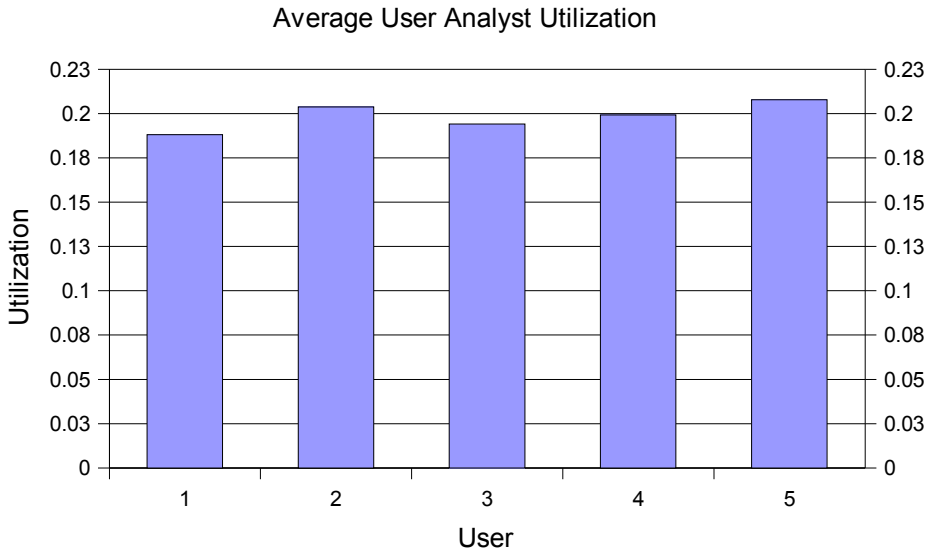

Figure 4.13 Baseline: Average utilization of user analysts

Resource utilization is dependent upon arrival rates, service times, and number of resources available. As such, the large NQ values above are likely to have an associated high resource utilization. However, small NQ values can also have high utilization rates. Unlike the queue statistics above, the resource statistics are broken into categories. Values over one occur because the statistic is more specifically scheduled utilization, some resources are based on schedules, and the schedule policy is set to have resources finish working on the current item at the time they are considered not available. For example, suppose a worker is scheduled to get off work at 1700 hours and is in the middle of working on a report. With the currently selected policy, the worker would work overtime to complete the report before going home resulting in a scheduled utilization greater than one. From the figures, it can be seen that the collection resources are on average completely utilized. This was intentional in the design of the notional data and is also reflected in the large NQ and WT values associated with collection. The utilization of communications resources after collection and processing resources are markedly low. It is unlikely that utilization would be so low in the real system and is an artifact of generating notional data. The breakdown of *Analyst_Src* indicates that for some sources the analysts are over-tasked, while others are under-tasked. The *AllSourceAnalysts* appear to be tasked at an appropriate, or slightly high, level even though they are used for both analysis and production. User analysts, on the other hand, are under utilized, but it must be taken into consideration that user integration was not included for this study. Utilization coupled with the previous measures provide a decent picture of the system performance. The insights gathered may be useful in fixing problems with the system but alone do not provide a solid measure of system performance impact on information needs.

To address the inability of standard process performance measures to adequately provide insight into the effect of system performance on information needs, additional measures were implemented. The additional measures are presented as

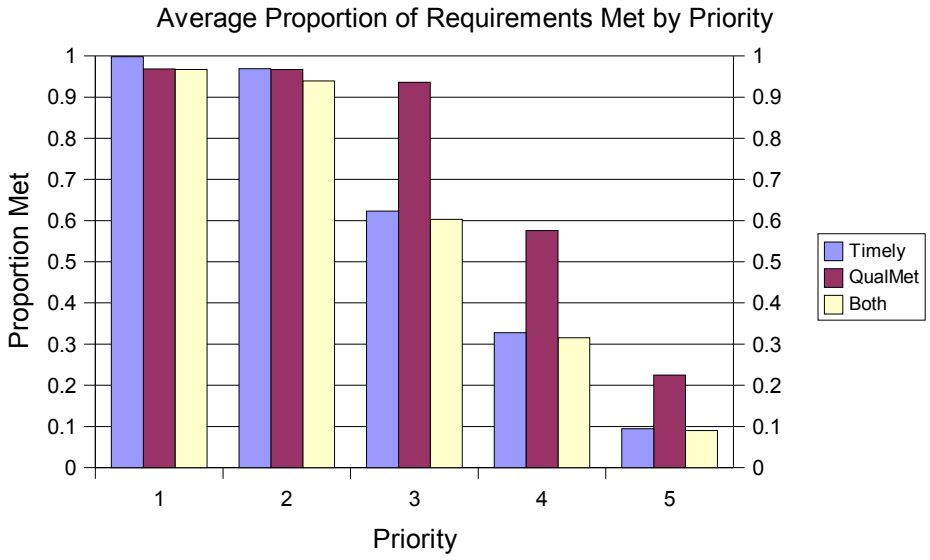

Figure 4.14 Baseline: Average proportion of requirements met by *Priority_User*

proportions of requirements met. These measures coupled with standard process performance measures should provide adequate insight into the effect of system performance on information needs. These measures are decomposed into various categories such as priority, source, and user. The average proportion of requirements met by *Priority_User* is shown in Figure 4.14. As would be expected with a priority based system, the highest priority requirements are on average almost always on time. As the priority goes down, so does the proportion that meet the timeliness requirement. This could be partially inferred from the average WT by priority in Figure 4.10. The proportion that met the quality requirement is also represented. It is interesting to note that for the baseline system that the proportion that met quality requirements also decreases as priority decreases. This may be caused partially by *RFI*s that never get information collected and partially for those that do get collected by not reaping the benefits of potential quality increases in the remainder of the system. The third proportion is that of meeting both the timeliness and quality requirements. This measure will never be higher than the lowest of the timely and quality proportions, but it will not necessarily be equal to the lowest. The fact that the proportion of

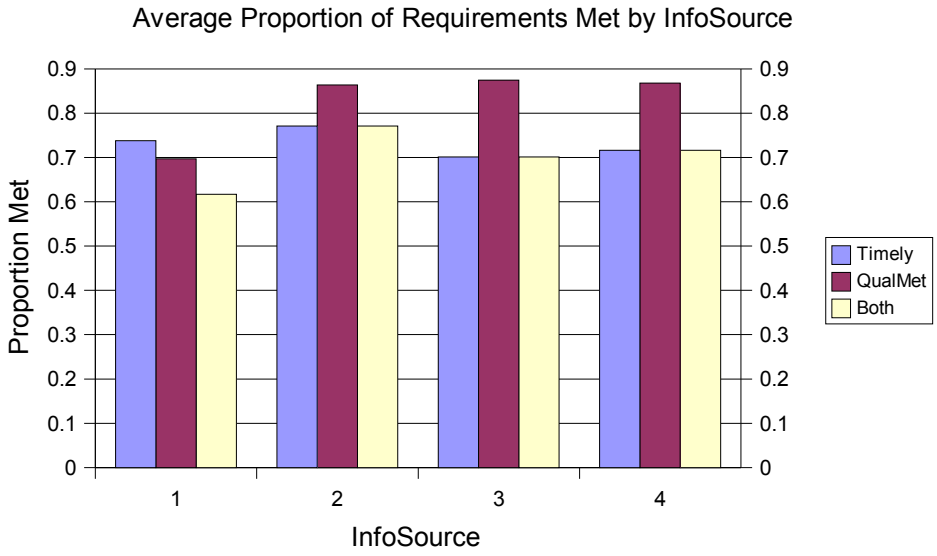

Figure 4.15 Baseline: Average proportion of requirements met by *InfoSource*

meeting both requirements drops as priority decreases aids in model verification. It is interesting to note that most of priority 1 and 2 requests are met with a relatively steep, nearly linear drop as priority continues to decrease.

The proportion of requirements met is also decomposed by *InfoSource* and shown in Figure 4.15. Since much of the required information for running the simulation is dependent upon the *InfoSource* attribute, this plot could be helpful in verification of input information. Note however that a source that provides "perfect" information quality may still not achieve 100% for meeting quality due to low priority items that never get collected. The results for the baseline system indicate that the first source meets a noticeably lower proportion of quality requirements yet a comparable proportion of timeliness requirements. The net result is a lower proportion of both requirements being met than for the other sources. This result is interesting since the major differences between the first and second sources are related to various times to complete tasks, rather than in quality updates within the system. For example, the collection time distributions are quite different even though the have the same means. The first source has exponential times and the

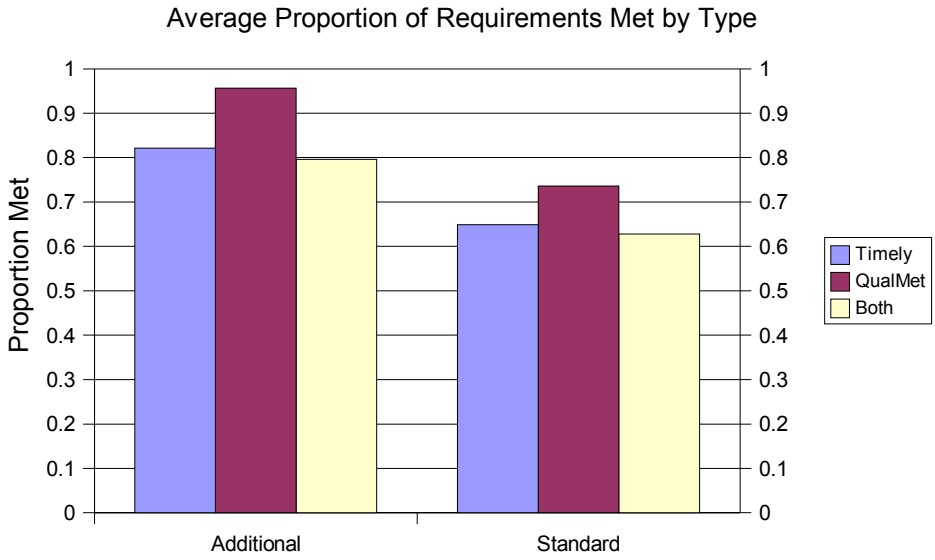

Figure 4.16 Baseline: Average proportion of requirements met, standard vs. additional

second source has constant times. The added variation in collection times is likely having an adverse affect on meeting quality requirements even though on average the same proportion of timeliness requirements are being met. One could do an entire study investigating the effects of varying collection time distributions.

Another way the requirements are partitioned is by the type of requirement, that is, standard or additional. The proportions for meeting timeliness, quality and both are shown in Figure 4.16. From the chart it can be seen that a greater proportion of additional requirements are met than standing requirements. One possible reason for this difference is that standard requirements must go through the collection process whereas some portion of the additional requirements are met without requiring collection.

Lastly, the proportion of requirements met is broken down by user. The results for the baseline system are given in Figure 4.17. The system was set up with five users who had identical processing requirements for items found in a library search not requiring additional collection but differed in the amount of processing required

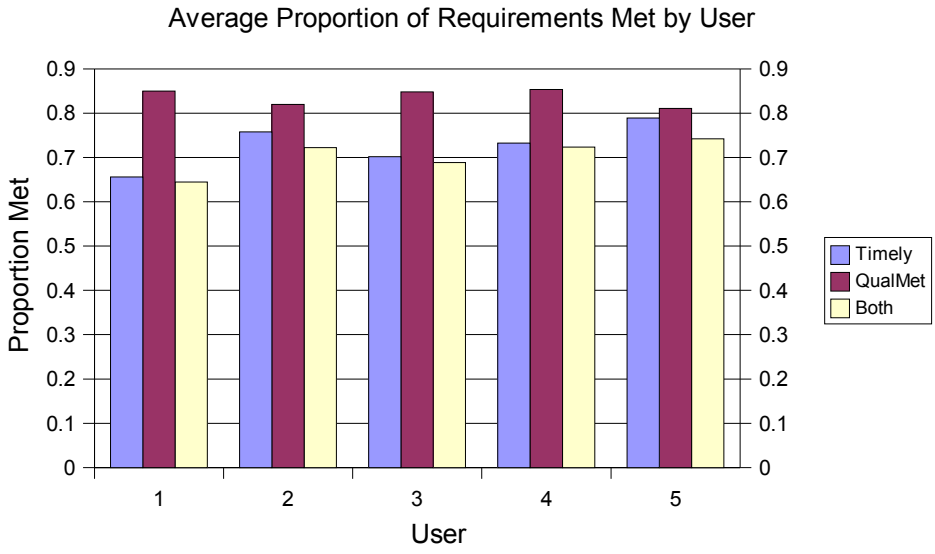

Figure 4.17 Baseline: Average proportion of requirements met by *User*

for any additional requirement requiring collection and standing (i.e. standard) requirements. Specifically, for any requirement needing collection *User* 1 required all steps, *User* 2 never required exploitation, *User* 3 never required analysis, *User* 4 never required analysis or production, and *User* 5 never required exploitation or production. In essence, the users are set to compare various architectures in a saturated system. Rather than examine a TPED like system independent of a TPPU like system, it would be more realistic that some requirements may always need to go through a complete process while other requirements may be able to skip various portions thereby resulting in a hybrid system that cannot be adequately characterized by either alone. For the baseline system, it can be seen that there is some variation in the proportion of quality requirements met is but it is fairly consistent in general. As one might expect, fewer of the timeliness requirements were met for *User* 1 than the other users. With the exception of *User* 1, the differences between the proportions for meeting both requirements may or may not be practically significant since the differences are at most a few percent. Conducting a paired-t test on the difference between *User* 1 and *User* 5 for *Timely*, *QualMet*, and *Both* as given

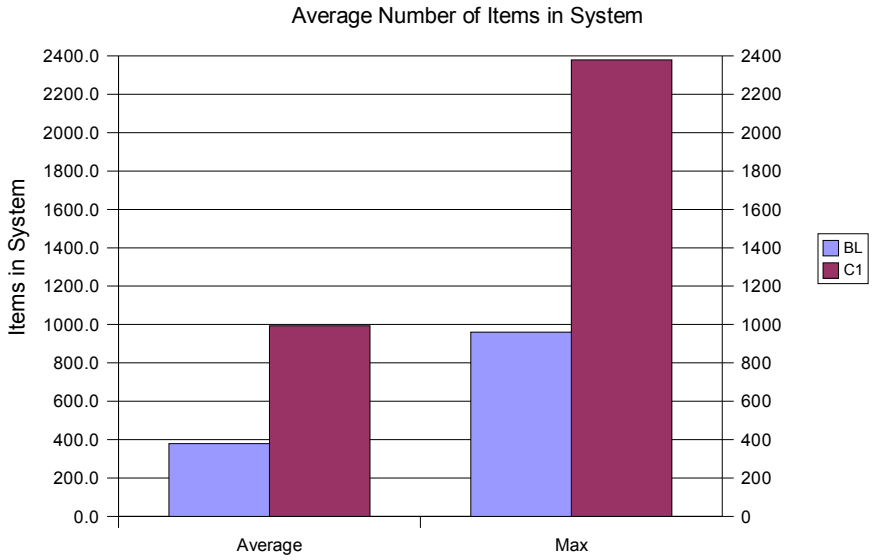

Figure 4.18 BL vs C1: Average and average maximum work in process

in Figure 4.17 results in p-values that are essentially zero (6.19E-38, 8.93E-29, and 1.67E-34, respectively). A selection of p-values for other t-test comparisons of *User* 1 and *User* 5 for this and other cases is given in Appendix F.9.

4.3.2 Case 1: Extreme Timely Threshold

Case 1 involved changing the value of the *Timely_Threshold* variable from 0 to 48 hours. This has the same effect as extending all *TimeR* values by 48 hours since the threshold value is used whenever a timeliness check is done. It is unlikely that such extensions for all requests would occur in real life, but this case is still useful for testing the system. Rather than restate all of the statistics reported for the baseline system in Section 4.3.1, only a selected portion that provide interesting results will be reported here. Additional details for Case 1 results can be found in Appendix F.2.

The first statistic of interest is the average number of items in system at any time or WIP. A plot of the average and average maximum values for the Baseline (BL) and Case 1 (C1) results are shown in Figure 4.18. As can be seen in the

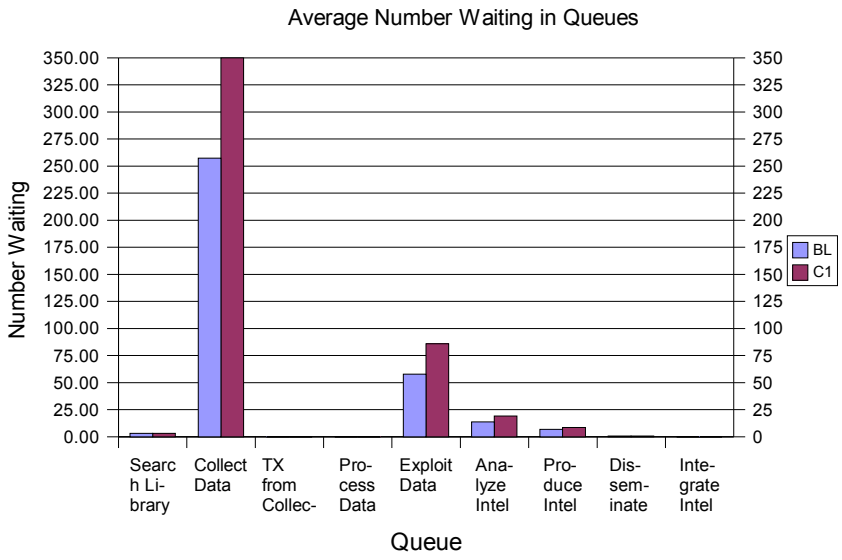

Figure 4.19 BL vs C1: Average number waiting in queues

figure, the WIP has doubled for both the average and the average maximum. This result is not unexpected as items are allowed to remain in the system longer, the number waiting would be expected to increase. The increase in the number of *RFI*s waiting in the various queues can be seen in Figure 4.19. Interestingly enough, there is no practically significant difference in the resource utilization. This is partially reflected in the fact that wait times for lower priority items has also increased (see Figure 4.20). Even with the 48 hour extension, the increased wait times for the lower priority items significantly exceed the required time. Since the *RFI*s are waiting so long before being serviced, they are thrown out of the system and do not cause increased resource utilization.

If we examine the proportion of requirements met, the added time for meeting requirements provides little benefit. The proportion of meeting both quality and timeliness requirements for each user is shown in Figure 4.21. Only a marginal increase is seen which likely has no practical significance. The results for partitioning the proportion of requirements met by *Priority_User* are shown in Figure 4.22 and indicate an interesting change from the baseline results. There is no practically

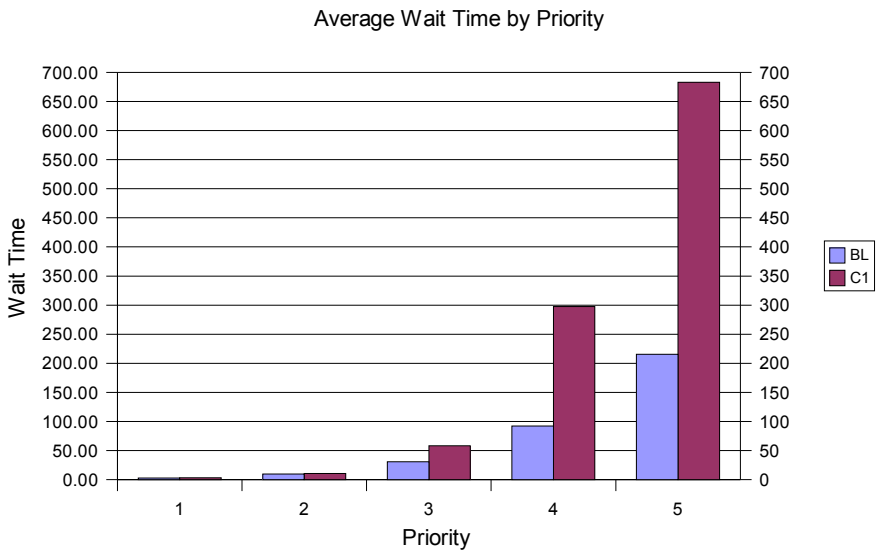

Figure 4.20 BL vs C1: Average total wait time (hours) by *Priority_User*

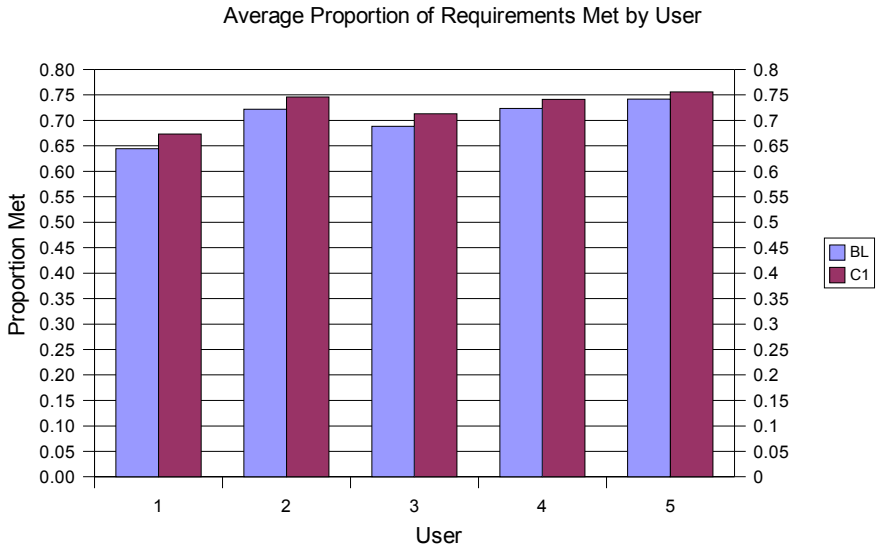

Figure 4.21 BL vs C1: Average proportion of requirements met by *User*

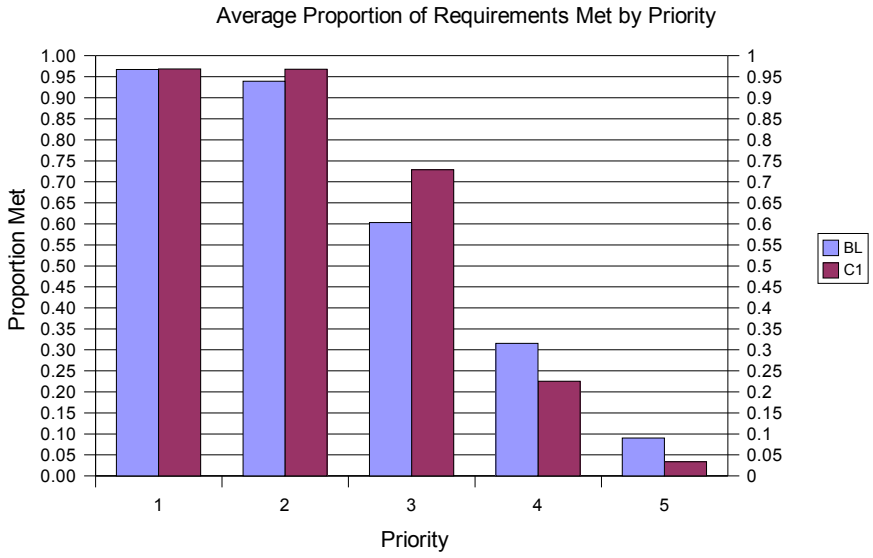

Figure 4.22 BL vs C1: Average proportion of requirements met by *Priority_User*

significant change in the proportion of priority 1 and 2 requirements met, but there is an increase in the proportion of priority 3 requirements met. This increase is offset by a decrease in the proportion of priority 4 and 5 requirements met.

Overall, the addition of 48 hours to the threshold for determining timeliness had a negative impact on the system. The only potential benefit was an increase in the proportion of priority 3 requirements met; however, this comes at a great cost. The disadvantages of increased WIP and wait times as well as meeting essentially none of the priority 5 requirements likely outweigh such a minor benefit. From the perspective of the users there is no real benefit, especially when one considers the fact that requirements may be met up to two days later than in the baseline system.

4.3.3 Case 2: Increased Timely Threshold

Case 2 involved changing the value of the *Timely_Threshold* variable from 0 to 12 hours. Similar to Case 1, this has the same effect as extending all *TimeR* values by 12 hours since the threshold value is used whenever a timeliness check is done. Although it is unlikely that such extensions would occur for all items in real life, it

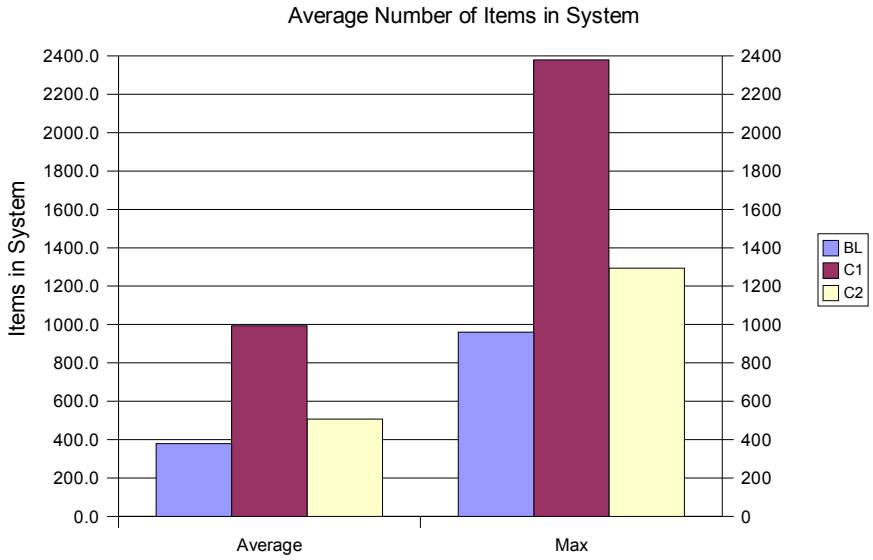

Figure 4.23 BL vs C2: Average and average maximum work in process

is more reasonable that Case 1. A 12 hour extension would be analogous to stating that if an item was due by close of business one day then it is OK to send it not later than start of business the next day. Rather than restate all of the statistics reported for the baseline system in Section 4.3.1, only the results reported for Case 1 will be re-examined here. Additional details for Case 2 results can be found in Appendix F.3.

The first statistic revisited is the average number of items in system at any time or WIP. A plot of the average and average maximum values for the BL, C1, and Case 2 (C2) results are shown in Figure 4.23. As might be expected, the WIP for C2 is between that of the BL and C1. The C2 increase over the BL results appears to be a little less than 1/4 of the C1 increase, so the change in WIP seems to be somewhat proportional for the region of the threshold examined. In addition to the NIS results, the results for number waiting in queues might also be expected to behave similarly. However, the increase in the number of *RFI*s waiting in the various queues for C2 (see Figure 4.24) appear to be at essentially the same levels as C1. As with C1, there is no practically significant difference in the resource utilization between the BL and

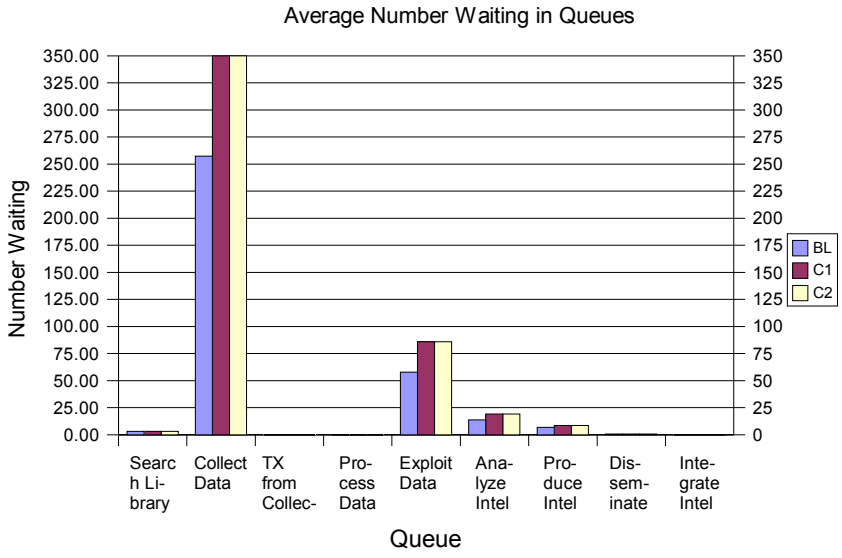

Figure 4.24 BL vs C2: Average number waiting in queues

C2. The lack of a change in the number waiting in queues is particularly interesting given that the wait times for lower priority items have decreased compared to C1 (see Figure 4.25). Unlike the WIP values, the increase in wait times for C1 and C2 do not appear to be proportional to the change in timeliness threshold. Even so, the large average wait times for priority 4 and 5 items for C2 give some indication that there is likely to be a relatively low proportion of requirements met for those priorities.

Examining the proportion of requirements met by *Priority_User* in Figure 4.26 indicate that the C2 results have a similar pattern as the C1 results. As expected, the C2 results fall between the BL and C1 results. It appears that some proportion of the meeting the priority 3 requirements has been redistributed to the lower priority items. Additionally, there is still no practically significant difference between the priority 1 and 2 proportions. Re-examining the proportion of requirements met for each *User* (see Figure 4.27) indicates that there is still no practically significant difference in the proportions of requirements met.

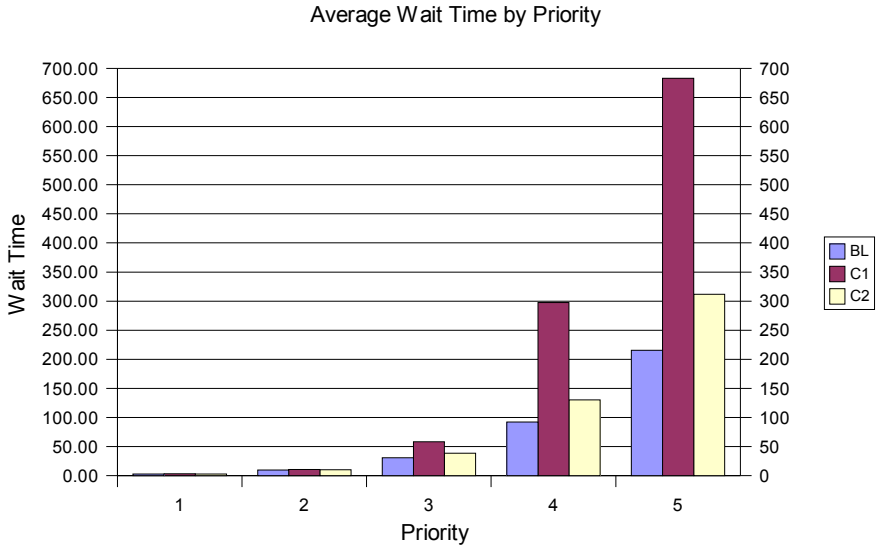

Figure 4.25 BL vs C2: Average total wait time (hours) by *Priority_User*

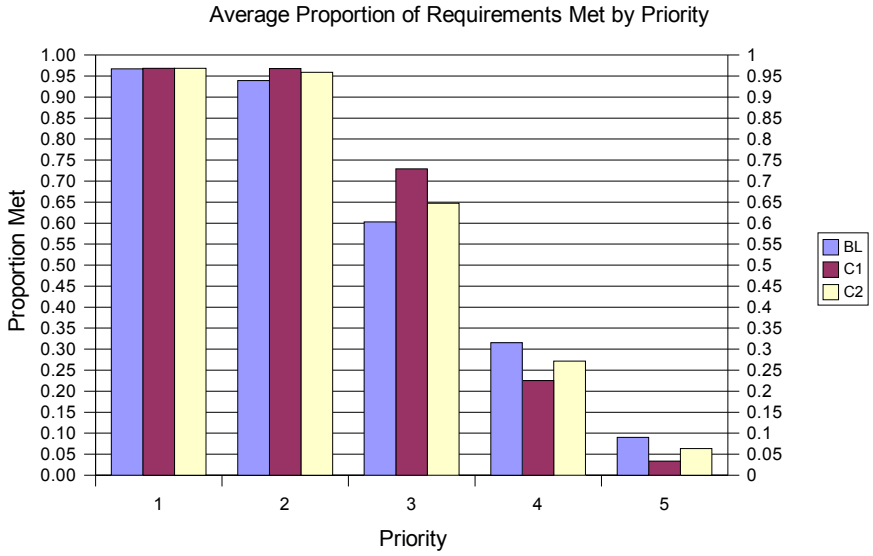

Figure 4.26 BL vs C2: Average proportion of requirements met by *Priority_User*

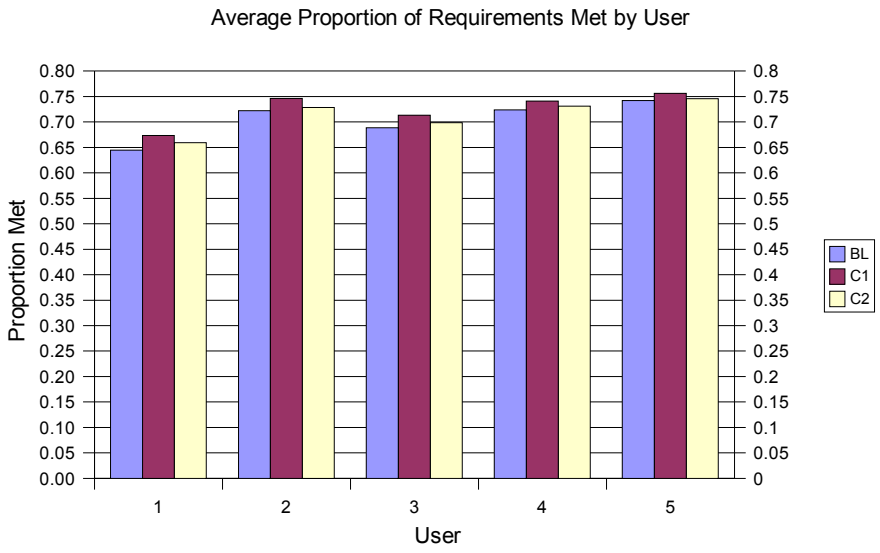

Figure 4.27 BL vs C2: Average proportion of requirements met by *User*

Overall, the addition of 12 hours to the threshold rather than 48 hours for determining timeliness had a reduced impact on the system. There is only a minor potential benefit in an increase in the proportion of priority 3 requirements met; however, this still comes at some cost. The same disadvantages that were present for C1 are still present for C2, except at reduced levels. From the perspective of the users there is no real benefit, especially when one considers the fact that requirements may be met later than in the baseline system. On the other hand, if users can live with later results, they may receive additional middle priority items but only at the cost of lower priority items. That may be an acceptable trade-off for the user but only if the remainder of the system can feasibly handle the increased number of items in the system.

4.3.4 Cases 3 and 4: Increased Quality Threshold

Case 3 involved changing the value of the *QualMet_Threshold* variable from 0 to 3 and Case 4 involved changing the *QualMet_Threshold* from 0 to 1. This should have a similar effect as either decreasing *QualR* or increasing *QualA* by 3 or 1 for

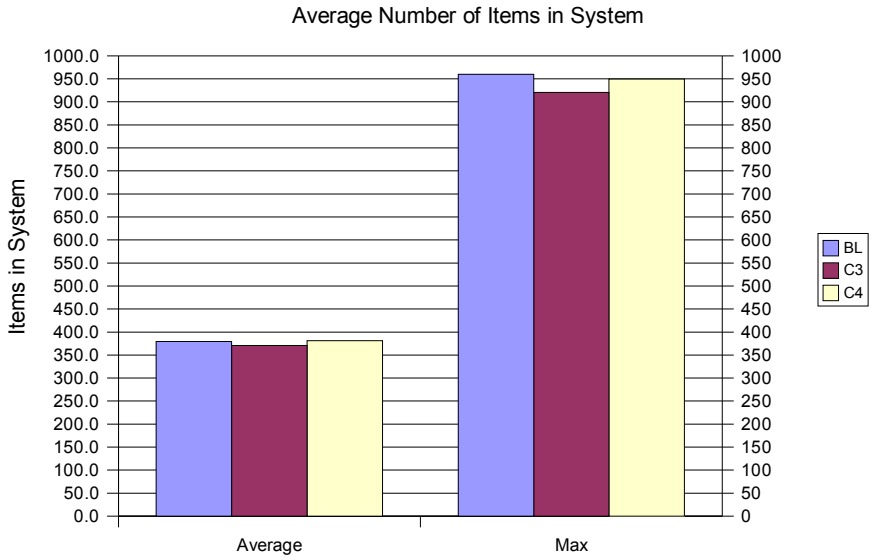

Figure 4.28 BL vs C3, C4: Average and average maximum work in process

Case 3 and 4, respectively. However, changing the threshold value does not actually alter the *QualR* or *QualA* attributes and would not be expected to impact timeliness as much as altering those values since Equation 3.3 is only used when assessing library search results and when assessing items as they leave the system. As with Case 1, this change for Case 3 is unrealistic but is useful for stressing the system. The change for Case 4 may be more realistic for selected items, but is unlikely to be as useful for studies as altering the *QualR* or *QualA* attributes. A value of 0.5 for *QualMet_Threshold* would be similar to rounding the *QualA* attribute to the nearest integer prior to comparing it with *QualR* which may be the largest reasonable value for the threshold unless one is modeling user overstatement of requirements. Only a selected portion of the statistics reported for the baseline system in Section 4.3.1 that provide interesting results will be reported. Additional details for Case 3 and Case 4 results can be found in Appendices F.4 and F.5, respectively.

The average number of *RFI*s in the system for the Baseline (BL), Case 3 (C3), and Case 4 (C4) is shown in Figure 4.28. As expected, there is minimal impact and to the WIP for either C3 or C4 and no practical difference. The negligible

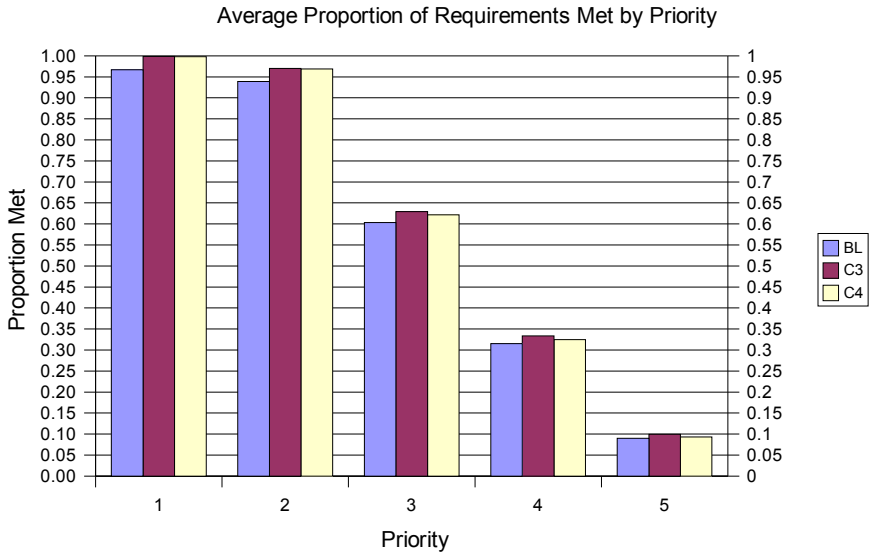

Figure 4.29 BL vs C3, C4: Average proportion of requirements met by *Priority_User*

apparent difference is likely due to a small increase in the proportion of items that met requirements in a library search; however, no statistics were collected to specifically test this.

Examining the impact of increasing *QualMet_Threshold* on the proportion of requirements met by *Priority_User* (see Figure 4.29) indicates that there is only minimal and practically insignificant increase for both C3 and C4. The results of partitioning the proportion of requirements met by *InfoSource* and *User* (Figures 4.30 and 4.31, respectively) indicate a similar phenomenon.Even though the is no practical difference between C3 and C4 for *User* 1, there is a statistically significant difference; a paired-t comparison results in a p-value of about 0.0042. Similarly, the difference between C3 and C4 for *User* 5 is statistically but not practically different; a paired-t test results in a p-value of about 0.0385. Additional results for selected paired-t comparisons are given in Appendix F.9.

The only exception is an increase in the proportion of requirements met for *InfoSource* 1 with no practical difference between C3 and C4. A closer examination

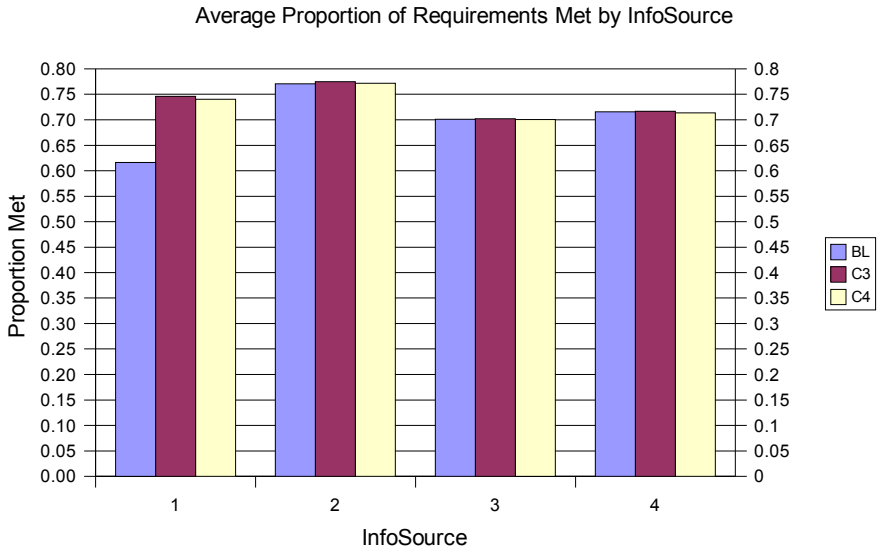

Figure 4.30 BL vs C3, C4: Average proportion of requirements met by *InfoSource*

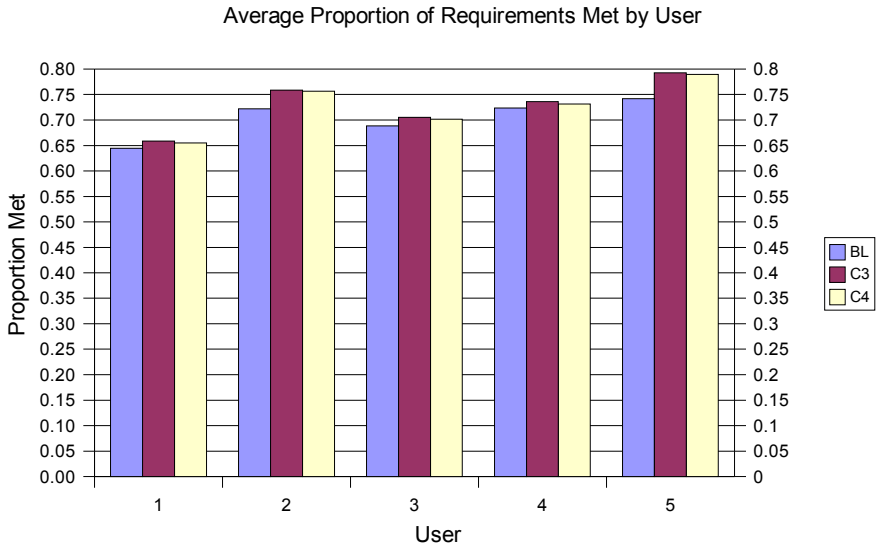

Figure 4.31 BL vs C3, C4: Average proportion of requirements met by *User*

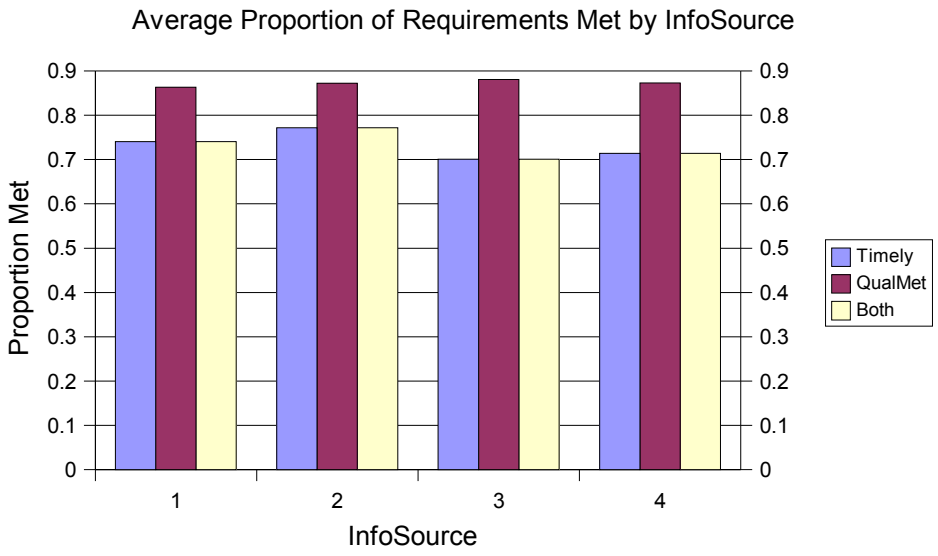

Figure 4.32 Case 4: Average proportion of requirements met by *InfoSource*

of some of C4 results will provide a better understanding of why a greater increase in the proportion of requirements met is not realized.

The breakdown of the C4 requirements by timeliness and quality shown in Figure 4.32 reveals the lack of a significant increase in the proportion of requirements met. As it can be seen, quality has been discounted enough that timeliness has become the limiting factor for meeting both requirements. Re-examination of the figures in Section 4.3.1 will reveal that the proportion for meeting the timeliness requirement is in most cases close to that for meeting both requirements resulting in no practical improvement.

4.3.5 Case 5: Increased Number of Additional Requirements

Case 5 (C5) involved increasing the number of additional requirements by about 50%. Rather than increasing the rate at at which batch arrivals occur, the number of requests per arrival was changed from a $DISC\,(.5, 10, 1, 15)$ distribution to a $DISC\,(.5, 15, 1, 23)$ distribution for all users, where the $DISC\,(\cdot)$ function describes a discrete cumulative distribution function. This has the effect of modeling

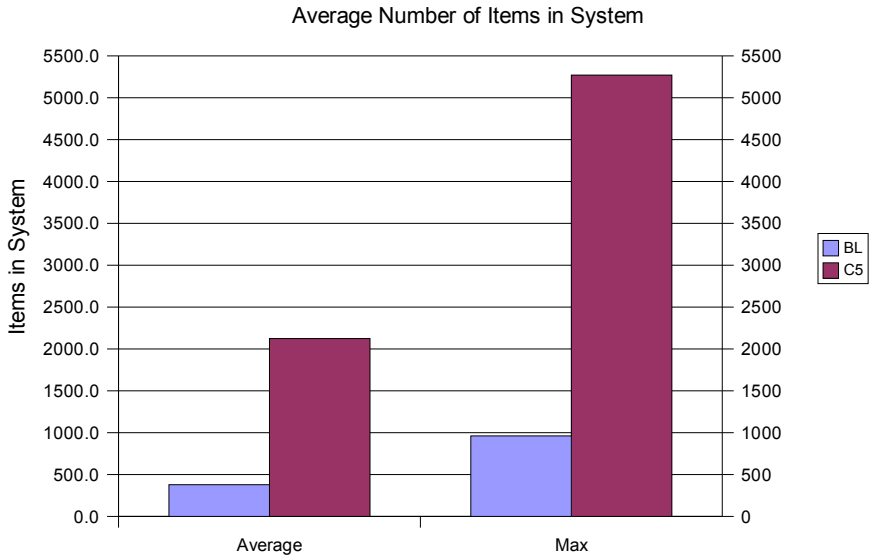

Figure 4.33 BL vs C5: Average and average maximum work in process

a sustained surge in additional requirements from all users. The surge is sustained because the distribution of requests per arrival and time between arrivals were set to remain the same throughout the entire simulation. As with the previous cases, only a portion of the statistics reported for the baseline system in Section 4.3.1 that provide interesting results will be revisited in this section. Additional details for Case 5 results can be found in Appendix F.6.

Since the number of entities arriving in the system has increased, it should come as no surprise that the average number of entities in the system has also increased. The average and average maximum WIP shown in Figure 4.33 indicate that there was more than a fourfold increase on average. Such a large increase gives some indication that there is a limiting factor or bottleneck in the model. Examining the average number of items waiting in various queues (see Figure 4.34) indicates that the limiting factor is collection. The large increase is likely due to the fact that the collection resources were already tasked at capacity prior to increasing the number of requirements (see Figure 4.12). Although it is interesting that the size of the exploitation queue appears to be slightly smaller, the difference is likely not of any

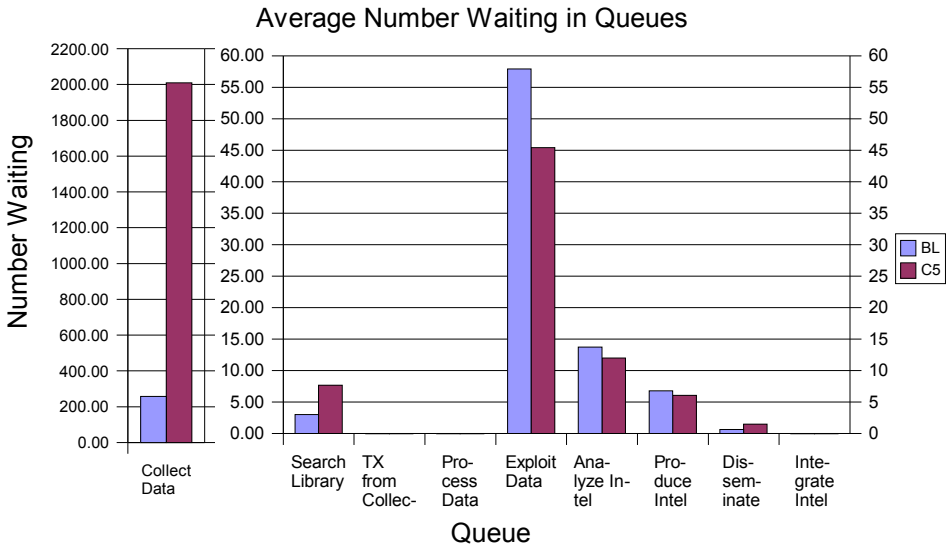

Figure 4.34 BL vs C5: Average number of *RFI*s waiting in queues

practical significance. Such a large average queue size and average WIP indicate that *RFI*s will spend longer in the system. Examining the average wait time by priority (see Figure 4.35) shows that the low priority items are spending tremendously more time waiting on average for C5 than for the BL. With such large average wait times, one would hardly expect many of the low priority requirements to be met. In fact, examining the average proportion of requirements met by *Priority_User* (see Figure 4.36) confirms the expected result. Furthermore, this figure highlights an interesting, but not unreasonable result. Since the system has become overloaded with requests, it can only maintain the proportion of requirements met for the top priority items. The proportion of requirements met for all other priorities has dropped significantly. Since the users have flooded the system with additional requests, and the collection resources cannot keep up, a lower proportion of their requirements are being met (see Figure 4.37). The reduction in the proportion of requirements met by *InfoSource* is also exhibited in Figure 4.38.

Overall, the result of a 50% increase in additional user requirements overloaded the system. The overload system was only able to keep up with the highest priority

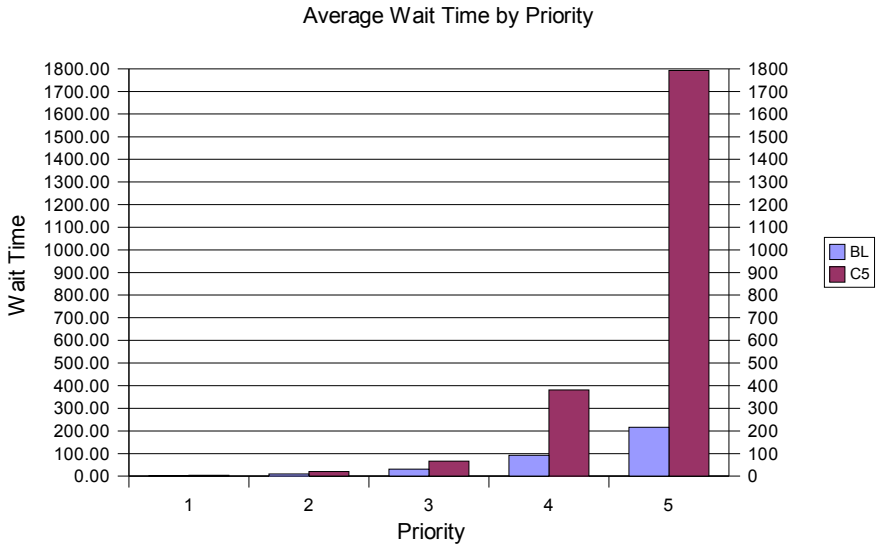

Figure 4.35 BL vs C5: Average total wait time by *Priority_User*

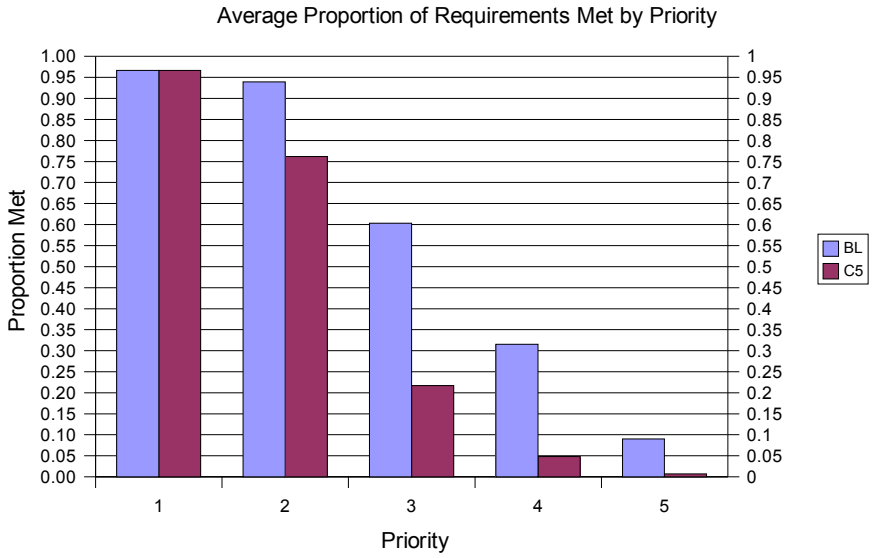

Figure 4.36 BL vs C5: Average proportion of requirements met by *Priority_User*

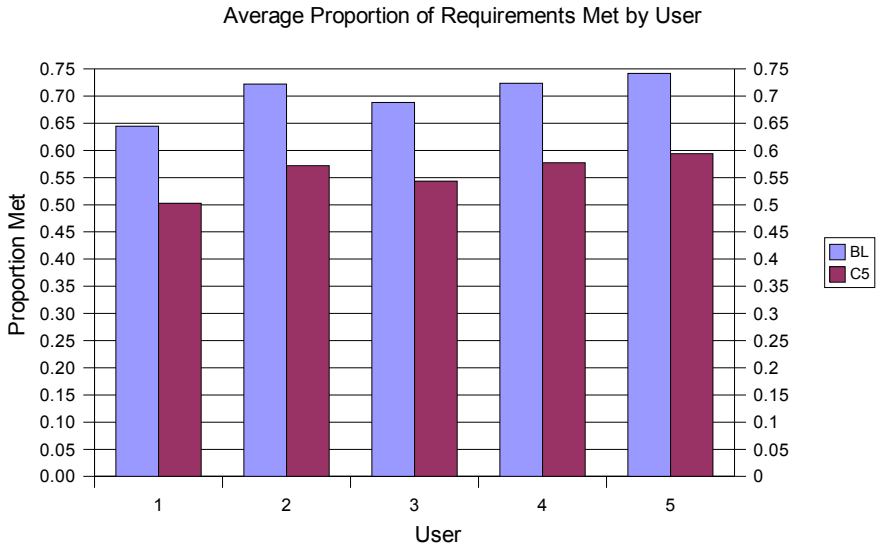

Figure 4.37 BL vs C5: Average proportion of requirements met by *User*

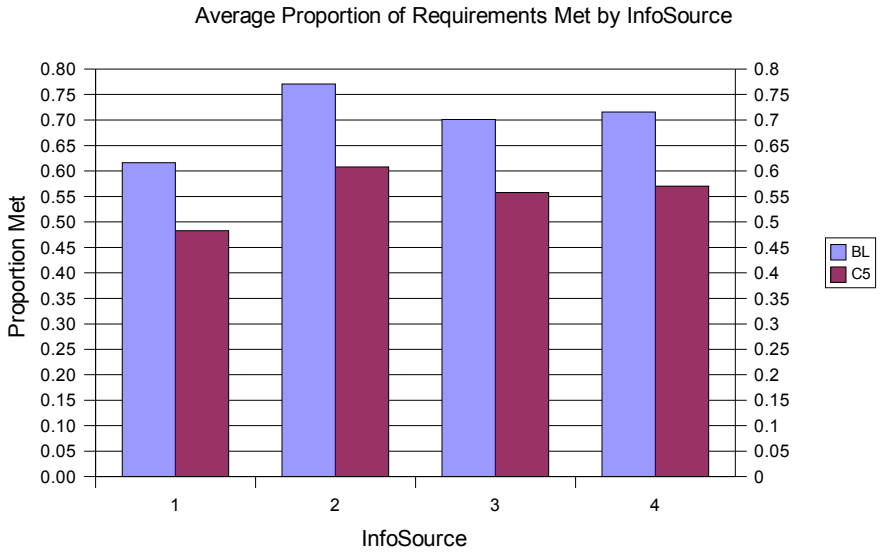

Figure 4.38 BL vs C5: Average proportion of requirements met by *InfoSource*

items. Furthermore, the general effect was that a smaller proportion of requests were satisfied. One could also examine the raw number of requirements that were satisfied (this was not done), but that might give an overly optimistic view of the overloaded system. This or a similar increase in requirements generated can be used to find bottlenecks or other weak points in the system if none are initially apparent from a given baseline.

4.3.6 Case 6: Increased Exploitation Times

Case 6 (C6) involved increasing all exploitation times by 50%. Rather than alter the distributions as was done for Case 5, a multiplicative factor of 1.5 was added to the block that assigns the delay for exploitation. This has the effect of increasing the delay for any *RFI* that undergoes exploitation. Partially done to stress the system, this sort of adjustment could be examined if there were reason to believe that the exploitation times were underestimated. However, such a large increase can artificially induce a bottleneck in the system as was the intent with this case. Similar to previous cases, only interesting results will be re-examined here rather than the complete set of statistics reported for the baseline system in Section 4.3.1. Additional details for Case 6 results can be found in Appendix F.7.

As expected, the increased time for exploitation has increased the total number of items in the system (see Figure 4.39) The majority of this increase is due to the number of items waiting in the exploitation queue (see Figure 4.40). As with C5, the large total number of items in system brings into question whether or not the system is any longer in steady state at the end of the 4 year replications or if a steady state any longer exists. A more detailed analysis of WIP would need to be accomplished in order to make this determination. Along with the larger number of items in queue and in system, we would expect higher wait times. Examining the wait times by priority shows that the lower priority items are indeed waiting longer on average. However, it is interesting to note that in Figure 4.41 that priority 4 items are waiting

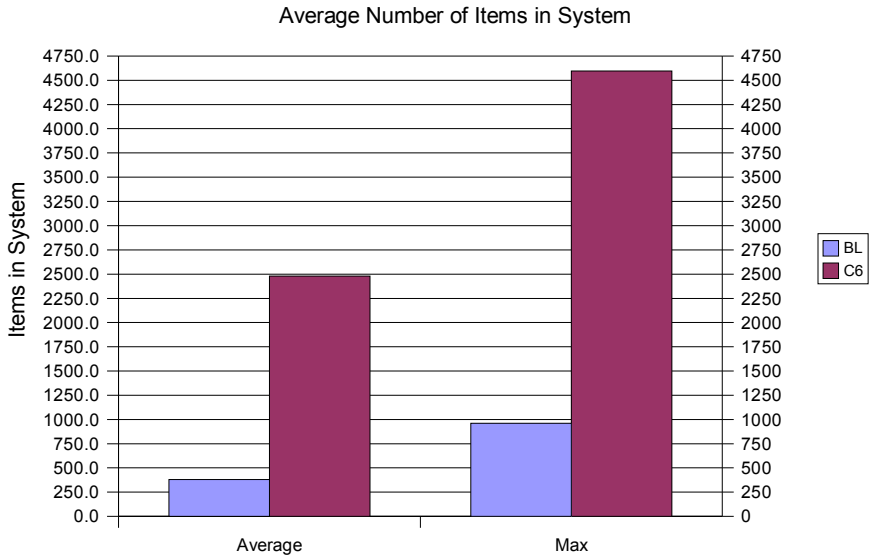

Figure 4.39 BL vs C6: Average and average maximum work in process

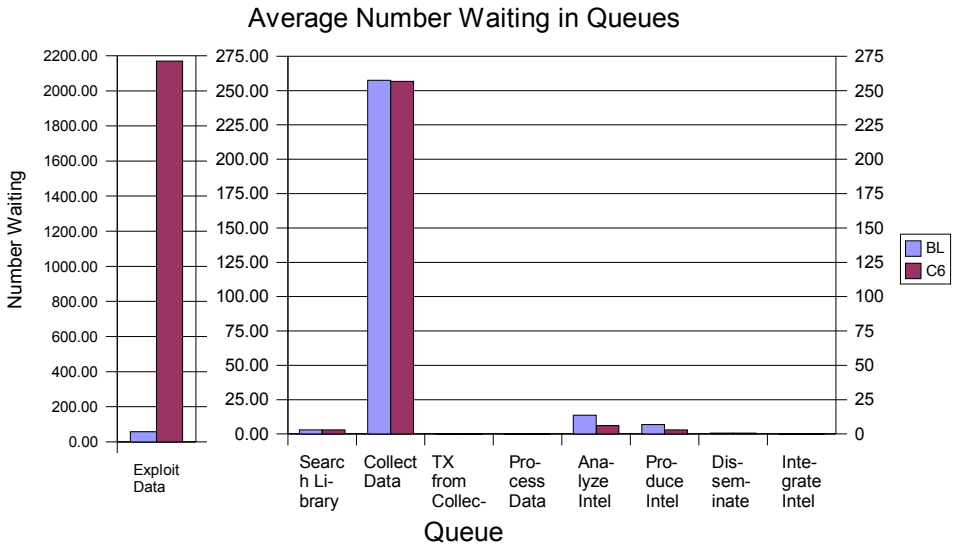

Figure 4.40 BL vs C6: Average number of *RFI*s waiting in queues

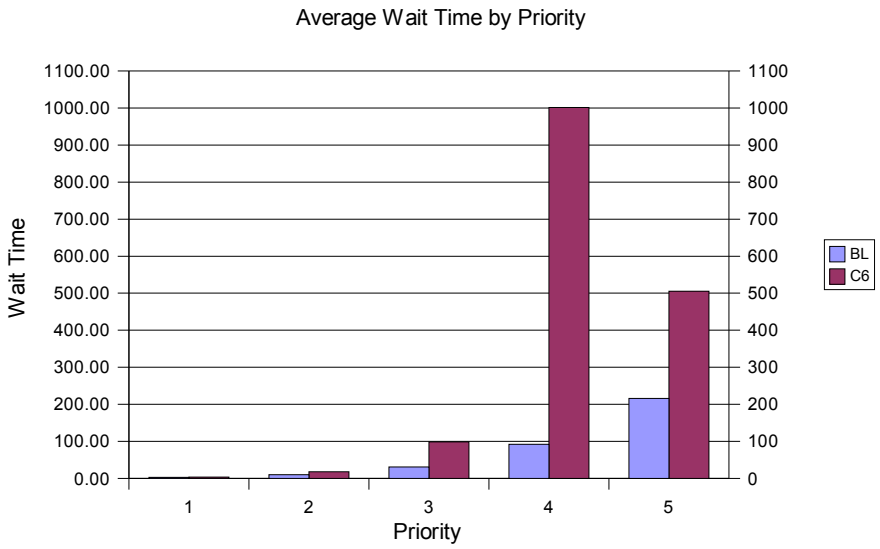

Figure 4.41 BL vs C6: Average total wait time (hours) by *Priority_User*

longer than priority 5 items on average. The average maximum for priority 5 items is nonetheless higher than the average maximum for priority 4 items. One possible explanation for this is that priority 5 items that are waiting in the exploitation queue may never get serviced and thereby will never exit the system, thereby biasing the results. Furthermore, the average proportion of requirements met by priority follows the expected trend of a lower proportion of priority 5 items being met than priority 4 items (see Figure 4.42). Similar to C5, the system can only keep up with the top priority items. Priority 2 and 3 items have suffered a practically significant decrease in proportion of requirements met, whereas the proportion of priority 4 and 5 items did not change in a practically significant manner. Breaking the proportion of requirements met down by *InfoSource* (see Figure 4.43) provides some interesting results. From the figure it can be seen that there is no practically significant difference in the proportion of requirements met for sources 1 and 2, but sources 3 and 4 have incurred a significant decrease. The reason for what might seem like a discrepancy can be found by examining the utilization of the *Analyst_Src* resources involved in performing exploitation. From the chart in Figure 4.44 it can be seen that there

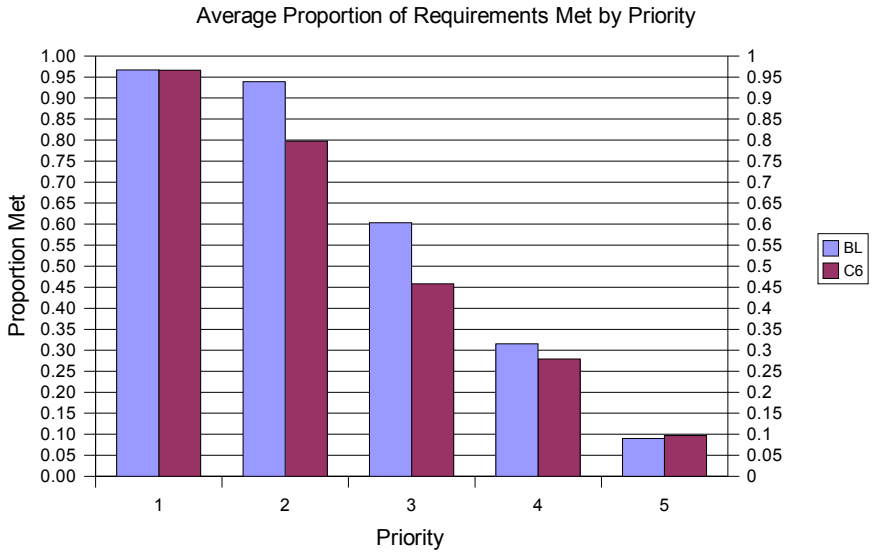

Figure 4.42 BL vs C6: Average proportion of requirements met by *Priority_User*

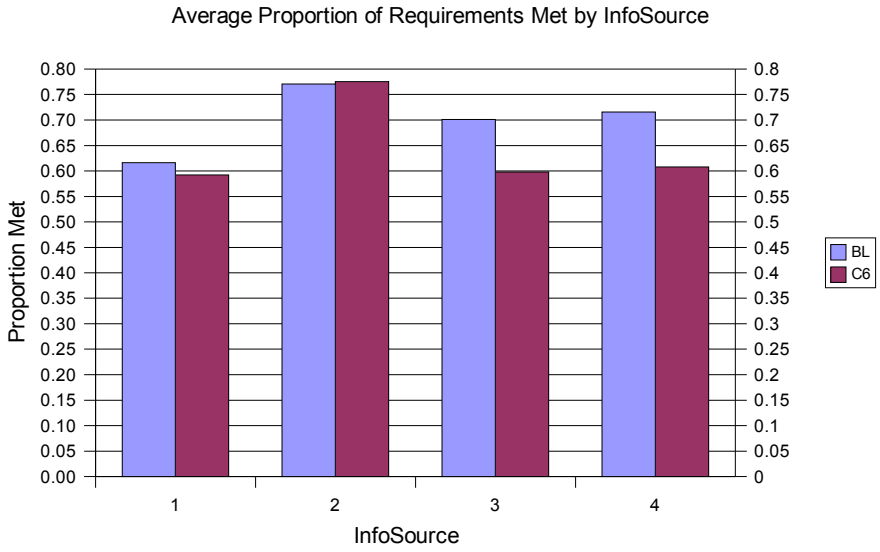

Figure 4.43 BL vs C6: Average proportion of requirements met by *InfoSource*

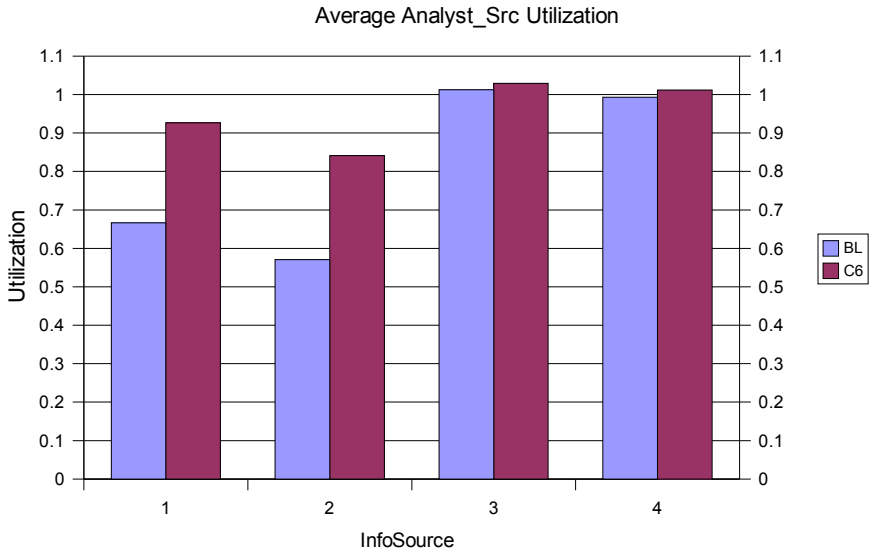

Figure 4.44 BL vs C6: Average utilization of *Analyst_Src* resources

was additional capacity for sources 1 and 2 whereas the resources were already at maximum utilization for sources 3 and 4. From the user perspective, we see what may be an initially surprising picture in Figure 4.45. However, as noted Section 4.3.1 discussing the BL system that users 2 and 5 do not require exploitation except for possibly a small portion of additional requirements. That distinction makes it clear as to why users 1, 3, and 4 have seen a practically significant reduction in their proportion of requirements met while users 2 and 5 have seen no significant change.

Overall, the 50% increase in delay times for exploitation successfully introduced a bottleneck into the system, albeit unrealistic. The end result of a higher WIP, longer wait times, and lower proportion of requirements met is consistent with expected results. The only possibly surprising result was the average wait times for priority 4 *RFI*s exceeded the average wait times for priority 5 *RFI*s. One possible, but not substantiated, explanation was that priority 5 *RFI*s are never leaving the exploitation queue, thereby biasing the result.

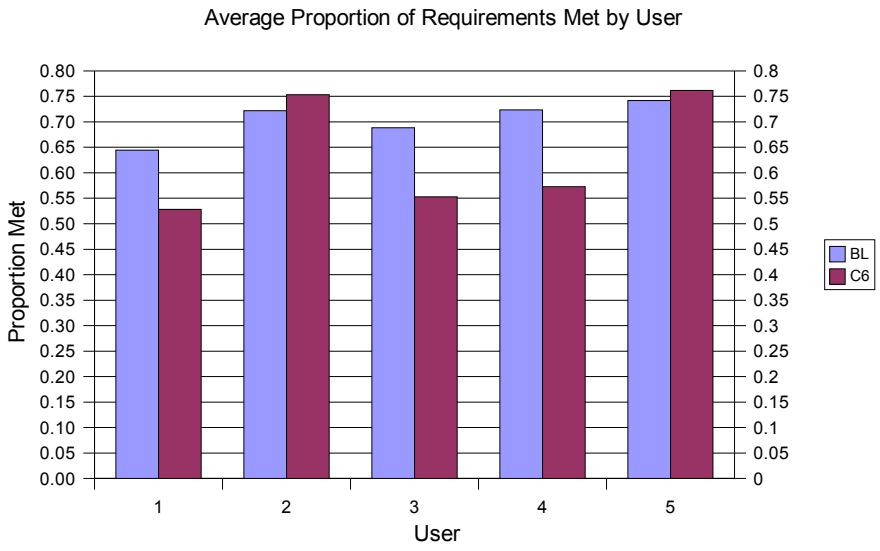

Figure 4.45 BL vs C6: Average proportion of requirements met by *User*

4.3.7 Case 7: Increased Analysis Times

Case 7 was created by increasing all analysis times by 50%. This was accomplished in a manner similar to Case 6. A simple multiplicative factor of 1.5 was added to the block that assigns the delay for analysis. This indiscriminately increases the analysis times for all *RFI*s that require analysis. The primary focus is again to stress the system and introduce an artificial bottleneck. Only a portion of the statistics reported for the baseline system in Section 4.3.1 that provide interesting results will be revisited here. Additional details for Case 7 results can be found in Appendix F.8.

Since the intent of this case was to introduce a bottleneck into the system, the first statistic examined was the average WIP. As can be seen in Figure 4.46 there is an expected increase in the both the average and average maximum number of items in system. Even though there appears to be an increase, the number of items in the C7 system are only a little less than twice that of the BL system. Focusing in on the average number of items in the various queues reveals that there is an expected backlog of items in the analysis queue (see Figure 4.47). Corresponding

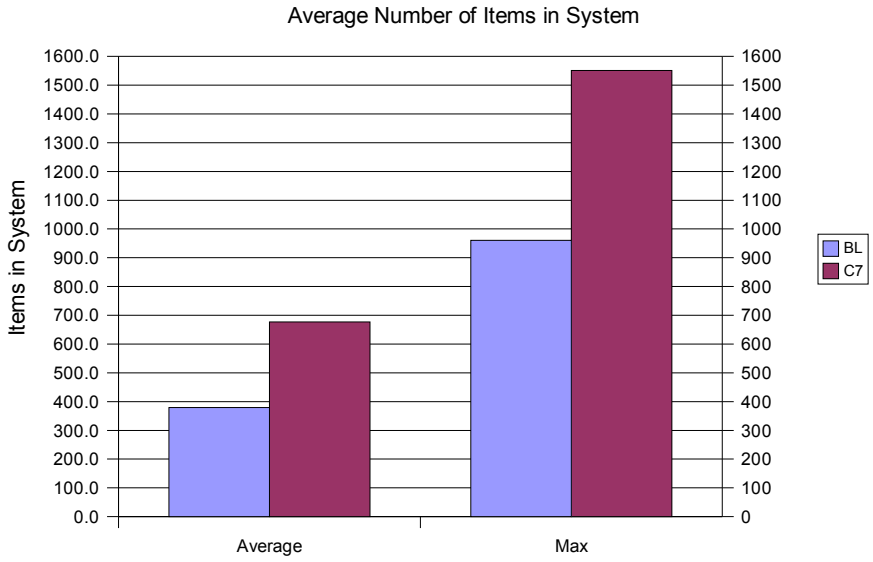

Figure 4.46 BL vs C7: Average and average maximum work in process

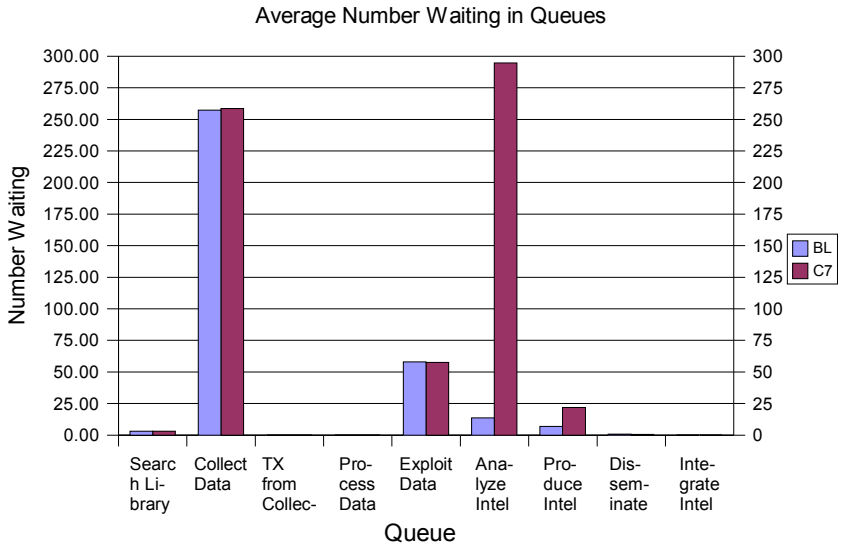

Figure 4.47 BL vs C7: Average number of *RFI*s waiting in queues

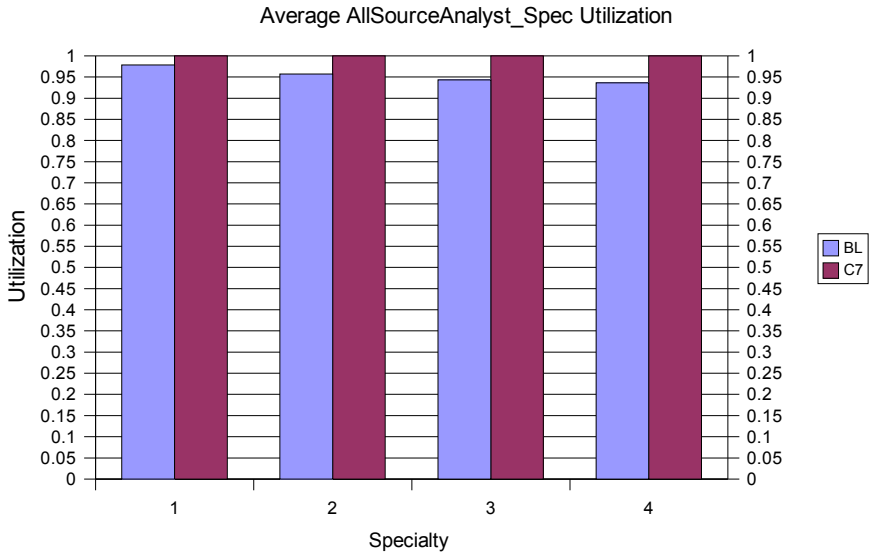

Figure 4.48 BL vs C7: Average utilization of *AllSourceAnalyst_Spec* resources

to the substantial queue size, the all source analyst resources are 100% utilized on average for C7 (see Figure 4.48). Note that this does not appear to be a practically significant change from the BL, but a small change near peak utilization may in reality be significant. It is interesting to note that even though the production process uses the same resources as the analysis process, the production queue does not have a large queue size on average. Since the production process has equal priority for using the all source analyst resources, this may be due to a reduced arrival rate of items to the production process. Related to the increased queue sizes and number in system, the average wait time by priority shows that the average wait time for lower priority items has increased (see Figure 4.49). For the lower priority items, it appears that the C7 wait times have nearly doubled compared to the BL wait times. Consequently, as can be seen in Figure 4.50 the proportion of requirements met by priority has significantly decreased for all priorities except the top priority items. When the proportion of requirements met is broken down by *InfoSource*, Figure 4.51 indicates that a practically significant decrease in the proportion of requirements met is realized for all sources. This is not unexpected since the analysis times were

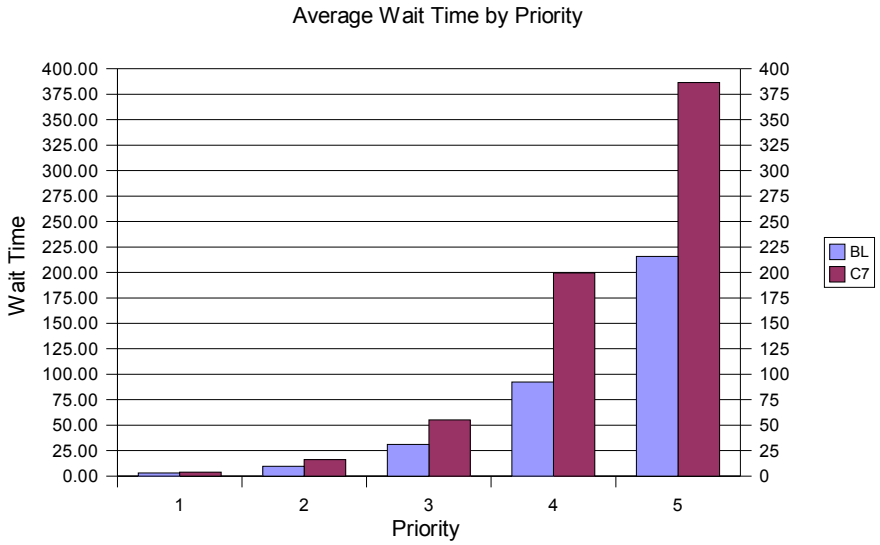

Figure 4.49 BL vs C7: Average total wait time (hours) by *Priority_User*

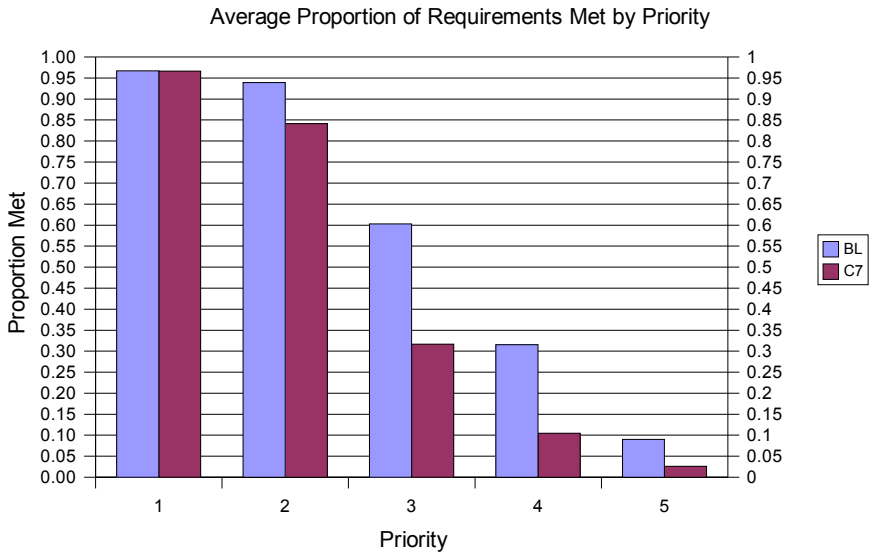

Figure 4.50 BL vs C7: Average proportion of requirements met by *Priority_User*

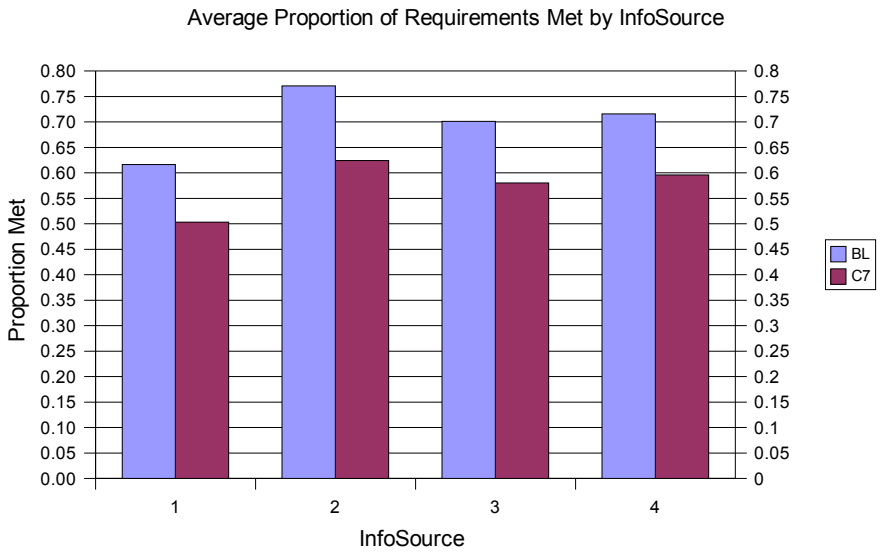

Figure 4.51 BL vs C7: Average proportion of requirements met by *InfoSource*

increased without regard to the source of information. From the users perspective, only *User* 4 maintained the proportion of requirements met in C7 that was obtained in the BL. Recalling the user setup discussed in Section 4.3.1, *User* 4 was the only user that required neither analysis nor production. All other users required at least one of analysis or production and as a result of the roughly 100% analyst utilization shown in Figure 4.48, ended up with a fewer proportion of requirements met.

Overall, the results of Case 7 are consistent with expectations given the 50% increase in analysis times. As expected, increased WIP, larger analysis queue size, and longer wait times for lower priority items were observed. Given the common resource usage between the analysis and production phases, a smaller proportion of requirements were met even for those users that did not require analysis so long as they required production.

4.4 Summary

The selected replication length and termination points allowed for an assessment of the baseline system near steady state responses. The seven cases provided

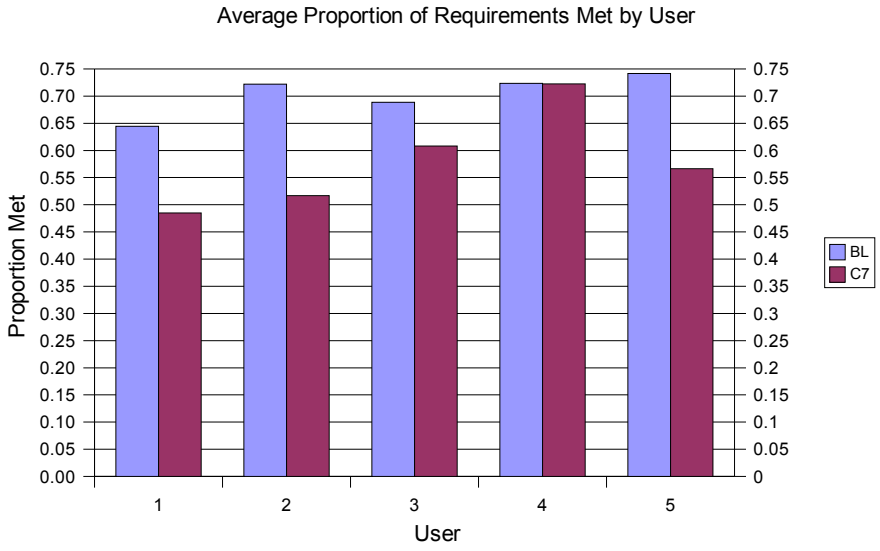

Figure 4.52 BL vs C7: Average proportion of requirements met by *User*

simplistic cases to stress the system to aid in model verification. Although more complex and less extreme changes may be more realistic, they may also provide results that are not as easy to interpret. The results of the seven cases consistently impacted the system in ways that would be expected given the nature of each individual change. In a few cases there were some slightly unexpected results but they were not unacceptable. In general an explanation of the unexpected results could be found by considering the interaction of other aspects of the model. The favorable results of the seven cases lay the foundation for investigation of more complex interactions within the model.

5. Conclusions

This chapter will provide a general discussion of the *Intelligence Process Model*. Both strengths and limitations of the model will be briefly reexamined. Additionally, some potential areas of application for the model and simulation will be presented. Finally, some areas of future research will be introduced.

5.1 Discussion

The initial focus of this study was on developing a high level simulation model to address the need to do quick turn studies on the ability of intelligence processes to support ISR systems. A review of prior efforts in modeling intelligence process generally focused on a specific implementation such as TPED or required too much detail for a high level model. As a result, the *Intelligence Process Model* was developed primarily from the high level perspective of the intelligence cycle presented in Joint Publications as well as SME input. The benefit of this approach is that it is grounded on documented processes while taking a top down view. As a high level model, the *IPM* requires less information to populate than a highly detailed model. Even so, the amount of information required to populate the model is still extensive. One challenge the high level perspective provides to analysts is appropriately aggregating existing detailed information.

The *IPM* was also developed in a modular fashion with emphasis on a core module tying all the other pieces together. This allows for additional detail to be later added to any particular submodel with minimal or no changes required for the rest of the model. Additionally, many details were organized in arrays of information to simplify the visual layout and aid in customization of the model for particular systems and scenarios. One benefit of using arrays is that the probability distributions and information for a selected scenario are input in only a few locations rather than requiring alterations throughout the model. Additionally, information

can be gathered and organized using a spreadsheet and easily transferred into Arena. An example of such a spreadsheet is given in Appendix D.

The validation and verification effort undertaken during and after model development lends credibility to the *IPM*. Multiple reviews of the conceptual model by SMEs helped to ensure that appropriate assumptions and details were included. Furthermore, multiple detailed walk-throughs of the implemented model helped to ensure that the implementation of the *IPM* in Arena was both conceptually and and technically correct. The use of a notional baseline system and seven case studies to stress the model lends additional credibility to the model given that the changes induced for the seven cases resulted in expected outcomes.

In spite of the effort undertaken to develop a complete, valid and credible model, two limitations still remain. The first limitation is with the detail included in the *Communications* submodel of the *IPM*. The *Communications* submodel contains a very simple representation of a traditional communications environment. That is, it only provides communications delays in relatively small number of areas and includes minimal communications resources. Although such details are desirable even in a high level model, various constraints prevented their inclusion for this study. The focus taken for the *Communications* submodel was that it correctly route entities between the submodels based on various entity attributes. The validation and verification effort as well as the case studies confirmed that this core functionality of the *Communications* submodel works correctly.

The second limitation involves the issue of information integration or fusion. This issue was intentionally left out during early model development. Later in model development there was some indication that addressing the fusion problem would be beneficial for some studies. The result was a minimal fusion model which required broad assumptions that were not easily integrated with the rest of the model. At this point we determined further development of the fusion capability was beyond the scope of this study. In order to appropriately address the fusion issue, one must

also determine a way to model the breakdown of large problems into the "minimal" pieces of information modeled in the *IPM*. Once that issue is addressed, reassembling the pieces becomes a simpler problem. The fusion portions of the model were not tested and are not recommended for use. The fusion portions are easily disabled, and in fact, were disabled for the baseline and all seven case studies.

Overall, the intent of developing a generalized framework of the intelligence process for simulation studies has been successful. The addition of relevant performance measures, such as proportion of requirements met, allows the *IPM* to be readily used for many simulation studies. Furthermore, the partitioning of those proportions into various categories allows additional insight into the process.

5.2 Application

Given that an analyst performing simulation studies using the *IPM* understands the limitations of the *Communications* submodel and how those limitations might impact simulation results, they could immediately begin using the *IPM* for simulation studies. One potential area of application for the *IPM* is to examine the TPED, TPPU, and other hybrid variations of the intelligence process in a "saturated" intelligence environment. In other words, rather than examine each of the variations independent of each other, replace a portion of the current process with that variation and examine the impact on system performance. Another potential area of application for the *IPM* is to examine the impact of various proposed changes to collection or other resources. Studies such as these could lend insight into which suggested changes might be more beneficial.

5.3 Future Research

Aside from the various studies that can be conducted using the model, the primary areas of future research involve modifying the model structure. The two

top candidates for additional research relate to the two concerns that were recapped in Section 5.1. The top candidate is the development of a more robust and realistic communications model. Such a communications model would need to maintain the functionality of the current *Communications* submodel which appropriately routes entities between other submodels while increasing the realism and robustness. The challenge for this research area is selecting an appropriate level of detail while maintaining the generalized nature of the *IPM*. The second candidate for additional research is an investigation into how high level problems are decomposed into smaller, "bite-sized" information requests that are currently handled by the *IPM*. In concert with such an investigation would be the development and addition of such a breakdown model as well as the development and addition of fusion within the model to reassemble information into the original requests. Other areas of research could involve adding additional detail to any particular submodel. However, care should be taken to not include too much detail, as it would defeat the purpose of a high level model to be used for quick look studies. Furthermore, additional details in these submodels may be of marginal benefit when compared to research into the communications model. The addition of these areas of study to the model could greatly broaden the scope of problems the simulation could be used to examine and allow additional insight into the intelligence process.

Appendix A. List of Abbreviations and Acronyms

BL	Baseline
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
CI	Counterintelligence
CONOPS	Concept of operations
COSMOS	C4ISR Space and Missile Operations Simulator
DES	Discrete event simulation
HUMINT	Human Intelligence
IMINT	Imagery Intelligence
INS	Information needs satisfaction
IPM	Intelligence Process Model
ISR	Intelligence, surveillance, and reconnaissance
ISR-PET	ISR Performance Evaluation Tool
ISRSIM	Intelligence, Surveillance and Reconnaissance Simulator
LVF	Lowest value first
MASINT	Measurement and Signature Intelligence
NIS	Number in system (see WIP)
NQ	Number in queue
NSS	National Security Strategy
NSSA	National Security Space Architect
OSINT	Open-Source Intelligence
PIR	Priority Information Request

QQT	Quality, quantity, and timeliness
QQTI	Quality, quantity, timeliness, and information needs satisfaction
QUICM	Quick ISR CONOPS Modeler
RFI	Request for information
SAR	Synthetic Aperture Radar
SIGINT	Signals Intelligence
SME	Subject Matter Expert
TECHINT	Technical Intelligence
TIS	Time in system
TPED	Tasking, processing, exploitation, dissemination
TPPU	Tasking, processing, posting, using
V&V	Validation and verification
VBA	Visual Basic for Applications
WIP	Work in process
WT	Wait time

Appendix B. Exported VBA Class File

The following *VBA* code was exported from the *Intelligence Process Model* version 2.7. It contains code for six *VBA* blocks used in the *Communications* submodel.

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "ThisDocument"
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
Private Sub VBA_Block_1_Fire()
' 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
'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
Private Sub VBA_Block_2_Fire()
' Determine NextStation after leaving Collection
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
'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_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
Private Sub VBA_Block_3_Fire()
' Determine NextStation after leaving Processing
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
'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_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
Private Sub VBA_Block_4_Fire()
' Determine NextStation after leaving Exploitation
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
'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_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
Private Sub VBA_Block_5_Fire()
' Determine NextStation after leaving Analysis
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
'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_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
Private Sub VBA_Block_6_Fire()
' Determine NextStation after leaving Production
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
'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_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
```

Appendix C. Model Flow Charts

The following figures contain flow charts used for model validation and verification. The first flow char depicts the potential flow of entities due to the modularity of the model. The current implementation is similar to the planned v2.5 implementation except that the Planning, Library Search, and Evaluation portions have been fully modularized. All possible routes depicted are not realized because of the core model logic within the *Communications* submodel. The core logic of the current v2.7 model is essentially the same as the planned v2.5 model except that some *RFI*s do not undergo a library search by default. Refer to Chapter 3 for details on the current model implementation.

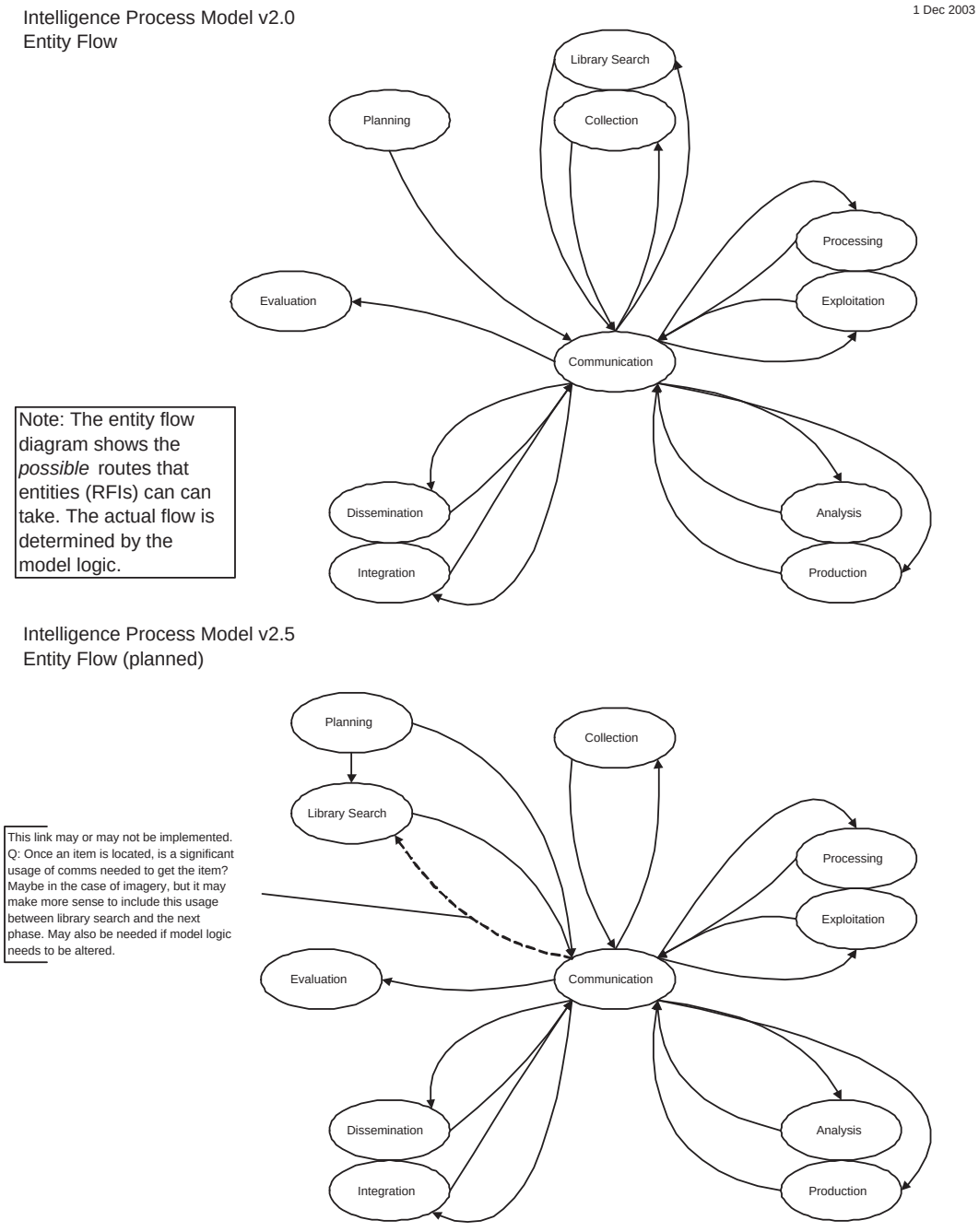

Figure C.1 Charts depicting possible entity flow in earlier model versions.

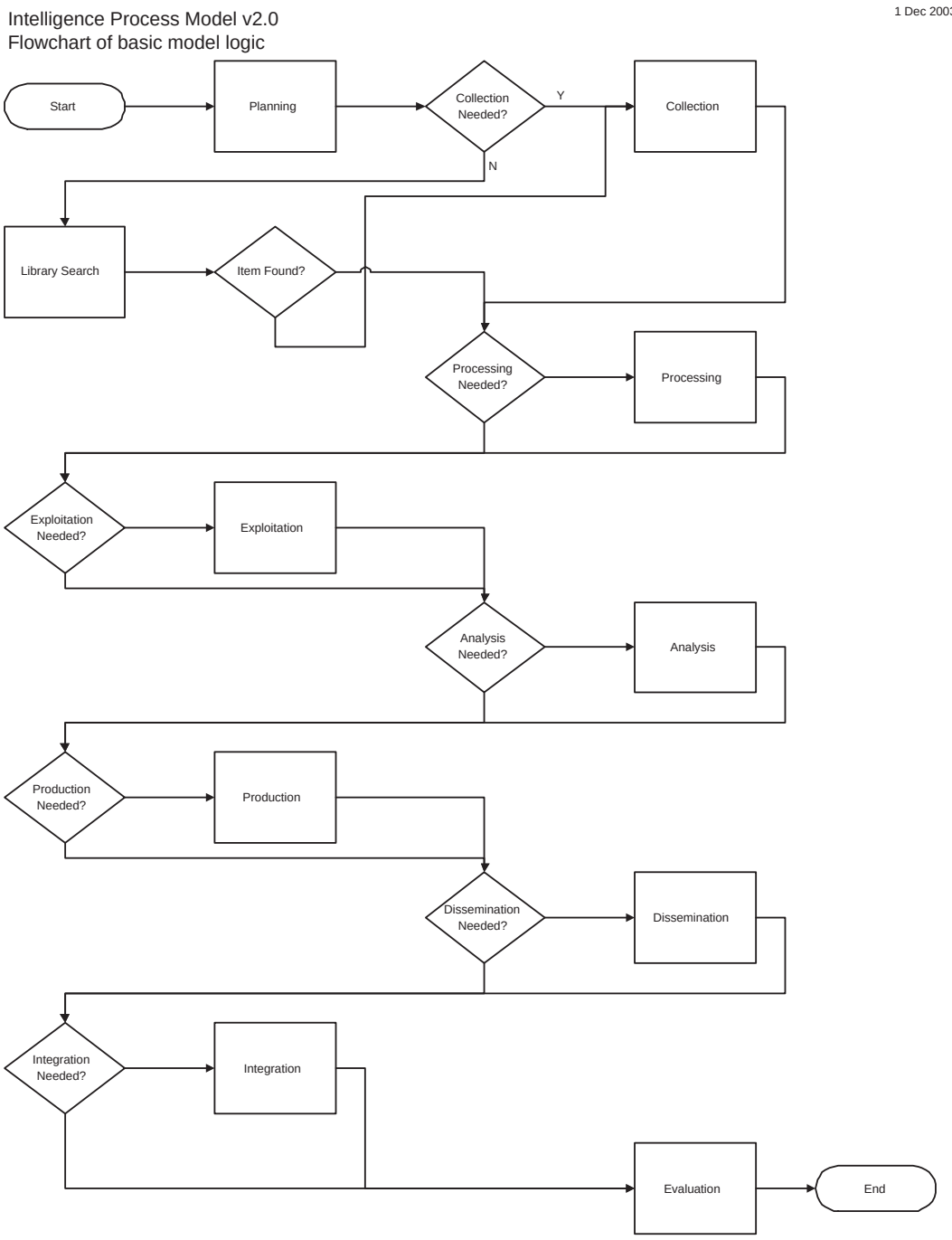

Figure C.2 Flow chart of the IPM v2.0 model logic.

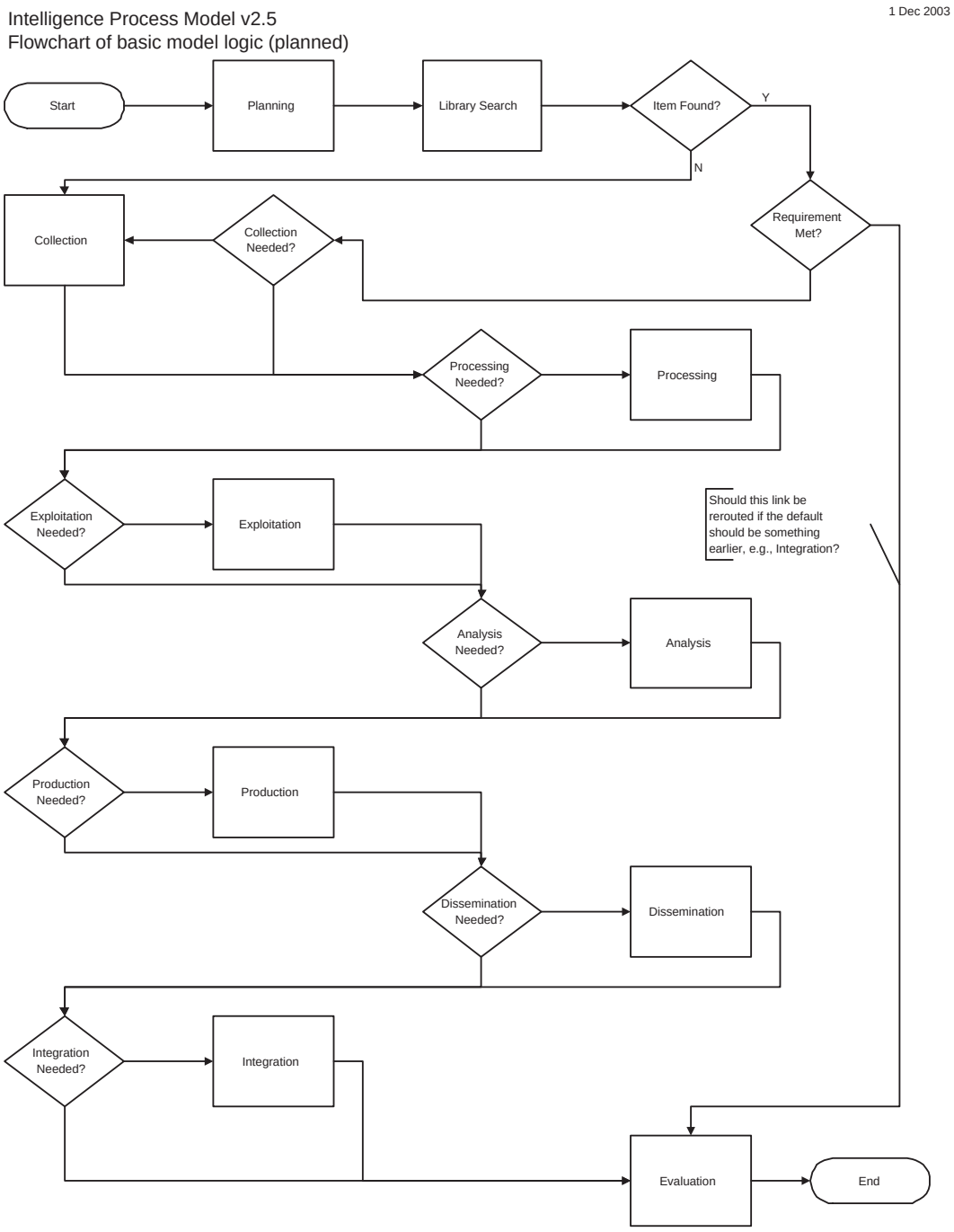

Figure C.3 Flow chart of the IPM v2.5 model logic.

Appendix D. Data Request Sheet with Data for Sample Study

The following figures contain a copy of the data request sheet used for to collect data for the sample study. It includes an exhaustive list of the data required for expected studies. The first page of the request sheet gives some basic information about the types of information expected for various attributes assigned in the remainder of the request sheet. Additional information about syntax and parameters of Arena probability distributions and expressions can be found in the software help.

Fixed Values

Name	Value	Comments
Max # of Users (User)	Max#U = 5	Hard coded into model. Can be changed, but requires significant effort. This impacts any portion of the model that is user dependent.
Max # of Info Sources (InfoSource)	Max#IS = 13	Hard coded into model. Can be changed, but requires significant effort. This impacts any portion of the model that is dependent on #IS.
Levels of Quality (QualR)	#QLR = 5	Hard coded into model. Can be changed, but requires significant effort. This impacts any portion of the model that is dependent on #QLR.
Levels of Quality (QualA)	#QLA = 6	Hard coded into model. Can be changed, but requires significant effort. This impacts any portion of the model that is dependent on #QLA.
Base Time Unit	Hours	Set to Hours as a default. Could be changed to Days, Hours, Minutes, or Seconds, but ensure this change is made *everywhere (i.e., in all delay, process, create, etc. blocks)* .

Misc Notes

All Arrays must be completely populated. However, any entries beyond #U and #IS can be populated with zeros.
Quality levels range from 0 to 5 with 5 being the best quality
Required Quality must have integer values from 1 to 5
Achieved Quality must be >= 0 (can be non-integral) but will be ranked from 0 to 5 for some actions in the model
Logical expressions (i.e., A == B or A>= B) evaluate to 0 if false or 1 if true

The information sources are given generically, but are intended to be used at a high aggregate level. Some samples are given below.

InfoSources	Sample Source
Source_01	IMINT
Source_02	HUMINT
Source_03	SIGINT
Source_04	MASINT
Source_05	OSINT
Source_06	TECHINT
Source_07	CI
Source_08	SBR
Source_09	MOVINT
Source_10	Radar
Source_11	Surveil
Source_12	Recon
Source_13	Coalition

Attributes of entites or other expressions can be referenced directly.

Attribute Name	Values	Comments
User	integer 1-5	Denotes user where item originated
Standard	integer 0-1	Denotes whether an item is a standing/standard request (1) or an additional request (0)
InfoSource	integer 1-13	Denotes the information source required for the item
TimeR	real > 0	Denotes time from creation of item when it is required
QualR	integer 1-5	Denotes required quality of item (5 is highest quality)
Priority_User	integer 1-5	Denotes user priority of an item (1 is highest priority)
Collect	integer 0-1	Denotes if collection should occur (1) or not (0) for this item
Process	integer 0-1	Denotes if processing should occur (1) or not (0) for this item
Exploit	integer 0-1	Denotes if exploitation should occur (1) or not (0) for this item
Analyze	integer 0-1	Denotes if analysis should occur (1) or not (0) for this item
Produce	integer 0-1	Denotes if production should occur (1) or not (0) for this item
Disseminate	integer 0-1	Denotes if dissemination should occur (1) or not (0) for this item
Integrate	integer 0-1	Denotes if user integration should occur (1) or not (0) for this item

Expression Name	Values	Expression Comments
Timely	integer 0-1	(TNOW - Entity.CreateTime) - TimeR <= 0 Evaluates if an item is timely or not
QualMet	integer 0-1	QualA >= QualR Evaluates if an item meets the quality requirement
AgeMet	integer 0-1	AgeMetInLibrary Evaluates if the age requirement was met during a library search
QualARank	integer 0-1	MN(MX(ANINT(QualA) , 0) + 1 , 6) Ranks actual/achieved quality into 5 levels 1-6 (corresponds to real levels 0-5) Functions: MN = min, MX = max, ANINT = round to nearest integer ANINT could be replaced by AINT (truncate) if it makes more sense

Supported Probability Distributions

Beta
Empirical Continuous
Empirical Discrete
k-Erlang
Exponential
Gamma
Johnson
Weibull
Lognormal
Normal
NSExpo
Poisson
Triangular
Uniform

Supported Mathmatical Expressions

Mathmatical expressions other than probability distributions can also be used in lieu of or in conjuction with probablility distributions if desired.
Contents of package (pdf) may need to be activated 2-3 times before it finds Acrobat.

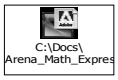

Figure D.1 Data Request Basic Information

NumUsers — Number of users for the study. This affects all arrays below with an User dimension.

	Value	Max
NumUsers	5	5

***Use streams 2,3,4,5 for InfoSource 1,2,3,4 draws respectively**
***Use stream 6 when indep. of InfoSource**

NumInfoSources — Number of information sources for the study. This affects all arrays below with an InfoSource dimension.

	Value	Max
NumInfoSources	4	13

NumAnalystSpecialties — Number of specialties for all source analysts. This affects all arrays below with an AnalystSpecialty dimension.

	Value	Max
NumAnalystSpecialties	4	13

**Note: If the values above are less than their max values, the remaing items in an array can be set to 0 (zero).
**Note: Some Sample entries have been given
**Note: Additional Variables or Expressions can be defined and used to fill out the datasheet if desired. Please add these items and defintions to the bottom of the sheet.

InfoCollectTimes — Each entry of the array corresponds to the distribution of time taken to collect information from a specified information source at a specified level of required quality.

QualR \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
1	EXPO(3,1)	3	UNIF(0.8,1.6,3)	EXPO(0.3,4)	0	0	0	0	0	0	0	0	0
2	EXPO(4,1)	4	UNIF(0.8,1.6,3)	EXPO(0.6,4)	0	0	0	0	0	0	0	0	0
3	EXPO(5,1)	5	UNIF(0.8,1.6,3)	EXPO(0.9,4)	0	0	0	0	0	0	0	0	0
4	EXPO(6,1)	6	UNIF(1.6,3.2,3)	EXPO(1.2,4)	0	0	0	0	0	0	0	0	0
5	EXPO(7,1)	7	UNIF(1.6,3.2,3)	EXPO(1.5,4)	0	0	0	0	0	0	0	0	0

LibrarySearchTimes — Each entry of the array corresponds to the distribution of time taken to search the "Library" of information available for information from a specified information source and level of required quality.

QualR \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
1	EXPO(0.08,1)	EXPO(0.08,2)	EXPO(0.08,3)	EXPO(0.08,4)	0	0	0	0	0	0	0	0	0
2	EXPO(0.1,1)	EXPO(0.1,2)	EXPO(0.1,3)	EXPO(0.1,4)	0	0	0	0	0	0	0	0	0
3	EXPO(0.12,1)	EXPO(0.12,2)	EXPO(0.12,3)	EXPO(0.12,4)	0	0	0	0	0	0	0	0	0
4	EXPO(0.14,1)	EXPO(0.14,2)	EXPO(0.14,3)	EXPO(0.14,4)	0	0	0	0	0	0	0	0	0
5	EXPO(0.16,1)	EXPO(0.16,2)	EXPO(0.16,3)	EXPO(0.16,4)	0	0	0	0	0	0	0	0	0

InfoProcessTimes — Each entry of the array corresponds to the distribution of time taken to process information (e.g., automated processing) from a specified source and a ranked achieved quality (0-5)

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.01	0.01	0.01	0.01	0	0	0	0	0	0	0	0	0
1	UNIF(0.01,0.02,1)	UNIF(0.01,0.02,2)	UNIF(0.01,0.02,3)	UNIF(0.01,0.02,4)	0	0	0	0	0	0	0	0	0
2	UNIF(0.01,0.02,1)	UNIF(0.01,0.02,2)	UNIF(0.01,0.02,3)	UNIF(0.01,0.02,4)	0	0	0	0	0	0	0	0	0
3	UNIF(0.01,0.02,1)	UNIF(0.01,0.02,2)	UNIF(0.01,0.02,3)	UNIF(0.01,0.02,4)	0	0	0	0	0	0	0	0	0
4	UNIF(0.02,0.03,1)	UNIF(0.02,0.03,2)	UNIF(0.02,0.03,3)	UNIF(0.02,0.03,4)	0	0	0	0	0	0	0	0	0
5	UNIF(0.02,0.03,1)	UNIF(0.02,0.03,2)	UNIF(0.02,0.03,3)	UNIF(0.02,0.03,4)	0	0	0	0	0	0	0	0	0

InfoDissemTimes — Each entry of the array corresponds to the distribution of time taken to disseminate information to a given user.

User	
1	EXPO(0.1,5)
2	EXPO(0.1,5)
3	EXPO(0.1,5)
4	EXPO(0.1,5)
5	EXPO(0.1,5)

InfoIntegTimes — Each entry of the array corresponds to the distribution of time taken for a specified user to integrate information into their processes

User	
1	0
2	0
3	0
4	0
5	0

Figure D.2 Data Request Sheet Page 1

InfoExploitTimes_QR_1 Each entry of the array corresponds to the distribution of time taken to exploit information from a given information source with a ranked actual quality and a required quality level 1

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	TRIA(1.4,6.1)	TRIA(1.3,5.5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
1	TRIA(1.3,5.5,5.1)	TRIA(1.3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
2	TRIA(1.3,5.1)	TRIA(1.3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
3	TRIA(1.3,5.1)	TRIA(1.2,5.4,5.2)	TRIA(1.1,5.3,5.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
4	TRIA(1.2,5.4,5.1)	TRIA(1.2,5.4,5.2)	TRIA(1.1,5.3,5.3)	TRIA(1.1,25,3.25,4	0	0	0	0	0	0	0	0	0
5	TRIA(1.2,4.1)	TRIA(1.2,4.2)	TRIA(1.1,5.3,5.3)	TRIA(1.1,25,3.25,4	0	0	0	0	0	0	0	0	0

*Min 1 hour, Max 8 hrs, range between

InfoExploitTimes_QR_2 Each entry of the array corresponds to the distribution of time taken to exploit information from a given information source with a ranked actual quality and a required quality level 2

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	TRIA(1.4,6.1)	TRIA(1.3,5.5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
1	TRIA(1.4,6.1)	TRIA(1.3,5.5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
2	TRIA(1.3,5.5,5.1)	TRIA(1.3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
3	TRIA(1.3,5.1)	TRIA(1.3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
4	TRIA(1.3,5.1)	TRIA(1.2,5.4,5.2)	TRIA(1.1,5.3,5.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
5	TRIA(1.2,5.4,5.1)	TRIA(1.2,5.4,5.2)	TRIA(1.1,5.3,5.3)	TRIA(1.1,25,3.25,4	0	0	0	0	0	0	0	0	0

InfoExploitTimes_QR_3 Each entry of the array corresponds to the distribution of time taken to exploit information from a given information source with a ranked actual quality and a required quality level 3

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	TRIA(3,5,7,1)	TRIA(1.4,5.6,5.2)	TRIA(1,3,5,3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
1	TRIA(1.4,6.1)	TRIA(1.3,5.5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
2	TRIA(1.4,6.1)	TRIA(1.3,5.5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
3	TRIA(1.3,5.5,5.1)	TRIA(1.3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
4	TRIA(1.3,5.1)	TRIA(1.3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
5	TRIA(1.3,5.1)	TRIA(1.2,5.4,5.2)	TRIA(1.1,5.3,5.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0

InfoExploitTimes_QR_4 Each entry of the array corresponds to the distribution of time taken to exploit information from a given information source with a ranked actual quality and a required quality level 4

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	TRIA(3,5,8,1)	TRIA(1.4,5.7,5.2)	TRIA(1.3,6.3)	TRIA(1.3,5.4)	0	0	0	0	0	0	0	0	0
1	TRIA(3,5,7,1)	TRIA(1.4,5.6,5.2)	TRIA(1.3,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
2	TRIA(1.4,6.1)	TRIA(1.3,5.5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
3	TRIA(1.4,6.1)	TRIA(1.3,5.5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
4	TRIA(1.3,5.5,5.1)	TRIA(1.3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
5	TRIA(1.3,5.1)	TRIA(1.3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0

InfoExploitTimes_QR_5 Each entry of the array corresponds to the distribution of time taken to exploit information from a given information source with a ranked actual quality and a required quality level 5

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	TRIA(3,5,8,1)	TRIA(1,5,8,2)	TRIA(1,4,7,3)	TRIA(1,3,6,4)	0	0	0	0	0	0	0	0	0
1	TRIA(3,5,8,1)	TRIA(1.4,5.6,5.2)	TRIA(1,3,6,3)	TRIA(1,3,5,4)	0	0	0	0	0	0	0	0	0
2	TRIA(3,5,7,1)	TRIA(1.4,5.6,5.2)	TRIA(1,3,5,3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
3	TRIA(1.4,6.1)	TRIA(1.3,5.5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
4	TRIA(1.4,6.1)	TRIA(1.3,5.5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
5	TRIA(1.3,5.5,5.1)	TRIA(1.3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0

InfoAnalyzeTimes_QR_1 Each entry of the array corresponds to the distribution of time taken to analyze information from a given information source with a ranked actual quality and a required quality level 1

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	TRIA(1.4,6.1)	TRIA(1.3,5.5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
1	TRIA(1.3,5.5,5.1)	TRIA(1.3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
2	TRIA(1.3,5.1)	TRIA(1.3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
3	TRIA(1.3,5.1)	TRIA(1.2,5.4,5.2)	TRIA(1.1,5.3,5.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
4	TRIA(1.2,5.4,5.1)	TRIA(1.2,5.4,5.2)	TRIA(1.1,5.3,5.3)	TRIA(1.1,25,3.25,4	0	0	0	0	0	0	0	0	0
5	TRIA(1.2,4.1)	TRIA(1.2,4.2)	TRIA(1.1,5.3,5.3)	TRIA(1.1,25,3.25,4	0	0	0	0	0	0	0	0	0

*Min 1 hour, Max 8 hrs, range between, Use same as Exploit times

Figure D.3 Data Request Sheet Page 2

InfoAnalyzeTimes_QR_2 Each entry of the array corresponds to the distribution of time taken to analyze information from a given information source with a ranked actual quality and a required quality level 2

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	TRIA(1.4,6.1)	TRIA(1.3,5.5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
1	TRIA(1.4,6.1)	TRIA(1.3,5.5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
2	TRIA(1.3,5.5,5.1)	TRIA(1,3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
3	TRIA(1.3,5.1)	TRIA(1,3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
4	TRIA(1.3,5.1)	TRIA(1.2,5.4,5.2)	TRIA(1.1,5.3,5.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
5	TRIA(1.2,5.4,5.1)	TRIA(1.2,5.4,5.2)	TRIA(1.1,5.3,5.3)	TRIA(1.1,25,3.25,4	0	0	0	0	0	0	0	0	0

InfoAnalyzeTimes_QR_3 Each entry of the array corresponds to the distribution of time taken to analyze information from a given information source with a ranked actual quality and a required quality level 3

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	TRIA(3,5,7,1)	TRIA(1,4,5,6,5.2)	TRIA(1,3,5,3)	TRIA(1,2,5,4,5,4)	0	0	0	0	0	0	0	0	0
1	TRIA(1.4,6.1)	TRIA(1.3,5,5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
2	TRIA(1.4,6.1)	TRIA(1.3,5,5,5.2)	TRIA(1.2,5.4,5.3)	TRIA(1.2,5.4,5.4)	0	0	0	0	0	0	0	0	0
3	TRIA(1.3,5,5,5.1)	TRIA(1,3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
4	TRIA(1.3,5.1)	TRIA(1,3,5.2)	TRIA(1.2,4.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0
5	TRIA(1.3,5.1)	TRIA(1.2,5.4,5.2)	TRIA(1.1,5.3,5.3)	TRIA(1.1,5.3,5.4)	0	0	0	0	0	0	0	0	0

InfoAnalyzeTimes_QR_4 Each entry of the array corresponds to the distribution of time taken to analyze information from a given information source with a ranked actual quality and a required quality level 4

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	TRIA(3,5,8,1)	TRIA(1,4,5,7,5.2)	TRIA(1,3,6,3)	TRIA(1,3,5,4)	0	0	0	0	0	0	0	0	0
1	TRIA(3,5,7,1)	TRIA(1,4,5,6,5.2)	TRIA(1,3,5,3)	TRIA(1,2,5,4,5,4)	0	0	0	0	0	0	0	0	0
2	TRIA(1.4,6.1)	TRIA(1,3,5,5,5.2)	TRIA(1,2,5,4,5,3)	TRIA(1,2,5,4,5,4)	0	0	0	0	0	0	0	0	0
3	TRIA(1.4,6.1)	TRIA(1,3,5,5,5.2)	TRIA(1,2,5,4,5,3)	TRIA(1,2,5,4,5,4)	0	0	0	0	0	0	0	0	0
4	TRIA(1.3,5,5,5.1)	TRIA(1,3,5.2)	TRIA(1,2,4,3)	TRIA(1,1,5,3,5,4)	0	0	0	0	0	0	0	0	0
5	TRIA(1.3,5.1)	TRIA(1,3,5.2)	TRIA(1,2,4,3)	TRIA(1,1,5,3,5,4)	0	0	0	0	0	0	0	0	0

InfoAnalyzeTimes_QR_5 Each entry of the array corresponds to the distribution of time taken to analyze information from a given information source with a ranked actual quality and a required quality level 5

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	TRIA(3,5,8,1)	TRIA(1,5,8,2)	TRIA(1,4,7,3)	TRIA(1,3,6,4)	0	0	0	0	0	0	0	0	0
1	TRIA(3,5,8,1)	TRIA(1,4,5,6,5.2)	TRIA(1,3,6,3)	TRIA(1,3,5,4)	0	0	0	0	0	0	0	0	0
2	TRIA(3,5,7,1)	TRIA(1.4,5,6,5.2)	TRIA(1,3,5,3)	TRIA(1,2,5,4,5,4)	0	0	0	0	0	0	0	0	0
3	TRIA(1.4,6.1)	TRIA(1,3,5,5,5.2)	TRIA(1,2,5,4,5,3)	TRIA(1,2,5,4,5,4)	0	0	0	0	0	0	0	0	0
4	TRIA(1.4,6.1)	TRIA(1,3,5,5,5.2)	TRIA(1,2,5,4,5,3)	TRIA(1,2,5,4,5,4)	0	0	0	0	0	0	0	0	0
5	TRIA(1.3,5,5,5.1)	TRIA(1,3,5.2)	TRIA(1,2,4,3)	TRIA(1,1,5,3,5,4)	0	0	0	0	0	0	0	0	0

InfoProduceTimes_QR_1 Each entry of the array corresponds to the distribution of time taken to produce information from a given information source with a ranked actual quality and a required quality level 1

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	EXPO(2.5,1)	EXPO(2.25,2)	EXPO(2,3)	EXPO(2,4)	0	0	0	0	0	0	0	0	0
1	EXPO(2,1)	EXPO(1.75,2)	EXPO(1.5,3)	EXPO(1.25,4)	0	0	0	0	0	0	0	0	0
2	EXPO(1.5,1)	EXPO(1.25,2)	EXPO(1.25,3)	EXPO(1,4)	0	0	0	0	0	0	0	0	0
3	EXPO(1,1)	EXPO(1,2)	EXPO(1,3)	EXPO(1,4)	0	0	0	0	0	0	0	0	0
4	EXPO(1,1)	EXPO(1,2)	EXPO(1,3)	EXPO(1,4)	0	0	0	0	0	0	0	0	0
5	EXPO(1,1)	EXPO(1,2)	EXPO(1,3)	EXPO(1,4)	0	0	0	0	0	0	0	0	0

*Min 1 hour, Max 6 hrs, range between, use EXPO

InfoProduceTimes_QR_2 Each entry of the array corresponds to the distribution of time taken to produce information from a given information source with a ranked actual quality and a required quality level 2

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	EXPO(2.5,1)	EXPO(2.25,2)	EXPO(2,3)	EXPO(2,4)	0	0	0	0	0	0	0	0	0
1	EXPO(2.5,1)	EXPO(2.25,2)	EXPO(2,3)	EXPO(2,4)	0	0	0	0	0	0	0	0	0
2	EXPO(2,1)	EXPO(1.75,2)	EXPO(1.5,3)	EXPO(1.25,4)	0	0	0	0	0	0	0	0	0
3	EXPO(1.5,1)	EXPO(1.25,2)	EXPO(1.25,3)	EXPO(1,4)	0	0	0	0	0	0	0	0	0
4	EXPO(1,1)	EXPO(1,2)	EXPO(1,3)	EXPO(1,4)	0	0	0	0	0	0	0	0	0
5	EXPO(1,1)	EXPO(1,2)	EXPO(1,3)	EXPO(1,4)	0	0	0	0	0	0	0	0	0

Figure D.4 Data Request Sheet Page 3

InfoProduceTimes_QR_3 Each entry of the array corresponds to the distribution of time taken to produce information from a given information source with a ranked actual quality and a required quality level 3

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	EXPO(3,1)	EXPO(2.75,2)	EXPO(2.5,3)	EXPO(2,4)	0	0	0	0	0	0	0	0	0
1	EXPO(2.5,1)	EXPO(2.25,2)	EXPO(2,3)	EXPO(2,4)	0	0	0	0	0	0	0	0	0
2	EXPO(2.5,1)	EXPO(2.25,2)	EXPO(2,3)	EXPO(2,4)	0	0	0	0	0	0	0	0	0
3	EXPO(2,1)	EXPO(1.75,2)	EXPO(1.5,3)	EXPO(1.25,4)	0	0	0	0	0	0	0	0	0
4	EXPO(1.5,1)	EXPO(1.25,2)	EXPO(1.25,3)	EXPO(1,4)	0	0	0	0	0	0	0	0	0
5	EXPO(1,1)	EXPO(1,2)	EXPO(1,3)	EXPO(1,4)	0	0	0	0	0	0	0	0	0

InfoProduceTimes_QR_4 Each entry of the array corresponds to the distribution of time taken to produce information from a given information source with a ranked actual quality and a required quality level 4

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	EXPO(3.5,1)	EXPO(3,2)	EXPO(2.75,3)	EXPO(2.5,4)	0	0	0	0	0	0	0	0	0
1	EXPO(3,1)	EXPO(2.75,2)	EXPO(2.5,3)	EXPO(2,4)	0	0	0	0	0	0	0	0	0
2	EXPO(2.5,1)	EXPO(2.25,2)	EXPO(2,3)	EXPO(2,4)	0	0	0	0	0	0	0	0	0
3	EXPO(2.5,1)	EXPO(2.25,2)	EXPO(2,3)	EXPO(2,4)	0	0	0	0	0	0	0	0	0
4	EXPO(2,1)	EXPO(1.75,2)	EXPO(1.5,3)	EXPO(1.25,4)	0	0	0	0	0	0	0	0	0
5	EXPO(1.5,1)	EXPO(1.25,2)	EXPO(1.25,3)	EXPO(1,4)	0	0	0	0	0	0	0	0	0

InfoProduceTimes_QR_5 Each entry of the array corresponds to the distribution of time taken to produce information from a given information source with a ranked actual quality and a required quality level 5

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0	EXPO(4,1)	EXPO(3.5,2)	EXPO(3.25,3)	EXPO(3,4)	0	0	0	0	0	0	0	0	0
1	EXPO(3.5,1)	EXPO(3,2)	EXPO(2.75,3)	EXPO(2.5,4)	0	0	0	0	0	0	0	0	0
2	EXPO(3,1)	EXPO(2.75,2)	EXPO(2.5,3)	EXPO(2,4)	0	0	0	0	0	0	0	0	0
3	EXPO(2.5,1)	EXPO(2.25,2)	EXPO(2,3)	EXPO(2,4)	0	0	0	0	0	0	0	0	0
4	EXPO(2.5,1)	EXPO(2.25,2)	EXPO(2,3)	EXPO(2,4)	0	0	0	0	0	0	0	0	0
5	EXPO(2,1)	EXPO(1.75,2)	EXPO(1.5,3)	EXPO(1.25,4)	0	0	0	0	0	0	0	0	0

LibrarySearchFound Each entry of the array corresponds to the probability that information from a specified information source is found by a given user. Required by the model to have a value of 0 (not found) or 1 (found)

User \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
1	DISC(0.8,0.1,1,1)	DISC(0.8,0.1,1,2)	DISC(0.8,0.1,1,3)	DISC(0.8,0.1,1,4)	0	0	0	0	0	0	0	0	0
2	DISC(0.8,0.1,1,1)	DISC(0.8,0.1,1,2)	DISC(0.8,0.1,1,3)	DISC(0.8,0.1,1,4)	0	0	0	0	0	0	0	0	0
3	DISC(0.8,0.1,1,1)	DISC(0.8,0.1,1,2)	DISC(0.8,0.1,1,3)	DISC(0.8,0.1,1,4)	0	0	0	0	0	0	0	0	0
4	DISC(0.8,0.1,1,1)	DISC(0.8,0.1,1,2)	DISC(0.8,0.1,1,3)	DISC(0.8,0.1,1,4)	0	0	0	0	0	0	0	0	0
5	DISC(0.8,0.1,1,1)	DISC(0.8,0.1,1,2)	DISC(0.8,0.1,1,3)	DISC(0.8,0.1,1,4)	0	0	0	0	0	0	0	0	0

LibrarySearchQualA Each entry of the array corresponds to the distribution of actual quality of information found during the library search (QualA).

User \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
1	UNIF(0.5,1)	UNIF(0.5,2)	UNIF(0.5,3)	UNIF(0.5,4)	0	0	0	0	0	0	0	0	0
2	UNIF(0.5,1)	UNIF(0.5,2)	UNIF(0.5,3)	UNIF(0.5,4)	0	0	0	0	0	0	0	0	0
3	UNIF(0.5,1)	UNIF(0.5,2)	UNIF(0.5,3)	UNIF(0.5,4)	0	0	0	0	0	0	0	0	0
4	UNIF(0.5,1)	UNIF(0.5,2)	UNIF(0.5,3)	UNIF(0.5,4)	0	0	0	0	0	0	0	0	0
5	UNIF(0.5,1)	UNIF(0.5,2)	UNIF(0.5,3)	UNIF(0.5,4)	0	0	0	0	0	0	0	0	0

LibrarySearchAgeMet Each entry of the array corresponds to the probability that information from a specified information source and a given user meets the age requirement

User \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
1	DISC(0.8,0.1,1,1)	DISC(0.8,0.1,1,2)	DISC(0.8,0.1,1,3)	DISC(0.8,0.1,1,4)	0	0	0	0	0	0	0	0	0
2	DISC(0.8,0.1,1,1)	DISC(0.8,0.1,1,2)	DISC(0.8,0.1,1,3)	DISC(0.8,0.1,1,4)	0	0	0	0	0	0	0	0	0
3	DISC(0.8,0.1,1,1)	DISC(0.8,0.1,1,2)	DISC(0.8,0.1,1,3)	DISC(0.8,0.1,1,4)	0	0	0	0	0	0	0	0	0
4	DISC(0.8,0.1,1,1)	DISC(0.8,0.1,1,2)	DISC(0.8,0.1,1,3)	DISC(0.8,0.1,1,4)	0	0	0	0	0	0	0	0	0
5	DISC(0.8,0.1,1,1)	DISC(0.8,0.1,1,2)	DISC(0.8,0.1,1,3)	DISC(0.8,0.1,1,4)	0	0	0	0	0	0	0	0	0

Figure D.5 Data Request Sheet Page 4

GlobalPriorities

Each entry of the array corresponds to the expression that determines the global priority of items for each user. This should probably contain Priority_User.
Setting all entries to Priority_User implies that all users have equal priority.
If example below is set for all users, Priority 1 requests will be processed first reguardless of the user. User 1 will have priority over other users for all other items.

User	
1	Priority_User
2	Priority_User
3	Priority_User
4	Priority_User
5	Priority_User

InfoCollectQualA

Each entry of the array corresponds to the distribution of quality achieved during the collection of information from a given information source at a required level of quality

	InfoSource												
QualR	1	2	3	4	5	6	7	8	9	10	11	12	13
1	TRIA(0.8,1.1,2.1)	TRIA(0.8,1.1,2.2)	TRIA(0.8,1.1,2.3)	TRIA(0.8,1.1,2.4)	0	0	0	0	0	0	0	0	0
2	TRIA(1.8,2.2,2.1)	TRIA(1.8,2.2,2.2)	TRIA(1.8,2.2,2.3)	TRIA(1.8,2.2,2.4)	0	0	0	0	0	0	0	0	0
3	TRIA(2.8,3.3,2.1)	TRIA(2.8,3.3,2.2)	TRIA(2.8,3.3,2.3)	TRIA(2.8,3.3,2.4)	0	0	0	0	0	0	0	0	0
4	TRIA(3.8,4.4,2.1)	TRIA(3.8,4.4,2.2)	TRIA(3.8,4.4,2.3)	TRIA(3.8,4.4,2.4)	0	0	0	0	0	0	0	0	0
5	TRIA(4.8,5.5,2.1)	TRIA(4.8,5.5,2.2)	TRIA(4.8,5.5,2.3)	TRIA(4.8,5.5,2.4)	0	0	0	0	0	0	0	0	0

InfoProcessQualA

Each entry of the array corresponds to the distribution of quality achieved for information from a given information source as altered by processing the information
Inclusion of QualA in these expressions is critical, otherwise the previous quality will be completely overwritten.

InfoSource	
1	QualA
2	QualA
3	QualA
4	QualA
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0

InfoExploitQualA

Each entry of the array corresponds to the distribution of quality achieved for information from a given information source as altered by exploitation of the information
Inclusion of QualA in these expressions is critical, otherwise the previous quality will be completely overwritten.

InfoSource	
1	QualA*UNIF(1.1,2.1)
2	QualA*UNIF(1.1,2.2)
3	QualA*UNIF(1.1,2.3)
4	QualA*UNIF(1.1,2.4)
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0

Figure D.6 Data Request Sheet Page 5

InfoAnalyzeQualA

Each entry of the array corresponds to the distribution of quality achieved for information from a given information source as altered by analysis by an all source analyst with a given specialty.
Inclusion of QualA in these expressions is critical, otherwise the previous quality will be completely overwritten.

AnalystSpecialty \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
1	QualA*UNIF(1.1.2.1)	QualA*UNIF(0.9.1.1	QualA*UNIF(0.9.1.1	QualA*UNIF(0.9.1.	0	0	0	0	0	0	0	0	0
2	QualA*UNIF(0.9,1.1.1)	QualA*UNIF(1.1.2.2	QualA*UNIF(0.9.1.1	QualA*UNIF(0.9.1.	0	0	0	0	0	0	0	0	0
3	QualA*UNIF(0.9,1.1.1)	QualA*UNIF(0.9.1.1	QualA*UNIF(1.1.2.2	QualA*UNIF(0.9.1.	0	0	0	0	0	0	0	0	0
4	QualA*UNIF(0.9,1.1.1)	QualA*UNIF(0.9.1.1	QualA*UNIF(0.9.1.1	QualA*UNIF(1.1.2.	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0

InfoProduceQualA

Each entry of the array corresponds to the distribution of quality achieved for information from a given information source as altered by production by an all source analyst with a given specialty.
Inclusion of QualA in these expressions is critical, otherwise the previous quality will be completely overwritten.

AnalystSpecialty \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
1	QualA*UNIF(1.1.2.1)	QualA*UNIF(0.9.1.1	QualA*UNIF(0.9.1.1	QualA*UNIF(0.9.1.	0	0	0	0	0	0	0	0	0
2	QualA*UNIF(0.9,1.1.1)	QualA*UNIF(1.1.2.2	QualA*UNIF(0.9.1.1	QualA*UNIF(0.9.1.	0	0	0	0	0	0	0	0	0
3	QualA*UNIF(0.9,1.1.1)	QualA*UNIF(0.9.1.1	QualA*UNIF(1.1.2.2	QualA*UNIF(0.9.1.	0	0	0	0	0	0	0	0	0
4	QualA*UNIF(0.9,1.1.1)	QualA*UNIF(0.9.1.1	QualA*UNIF(0.9.1.1	QualA*UNIF(1.1.2.	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0

InfoDissemQualA

Each entry of the array corresponds to the distribution of quality achieved for information from a given information source as altered during dissemination
Inclusion of QualA in these expressions is critical, otherwise the previous quality will be completely overwritten.

InfoSource	1
1	QualA
2	QualA
3	QualA
4	QualA
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0

Figure D.7 Data Request Sheet Page 6

InfoIntegQualA

Each entry of the array corresponds to the distribution of quality achieved for information from a given source as altered during user integration
Inclusion of QualA in these expressions is critical, otherwise the previous quality will be completely overwritten.

InfoSource

1	QualA
2	QualA
3	QualA
4	QualA
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0

RFI_StdAttrib_U1

Attributes for standing/standard information requirements for User 1. The assumption is that these items will require collection.

Proportion of information requirements from each information source

InfoSource

1	2	3	4	5	6	7	8	9	10	11	12	13
0.19	0.19	0.23	0.39	0	0	0	0	0	0	0	0	0
0.19	0.38	0.61	1									

DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)

Proportion of information requirements that have a given required quality

QualR

1	2	3	4	5
0.05	0.05	0.5	0.3	0.1
0.05	0.1	0.6	0.9	1

DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)

Distribution of time until items are required (must be > 0)

TimeR

TRIA(24,48,120,5)

Distribution of user priorities (must be >= 1)

Priority_User	0.2	0.2	0.2	0.2
DISC(.2,1,.4,2,.6,3,.8,4,1,5,5)	0.2	0.4	0.6	0.8

InfoSourc DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)
TimeR TRIA(24,48,120,5)
QualR DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)
Priority_U DISC(.2,1,.4,2,.6,3,.8,4,1,5,5)

RFI_AddAttrib_U1

Attributes for additional information requirements for User 1. The assumption is that these items will only require collection if an adequate item is not found in the library.

Proportion of information requirements from each information source

InfoSource

1	2	3	4	5	6	7	8	9	10	11	12	13
0.19	0.19	0.23	0.39	0	0	0	0	0	0	0	0	0
0.19	0.38	0.61	1									

DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)

Proportion of information requirements that have a given required quality

QualR

1	2	3	4	5
0.05	0.05	0.5	0.3	0.1
0.05	0.1	0.6	0.9	1

DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)

Distribution of time until items are required (must be > 0)

TimeR

TRIA(24,36,48,5)

Distribution of user priorities (must be >= 1)

Priority_User	0.4	0.3	0.3	0
DISC(.4,1,.7,2,1,3,5)	0.4	0.7	1	1

InfoSourc DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)
TimeR TRIA(24,36,48,5)
QualR DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)
Priority_U DISC(.4,1,.7,2,1,3,5)

Figure D.8 Data Request Sheet Page 7

RFI_StdSteps_U1

Required steps for standing/standard information requirements. Proportion of items that must accomplish each step.

Steps	Percent of items required to complete
Collect	100% 1
Process	100% 1
Exploit	100% 1
Analyze	100% 1
Produce	100% 1
Disseminate	100% 1
Integrate	0% 0

RFI_AddSteps_U1

Required steps for additional requirements where an adequate item was *NOT* found during the library search. Assumption is that these require additional collection.

Steps	Percent of items required to complete
Collect	100% 1
Process	100% 1
Exploit	100% 1
Analyze	100% 1
Produce	100% 1
Disseminate	100% 1
Integrate	0% 0

RFI_AddSteps_NC_U1

Required steps for additional requirements where an adequate item was found during the library search (found, met quality requirement, met age requirement).

Steps	Percent of items required to complete
Collect	0% 0
Process	0% 0
Exploit	10% DISC(.9,0.1,1.1.5)
Analyze	20% DISC(.8,0.1,1.1.5)
Produce	50% DISC(.5,0.1,1.1.5)
Disseminate	Exploit || Analyze || Exploit || Analyze || Produce
Integrate	0% 0

RFI_StdAttrib_U2

Attributes for standing/standard information requirements for User 1. The assumption is that these items will require collection.

Proportion of information requirements from each information source

	InfoSource												
1	2	3	4	5	6	7	8	9	10	11	12	13	
	0.19	0.19	0.23	0.39	0	0	0	0	0	0	0	0	0
	0.19	0.38	0.61	1									

DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)

Proportion of information requirements that have a given required quality

	QualR					
1	2	3	4	5		
	0.05	0.05	0.5	0.3	0.1	DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)
	0.05	0.1	0.6	0.9	1	

Distribution of time until items are required (must be > 0)

TimeR

TRIA(24,48,120,5)

Distribution of user priorities (must be >= 1)

Priority_User

DISC(.2,1,.4,2,.6,3,.8,4,1,5,5)

0.2	0.2	0.2	0.2		
0.2	0.4	0.6	0.8	1	

InfoSourc DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)
TimeR TRIA(24,48,120,5)
QualR DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)
Priority_U DISC(.2,1,.4,2,.6,3,.8,4,1,5,5)

Figure D.9 Data Request Sheet Page 8

RFI_AddAttrib_U2

Attributes for additional information requirements for User 1. The assumption is that these items will only require collection if an adequate item is not found in the library.

Proportion of information requirements from each information source

InfoSource

1	2	3	4	5	6	7	8	9	10	11	12	13
0.19	0.19	0.23	0.39	0	0	0	0	0	0	0	0	0
0.19	0.38	0.61	1									

DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)

Proportion of information requirements that have a given required quality

QualR

1	2	3	4	5	6
0.05	0.05	0.5	0.3	0.1	DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)
0.05	0.1	0.6	0.9	1	

Distribution of time until items are required (must be > 0)

TimeR

TRIA(24,36,48,5)

Distribution of user priorities (must be >= 1)

Priority_User

DISC(.4,1,.7,2,1,3,5)

InfoSource DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)
TimeR TRIA(24,36,48,5)
QualR DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)
Priority_U DISC(.4,1,.7,2,1,3,5)

RFI_StdSteps_U2

Required steps for standing/standard information requirements. Proportion of items that must accomplish each step.

Steps	Percent of items required to complete	
Collect	100%	1
Process	100%	1
Exploit	0%	0
Analyze	100%	1
Produce	100%	1
Disseminate	100%	1
Integrate	0%	0

RFI_AddSteps_U2

Required steps for additional requirements where an adequate item was *NOT* found during the library search. Assumption is that these require additional collection.

Steps	Percent of items required to complete	
Collect	100%	1
Process	100%	1
Exploit	0%	0
Analyze	100%	1
Produce	100%	1
Disseminate	100%	1
Integrate	0%	0

RFI_AddSteps_NC_U2

Required steps for additional requirements where an adequate item was found during the library search (found, met quality requirement, met age requirement).

Steps	Percent of items required to complete	
Collect	0%	0
Process	0%	0
Exploit	10%	DISC(.9,0,1,1,5)
Analyze	20%	DISC(.8,0,1,1,5)
Produce	50%	DISC(.5,0,1,1,5)
Disseminate	Exploit || Analyze || Exploit || Analyze || Produce	
Integrate	0%	0

Figure D.10 Data Request Sheet Page 9

RFI_StdAttrib_U3 Attributes for standing/standard information requirements for User 1. The assumption is that these items will require collection.

Proportion of information requirements from each information source

	InfoSource												
	1	2	3	4	5	6	7	8	9	10	11	12	13
		0.19	0.19	0.23	0.39	0	0	0	0	0	0	0	0
		0.19	0.38	0.61	1								

DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)

Proportion of information requirements that have a given required quality

	QualR												
	1	2	3	4	5	6	7	8	9	10	11	12	13
		0.18	0.18	0.21	0.43	0.1	0	0	0	0	0	0	0
		0.18	0.36	0.57	1	1.1							

DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)

Distribution of time until items are required (must be > 0)

TimeR

TRIA(24,48,120,5)

Distribution of user priorities (must be >= 1)

	InfoSource												
	1	2	3	4	5	6	7	8	9	10	11	12	13
		0.2	0.2	0.2	0.2	0.2	0	0	0	0	0	0	0
		0.2	0.4	0.6	0.8	1							

DISC(.2,1,.4,2,.6,3,.8,4,1,5,5)

InfoSourc DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)
TimeR TRIA(24,48,120,5)
QualR DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)
Priority_U DISC(.2,1,.4,2,.6,3,.8,4,1,5,5)

RFI_AddAttrib_U3 Attributes for additional information requirements for User 1. The assumption is that these items will only require collection if an adequate item is not found in the library.

Proportion of information requirements from each information source

	InfoSource												
	1	2	3	4	5	6	7	8	9	10	11	12	13
		0.19	0.19	0.23	0.39	0	0	0	0	0	0	0	0
		0.19	0.38	0.61	1								

DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)

Proportion of information requirements that have a given required quality

	QualR												
	1	2	3	4	5	6	7	8	9	10	11	12	13
		0.05	0.05	0.5	0.3	0.1	0	0	0	0	0	0	0
		0.05	0.1	0.6	0.9	1							

DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)

Distribution of time until items are required (must be > 0)

TimeR

TRIA(24,36,48,5)

Distribution of user priorities (must be >= 1)

	0.4	0.3	0.3	0	0	
	0.4	0.7	1	1	1	

DISC(.4,1,.7,2,1,3,5)

InfoSourc DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)
TimeR TRIA(24,36,48,5)
QualR DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)
Priority_U DISC(.4,1,.7,2,1,3,5)

RFI_StdSteps_U3 Required steps for standing/standard information requirements. Proportion of items that must accomplish each step.

Steps	Percent of items required to complete
Collect	100%
Process	100%
Exploit	100%
Analyze	0%
Produce	100%
Disseminate	100%
Integrate	0%

	1
	1
	1
	0
	1
	1
	0

RFI_AddSteps_U3 Required steps for additional requirements where an adequate item was *NOT* found during the library search. Assumption is that these require additional collection.

Steps	Percent of items required to complete
Collect	100%
Process	100%
Exploit	100%
Analyze	0%
Produce	100%
Disseminate	100%
Integrate	0%

	1
	1
	1
	0
	1
	1
	0

Figure D.11 Data Request Sheet Page 10

RFI_AddSteps_NC_U3 Required steps for additional requirements where an adequate item was found during the library search (found, met quality requirement, met age requirement).

Steps	Percent of items required to complete
Collect	0%
Process	0%
Exploit	10% DISC(.9,0.1,1,5)
Analyze	20% DISC(.8,0.1,1,5)
Produce	50% DISC(.5,0.1,1,5)
Disseminate	Exploit || Analyze || Exploit || Analyze || Produce
Integrate	0%

RFI_StdAttrib_U4 Attributes for standing/standard information requirements for User 1. The assumption is that these items will require collection.

Proportion of information requirements from each information source

InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
	0.19	0.19	0.23	0.39	0	0	0	0	0	0	0	0	0
	0.19	0.38	0.61	1									

DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)

Proportion of information requirements that have a given required quality

QualR	1	2	3	4	5
	0.05	0.05	0.5	0.3	0.1
	0.05	0.1	0.6	0.9	1

InfoSourc	DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)
TimeR	TRIA(24,48,120,5)
QualR	DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)
Priority_U	DISC(.2,1,.4,2,.6,3,.8,4,1,5,5)

Distribution of time until items are required (must be > 0)
TimeR
TRIA(24,48,120,5)
Distribution of user priorities (must be >= 1)
Priority_User
DISC(.2,1,.4,2,.6,3,.8,4,1,5,5)

RFI_AddAttrib_U4 Attributes for additional information requirements for User 1. The assumption is that these items will only require collection if an adequate item is not found in the library.

Proportion of information requirements from each information source

InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
	0.19	0.19	0.23	0.39	0	0	0	0	0	0	0	0	0
	0.19	0.38	0.61	1									

DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)

Proportion of information requirements that have a given required quality

QualR	1	2	3	4	5
	0.05	0.05	0.5	0.3	0.1
	0.05	0.1	0.6	0.9	1

InfoSourc	DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)
TimeR	TRIA(24,36,48,5)
QualR	DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)
Priority_U	DISC(.4,1,.7,2,1,3,5)

Distribution of time until items are required (must be > 0)
TimeR
TRIA(24,36,48,5)
Distribution of user priorities (must be >= 1)
Priority_User
DISC(.4,1,.7,2,1,3,5)

RFI_StdSteps_U4 Required steps for standing/standard information requirements. Proportion of items that must accomplish each step.

Steps	Percent of items required to complete
Collect	100%
Process	100%
Exploit	100%
Analyze	0%
Produce	0%
Disseminate	100%
Integrate	0%

Figure D.12 Data Request Sheet Page 11

RFI_AddSteps_U4 Required steps for additional requirements where an adequate item was *NOT* found during the library search. Assumption is that these require additional collection.

Steps	Percent of items required to complete	
Collect	100%	1
Process	100%	1
Exploit	100%	1
Analyze	0%	0
Produce	0%	0
Disseminate	100%	1
Integrate	0%	0

RFI_AddSteps_NC_U4 Required steps for additional requirements where an adequate item was found during the library search (found, met quality requirement, met age requirement).

Steps	Percent of items required to complete	
Collect	0%	0
Process	0%	0
Exploit	10%	DISC(.9,0,1,1,5)
Analyze	20%	DISC(.8,0,1,1,5)
Produce	50%	DISC(.5,0,1,1,5)
Disseminate	Exploit || Analyze || Exploit || Analyze || Produce	
Integrate	0%	0

RFI_StdAttrib_U5 Attributes for standing/standard information requirements for User 1. The assumption is that these items will require collection.

Proportion of information requirements from each information source

InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
	0.19	0.19	0.23	0.39	0	0	0	0	0	0	0	0	0
	0.19	0.38	0.61	1									

DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)

Proportion of information requirements that have a given required quality

QualR	1	2	3	4	5	6	7	8	9	10	11	12	13
	0.05	0.05	0.5	0.3	0.1	0	0	0	0	0	0	0	0
	0.05	0.1	0.6	0.9	1								

Distribution of time until items are required (must be > 0)

TimeR

TRIA(24,48,120,5)

Distribution of user priorities (must be >= 1)

Priority_User	1	2	3	4	5	6	7	8	9	10	11	12	13
	0.2	0.2	0.2	0.2	0.2	0	0	0	0	0	0	0	0
	0.2	0.4	0.6	0.8	1								

DISC(.2,1,.4,2,.6,3,.8,4,1,5,5)

RFI_AddAttrib_U5 Attributes for additional information requirements for User 1. The assumption is that these items will only require collection if an adequate item is not found in the library.

Proportion of information requirements from each information source

InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
	0.19	0.19	0.23	0.39	0	0	0	0	0	0	0	0	0
	0.19	0.38	0.61	1									

DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)

Proportion of information requirements that have a given required quality

QualR	1	2	3	4	5	6	7	8	9	10	11	12	13
	0.05	0.05	0.5	0.3	0.1	0	0	0	0	0	0	0	0
	0.05	0.1	0.6	0.9	1								

Distribution of time until items are required (must be > 0)

TimeR

TRIA(24,36,48,5)

Distribution of user priorities (must be >= 1)

Priority_User	1	2	3	4	5	6	7	8	9	10	11	12	13
	0.4	0.4	0.3	0.3	0	0	0						
	0.4	0.7	1	1	1								

DISC(.4,1,.7,2,1,3,5)

InfoSourc DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)
TimeR TRIA(24,48,120,5)
QualR DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)
Priority_U DISC(.2,1,.4,2,.6,3,.8,4,1,5,5)

InfoSourc DISC(0.19, 1, 0.38, 2, 0.61, 3, 1, 4, 5)
TimeR TRIA(24,36,48,5)
QualR DISC(.05,1,.1,2,.6,3,.9,4,1,5,5)
Priority_U DISC(.4,1,.7,2,1,3,5)

Figure D.13 Data Request Sheet Page 12

RFI_StdSteps_U5 — Required steps for standing/standard information requirements. Proportion of items that must accomplish each step.

Steps	Percent of items required to complete	
Collect	100%	1
Process	100%	1
Exploit	0%	0
Analyze	100%	1
Produce	0%	0
Disseminate	100%	1
Integrate	0%	0

RFI_AddSteps_U5 — Required steps for additional requirements where an adequate item was *NOT* found during the library search. Assumption is that these require additional collection.

Steps	Percent of items required to complete	
Collect	100%	1
Process	100%	1
Exploit	0%	0
Analyze	100%	1
Produce	0%	0
Disseminate	100%	1
Integrate	0%	0

RFI_AddSteps_NC_U5 — Required steps for additional requirements where an adequate item was found during the library search (found, met quality requirement, met age requirement).

Steps	Percent of items required to complete	
Collect	0%	0
Process	0%	0
Exploit	10%	DISC(.9,0,1,1.5)
Analyze	20%	DISC(.8,0,1,1.5)
Produce	50%	DISC(.5,0,1,1.5)
Disseminate	Exploit || Analyze || Exploit || Analyze || Produce	
Integrate	0%	0

TimeBtwnArrivals_StdReq — Distribution of time between arrivals of standing/standard information requirements for each user. These items will not undergo a library search.

User	
1	UNIF(20,28.5)
2	UNIF(20,28.5)
3	UNIF(20,28.5)
4	UNIF(20,28.5)
5	UNIF(20,28.5)

TimeBtwnArrivals_AddReq — Distribution of time between arrivals of additional information requirements for each user. These items will undergo a library search.

User	
1	EXPO(18.5)
2	EXPO(18.5)
3	EXPO(18.5)
4	EXPO(18.5)
5	EXPO(18.5)

EntitiesPerArrival_StdReq — Number (or distribution) of entities generated for each arrival (e.g., if requests arrive in batches rather than individually)
Note that each of the items that arrive are given their own properties

User	
1	20
2	20
3	20
4	20
5	20

Figure D.14 Data Request Sheet Page 13

EntitiesPerArrival_AddReq

Number (or distribution) of entities generated for each arrival (e.g., if requests arrive in batches rather than individually)
Note that each of the items that arrive are given their own properties

User	Case 5
1	DISC(.5,10,1.15,5)
2	DISC(.5,10,1.15,5)
3	DISC(.5,10,1.15,5)
4	DISC(.5,10,1.15,5)
5	DISC(.5,10,1.15,5)

InfoProcessFusionTimes

Distribution of additional time taken during processing to perform fusion. Assumes all additional information for fusion is available.
*Not Required or planned for use during study

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0		0	0	0	0	0	0	0	0	0	0	0	0
1		0	0	0	0	0	0	0	0	0	0	0	0
2		0	0	0	0	0	0	0	0	0	0	0	0
3		0	0	0	0	0	0	0	0	0	0	0	0
4		0	0	0	0	0	0	0	0	0	0	0	0
5		0	0	0	0	0	0	0	0	0	0	0	0

InfoProcessFusionQualA

Distribution of altered achieved quality during processing as a result of fusion
Inclusion of QualA in these expressions is critical, otherwise the previous quality will be completely overwritten.
*Not Required or planned for use during study

InfoSource	QualA
1	QualA
2	QualA
3	QualA
4	QualA
5	QualA
6	QualA
7	QualA
8	QualA
9	QualA
10	QualA
11	QualA
12	QualA
13	QualA

InfoExploitFusionTimes

Distribution of additional time taken during processing to perform fusion. Assumes all additional information for fusion is available.
*Not Required or planned for use during study

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0		0	0	0	0	0	0	0	0	0	0	0	0
1		0	0	0	0	0	0	0	0	0	0	0	0
2		0	0	0	0	0	0	0	0	0	0	0	0
3		0	0	0	0	0	0	0	0	0	0	0	0
4		0	0	0	0	0	0	0	0	0	0	0	0
5		0	0	0	0	0	0	0	0	0	0	0	0

Figure D.15 Data Request Sheet Page 14

InfoExploitFusionQualA

Distribution of altered achieved quality during exploitation as a result of fusion

Inclusion of QualA in these expressions is critical, otherwise the previous quality will be completely overwritten.

***Not Required or planned for use during study**

InfoSource	
1	QualA
2	QualA
3	QualA
4	QualA
5	QualA
6	QualA
7	QualA
8	QualA
9	QualA
10	QualA
11	QualA
12	QualA
13	QualA

InfoAnalyzeFusionTimes

InfoAnalyzeFusionTimes (QualARank, InfoSource)

***Not Required or planned for use during study**

QualARank \ InfoSource	1	2	3	4	5	6	7	8	9	10	11	12	13
0		0	0	0	0	0	0	0	0	0	0	0	0
1		0	0	0	0	0	0	0	0	0	0	0	0
2		0	0	0	0	0	0	0	0	0	0	0	0
3		0	0	0	0	0	0	0	0	0	0	0	0
4		0	0	0	0	0	0	0	0	0	0	0	0
5		0	0	0	0	0	0	0	0	0	0	0	0

InfoAnalyzeFusionQualA

InfoAnalyzeFusionQualA (InfoSource)

Inclusion of QualA in these expressions is critical, otherwise the previous quality will be completely overwritten.

***Not Required or planned for use during study**

InfoSource	
1	QualA
2	QualA
3	QualA
4	QualA
5	QualA
6	QualA
7	QualA
8	QualA
9	QualA
10	QualA
11	QualA
12	QualA
13	QualA

FusionProbabilities

FusionProbabilities (i)

***Not Required or planned for use during study**

Step	Percent of items to undergo fusion
Processing	0%
Exploitation	0%
Analysis	0%

Figure D.16 Data Request Sheet Page 15

Figure D.17 Data Request Sheet Page 16

CommsPostExploitTimes — Distribution of time taken to post information from a given source to centralized repositories after exploitation

InfoSource	
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0

Resource Sets

The capacity of a resource can be specified as a fixed value or as a schedule.
Usage is assumed to be one resource per item that is processed due the the high level aggregated nature of the model, capacities should be adjusted accordingly
The distribution of a capacity is entered for a given amount of time, at the end of a schedule the entire schedule is repeated
Note that the schedules are not limited to 6 entries and can be as large as needed.

Sources — Information sources used during the collection phase. These are tasked based on the InfoSource assigned to an entity.

InfoSource	Capacity	Time	Capacity	Time	Capacity	Time	Capacity	Time	Capacity	Time	Capacity	Time
1	7		12	6	12							
2	7		12	6	12				6.5	2.6		
3	3		12	2	12				2.5			
4	3		12	2	12							
5	0											
6	0						5	3	4	2.66667		
7	0						2	1	1.5			
8	0											
9	0											
10	0											
11	0											
12	0											
13	0											

InfoProcessors — Processors are only used during the processing phase and are tasked according to the InfoSource attribute.

InfoSource	Capacity	Time	Capacity	Time	Capacity	Time	Capacity	Time	Capacity	Time	Capacity	Time
1	1	Infinite										
2	1	Infinite										
3	1	Infinite										
4	1	Infinite										
5	0											
6	0											
7	0											
8	0											
9	0											
10	0											
11	0											
12	0											
13	0											

InfoSpecialists — Analysts (InfoSpecialists) are only used during exploitation and are tasked according to their specialty in a given information source

InfoSource	Capacity	Time	Capacity	Time	Capacity	Time	Capacity	Time	Capacity	Time	Capacity	Time
1	4		12	3	12							
2	4		12	3	12							
3	2		12	1	12							
4	3		12	2	12							
5	0											
6	0											
7	0											
8	0											
9	0											
10	0											
11	0											
12	0											
13	0											

Figure D.18 Data Request Sheet Page 17

UserAnalysts

User analysts are used during the library search, dissemination, and integration phases.

User	Capacity	Time
1		1
2		1
3		1
4		1
5		1

AllSourceAnalysts

All source analysts are categorized according to their specialty (this may or may not correspond with the information sources) and used during analysis and production.

Specialty	Capacity	Time
1		5
2		4
3		2
4		2
5		0
6		0
7		0
8		0
9		0
10		0
11		0
12		0
13		0

CommsCollection

This set of resources is tasked based on information source during transmission of information after it is collected.

InfoSource	Capacity	Time
1		1
2		1
3		1
4		1
5		0
6		0
7		0
8		0
9		0
10		0
11		0
12		0
13		0

CommsPost

This resource is used during the position of information after collection, processing, and exploitation.

Capacity	Time
1	0

Additional Variables and/or Expressions

Name	Value	
PostInfo	0	Enables (1) or disables (0) use of post queues/resources in communicationss submodel

Figure D.19 Data Request Sheet Page 18

Appendix E. Implemented Statistics

The following table is a list of all the statistics that were created to gather information about the system. Note that this list does not include any of the built in statistics such as resource utilization or wait times in queues. These statistics are defined using the data collected in the variables given in Table 3.4.

Statistic	Definition
Pct_Timely_Pr_1	S_Num_Timely_Priority(1) / S_Num_Out_Priority(1)
Pct_Timely_Pr_2	S_Num_Timely_Priority(2) / S_Num_Out_Priority(2)
Pct_Timely_Pr_3	S_Num_Timely_Priority(3) / S_Num_Out_Priority(3)
Pct_Timely_Pr_4	S_Num_Timely_Priority(4) / S_Num_Out_Priority(4)
Pct_Timely_Pr_5	S_Num_Timely_Priority(5) / S_Num_Out_Priority(5)
Pct_Timely_Usr_1	S_Num_Timely_User(1) / S_Num_Out_User(1)
Pct_Timely_Usr_2	S_Num_Timely_User(2) / S_Num_Out_User(2)
Pct_Timely_Usr_3	S_Num_Timely_User(3) / S_Num_Out_User(3)
Pct_Timely_Usr_4	S_Num_Timely_User(4) / S_Num_Out_User(4)
Pct_Timely_Usr_5	S_Num_Timely_User(5) / S_Num_Out_User(5)
Pct_Timely_Add	S_Num_Timely_Standard(1) / S_Num_Out_Standard(1)
Pct_Timely_Std	S_Num_Timely_Standard(2) / S_Num_Out_Standard(2)
Pct_QualMet_Pr_1	S_Num_Qual_Priority(1) / S_Num_Out_Priority(1)
Pct_QualMet_Pr_2	S_Num_Qual_Priority(2) / S_Num_Out_Priority(2)
Pct_QualMet_Pr_3	S_Num_Qual_Priority(3) / S_Num_Out_Priority(3)
Pct_QualMet_Pr_4	S_Num_Qual_Priority(4) / S_Num_Out_Priority(4)
Pct_QualMet_Pr_5	S_Num_Qual_Priority(5) / S_Num_Out_Priority(5)
Pct_QualMet_Usr_1	S_Num_Qual_User(1) / S_Num_Out_User(1)
Pct_QualMet_Usr_2	S_Num_Qual_User(2) / S_Num_Out_User(2)
Pct_QualMet_Usr_3	S_Num_Qual_User(3) / S_Num_Out_User(3)

Statistic	Definition
Pct_QualMet_Usr_4	S_Num_Qual_User(4) / S_Num_Out_User(4)
Pct_QualMet_Usr_5	S_Num_Qual_User(5) / S_Num_Out_User(5)
Pct_QualMet_Add	S_Num_Qual_Standard(1) / S_Num_Out_Standard(1)
Pct_QualMet_Std	S_Num_Qual_Standard(2) / S_Num_Out_Standard(2)
Pct_QT_Pr_1	S_Num_QT_Priority(1) / S_Num_Out_Priority(1)
Pct_QT_Pr_2	S_Num_QT_Priority(2) / S_Num_Out_Priority(2)
Pct_QT_Pr_3	S_Num_QT_Priority(3) / S_Num_Out_Priority(3)
Pct_QT_Pr_4	S_Num_QT_Priority(4) / S_Num_Out_Priority(4)
Pct_QT_Pr_5	S_Num_QT_Priority(5) / S_Num_Out_Priority(5)
Pct_QT_Usr_1	S_Num_QT_User(1) / S_Num_Out_User(1)
Pct_QT_Usr_2	S_Num_QT_User(2) / S_Num_Out_User(2)
Pct_QT_Usr_3	S_Num_QT_User(3) / S_Num_Out_User(3)
Pct_QT_Usr_4	S_Num_QT_User(4) / S_Num_Out_User(4)
Pct_QT_Usr_5	S_Num_QT_User(5) / S_Num_Out_User(5)
Pct_QT_Add	S_Num_QT_Standard(1) / S_Num_Out_Standard(1)
Pct_QT_Std	S_Num_QT_Standard(2) / S_Num_Out_Standard(2)
S_Num_T_Pr_1	S_Num_Timely_Priority(1)
S_Num_T_Pr_2	S_Num_Timely_Priority(2)
S_Num_T_Pr_3	S_Num_Timely_Priority(3)
S_Num_T_Pr_4	S_Num_Timely_Priority(4)
S_Num_T_Pr_5	S_Num_Timely_Priority(5)
S_Num_T_Usr_1	S_Num_Timely_User(1)
S_Num_T_Usr_2	S_Num_Timely_User(2)
S_Num_T_Usr_3	S_Num_Timely_User(3)
S_Num_T_Usr_4	S_Num_Timely_User(4)
S_Num_T_Usr_5	S_Num_Timely_User(5)

Statistic	Definition
S_Num_T_Add	S_Num_Timely_Standard(1)
S_Num_T_Std	S_Num_Timely_Standard(2)
S_Num_Q_Pr_1	S_Num_Qual_Priority(1)
S_Num_Q_Pr_2	S_Num_Qual_Priority(2)
S_Num_Q_Pr_3	S_Num_Qual_Priority(3)
S_Num_Q_Pr_4	S_Num_Qual_Priority(4)
S_Num_Q_Pr_5	S_Num_Qual_Priority(5)
S_Num_Q_Usr_1	S_Num_Qual_User(1)
S_Num_Q_Usr_2	S_Num_Qual_User(2)
S_Num_Q_Usr_3	S_Num_Qual_User(3)
S_Num_Q_Usr_4	S_Num_Qual_User(4)
S_Num_Q_Usr_5	S_Num_Qual_User(5)
S_Num_Q_Add	S_Num_Qual_Standard(1)
S_Num_Q_Std	S_Num_Qual_Standard(2)
S_Num_QT_Pr_1	S_Num_QT_Priority(1)
S_Num_QT_Pr_2	S_Num_QT_Priority(2)
S_Num_QT_Pr_3	S_Num_QT_Priority(3)
S_Num_QT_Pr_4	S_Num_QT_Priority(4)
S_Num_QT_Pr_5	S_Num_QT_Priority(5)
S_Num_QT_Usr_1	S_Num_QT_User(1)
S_Num_QT_Usr_2	S_Num_QT_User(2)
S_Num_QT_Usr_3	S_Num_QT_User(3)
S_Num_QT_Usr_4	S_Num_QT_User(4)
S_Num_QT_Usr_5	S_Num_QT_User(5)
S_Num_QT_Add	S_Num_QT_Standard(1)
S_Num_QT_Std	S_Num_QT_Standard(2)

Statistic	Definition
S_Num_Out_Pr_1	S_Num_Out_Priority(1)
S_Num_Out_Pr_2	S_Num_Out_Priority(2)
S_Num_Out_Pr_3	S_Num_Out_Priority(3)
S_Num_Out_Pr_4	S_Num_Out_Priority(4)
S_Num_Out_Pr_5	S_Num_Out_Priority(5)
S_Num_Out_Usr_1	S_Num_Out_User(1)
S_Num_Out_Usr_2	S_Num_Out_User(2)
S_Num_Out_Usr_3	S_Num_Out_User(3)
S_Num_Out_Usr_4	S_Num_Out_User(4)
S_Num_Out_Usr_5	S_Num_Out_User(5)
S_Num_Out_Add	S_Num_Out_Standard(1)
S_Num_Out_Std	S_Num_Out_Standard(2)
Pct_Timely_Src_01	S_Num_Timely_Src(1) / S_Num_Out_Src(1)
Pct_Timely_Src_02	S_Num_Timely_Src(2) / S_Num_Out_Src(2)
Pct_Timely_Src_03	S_Num_Timely_Src(3) / S_Num_Out_Src(3)
Pct_Timely_Src_04	S_Num_Timely_Src(4) / S_Num_Out_Src(4)
Pct_Timely_Src_05	S_Num_Timely_Src(5) / S_Num_Out_Src(5)
Pct_Timely_Src_06	S_Num_Timely_Src(6) / S_Num_Out_Src(6)
Pct_Timely_Src_07	S_Num_Timely_Src(7) / S_Num_Out_Src(7)
Pct_Timely_Src_08	S_Num_Timely_Src(8) / S_Num_Out_Src(8)
Pct_Timely_Src_09	S_Num_Timely_Src(9) / S_Num_Out_Src(9)
Pct_Timely_Src_10	S_Num_Timely_Src(10) / S_Num_Out_Src(10)
Pct_Timely_Src_11	S_Num_Timely_Src(11) / S_Num_Out_Src(11)
Pct_Timely_Src_12	S_Num_Timely_Src(12) / S_Num_Out_Src(12)
Pct_Timely_Src_13	S_Num_Timely_Src(13) / S_Num_Out_Src(13)
Pct_QualMet_Src_01	S_Num_Qual_Src(1) / S_Num_Out_Src(1)

Statistic	Definition
Pct_QualMet_Src_02	S_Num_Qual_Src(2) / S_Num_Out_Src(2)
Pct_QualMet_Src_03	S_Num_Qual_Src(3) / S_Num_Out_Src(3)
Pct_QualMet_Src_04	S_Num_Qual_Src(4) / S_Num_Out_Src(4)
Pct_QualMet_Src_05	S_Num_Qual_Src(5) / S_Num_Out_Src(5)
Pct_QualMet_Src_06	S_Num_Qual_Src(6) / S_Num_Out_Src(6)
Pct_QualMet_Src_07	S_Num_Qual_Src(7) / S_Num_Out_Src(7)
Pct_QualMet_Src_08	S_Num_Qual_Src(8) / S_Num_Out_Src(8)
Pct_QualMet_Src_09	S_Num_Qual_Src(9) / S_Num_Out_Src(9)
Pct_QualMet_Src_10	S_Num_Qual_Src(10) / S_Num_Out_Src(10)
Pct_QualMet_Src_11	S_Num_Qual_Src(11) / S_Num_Out_Src(11)
Pct_QualMet_Src_12	S_Num_Qual_Src(12) / S_Num_Out_Src(12)
Pct_QualMet_Src_13	S_Num_Qual_Src(13) / S_Num_Out_Src(13)
Pct_QT_Src_01	S_Num_QT_Src(1) / S_Num_Out_Src(1)
Pct_QT_Src_02	S_Num_QT_Src(2) / S_Num_Out_Src(2)
Pct_QT_Src_03	S_Num_QT_Src(3) / S_Num_Out_Src(3)
Pct_QT_Src_04	S_Num_QT_Src(4) / S_Num_Out_Src(4)
Pct_QT_Src_05	S_Num_QT_Src(5) / S_Num_Out_Src(5)
Pct_QT_Src_06	S_Num_QT_Src(6) / S_Num_Out_Src(6)
Pct_QT_Src_07	S_Num_QT_Src(7) / S_Num_Out_Src(7)
Pct_QT_Src_08	S_Num_QT_Src(8) / S_Num_Out_Src(8)
Pct_QT_Src_09	S_Num_QT_Src(9) / S_Num_Out_Src(9)
Pct_QT_Src_10	S_Num_QT_Src(10) / S_Num_Out_Src(10)
Pct_QT_Src_11	S_Num_QT_Src(11) / S_Num_Out_Src(11)
Pct_QT_Src_12	S_Num_QT_Src(12) / S_Num_Out_Src(12)
Pct_QT_Src_13	S_Num_QT_Src(13) / S_Num_Out_Src(13)
S_Avg_WaitTime_Pr_1	S_Tot_WaitTime_Priority(1) / S_Num_Out_Priority(1)

Statistic	Definition
S_Avg_WaitTime_Pr_2	S_Tot_WaitTime_Priority(2) / S_Num_Out_Priority(2)
S_Avg_WaitTime_Pr_3	S_Tot_WaitTime_Priority(3) / S_Num_Out_Priority(3)
S_Avg_WaitTime_Pr_4	S_Tot_WaitTime_Priority(4) / S_Num_Out_Priority(4)
S_Avg_WaitTime_Pr_5	S_Tot_WaitTime_Priority(5) / S_Num_Out_Priority(5)
S_Max_WaitTime_Pr_1	S_Max_WaitTime_Priority(1)
S_Max_WaitTime_Pr_2	S_Max_WaitTime_Priority(2)
S_Max_WaitTime_Pr_3	S_Max_WaitTime_Priority(3)
S_Max_WaitTime_Pr_4	S_Max_WaitTime_Priority(4)
S_Max_WaitTime_Pr_5	S_Max_WaitTime_Priority(5)

Appendix F. Additional Simulation Results

The following sections include additional simulation results that were not presented in Chapter 4.

F.1 Baseline Results

alpha 0.05

Requirements

Averages

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.9983	0.9684	0.9671	1	0.6558	0.8493	0.6445
2	0.9693	0.9675	0.9393	2	0.7575	0.8193	0.7220
3	0.6229	0.9358	0.6032	3	0.7016	0.8480	0.6885
4	0.3275	0.5755	0.3153	4	0.7323	0.8532	0.7235
5	0.0946	0.2250	0.0903	5	0.7889	0.8106	0.7419

Half-Widths

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.000011	0.000049	0.000048	1	0.000286	0.000257	0.000292
2	0.000181	0.000060	0.000182	2	0.000328	0.000173	0.000301
3	0.000766	0.000128	0.000741	3	0.000353	0.000262	0.000342
4	0.000702	0.000975	0.000684	4	0.000275	0.000238	0.000294
5	0.000503	0.001012	0.000471	5	0.000364	0.000227	0.000344

Averages

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.7380	0.6966	0.6165	Additional	0.8213	0.9566	0.7960
2	0.7710	0.8638	0.7710	Standard	0.6488	0.7357	0.6275
3	0.7011	0.8744	0.7011				
4	0.7160	0.8681	0.7160				

Half-Widths

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.000460	0.000374	0.000390	Additional	0.000297	0.000050	0.000276
2	0.000368	0.000280	0.000368	Standard	0.000305	0.000383	0.000297
3	0.000340	0.000284	0.000340				
4	0.000321	0.000277	0.000321				

Figure F.1 Baseline simulation results part 1.

WIP

Averages

	Average	Max	
RFI	379.55	960.04	789 min(max)

Half-Widths

	Average	Max
RFI	0.9098	9.2790

Resource Utilization

CommsPost		Averages	Half-Widths
Averages	User	Analyst_User	Analyst_User
0	1	0.1880	0.000233
Half-Width	2	0.2037	0.000218
0	3	0.1941	0.000236
	4	0.1991	0.000275
	5	0.2077	0.000287

Averages

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.9983	0.0121	0.0223	0.6664	0.9784
InfoSource 2	1.0011	0.0121	0.0290	0.5705	0.9568
InfoSource 3	1.0023	0.0148	0.0361	1.0123	0.9434
InfoSource 4	1.0019	0.0249	0.0617	0.9934	0.9361

Half-Widths

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.000137	0.000007	0.000008	0.000399	0.000115
InfoSource 2	0.000068	0.000006	0.000004	0.000331	0.000240
InfoSource 3	0.000091	0.000005	0.000005	0.000119	0.000321
InfoSource 4	0.000083	0.000009	0.000015	0.000190	0.000352

Figure F.2 Baseline simulation results part 2.

Queue Number Waiting

Averages

	Average	Max
Search Library	3.04	59.04
Collect Data	257.43	842.04
TX from Collection	0.00	3.92
Process Data	0.00	3.68
Exploit Data	57.91	212.96
Analyze Intel	13.73	118.84
Produce Intel	6.82	68.48
Disseminate Intel	0.66	19.32
Integrate Intel	0.00	0.00

Half-Widths

	Average	Max
Search Library	0.0025	0.3998
Collect Data	0.8823	9.4037
TX from Collection	0.0000	0.0627
Process Data	0.0000	0.0518
Exploit Data	0.2594	2.3148
Analyze Intel	0.0701	1.4936
Produce Intel	0.0183	0.7288
Disseminate Intel	0.0007	0.2232
Integrate Intel	0.0000	0.0000

Queue Wait Time

Averages

	Average	Max
Search Library	0.8763	8.7080
Collect Data	34.1546	1309.7055
TX from Collection	0.0003	0.0958
Process Data	0.0007	0.0893
Exploit Data	15.2552	1696.3219
Analyze Intel	3.7641	288.8713
Produce Intel	2.0291	137.7551
Disseminate Intel	0.1195	9.6560
Integrate Intel	=	=

Half-Widths

	Average	Max
Search Library	0.000436	0.103764
Collect Data	0.107294	18.737710
TX from Collection	0.000003	0.000976
Process Data	0.000004	0.001392
Exploit Data	0.061486	36.319224
Analyze Intel	0.017433	3.809782
Produce Intel	0.005114	1.966274
Disseminate Intel	0.000133	0.153196
Integrate Intel	=	=

Wait Time by Priority

Averages

Priority	Average	Max
1	2.98	22.41
2	9.68	71.86
3	31.09	277.74
4	92.35	883.76
5	215.63	1773.62

Half-Widths

Priority	Average	Max
1	0.0019	0.1965
2	0.0116	0.9443
3	0.0447	2.8621
4	0.1916	15.9196
5	0.8180	35.9756

Figure F.3 Baseline simulation results part 3.

F.2 Case 1 Results

alpha 0.05

Requirements

Averages

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	1.0000	0.9687	0.9687	1	0.6856	0.8408	0.6731
2	0.9996	0.9685	0.9681	2	0.7837	0.8108	0.7461
3	0.7554	0.9594	0.7290	3	0.7271	0.8402	0.7130
4	0.2347	0.5491	0.2251	4	0.7503	0.8452	0.7412
5	0.0356	0.1216	0.0337	5	0.8054	0.8010	0.7560

Half-Widths

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.000000	0.000047	0.000047	1	0.000304	0.000260	0.000296
2	0.000030	0.000046	0.000053	2	0.000340	0.000287	0.000331
3	0.001014	0.000130	0.001005	3	0.000335	0.000230	0.000324
4	0.001060	0.001862	0.001038	4	0.000448	0.000346	0.000448
5	0.000440	0.000934	0.000415	5	0.000327	0.000279	0.000330

Averages

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.7870	0.7016	0.6577	Additional	0.9006	0.9654	0.8716
2	0.8029	0.8539	0.8029	Standard	0.6256	0.7131	0.6049
3	0.7149	0.8645	0.7149				
4	0.7281	0.8543	0.7281				

Half-Widths

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.000474	0.000377	0.000407	Additional	0.000358	0.000061	0.000333
2	0.000489	0.000376	0.000489	Standard	0.000357	0.000503	0.000355
3	0.000424	0.000361	0.000424				
4	0.000368	0.000353	0.000368				

Figure F.4 Case 1 simulation results part 1.

WIP

Averages

	Average	Max	
RFI	993.49	2379.72	1820 min(max)

Half-Widths

	Average	Max
RFI	6.3689	25.5148

Resource Utilization

CommsPost		Averages	Half-Widths
Averages	User	Analyst_User	Analyst_User
0	1	0.1925	0.000257
	2	0.2070	0.000313
Half-Width	3	0.1983	0.000260
0	4	0.2011	0.000264
	5	0.2110	0.000325

Averages

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	1.0034	0.0122	0.0229	0.6877	0.9901
InfoSource 2	1.0036	0.0122	0.0295	0.5803	0.9803
InfoSource 3	1.0063	0.0149	0.0364	1.0190	0.9742
InfoSource 4	1.0052	0.0249	0.0620	1.0046	0.9707

Half-Widths

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.000059	0.000007	0.000010	0.000418	0.000077
InfoSource 2	0.000027	0.000004	0.000003	0.000347	0.000153
InfoSource 3	0.000032	0.000006	0.000005	0.000045	0.000210
InfoSource 4	0.000025	0.000008	0.000019	0.000121	0.000239

Figure F.5 Case 1 simulation results part 2.

Queue Number Waiting

Averages

	Average	Max
Search Library	3.05	60.16
Collect Data	349.87	1152.00
TX from Collection	0.00	3.84
Process Data	0.00	3.36
Exploit Data	85.80	286.76
Analyze Intel	19.24	140.64
Produce Intel	8.52	80.32
Disseminate Intel	0.68	19.84
Integrate Intel	0.00	0.00

Half-Widths

	Average	Max
Search Library	0.0040	0.4841
Collect Data	1.7445	15.0702
TX from Collection	0.0000	0.0702
Process Data	0.0000	0.0469
Exploit Data	0.6464	4.3303
Analyze Intel	0.1053	1.5529
Produce Intel	0.0338	1.0107
Disseminate Intel	0.0011	0.2180
Integrate Intel	0.0000	0.0000

Queue Wait Time

Averages

	Average	Max
Search Library	0.8766	9.1206
Collect Data	46.3907	1799.5178
TX from Collection	0.0003	0.0979
Process Data	0.0008	0.0839
Exploit Data	22.4632	3012.5203
Analyze Intel	5.2179	392.5094
Produce Intel	2.5045	170.2537
Disseminate Intel	0.1205	9.4541
Integrate Intel	=	=

Half-Widths

	Average	Max
Search Library	0.000497	0.083905
Collect Data	0.200104	23.890979
TX from Collection	0.000004	0.000912
Process Data	0.000005	0.001214
Exploit Data	0.160679	75.878188
Analyze Intel	0.026361	3.833825
Produce Intel	0.009557	2.636227
Disseminate Intel	0.000196	0.167818
Integrate Intel	=	=

Wait Time by Priority

Averages

Priority	Average	Max
1	3.16	21.84
2	10.84	91.78
3	58.43	459.04
4	297.66	4033.54
5	683.02	11331.6

Half-Widths

Priority	Average	Max
1	0.0014	0.2093
2	0.0254	0.8116
3	0.1176	7.9202
4	2.2884	77.2516
5	6.1770	324.0495

Figure F.6 Case 1 simulation results part 3.

F.3 Case 2 Results

alpha 0.05

Requirements

Averages

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.9998	0.9685	0.9683	1	0.6713	0.8451	0.6593
2	0.9907	0.9678	0.9591	2	0.7647	0.8148	0.7283
3	0.6694	0.9449	0.6476	3	0.7120	0.8436	0.6985
4	0.2820	0.5534	0.2713	4	0.7397	0.8483	0.7308
5	0.0662	0.1825	0.0631	5	0.7938	0.8060	0.7458

Half-Widths

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.000004	0.000069	0.000070	1	0.000417	0.000325	0.000414
2	0.000121	0.000066	0.000126	2	0.000523	0.000277	0.000494
3	0.001053	0.000174	0.001024	3	0.000371	0.000287	0.000366
4	0.001205	0.001573	0.001170	4	0.000358	0.000308	0.000354
5	0.000576	0.001251	0.000551	5	0.000439	0.000307	0.000450

Averages

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.7561	0.6953	0.6312	Additional	0.8558	0.9599	0.8286
2	0.7833	0.8595	0.7833	Standard	0.6367	0.7243	0.6157
3	0.7064	0.8696	0.7064				
4	0.7215	0.8620	0.7215				

Half-Widths

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.000327	0.000310	0.000328	Additional	0.000418	0.000069	0.000411
2	0.000533	0.000307	0.000533	Standard	0.000473	0.000546	0.000463
3	0.000510	0.000396	0.000510				
4	0.000408	0.000312	0.000408				

Figure F.7 Case 2 simulation results part 1.

WIP

Averages

	Average	Max	
RFI	507.34	1293.56	1022 min(max)

Half-Widths

	Average	Max
RFI	1.6942	13.3594

Resource Utilization

CommsPost		Averages	Half-Widths
Averages	User	Analyst_User	Analyst_User
0	1	0.1908	0.000256
Half-Width	2	0.2057	0.000278
0	3	0.1957	0.000257
	4	0.1995	0.000240
	5	0.2099	0.000262

Averages

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	1.0007	0.0121	0.0226	0.6743	0.9838
InfoSource 2	1.0024	0.0122	0.0292	0.5733	0.9677
InfoSource 3	1.0047	0.0149	0.0363	1.0162	0.9574
InfoSource 4	1.0035	0.0249	0.0619	0.9973	0.9519

Half-Widths

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.000099	0.000007	0.000010	0.000388	0.000118
InfoSource 2	0.000056	0.000004	0.000004	0.000219	0.000236
InfoSource 3	0.000062	0.000005	0.000006	0.000082	0.000308
InfoSource 4	0.000047	0.000010	0.000016	0.000118	0.000316

Figure F.8 Case 2 simulation results part 2.

Queue Number Waiting

Averages

	Average	Max
Search Library	3.05	60.16
Collect Data	349.87	1152.00
TX from Collection	0.00	3.84
Process Data	0.00	3.36
Exploit Data	85.80	286.76
Analyze Intel	19.24	140.64
Produce Intel	8.52	80.32
Disseminate Intel	0.68	19.84
Integrate Intel	0.00	0.00

Half-Widths

	Average	Max
Search Library	0.0040	0.4841
Collect Data	1.7445	15.0702
TX from Collection	0.0000	0.0702
Process Data	0.0000	0.0469
Exploit Data	0.6464	4.3303
Analyze Intel	0.1053	1.5529
Produce Intel	0.0338	1.0107
Disseminate Intel	0.0011	0.2180
Integrate Intel	0.0000	0.0000

Queue Wait Time

Averages

	Average	Max
Search Library	0.8766	9.1206
Collect Data	46.3907	1799.5178
TX from Collection	0.0003	0.0979
Process Data	0.0008	0.0839
Exploit Data	22.4632	3012.5203
Analyze Intel	5.2179	392.5094
Produce Intel	2.5045	170.2537
Disseminate Intel	0.1205	9.4541
Integrate Intel	=	=

Half-Widths

	Average	Max
Search Library	0.000497	0.083905
Collect Data	0.200104	23.890979
TX from Collection	0.000004	0.000912
Process Data	0.000005	0.001214
Exploit Data	0.160679	75.878188
Analyze Intel	0.026361	3.833825
Produce Intel	0.009557	2.636227
Disseminate Intel	0.000196	0.167818
Integrate Intel	=	=

Wait Time by Priority

Averages

Priority	Average	Max
1	3.06	22.08
2	10.29	75.5
3	38.73	320.57
4	130.3	1318.65
5	311.74	3091.8

Half-Widths

Priority	Average	Max
1	0.0025	0.3078
2	0.0200	0.7831
3	0.0773	3.2670
4	0.3977	22.5456
5	1.4061	75.8207

Figure F.9 Case 2 simulation results part 3.

F.4 Case 3 Results

alpha 0.05

Requirements

Averages

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.9984	1.0000	0.9984	1	0.6585	0.9487	0.6585
2	0.9699	0.9999	0.9699	2	0.7589	0.9487	0.7589
3	0.6291	0.9931	0.6291	3	0.7051	0.9489	0.7051
4	0.3337	0.8456	0.3337	4	0.7360	0.9485	0.7360
5	0.0996	0.6997	0.0996	5	0.7927	0.9488	0.7927

Half-Widths

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.000012	0.000001	0.000012	1	0.000261	0.000112	0.000261
2	0.000130	0.000003	0.000130	2	0.000445	0.000160	0.000445
3	0.000887	0.000055	0.000887	3	0.000404	0.000135	0.000404
4	0.001091	0.000602	0.001091	4	0.000424	0.000122	0.000424
5	0.000577	0.000572	0.000577	5	0.000385	0.000136	0.000385

Averages

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.7464	0.9473	0.7464	Additional	0.8244	0.9985	0.8244
2	0.7752	0.9482	0.7752	Standard	0.6516	0.9072	0.6516
3	0.7022	0.9509	0.7022				
4	0.7170	0.9483	0.7170				

Half-Widths

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.000563	0.000191	0.000563	Additional	0.000377	0.000017	0.000377
2	0.000436	0.000129	0.000436	Standard	0.000409	0.000220	0.000409
3	0.000352	0.000113	0.000352				
4	0.000445	0.000148	0.000445				

Figure F.10 Case 3 simulation results part 1.

WIP

Averages

	Average	Max	
RFI	370.58	920.48	718 min(max)

Half-Widths

	Average	Max
RFI	1.0255	9.5987

Resource Utilization

CommsPost		Averages	Half-Widths
Averages	User	Analyst_User	Analyst_User
0	1	0.1882	0.000258
	2	0.2038	0.000219
Half-Width	3	0.1961	0.000206
0	4	0.1989	0.000216
	5	0.2084	0.000266

Averages

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.9973	0.0121	0.0223	0.6677	0.9797
InfoSource 2	1.0008	0.0121	0.0290	0.5712	0.9595
InfoSource 3	1.0022	0.0149	0.0361	1.0131	0.9467
InfoSource 4	1.0017	0.0249	0.0617	0.9936	0.9402

Half-Widths

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.000099	0.000006	0.000009	0.000419	0.000086
InfoSource 2	0.000071	0.000006	0.000004	0.000228	0.000187
InfoSource 3	0.000091	0.000006	0.000005	0.000078	0.000251
InfoSource 4	0.000089	0.000011	0.000018	0.000123	0.000290

Figure F.11 Case 3 simulation results part 2.

Queue Number Waiting

Averages

	Average	Max
Search Library	3.05	59.16
Collect Data	245.88	805.44
TX from Collection	0.00	3.64
Process Data	0.00	3.36
Exploit Data	59.37	206.28
Analyze Intel	14.42	118.40
Produce Intel	7.20	71.92
Disseminate Intel	0.67	19.68
Integrate Intel	0.00	0.00

Half-Widths

	Average	Max
Search Library	0.0031	0.6727
Collect Data	1.0529	9.6125
TX from Collection	0.0000	0.0404
Process Data	0.0000	0.0469
Exploit Data	0.2162	1.4740
Analyze Intel	0.0571	1.3578
Produce Intel	0.0220	1.3800
Disseminate Intel	0.0010	0.2307
Integrate Intel	0.0000	0.0000

Queue Wait Time

Averages

	Average	Max
Search Library	0.8786	8.9236
Collect Data	32.7077	1214.0334
TX from Collection	0.0003	0.0906
Process Data	0.0007	0.0839
Exploit Data	15.6284	1684.6015
Analyze Intel	3.9453	286.4975
Produce Intel	2.1299	151.2150
Disseminate Intel	0.1209	10.1141
Integrate Intel	=	=

Half-Widths

	Average	Max
Search Library	0.000556	0.124577
Collect Data	0.132673	20.079266
TX from Collection	0.000003	0.000894
Process Data	0.000005	0.001037
Exploit Data	0.054748	25.128307
Analyze Intel	0.014454	4.086079
Produce Intel	0.006356	2.603467
Disseminate Intel	0.000187	0.200957
Integrate Intel	=	=

Wait Time by Priority

Averages

Priority	Average	Max
1	2.98	20.82
2	9.61	67.86
3	30.71	258.58
4	91.18	867.8
5	206.57	1746.68

Half-Widths

Priority	Average	Max
1	0.0015	0.2299
2	0.0106	0.6756
3	0.0535	3.9127
4	0.2538	14.1835
5	0.9477	24.2340

Figure F.12 Case 3 simulation results part 3.

F.5 Case 4 Results

alpha 0.05

Requirements

Averages

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.9984	1.0000	0.9984	1	0.6549	0.8725	0.6549
2	0.9690	0.9998	0.9690	2	0.7565	0.8725	0.7565
3	0.6220	0.9754	0.6220	3	0.7019	0.8729	0.7019
4	0.3244	0.6158	0.3244	4	0.7317	0.8726	0.7317
5	0.0932	0.2710	0.0932	5	0.7894	0.8729	0.7894

Half-Widths

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.000010	0.000001	0.000010	1	0.000393	0.000320	0.000393
2	0.000189	0.000009	0.000190	2	0.000356	0.000234	0.000355
3	0.000754	0.000143	0.000754	3	0.000320	0.000298	0.000320
4	0.001037	0.001220	0.001037	4	0.000335	0.000281	0.000335
5	0.000638	0.001327	0.000638	5	0.000444	0.000321	0.000443

Averages

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.7406	0.8632	0.7406	Additional	0.8216	0.9924	0.8215
2	0.7718	0.8724	0.7718	Standard	0.6476	0.7725	0.6476
3	0.7006	0.8807	0.7006				
4	0.7138	0.8728	0.7138				

Half-Widths

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.000545	0.000372	0.000545	Additional	0.000327	0.000045	0.000327
2	0.000432	0.000289	0.000432	Standard	0.000425	0.000518	0.000425
3	0.000388	0.000312	0.000388				
4	0.000322	0.000301	0.000322				

Figure F.13 Case 4 simulation results part 1.

WIP

Averages

	Average	Max	
RFI	381.31	949.36	809 min(max)

Half-Widths

	Average	Max
RFI	1.0522	8.6682

Resource Utilization

CommsPost		Averages	Half-Widths
Averages	User	Analyst_User	Analyst_User
0	1	0.1882	0.000266
Half-Width	2	0.2032	0.000246
0	3	0.1953	0.000320
	4	0.1999	0.000261
	5	0.2091	0.000316

Averages

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.9979	0.0121	0.0223	0.6677	0.9782
InfoSource 2	1.0010	0.0121	0.0290	0.5707	0.9563
InfoSource 3	1.0026	0.0149	0.0361	1.0128	0.9428
InfoSource 4	1.0020	0.0248	0.0616	0.9934	0.9356

Half-Widths

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.000090	0.000008	0.000009	0.000332	0.000113
InfoSource 2	0.000074	0.000005	0.000004	0.000236	0.000225
InfoSource 3	0.000094	0.000005	0.000006	0.000125	0.000299
InfoSource 4	0.000078	0.000009	0.000017	0.000181	0.000393

Figure F.14 Case 4 simulation results part 2.

Queue Number Waiting

Averages

	Average	Max
Search Library	3.05	60.44
Collect Data	258.87	828.68
TX from Collection	0.00	3.60
Process Data	0.00	3.48
Exploit Data	58.17	205.08
Analyze Intel	13.74	112.36
Produce Intel	6.85	68.68
Disseminate Intel	0.67	19.00
Integrate Intel	0.00	0.00

Half-Widths

	Average	Max
Search Library	0.0033	0.6754
Collect Data	1.1208	7.9886
TX from Collection	0.0000	0.0477
Process Data	0.0000	0.0421
Exploit Data	0.2967	1.5637
Analyze Intel	0.0542	1.2131
Produce Intel	0.0209	0.7795
Disseminate Intel	0.0010	0.2347
Integrate Intel	0.0000	0.0000

Queue Wait Time

Averages

	Average	Max
Search Library	0.8755	8.7620
Collect Data	34.3492	1274.3049
TX from Collection	0.0003	0.0979
Process Data	0.0007	0.0855
Exploit Data	15.2752	1683.6870
Analyze Intel	3.7639	263.0930
Produce Intel	2.0384	137.8200
Disseminate Intel	0.1205	9.9034
Integrate Intel	=	=

Half-Widths

	Average	Max
Search Library	0.000400	0.117231
Collect Data	0.136397	15.285306
TX from Collection	0.000003	0.000848
Process Data	0.000005	0.000965
Exploit Data	0.070643	36.890073
Analyze Intel	0.013363	3.542768
Produce Intel	0.005781	2.553720
Disseminate Intel	0.000186	0.161694
Integrate Intel	=	=

Wait Time by Priority

Averages

Priority	Average	Max
1	2.98	21.39
2	9.69	67.25
3	31.12	267.87
4	92.5	897.75
5	217.43	1765.8

Half-Widths

Priority	Average	Max
1	0.0021	0.1792
2	0.0135	0.5753
3	0.0414	4.6343
4	0.2280	17.6601
5	0.9201	35.1022

Figure F.15 Case 4 simulation results part 3.

F.6 Case 5 Results

alpha 0.05

Requirements

Averages

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.9974	0.9688	0.9668	1	0.5112	0.7744	0.5028
2	0.7858	0.9580	0.7616	2	0.5992	0.7493	0.5722
3	0.2240	0.7944	0.2167	3	0.5536	0.7722	0.5435
4	0.0510	0.1069	0.0485	4	0.5841	0.7773	0.5772
5	0.0071	0.0172	0.0067	5	0.6313	0.7416	0.5941

Half-Widths

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.000012	0.000038	0.000037	1	0.000437	0.000363	0.000430
2	0.000809	0.000111	0.000791	2	0.000457	0.000394	0.000422
3	0.000781	0.000494	0.000760	3	0.000498	0.000416	0.000488
4	0.000480	0.000898	0.000462	4	0.000435	0.000415	0.000430
5	0.000155	0.000299	0.000146	5	0.000513	0.000478	0.000483

Averages

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.5771	0.5984	0.4830	Additional	0.6424	0.9292	0.6228
2	0.6080	0.7894	0.6080	Standard	0.4911	0.5514	0.4753
3	0.5575	0.8045	0.5575				
4	0.5703	0.8060	0.5703				

Half-Widths

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.000478	0.000378	0.000411	Additional	0.000441	0.000086	0.000427
2	0.000553	0.000465	0.000553	Standard	0.000608	0.000682	0.000589
3	0.000478	0.000359	0.000478				
4	0.000560	0.000604	0.000560				

Figure F.16 Case 5 simulation results part 1.

WIP

Averages

	Average	Max	
RFI	2123.01	5269.68	3279 min(max)
			7526 max(max)

Half-Widths

	Average	Max
RFI	27.3718	85.5808

Resource Utilization

CommsPost		Averages	Half-Widths
	User	Analyst_User	Analyst_User
Averages	1	0.2312	0.000347
0	2	0.2454	0.000388
Half-Width	3	0.2370	0.000436
0	4	0.2445	0.000321
	5	0.2514	0.000428

Averages

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	1.0044	0.0122	0.0221	0.6589	0.9681
InfoSource 2	1.0039	0.0122	0.0286	0.5633	0.9352
InfoSource 3	1.0069	0.0149	0.0361	1.0047	0.9164
InfoSource 4	1.0056	0.0249	0.0616	0.9831	0.9063

Half-Widths

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.000020	0.000006	0.000009	0.000387	0.000139
InfoSource 2	0.000010	0.000005	0.000004	0.000353	0.000271
InfoSource 3	0.000016	0.000007	0.000005	0.000144	0.000335
InfoSource 4	0.000018	0.000011	0.000017	0.000177	0.000355

Figure F.17 Case 5 simulation results part 2.

Queue Number Waiting

Averages

	Average	Max
Search Library	7.69	107.00
Collect Data	2010.36	5158.88
TX from Collection	0.01	4.80
Process Data	0.01	4.12
Exploit Data	45.43	208.48
Analyze Intel	12.03	116.96
Produce Intel	6.08	70.24
Disseminate Intel	1.51	33.24
Integrate Intel	0.00	0.00

Half-Widths

	Average	Max
Search Library	0.0098	0.9380
Collect Data	27.4263	84.9515
TX from Collection	0.0002	0.0674
Process Data	0.0002	0.0550
Exploit Data	0.1712	2.5941
Analyze Intel	0.0455	2.1581
Produce Intel	0.0189	0.7738
Disseminate Intel	0.0022	0.2941
Integrate Intel	0.0000	0.0000

Queue Wait Time

Averages

	Average	Max
Search Library	1.4555	15.9069
Collect Data	207.9107	8641.5306
TX from Collection	0.0008	0.1149
Process Data	0.0014	0.0985
Exploit Data	11.9811	1053.0225
Analyze Intel	3.3242	236.9269
Produce Intel	1.8438	131.7508
Disseminate Intel	0.2789	19.4595
Integrate Intel	=	=

Half-Widths

	Average	Max
Search Library	0.000698	0.198324
Collect Data	3.177007	210.679166
TX from Collection	0.000026	0.001575
Process Data	0.000036	0.000988
Exploit Data	0.040568	20.118175
Analyze Intel	0.011307	4.372985
Produce Intel	0.005358	3.244813
Disseminate Intel	0.000404	0.343637
Integrate Intel	=	=

Wait Time by Priority

Averages

Priority	Average	Max
1	4.23	31.42
2	20.12	139.46
3	66.48	625.05
4	380.97	2332.61
5	1792.72	8641.53

Half-Widths

Priority	Average	Max
1	0.0038	0.1988
2	0.0349	2.5756
3	0.1593	10.4808
4	3.3455	40.2363
5	32.1296	210.6792

Figure F.18 Case 5 simulation results part 3.

F.7 Case 6 Results

alpha 0.05

Requirements

Averages

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.9970	0.9691	0.9666	1	0.5385	0.8418	0.5281
2	0.8268	0.9674	0.7976	2	0.7896	0.8216	0.7536
3	0.4779	0.9306	0.4581	3	0.5649	0.8413	0.5531
4	0.2918	0.5611	0.2790	4	0.5811	0.8457	0.5730
5	0.1024	0.1702	0.0971	5	0.8096	0.8118	0.7617

Half-Widths

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.000024	0.000044	0.000044	1	0.000468	0.000242	0.000462
2	0.000462	0.000060	0.000456	2	0.000360	0.000284	0.000348
3	0.000605	0.000192	0.000569	3	0.000440	0.000239	0.000435
4	0.000898	0.001246	0.000862	4	0.000449	0.000255	0.000435
5	0.000679	0.001650	0.000656	5	0.000409	0.000276	0.000390

Averages

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.7122	0.6859	0.5922	Additional	0.7347	0.9546	0.7099
2	0.7753	0.8641	0.7753	Standard	0.5923	0.7287	0.5709
3	0.5978	0.8723	0.5978				
4	0.6080	0.8654	0.6080				

Half-Widths

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.000503	0.000397	0.000437	Additional	0.000332	0.000073	0.000324
2	0.000359	0.000257	0.000359	Standard	0.000537	0.000425	0.000518
3	0.000630	0.000233	0.000630				
4	0.000550	0.000322	0.000550				

Figure F.19 Case 6 simulation results part 1.

WIP

Averages

	Average	Max	
RFI	2480.28	4596.72	2990 min(max)

Half-Widths

	Average	Max
RFI	32.3647	66.5620

Resource Utilization

CommsPost		Averages	Half-Widths
Averages	User	Analyst_User	Analyst_User
0	1	0.1687	0.000289
Half-Width	2	0.2086	0.000318
0	3	0.1726	0.000215
	4	0.1750	0.000237
	5	0.2121	0.000222

Averages

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.9980	0.0121	0.0223	0.9262	0.9505
InfoSource 2	1.0010	0.0121	0.0290	0.8414	0.8985
InfoSource 3	1.0025	0.0149	0.0361	1.0293	0.8683
InfoSource 4	1.0021	0.0249	0.0617	1.0122	0.8533

Half-Widths

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.000123	0.000006	0.000009	0.000410	0.000179
InfoSource 2	0.000079	0.000006	0.000004	0.000375	0.000374
InfoSource 3	0.000086	0.000006	0.000006	0.000040	0.000463
InfoSource 4	0.000057	0.000009	0.000020	0.000025	0.000519

Figure F.20 Case 6 simulation results part 2.

Queue Number Waiting

Averages

	Average	Max
Search Library	3.03	58.96
Collect Data	256.80	846.52
TX from Collection	0.00	3.60
Process Data	0.00	3.68
Exploit Data	2169.46	4088.24
Analyze Intel	6.21	75.24
Produce Intel	3.08	45.52
Disseminate Intel	0.58	18.76
Integrate Intel	0.00	0.00

Half-Widths

	Average	Max
Search Library	0.0029	0.4206
Collect Data	0.9797	9.8619
TX from Collection	0.0000	0.0477
Process Data	0.0000	0.0570
Exploit Data	32.3666	64.8806
Analyze Intel	0.0299	0.6933
Produce Intel	0.0092	0.5700
Disseminate Intel	0.0007	0.2054
Integrate Intel	0.0000	0.0000

Queue Wait Time

Averages

	Average	Max
Search Library	0.8729	8.3167
Collect Data	34.0720	1286.2925
TX from Collection	0.0003	0.0999
Process Data	0.0007	0.0828
Exploit Data	316.0030	22116.5137
Analyze Intel	1.8240	159.7927
Produce Intel	1.0468	76.8210
Disseminate Intel	0.1170	9.4148
Integrate Intel	=	=

Half-Widths

	Average	Max
Search Library	0.000348	0.072272
Collect Data	0.121317	21.771827
TX from Collection	0.000003	0.001117
Process Data	0.000005	0.000919
Exploit Data	8.555686	673.193582
Analyze Intel	0.008161	3.068029
Produce Intel	0.002794	0.995185
Disseminate Intel	0.000150	0.119430
Integrate Intel	=	=

Wait Time by Priority

Averages

Priority	Average	Max
1	3.85	35.65
2	17.8	200.75
3	98.71	1391.17
4	1001.44	15520.27
5	505.49	16799.32

Half-Widths

Priority	Average	Max
1	0.0042	0.2979
2	0.0315	2.2636
3	0.5795	23.8858
4	25.1839	419.2029
5	26.5165	1048.4747

Figure F.21 Case 6 simulation results part 3.

F.8 Case 7 Results

alpha 0.05

Requirements

Averages

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.9974	0.9681	0.9661	1	0.4932	0.8498	0.4852
2	0.8667	0.9659	0.8411	2	0.5420	0.8105	0.5168
3	0.3253	0.9306	0.3171	3	0.6194	0.8488	0.6082
4	0.1075	0.5681	0.1050	4	0.7313	0.8531	0.7225
5	0.0268	0.2191	0.0260	5	0.6021	0.8044	0.5664

Half-Widths

Priority	Timely	QualMet	Both	User	Timely	QualMet	Both
1	0.000017	0.000042	0.000047	1	0.000255	0.000293	0.000262
2	0.000566	0.000046	0.000537	2	0.000360	0.000300	0.000352
3	0.000646	0.000191	0.000613	3	0.000307	0.000337	0.000304
4	0.000348	0.001405	0.000334	4	0.000330	0.000326	0.000325
5	0.000141	0.001135	0.000137	5	0.000323	0.000227	0.000292

Averages

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.5968	0.6845	0.5031	Additional	0.6883	0.9543	0.6683
2	0.6241	0.8644	0.6241	Standard	0.5222	0.7324	0.5062
3	0.5801	0.8744	0.5801				
4	0.5957	0.8665	0.5957				

Half-Widths

InfoSource	Timely	QualMet	Both		Timely	QualMet	Both
1	0.000383	0.000397	0.000318	Additional	0.000375	0.000069	0.000355
2	0.000395	0.000347	0.000395	Standard	0.000265	0.000523	0.000254
3	0.000279	0.000323	0.000279				
4	0.000261	0.000359	0.000261				

Figure F.22 Case 7 simulation results part 1.

WIP

Averages

	Average	Max	
RFI	676.67	1550.44	1977 max(max)
			1278 min(max)

Half-Widths

	Average	Max
RFI	3.4535	14.0264

Resource Utilization

CommsPost		Averages	Half-Widths
Averages	User	Analyst_User	Analyst_User
0	1	0.1622	0.000216
	2	0.1704	0.000244
Half-Width	3	0.1827	0.000256
0	4	0.1991	0.000259
	5	0.1795	0.000198

Averages

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.9984	0.0121	0.0224	0.6685	0.9999
InfoSource 2	1.0012	0.0121	0.0290	0.5700	0.9998
InfoSource 3	1.0026	0.0149	0.0361	1.0128	0.9998
InfoSource 4	1.0018	0.0248	0.0616	0.9930	0.9998

Half-Widths

	Source	CommsCollectTX	Processor_Src	Analyst_Src	AllSourceAnalyst_Spec
InfoSource 1	0.000113	0.000006	0.000009	0.000345	0.000004
InfoSource 2	0.000062	0.000005	0.000004	0.000246	0.000009
InfoSource 3	0.000082	0.000005	0.000005	0.000131	0.000012
InfoSource 4	0.000077	0.000011	0.000018	0.000171	0.000011

Figure F.23 Case 7 simulation results part 2.

Queue Number Waiting

Averages

	Average	Max
Search Library	3.02	57.44
Collect Data	258.53	819.20
TX from Collection	0.00	3.56
Process Data	0.00	3.28
Exploit Data	57.69	217.80
Analyze Intel	294.66	920.64
Produce Intel	21.87	126.28
Disseminate Intel	0.52	17.80
Integrate Intel	0.00	0.00

Half-Widths

	Average	Max
Search Library	0.0032	0.2949
Collect Data	1.0943	8.0302
TX from Collection	0.0000	0.0418
Process Data	0.0000	0.0378
Exploit Data	0.3372	3.1738
Analyze Intel	3.3035	12.4762
Produce Intel	0.0637	0.9537
Disseminate Intel	0.0006	0.1994
Integrate Intel	0.0000	0.0000

Queue Wait Time

Averages

	Average	Max
Search Library	0.8711	8.6644
Collect Data	34.2605	1292.3343
TX from Collection	0.0003	0.0981
Process Data	0.0007	0.0838
Exploit Data	15.1912	1644.2010
Analyze Intel	79.0979	5739.7372
Produce Intel	7.6395	476.6527
Disseminate Intel	0.1139	8.5753
Integrate Intel	=	=

Half-Widths

	Average	Max
Search Library	0.000416	0.115551
Collect Data	0.137541	15.211803
TX from Collection	0.000004	0.001067
Process Data	0.000006	0.000958
Exploit Data	0.080230	38.140094
Analyze Intel	0.975802	129.167324
Produce Intel	0.021458	7.927161
Disseminate Intel	0.000135	0.121276
Integrate Intel	=	=

Wait Time by Priority

Averages

Priority	Average	Max
1	3.88	24.29
2	16.18	84.14
3	55.3	370.27
4	199.46	1981.87
5	386.49	5799.56

Half-Widths

Priority	Average	Max
1	0.0032	0.2742
2	0.0245	0.6406
3	0.0913	4.6094
4	0.9948	29.4104
5	4.0545	129.6670

Figure F.24 Case 7 simulation results part 3.

F.9 Selected Paired-t tests

alpha = 0.05

*Insignificant differences **highlighted** based on alpha*

Paired-t test p-values for User 1 vs User 5 over 25 replications

Case	Timely	QualMet	Both
BL	0.00000000	0.00000000	0.00000000
C1	0.00000000	0.00000000	0.00000000
C2	0.00000000	0.00000000	0.00000000
C3	0.00000000	***0.68767872***	0.00000000
C4	0.00000000	***0.56661909***	0.00000000
C5	0.00000000	0.00000000	0.00000000
C6	0.00000000	0.00000000	0.00000000
C7	0.00000000	0.00000000	0.00000000

Paired-t test p-values Case comparisons for User 1 over 25 reps

	Timely	QualMet	Both
BL – C1	0.00000000	0.00000000	0.00000000
BL – C2	0.00000000	0.00099752	0.00000000
BL – C3	0.00226316	0.00000000	0.00000000
BL – C4	***0.49213783***	0.00000000	0.00000004
BL – C5	0.00000000	0.00000000	0.00000000
BL – C6	0.00000000	0.00000007	0.00000000
BL – C7	0.00000000	***0.61116139***	0.00000000
C1 – C2	0.00000000	0.00001316	0.00000000
C3 – C4	0.00428745	0.00000000	0.00421697

Paired-t test p-values Case comparisons for User 5 over 25 reps

	Timely	QualMet	Both
BL – C1	0.00000000	0.00000000	0.00000000
BL – C2	0.00055935	0.00000913	0.00513163
BL – C3	0.00682321	0.00000000	0.00000000
BL – C4	***0.70841705***	0.00000000	0.00000000
BL – C5	0.00000000	0.00000000	0.00000000
BL – C6	0.00000000	***0.12096061***	0.00000000
BL – C7	0.00000000	0.00000007	0.00000000
C1 – C2	0.00000000	0.00008434	0.00000007
C3 – C4	0.03908054	0.00000000	0.03849555

Figure F.25 p-values for selected paired-t tests of *User* 1 and *User* 5.

Bibliography

C4ISR Model Report, 2000. Department of the Air Force. *Air Force Command, Control, Communications, Computers, and Intelligence Surveillance Reconnaissance (C4ISR) Modeling Special Task Group Report on C4ISR Modeling*. Washington: HQ USAF, October 2000.

JP 1, 2000. Department of Defense. *Joint Warfare of the Armed Forces of the United States*. Joint Publication 1. Washington: GPO, 14 November 2000.

JP 1-02, 2003. Department of Defense. *Department of Defense Dictionary of Military and Associated Terms*. Joint Publication 1-02. Washington: GPO, 12 April 2001 (As Amended Through 5 September 2003).

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

JP 2-01, 1996. Department of Defense. *Joint Intelligence Support to Military Operations*. Joint Publication 2-01. Washington: GPO, 20 November 1996.

JP 2-01.1, 2003. Department of Defense. *Joint Tactics, Techniques, and Procedures for Intelligence Support to Targeting*. Joint Publication 2-01.1. Washington: GPO, 9 January 2003.

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

JP 2-02, 1998. Department of Defense. *National Intelligence Support to Joint Operations*. Joint Publication 2-02. Washington: GPO, 28 September 1998.

JP 2-03, 1999. Department of Defense. *Joint Tactics, Techniques, and Procedures for Geospatial Information and Services Support to Joint Operations*. Joint Publication 2-03. Washington: GPO, 31 March 1999.

NSSA, 2003. National Security Space Architect. *TPED/TPPU Thesis Proposal*. February 2003.

NSSA Meeting, 2003. National Security Space Architect. Meeting. 23 September 2003.

Welch, 1983. Welch, Peter D. "The Statistical Analysis of Simulation Results," *Computer Performance Modeling Handbook*. Academic Press, Inc., 1983.

White House, 2002. White House. *The National Security Strategy of the United States of America*. Washington: GPO, September 2002.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
05-03-2004	**Master's Thesis**	Jul 2003 – Mar 2004

4. TITLE AND SUBTITLE

MODELING AND SIMULATION OF THE MILITARY INTELLIGENCE PROCESS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Pawling, Carl R., Captain, USAF

5d. PROJECT NUMBER
ENR#2003-107

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Street, Building 642
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/04-09

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Security Space Architect (NSSA)
Attn: Major Darren Sene
P.O. Box 222310
Chantilly, VA 20153-2310 Darren.Sene@osd.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

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

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

There is concern within U.S space and intelligence organizations that the current Tasking, Processing, Exploitation, and Dissemination processes may be insufficient to support current and future Intelligence, Surveillance, and Reconnaissance systems. As part of a larger intelligence process, more detailed analysis becomes critical to determine what portions need to be improved. This analysis can be accomplished by simulation, which is appropriate due to the complexity of the process and the ability to compare variations in the process. We construct a high level model of a generalized military intelligence process based in part on the Intelligence Cycle outlined in Joint Publications. Using the Arena process oriented simulation software, our modular simulation can be used for quick turn studies on changes to the process, specifically with respect to classical measures such as quality, quantity, and timeliness. A sample study using the basic framework of the intelligence process with statistical analysis is also conducted.

15. SUBJECT TERMS

SIMULATION, MILITARY INTELLIGENCE, TPED, TPPU

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	173	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4326; e-mail: John.Miller@afit.edu