3-24-2016

Improving System Design Through the Integration of Human Systems and Systems Engineering Models

Michael E. Watson

IMPROVING SYSTEM DESIGN THROUGH THE INTEGRATION OF HUMAN SYSTEMS AND SYSTEMS ENGINEERING MODELS

THESIS

Michael E. Watson, Captain, USAF

AFIT-ENV-MS-16-M-190

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-16-M-190

IMPROVING SYSTEM DESIGN THROUGH THE INTEGRATION OF HUMAN SYSTEMS AND SYSTEMS ENGINEERING MODELS

THESIS

Presented to the Faculty

Department of Systems Engineering and Management

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 Systems Engineering

Michael E. Watson, BS

Captain, USAF

March 2016

AFIT-ENV-MS-16-M-190

IMPROVING SYSTEM DESIGN THROUGH THE INTEGRATION OF HUMAN SYSTEMS AND SYSTEMS ENGINEERING MODELS

Michael E. Watson, BS

Captain, USAF

AFIT-ENV-MS-16-M-190

Abstract

The human is a critical aspect of many systems, but frequently there is a failure to properly account for human capabilities and involvement during system design. This inattention results in systems with higher lifecycle costs, decreased user compatibility, and the potential to produce disastrous consequences. This research presents an approach to integrating the human into system models by using two methods: static and dynamic modeling. The static method uses a user-centered design framework to create system- and human-centered models that deconstruct the system and user into their respective components. These models are integrated to create system models that include relevant information about the human and highlight potentially conflicting tasks. The dynamic method uses a human performance modeling tool to create a discrete event simulation (DES) of the system. This DES model is used to perform an analysis between system trades, by which constraints and assumptions placed on the human are verified. Data gained from the analysis are integrated back into system models in order to reflect true system performance. By applying these two integration methods early in the system's lifecycle, system models can more effectively account for the human as a critical component of the system, thus improving system design.

AFIT-ENV-MS-16-M-190

To my wife for her never-ending patience and support

Acknowledgments

I would like to express my sincere gratitude to my research advisor, Major Christina Rusnock, for her continuous mentorship throughout this thesis effort. I would also like to thank my committee members, Dr. John Colombi and Dr. Michael Miller, for sharing their unique insights and feedback along the way. My appreciation goes to Mr. Michael Hoepf at the 711th Human Performance Wing for familiarizing me with Vigilant Spirit and for his subject-matter expertise. Finally, special thanks go to the Human Research and Engineering Directorate at the Army Research Laboratory for making this research possible.

Michael E. Watson

Table of Contents

List of Figures

List of Tables

IMPROVING SYSTEM DESIGN THROUGH THE INTEGRATION OF HUMAN SYSTEMS AND SYSTEMS ENGINEERING MODELS

I. Introduction

Chapter Overview

This chapter begins by explaining the background and problem relating to systems engineering (SE) practices and the lack of integrated models for considering Human Systems Integration (HSI). The chapter then discusses a solution of integrating the HSI and SE processes, followed by focusing on questions regarding how to successfully do so. Next, the chapter addresses the methodology that will be undertaken to answer these questions, highlights assumptions unique to this research, and concludes with an overview of the remaining chapters.

Background

Systems engineering (SE) has become an increasingly important part of the lifecycle management of Department of Defense (DoD) systems, and has even been institutionalized as the disciplinary approach to acquisition program development (Department of Defense, 2015). SE is a unique approach to system development that concentrates on a holistic, top-down view of the system and breaks it down into its multiple components (International Council on Systems Engineering, 2015). It is important to consider the human as one of the system's components, as nearly every system is constructed to fulfill a human need and therefore requires some manner of human involvement. The process by which the human can be effectively accounted during system development is called Human Systems Integration (HSI) (U.S. Air Force, 2010).

It is important to consider HSI during the SE process, as failure to do so can lead to serious consequences. For example, independent investigations found that inadequate design of the Global Hawk unmanned aerial vehicle and Patriot air and missile defense (AMD) systems led to the Global Hawk and Patriot operators' over-reliance on automation and a lack of situation awareness, resulting in mission degradation and two Patriot AMD fratricide incidents (Hopcroft, Burchat, & Vince, 2006; Hawley, 2011). Poor HSI also contributed to the partial nuclear meltdown of the Three Mile Island power plant in 1979. A combination of confusing warning display design, absence of instruments displaying critical information, and improper operator training led to the worst nuclear power plant accident in U.S. history (U.S. Nuclear Regulatory Commission, 2014).

DoD Instruction 5000.02 (2015) recognizes the need to consider HSI during system development, mandating that HSI be taken into account for new acquisition systems and continue to be considered throughout the system's lifecycle. However, HSI implementation has received criticism as neither occurring at a sufficient level of detail, nor at the correct time (Handley & Knapp, 2014). Others have suggested that systems engineers do not apply HSI during early design stages of the system's lifecycle as they should (Orellana & Madni, 2014; Hardman & Colombi, 2012). This failure to integrate the HSI and SE processes has been attributed to producing systems that have higher costs later in the lifecycle and are less compatible with the user (Mitchell, Agan, & Samms, 2011; Hardman & Colombi, 2012).

Problem Statement

There have been several efforts to better integrate the HSI process into SE (Bodenhamer, 2012; Handley & Smillie, 2009; Hardman, Colombi, Jacques, & Miller, 2008; Mitchell, Agan, & Samms, 2011; Orellana & Madni, 2014). These efforts strive to solve the problem by focusing on the issue at varying levels of scope: the process, methods, and tools levels. The relationship between these three levels is defined in terms of detail: a process is supported by methods, which are facilitated by tools (Martin, 1996). For example, at the process level, Chua and Feigh (2011) offer general ways the human may be considered in early system design. At the tools level, other researchers propose the use of modeling and simulation (Boy & Narkevicius, 2013; Mitchell D. K., 2005) or system diagramming techniques (Ramos, Ferreira, & Barcelo, 2013). However, integration efforts at the tools level have been relatively few. Of those efforts, an integration plan is lacking to inform system models following human analysis. Additionally, most efforts direct focus on integration only in the concept and architectural phases of system development.

This research addresses integration by focusing on HSI and SE at their respective tools levels of scope – as opposed to the higher process and methods levels – through the use of models. The research also looks at integration primarily during the preliminary or detailed design phases of the system's lifecycle, later than the concept and architectural phases seen in many other studies. By focusing on integration slightly later in system design, more system information is known, thus enabling effective human and system consideration at the tools level of detail. However, current SE and HSI models are disjointed from each other with no clear integration path. For an extended review of the definitions and integration efforts relating to SE and HSI, see Appendix A.

Research Objective

This research effort has several objectives. First it must be determined if it is possible for HSI to be incorporated into SE practices. The next objective is to determine how to effectively perform this integration. The final objective is to demonstrate the value of integration. SE and HSI practitioners conduct system tradeoff analyses when designing the system. In order to meet these objectives, first it is important to understand what tradeoffs are associated with each practice and how these tradeoff analyses are being performed. Additionally, a review of the literature is necessary to establish the extent of efforts to integrate the SE and HSI processes thus far. Knowing this information aids this research in completing the first two objectives by understanding the possibility of integration and potential methods to perform this integration. To complete the final objective, this research effort makes use of a selected military-relevant scenario combined with an associated trade study in order to demonstrate the value of integration. The Systems Modeling Language (SysML) is chosen as the SE modeling tool with which to depict the system-centered models. Software called the Improved Performance Research Integration Tool (IMPRINT) is chosen as the HSI tool with which to create human-centered models. By completing these objectives, this research seeks to contribute to the SE and HSI disciplines by increasing the knowledge related to integrating these modeling perspectives and demonstrating a method for integration. This integration will allow human considerations to be effectively considered during development, thus improving system design.

Research Question / Investigative Questions

In order to guide this research effort's objectives, the following research question is formulated: ***How can HSI models be integrated with SE models in order to perform system design tradeoffs?***

Furthermore, the research question is divided into the following four investigative questions:

1. What information should be captured in human-centric and system-centric models to enable effective integration?
2. What considerations and decisions must be made when integrating between human-centric and system-centric models?
3. What information can currently be passed from IMPRINT to SysML models?
4. What information do SysML models need from IMPRINT to effectively inform tradeoff analyses?

Methodology

The software tool MagicDraw is used to create the SysML models of the system. Additionally, the software tool IMPRINT is used to model the human. To perform the scenario-specific trade study, a human subjects experiment from the 711th Human Performance Wing (HPW) at Wright-Patterson Air Force Base is utilized. The 711th HPW's study involves a synthetic task environment called Vigilant Spirit, which simulates a remotely-piloted aircraft surveillance mission. This research effort uses the

Vigilant Spirit simulation and the experiment data collected by the 711th HPW as the scenario with which to demonstrate integration.

Assumptions / Limitations

Context Assumptions

This research focuses on improving integration between the SE and HSI processes via system models. Therefore, it is assumed that the developmental lifecycle stages for a typical system employ a model-based SE approach, and further utilize SysML as the preferred modeling language.

It is assumed that the HSI professional's view of the system is unique from other engineering disciplines, therefore necessitating the creation of more human-centered viewpoints of the system in order to capture this unique perspective. However, this research also assumes that HSI professionals prefer to develop human-centered models that are inherently useful for them, as opposed to creating specific models with utility solely for systems engineers. As such, the best way to perform integration is to utilize information that is also beneficial to the HSI professional.

Though this research specifically focuses upon the human factors engineering HSI domain for analysis, it is assumed that this analysis may also be extended to some or all of the remaining eight HSI domains in order to achieve effective integration.

Model Assumptions

It is assumed that this research's case study involving the Vigilant Spirit simulation is an accurate representation of the typical tradeoff analyses that systems engineers perform during system design and development. Further, elements of this research are predicated upon the experiment performed and associated data collected by the 711th HPW. As such, it is assumed that the data used for the purposes of this research are correct and that the source of the data is reliable. Additionally, assumptions concerning the baseline and alternative Vigilant Spirit models created in IMPRINT may be found in Appendices B and D, respectively.

Preview

This chapter explained the background relating to SE and HSI, the problem of a lack of integration between the two processes, and the necessity of integration. Chapter II addresses the first two investigative questions by providing a static method for integrating user-centered design into system models. Chapter III addresses the last two investigative questions and re-addresses the first two questions from a dynamic standpoint, using human performance modeling to perform system-related tradeoff analyses. Chapter IV summarizes significant findings and insights from the research, draws conclusions, and offers recommendations for future research related to SE and HSI process integration.

II. Improving System Models by Integrating User-Centered Design

Chapter Overview

This chapter begins to address the first two investigative questions of the research. Specifically, it addresses what information should be captured in both human- and system-centric models to enable effective integration, and what considerations and decisions need to be made when integrating these two modeling perspectives. The chapter presents a static method of integration, where systems engineers can use the concept of user-centered design to integrate relevant information about the human into system design models while staying within the SE toolset. The work presented in this chapter is an adaptation of a journal article in submission.

Abstract

The human user is important to consider during system design. However, failure to properly integrate the user in system design is still commonplace. This research presents a method for integrating human factors considerations into system models through user-centered design (UCD). First, a task analysis is performed on the as-is system and used to build Systems Modeling Language (SysML) diagrams. System-centered diagrams are created from the systems engineer's perspective; then, using UCD, human-centered SysML diagrams are created from the human factors engineer's perspective. These diagrams are compared and, using common anchor-points between the two, new diagrams are created which incorporate both the system and user into one integrated set. These new system models capture the important aspects of the human,

allowing both systems and human factors engineers to effectively account for the user in system design.

Introduction

Systems engineering (SE) approaches design and development by focusing on the complete system and deconstructing it into its multiple components (International Council on Systems Engineering, 2015). In many systems, the human operators and maintainers are among these components, and often the most important to consider. Systems engineers can design for the human through a process called Human Systems Integration (HSI) (U.S. Air Force, 2010). Because nearly every system requires a human user in some manner, failure to consider the user in system design could yield negative consequences.

Failures in system design allowed for human error to cause catastrophic and fatal results in the SpaceShipTwo crash in 2014. The accident occurred when the co-pilot initiated the spacecraft's re-entry system too early, causing the crash and killing the co-pilot. However, National Transportation Safety Board Chairman Christopher Hart suggests that the system's designers were ultimately responsible due to a "failure to consider and protect against the possibility that a single human error could result in a catastrophic hazard" (Malik, 2015).

As a historic example, in 1979, a partial nuclear meltdown occurred at the Three Mile Island power plant near Harrisburg, Pennsylvania. The precipitating cause of the incident was a relief valve stuck in an open position. Because of a combination of confusing warning display design, absence of instruments displaying critical information, and improper training, operators misinterpreted the nature of the malfunction and took actions that actually worsened the situation, leading to the worst nuclear power plant accident in U.S. history (U.S. Nuclear Regulatory Commission, 2014).

These examples illustrate the importance of integrating the HSI process into SE by focusing on the user. A failure to integrate the human with the system could cause a variety of issues, to include ineffective system interfaces, increased developmental and operational costs, and loss of user safety. Unfortunately, lack of HSI is still commonplace during development. This paper seeks to counteract this problem by proposing a method of integrating the HSI and SE processes in order to improve system designs, and does so by focusing specifically on integrating user-centered design into system modeling products. By performing this integration method, human factors engineers gain a way to incorporate their user-centered design practices into the SE process, and systems engineers gain a way to incorporate human considerations into their existing SE framework, thus improving system design.

Background

Processes, Methods, and Tools

For the purposes of this paper, a process is defined as a philosophical approach defining *what* activities should be accomplished to achieve an objective. Methods support processes by defining in greater detail *how* to accomplish those activities. Tools are the *enabling* mechanisms that facilitate and enhance the implementation of a given method (Martin, 1996). There may be more than one tool capable of supporting a particular method, and there likewise could be multiple methods capable of supporting a process.

Systems Engineering

SE is a process that has become an increasingly important part of the overall lifecycle management of Department of Defense (DoD) systems, to the point of becoming an institutionalized disciplinary approach to the development of defense acquisition programs (Department of Defense, 2015). The International Council on Systems Engineering (INCOSE) (2015) defines a system as “an integrated set of elements, subsystems, or assemblies that accomplish a defined objective,” whereas these elements could not otherwise produce the same results by themselves. Elements may include hardware, software, people, information, and facilities. SE offers a holistic approach to developing these systems by integrating the many disciplines involved and thereby accounting for factors such as requirements, cost, and schedule early in the system’s lifecycle and continuing through development, operation, and eventually disposal. To support this consideration throughout the lifecycle, the SE process is composed of 14 technical sub-processes ranging from stakeholder requirements definition to system disposal (International Council on Systems Engineering, 2015).

There are many different methods of practicing SE. Model-based systems engineering (MBSE) is an emerging method with which to perform SE. Whereas the traditional document-based method is driven by the development of a set of disjointed documents, each separately detailing system-related information such as requirements or design specifications, MBSE allows for the development of the same information through a series of interrelated models that together form a complete system model (Friedenthal, Moore, & Steiner, 2014). The MBSE method results in improved team communication, increased quality of the system's specification and design, and the ability to reuse the model throughout the system's lifecycle (Friedenthal, Moore, & Steiner, 2014).

If MBSE is a method of practicing SE, then the Systems Modeling Language (SysML) is a tool with which to implement MBSE. There are several graphical modeling languages available for SE applications, SysML being one of them (Delligatti, 2014). SysML provides a means of communicating system information via a selection of uniquely-purposed diagrams. These diagrams allow the modeler to represent requirements as well as behavioral and structural aspects of the system, as shown in Figure 1 (Delligatti, 2014).

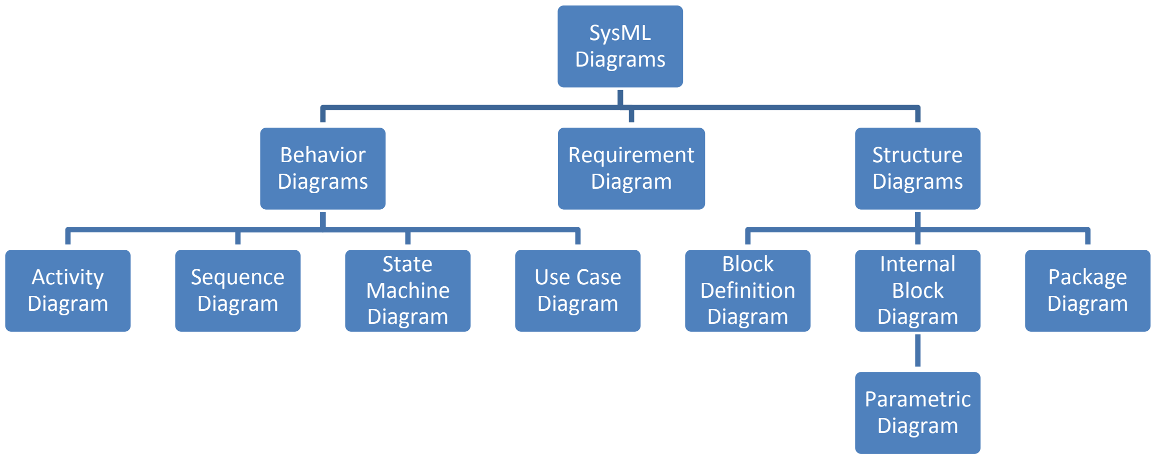

Figure 1: SysML Diagram Taxonomy – adapted from (Delligatti, 2014)

Human Systems Integration

The human should also be a critical consideration during system development and the SE process in general. The Air Force HSI Handbook defines HSI as the "process by which to design and develop systems that effectively and affordably integrate human capabilities and limitations" (U.S. Air Force, 2010). This approach is necessary because humans who operate, maintain, and support the system are an integral part of the total system itself (Department of Defense, 2013). HSI is divided into nine domains: manpower, personnel, training, human factors engineering, environment, safety, occupational health, survivability, and habitability (U.S. Air Force, 2010).

Human factors engineering is the primary HSI domain with which to focus on integration (U.S. Air Force, 2010). Human factors engineering, also called ergonomics, is the study of the interactions between the human and system, and the efficiency of those interactions (International Ergonomics Association, 2016).

There are various methods of practicing human factors engineering, one of which is through user-centered design (UCD). UCD is the idea of designing a system with a focus primarily on the user and involving the user in the design process. By focusing on the user's goals, preferences, tools needed, and tasks performed, the goal is that the end-system will be best suited for what the user needs (Norman & Draper, 1986).

The DoD recognizes the need for consideration of HSI during system development. DoD Instruction 5000.02 (2015) mandates that HSI be taken into account for each new acquisition system and continue to be considered throughout the system's lifecycle. The goal of incorporating HSI into system development is to increase the potential for the user to successfully and efficiently conduct the mission while maximizing system performance (Department of Defense, 2015). However, HSI implementation has been criticized as neither occurring at a sufficient level of detail, nor at the correct time (Handley & Knapp, 2014). Within the SE literature, it has been acknowledged that systems engineers do not apply HSI during early design stages of the system's lifecycle as they should (Orellana & Madni, 2014; Hardman & Colombi, 2012). Furthermore, even the DoD's template for acquisition system planning and development, called the DoD Architecture Framework (DoDAF), only minimally accounts for humans in its products. The DoDAF treats humans either as an invisible part of the system, as generic "performers," or as elements of larger organizations (Department of Defense, 2009). Thus, the DoDAF has been criticized for not properly accounting for human impacts on system performance or the human specifications needed to operate and maintain the system (Handley & Knapp, 2014). This failure to integrate the HSI process with SE results in a system that has higher lifecycle costs and is less compatible with the user (Mitchell, Agan, & Samms, 2011). In addition, the system may have imbedded design flaws that could manifest in disastrous consequences during operation, as illustrated in previous examples.

There have been several efforts to better integrate human systems considerations into the SE process. These efforts strive to solve the problem by focusing on the issue at varying levels of scope: the process level, the methods level, and the tools level.

Process-Level Integration

Integration efforts at the process level strive to fundamentally change or augment the SE and/or HSI process itself. Chua and Feigh (2011) offer various ways in which human factors may be generally included in early system development. They organize their ideas according to four system design stages: requirements acquisition, concept generation, preliminary, and detailed. Admittedly at a high level of detail, Chua and Feigh provide general suggestions in an effort to encourage communication between systems engineers and human factors engineers, and to promote awareness of human factors during system design.

Hardman and Colombi (2012) extend the idea of augmenting the SE process by highlighting the necessity for quantitative methods of expressing HSI requirements in order to be properly considered by program management during system development. As such, Hardman and Colombi outline areas in which to emphasize HSI throughout the early requirements analysis, function allocation, and design stages of system development, and further suggest the usage of empirical measures such as safety and human subjects data to minimize subjectivity.

Another process-level idea is to standardize the terminology between SE and HSI. Hardman, Colombi, Jacques, and Miller (2008) clarify the HSI terminology across the DoD and HSI communities due to inconsistencies between numerous DoD and HSI publications, such as between the DoDAF, Defense Acquisition Guide, and INCOSE's handbook. The idea of standardization may be extended from the DoD to the entire SE community (Madni, 2009; Orellana & Madni, 2014). Orellana and Madni (2014) argue that the reason why there is a lack of integration between the SE and HSI processes is because differences in terminology prevent systems engineers and those untrained in HSI from communicating with those who are trained. A proposed solution is to build a common HSI ontology to connect the semantics of the two fields, thus providing a means to address HSI concerns during system design (Madni, 2009; Orellana & Madni, 2014). Bruseberg (2008) corroborates Orellana and Madni's claim, citing several examples of differences between HSI and SE's interpretations of terminology. For instance, whereas the term “activity” has a high-level connotation to systems engineers, its scope is more low-level and detailed to human factors engineers.

Methods-Level Integration

Efforts at the methods level strive to enhance integration by improving one of the existing SE design or analysis methods, or by proposing a new method. Crisp, Hoang, Karangelen, and Britton (2000) do the latter. Continuing the ideas put forth by Hardman et al. (2008), Orellana and Madni (2014), and Bruseberg (2008), once a common language between SE and HSI is established, Crisp et al. propose a way to further establish an effective integration. Due to the need for systems engineers to synchronize multiple disciplines, a central software interchange could implement this common language as a data schema in order to translate information between disciplines’ software tools and allow communication.

Hardman et al. (2008) propose an augmentation to the DoDAF to improve integration. They examine how each of the nine HSI domains can be addressed in the existing DoDAF products. Each HSI domain lends itself to a DoDAF capability. For example, since the manpower and personnel domains deal with the numbers of users and associated knowledge and skills needed to operate the system, these domains may be addressed by the DoDAF’s Operational or Services Views. A properly developed use case can also address manpower in addition to addressing the training domain. Human factors engineering is a key domain to address in system development since it addresses system limitations as a result of human involvement. As such, there are many DoDAF products that may be used to identify problem areas or tradeoff opportunities, such as the

Systems Interface Description (SV-1), Systems-Systems Matrix (SV-3), and the Systems Functionality Description (SV-4).

Piaszczyk (2011) proposes a method of integration similar to Hardman et al.'s (2008) DoDAF augmentation. However, Piaszczyk uses a MBSE approach instead, focusing on the DoDAF's graphical products to represent the human. He describes how to factor the human into existing DoDAF views in order to derive human-related requirements and drive system design throughout the acquisition lifecycle. These product re-scopes encompass the DoDAF's Operational and System Views. For example, the Operational Architectural Diagram (OV-2) is used to derive system operator requirements and the Organizational Relationships Diagram (OV-4) is used to define the human's roles with regard to the system. The methods proposed by Hardman et al. (2008) and Piaszczyk (2011) present ways to include HSI in the DoDAF without developing new products.

Another integration method is to create a new, human-focused product to augment existing architecture frameworks. In 2007, representatives from the United States, United Kingdom, Canada, and the Netherlands convened the North Atlantic Treaty Organization (NATO) Human View Panel in order to examine the current state of Human View presence within architecture frameworks around the world, and to propose a standard Human View that could be adopted by any architecture framework (Handley & Smillie, 2008). The resultant NATO Human View is comprised of eight products:

- HV-A: Concept
- HV-B: Constraints
- HV-C: Tasks

- HV-D: Roles
- HV-E: Human Network
- HV-F: Training
- HV-G: Metrics
- HV-H: Human Dynamics

All of these products are designed to address different human aspects that are important to consider during system design and development. For example, the Concept (HV-A) offers a high-level look at the human component of the system, while Constraints (HV-B) focuses on weaknesses or limitations the human brings that affect the system. HV-B can be further subdivided into subviews such as Manpower Projection Constraints and Personnel Policy Constraints. Since most of these views are static by nature, Human Dynamics (HV-H) is designed to address the dynamic aspects from each of the other views, to include state changes, conditions, time units, and performance measures. The Human View is intended to force systems architects to consider the human in its own architecture framework view instead of arbitrarily adding human considerations into other views. Another goal of adding a Human View directly into an architecture framework is to enable systems engineers and HSI analysts to collaborate early in system development, thus contributing more effectively to design (Smillie & Handley, 2009).

Furthermore, Handley and Knapp (2014) detail four stages by which to compile the Human View products, with each stage focusing on certain sets of models at a time. Moving to the next stage shifts focus to another model, while still reiterating through previous models in order to ensure a complete product is formed. Figure 2 shows the completed Human View development (Handley & Knapp, 2014).

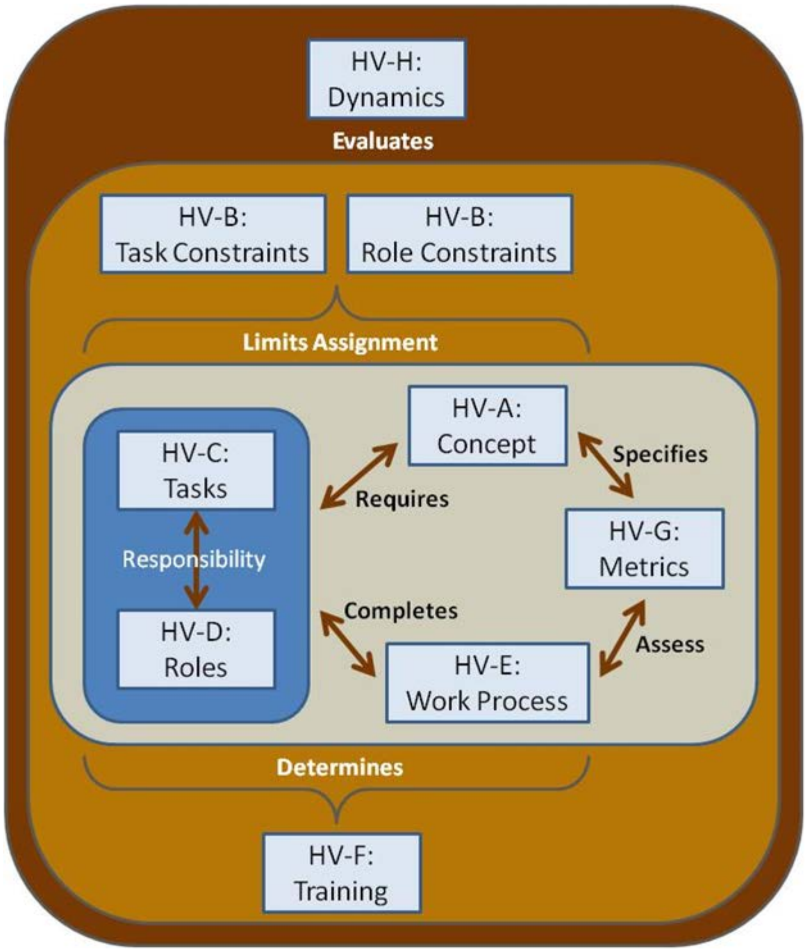

Figure 2: Human View Development – adapted from (Handley & Knapp, 2014)

Handley (2011) made an effort to further adapt the NATO Human View specifically to the DoDAF. The DoDAF 2.0, released in 2009, allows for easier integration of human-centered information within the framework, mainly due to the inclusion of the DoDAF 2.0 Meta Model (DM2). Since the DM2 allows the system architect to create "Fit for Purpose" views to augment the existing architecture framework, Handley claims that the NATO Human View may be mapped to the DM2 more easily than in previous DoDAFs.

Similarly, Bruseberg (2008) proposed a Human View specifically for the British Ministry of Defence Architecture Framework (MODAF). Listing several of the same human-related shortcomings in the MODAF as does Handley (2014) for the DoDAF, Bruseberg (2008) details ways in which her Human View can improve the MODAF's representation of the human during system development. She argues that human views aid in modeling the "soft systems" human side of system development, thus bridging the communication gap between systems engineers and human factors engineers. The MODAF Human View is comprised of seven products, HV-A through G. These products largely parallel the NATO Human View's eight products. For example, the MODAF Human View also has products capturing human functions and tasks (HV-E), roles and competencies (HV-F), and dynamic aspects of human behavior (HV-G). Though similar to the DoDAF-centered Human View, development of the MODAF Human View predates Handley's work and even the NATO Human View.

Sharples (2014) put the NATO Human View into practice to solve a real-world problem for German-based Airbus Defence and Space. Sharples integrated the Human View with Airbus's existing architecture for a remotely-piloted aircraft (RPA) system in order to identify human-related deficiencies and refine the architecture. By taking the Human View's separate products and augmenting the operational and system views from the existing RPA architecture, Sharples was able to identify system gaps such as the absence of several roles from the original model.

Tools-Level Integration

The most in-depth, narrowly-scoped way to integrate the HSI and SE processes is to approach integration at a tools level. Efforts at this level focus on improving the way in which tools such as SysML can be used to incorporate the human into SE. While some researchers advocate the use of modeling and simulation in general to consider HSI (Boy & Narkevicius, 2013), some efforts have specifically used MBSE modeling to accomplish this task. Bodenhamer (2012) states that to understand the human's interaction with the system, the human must first be deconstructed into the functional components necessary to operate the system. These components include sensory channels, cognitive processing, psychomotor capabilities, and physical interfaces. The system itself must also be deconstructed into its components, treating the user as one of these components. Using a landmine detector system as a case study, Bodenhamer created a high-level architectural concept of the system to demonstrate this concept. He modeled the behavioral aspects of the system by creating activity and sequence diagrams. These diagrams visually highlight the human-system interaction that is necessary for mission success. By doing so, Bodenhamer claims that the modeler can identify HSI-related problems that could affect system performance or mission success.

Ramos, Ferreira, and Barceló (2013) address human integration from the process, methods, and tools levels. As part of their larger effort to enhance the overall SE process they amalgamate aspects from a variety of methodologies in order to present a revised, more agile MBSE methodology. However, their main focus is at the tools level. HSI is considered as a part of the overall methodology, in which Ramos et al. advocate a systems engineer-focused implementation of HSI via SysML diagrams such as activity and internal block diagrams.

Orellana and Madni (2014) also address integration from multiple levels of scope. After proposing their process-level HSI ontology, they narrow to the tools level. Orellana and Madni's ontology is influenced by defining the human in terms of SysML diagrams. The goal of the ontology is to "bridge the gap" between systems engineers and human factors engineers by allowing systems engineers to define the human using their own MBSE modeling methods. Orellana and Madni provide a high-level description of ways in which the human can generally be represented through SysML diagrams. Ahram and Karwowski (2009) also recommend a common language by incorporating a HSI framework into systems engineers' SysML modeling practices.

Research Gap

There have been several efforts to integrate the HSI and SE processes. These efforts have addressed the integration problem from various standpoints: the process level, methods level, and tools level. Numerous processes and methods have been proposed, but relatively few efforts have tried to integrate by addressing SE at the tools level. Additionally, most efforts have focused on integration only at the concept or architecture phase of the system's lifecycle.

The purpose of this paper is to present a different integration approach at the tools level to be used during the preliminary or detailed design phases, later in the system's lifecycle. Ideally, this approach should be used in combination with the other integration efforts so the human is considered at each phase. By focusing on integrating HSI methods like UCD directly with MBSE tools like SysML, the resulting system models allow for human consideration at a lower level of system detail. This system detail is enabled by focusing on integration later in the system design lifecycle phases when more information about the system is known, thus allowing for a more thorough consideration of the human. Therefore, all aspects of the human can be effectively considered during system design. Figure 3 shows that this paper's research lies in the preliminary and detailed design phases of the SE Vee process model. By contrast, other integration efforts from the literature tend to focus only on the conceptual phase.

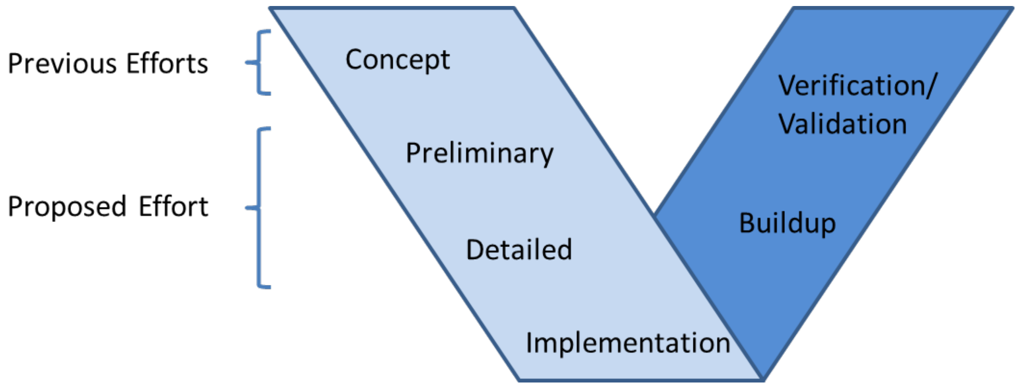

Figure 3: Integration Efforts in the Vee Process Model – adapted from (Forsberg, Mooz, & Cotterman, 2005)

This paper integrates human considerations with system considerations by creating SysML diagrams to represent the human as an equal component of the system. These diagrams are sequentially built using various perspectives and the concept of UCD within human factors engineering. The premise is that by correcting the integration problem at the SysML level through UCD, the higher-level SE and HSI processes are thereby integrated as well, thus improving system design.

Methodology

This study demonstrates a method of integrating UCD with SysML using a series of steps. First, a task analysis is performed on an example system as a case study. Next, both system- and human-centered diagrams are created to represent different viewpoints of the system. These diagrams are compared and analyzed with each other, and then new diagrams are created which incorporate both system and human considerations together into one integrated set of diagrams.

Perform Task Analysis

First, a task analysis is performed on the system to identify the relevant processes and activities, permitting them to be accurately represented in subsequent models. With UCD in mind, the task analysis may also include tasks required of the operator to perform the mission that are either within or outside the boundary of the system under design.

This study uses a synthetic task environment called Vigilant Spirit as the system case scenario with which to demonstrate integration. Vigilant Spirit is used by the 711th Human Performance Wing (HPW) at Wright-Patterson Air Force Base, Ohio to conduct human-in-the-loop experiments studying the effects of certain tasks on participants' performance, workload, and physiology (Hoepf, Middendorf, Epling, & Galster, 2015). Vigilant Spirit was designed to simulate RPA missions with a single operator controlling the RPA(s). This synthetic task environment is a suitable case study because the overall system requires a combination of both human and system activities. Vigilant Spirit is also a relatively simple system, allowing the study to remain narrow in scope in order to focus on the methodology involved instead of the intricate details of a complex system. This scope allows for results to be more easily generalized to other systems.

Using Vigilant Spirit, the pilot performs a surveillance mission attempting to locate and follow a high value target (HVT) walking through an urban marketplace. The HVT is identified by a rifle he is carrying, but there are also distractors walking throughout the market who are either unarmed or carrying pistols or shovels. Participants are seated in front of two monitors that display the simulated RPA camera feed and a communications window. They use a computer mouse to click within the camera feed window to move the camera and re-center the RPA's loiter circle, and scroll the mouse wheel to zoom the camera in and out. Subjects use a keyboard to indicate to the system when they locate the HVT. Beside the primary task of surveilling the HVT, there is also a secondary communication task. Throughout the mission the operator is asked a series of route navigation, math-based questions through a headset. To answer the questions, the operator uses the keyboard to open a communication line to orally respond via headset. Figure 4 shows the 711th HPW's experimental setup for the Vigilant Spirit environment.

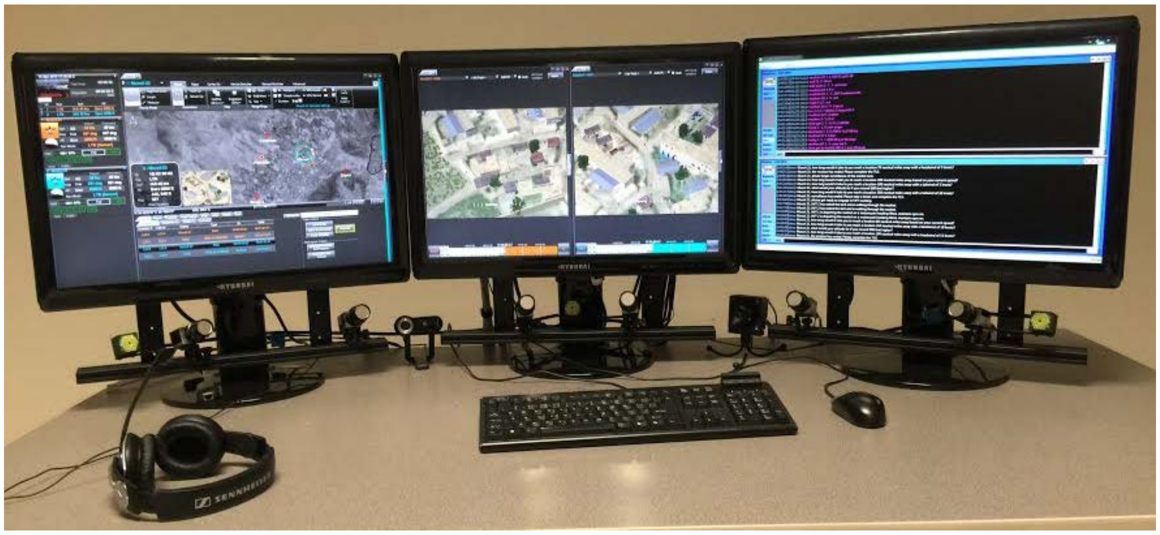

Figure 4: Vigilant Spirit Experiment Setup

This study's task analysis of Vigilant Spirit is accomplished through physical observation of the simulation itself during an experimental dry-run, as well as through analysis of the human subjects' data collected by the 711th HPW. The behavioral dataset from the 711th HPW's experiment is used to analyze the activities the subjects accomplished, the order activities were performed, and tasks in which the subjects succeeded or failed. The analysis is used to build task networks as a way of visually representing system and human tasks.

Build System-Centered Diagrams

In order to build the SysML diagrams of the system, the necessary information must first be identified and collected. The requisite information is dictated to some extent by the focus of the modeler, whether looking at structural/physical data or behavioral aspects of the system. Regardless of focus, at a minimum the information collected will include identification of relevant subsystems, how they communicate with each other and with external entities, and what information is passed back and forth therein. The task analysis' identification of the system's internal tasks and processes is particularly useful for building activity or sequence diagrams. This is because the focus of such diagrams is to represent the activities involved in performing a certain mission, with varying levels of detail. Within this study's human context, the most relevant diagrams will be behavioral and activity-based.

Build Human-Centered Diagrams

Building the human-centered diagrams is accomplished similarly to the system-centered diagrams. Whereas the system-centered diagrams represent the system primarily from the systems engineer's point of view, the human-centered diagrams instead represent that same system from the unique perspective of the human factors engineer. Building these human-centered diagrams can be accomplished by using UCD concepts. By interacting directly with the end-user, modelers can see how the system is experienced from the user's perspective and define human considerations more easily. In this manner, aspects of the user and its interaction with the system may be uncovered that would otherwise go unaccounted. This approach is similar to a human factors engineer's approach to defining the system. A key factor upon which to focus when generating these diagrams is how the user communicates with the system. What interfaces are used to communicate with the system, and which of the user's senses are utilized to interact with those interfaces? Cognitive processes should also be analyzed with respect to these interactions, including: the choices or decisions the user makes, how the interface design affects the user's workflow, and the user's desired workflow. The focus is now specifically on the user as part of the system. As part of the UCD approach, the process of understanding and modeling the system may involve some iterations of user interaction in order to get the diagrams to a sufficient point to move forward.

This study will build the human-centered diagrams by also using SysML. SysML is the standard modeling framework in SE, thus we are intentionally modeling the human using this existing framework in order to encourage wider adoption and communication with the SE community, and to facilitate rapid integration. If SysML is able to meet systems engineers' needs and sufficiently represent HSI considerations, then it would greatly facilitate integration between the two disciplines. Because the human aspect of the system is more behavioral in nature, SysML activity and sequence diagrams play an important role in the human's representation.

Compare and Analyze the Differences

The generated system- and human-centered diagrams are qualitatively compared to identify and analyze the differences between them. The relevant diagrams that are generated from both the system's and human's focus, in this case activity and sequence diagrams, tend to have similarities in overall mission-related activities and tasks performed. These similarities found in both sets of diagrams may be used as common anchor-points with which to compare the system's handling of tasks versus that of the human. For example, a single task may include both human and system involvement, therefore the task will appear on each of the separate diagrams. This common task would then serve as an anchor-point, connecting the separate human and system inputs that feed into that task. This visual comparison of the system- and human-centered diagrams aids in determining perspective-related differences.

Create an Integrated Set of Diagrams

The differences noted by comparing the separate system- and human-centered diagrams can be used to create new diagrams which integrate both the system and human perspectives. Specific areas highlighted by one set of diagrams can be used to augment the other diagrams that do not focus on the same areas. The result is a single set of diagrams that accounts for both the system and human by combining the system- and human-centered diagrams' strengths while minimizing their individual weaknesses. Re-iteration using the same UCD concepts as in the human-centered diagrams may also be helpful with these integrated diagrams in order to ensure relevant human considerations have been maintained.

Results and Analysis

The process of a subject participating in the Vigilant Spirit simulation was analyzed to identify the various tasks performed by the system and operator throughout the procedure. The results of the task analysis revealed essentially three separate processes occurring during the simulation: two system processes and one human process. The system has an independent set of activities it performs for the surveillance and communication tasks, each of which precipitate response activities from the human operator. For example, the system spawns a HVT at the beginning of each of four iterations, for which the operator must search, indicate if found, and then zoom in and follow. The system also asks four iterations of communications questions, prompting calculations and answers from the operator. A SysML activity diagram was selected to represent these tasks, shown in Figure 5. Activity diagrams are conducive to representing task analyses during early system design because they are able to visually depict mission activities at a high level, allowing modelers to consider the actors, decisions, and task flows involved.

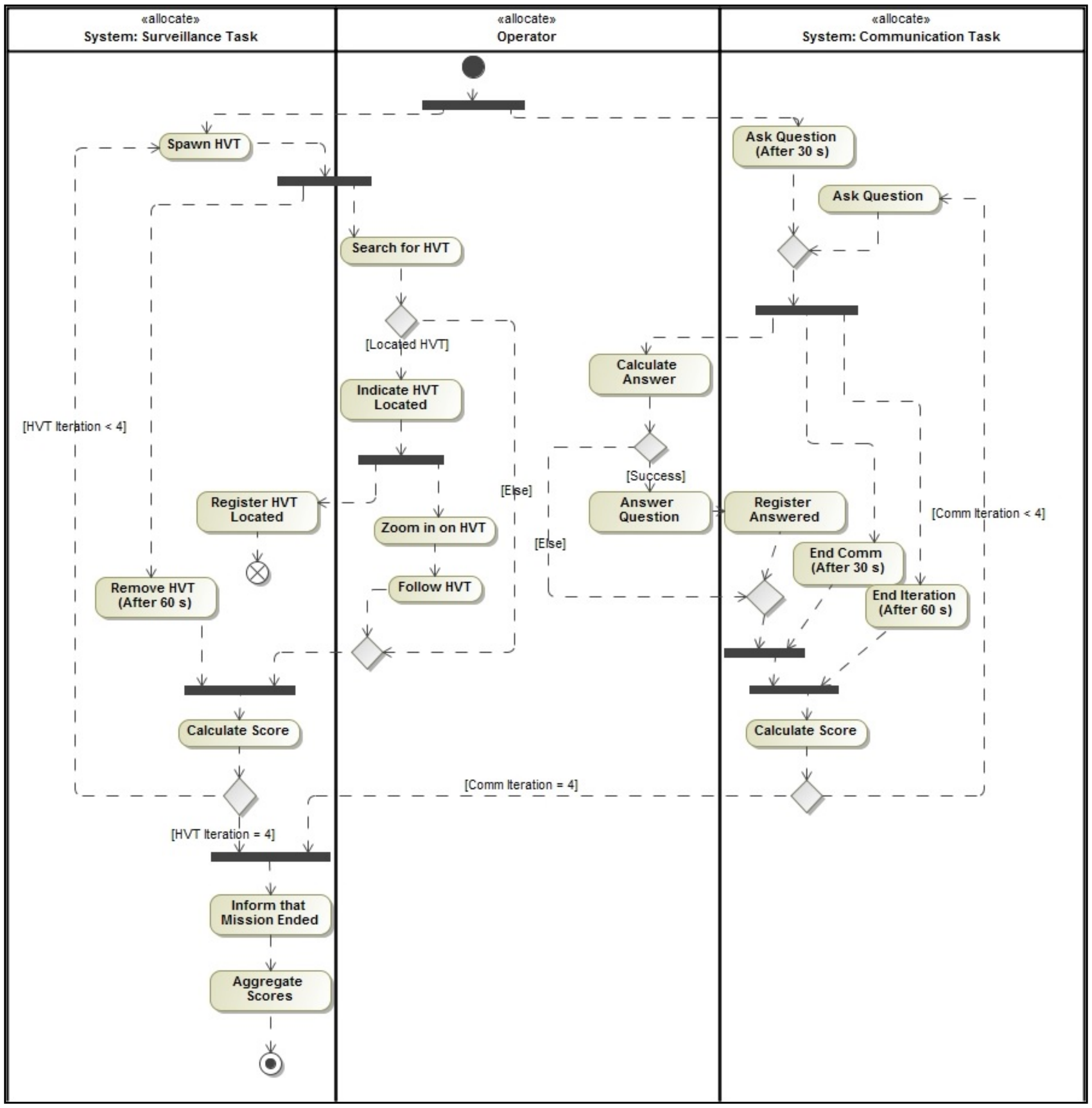

Figure 5: Activity Diagram of Vigilant Spirit Tasks

The activity diagram in Figure 5 also offered our first look at representing the system through SysML diagrams. Its broad depiction of the system's activities and interactions served as a basis with which to expand upon and incorporate more details in new diagrams. Sequence diagrams were used for this purpose, as they are better suited for illustration of subsystem activities and inter-system communication.

When creating the sequence diagrams, it became clear that the Vigilant Spirit system is actually composed of two separate subsystems: surveillance and communication. The surveillance and communication tasks occur independently of each other from a system standpoint, and would likely use different hardware in a real-world implementation. Therefore, we divided each of the surveillance and communication subsystems into their individual components with separate sequence diagrams instead of representing them as one system. Doing so allows for a functional allocation of who or what will be handling these different system aspects. The system-focused sequence diagrams of the surveillance and communication tasks are shown in Figure 6 and Figure 7, respectively.

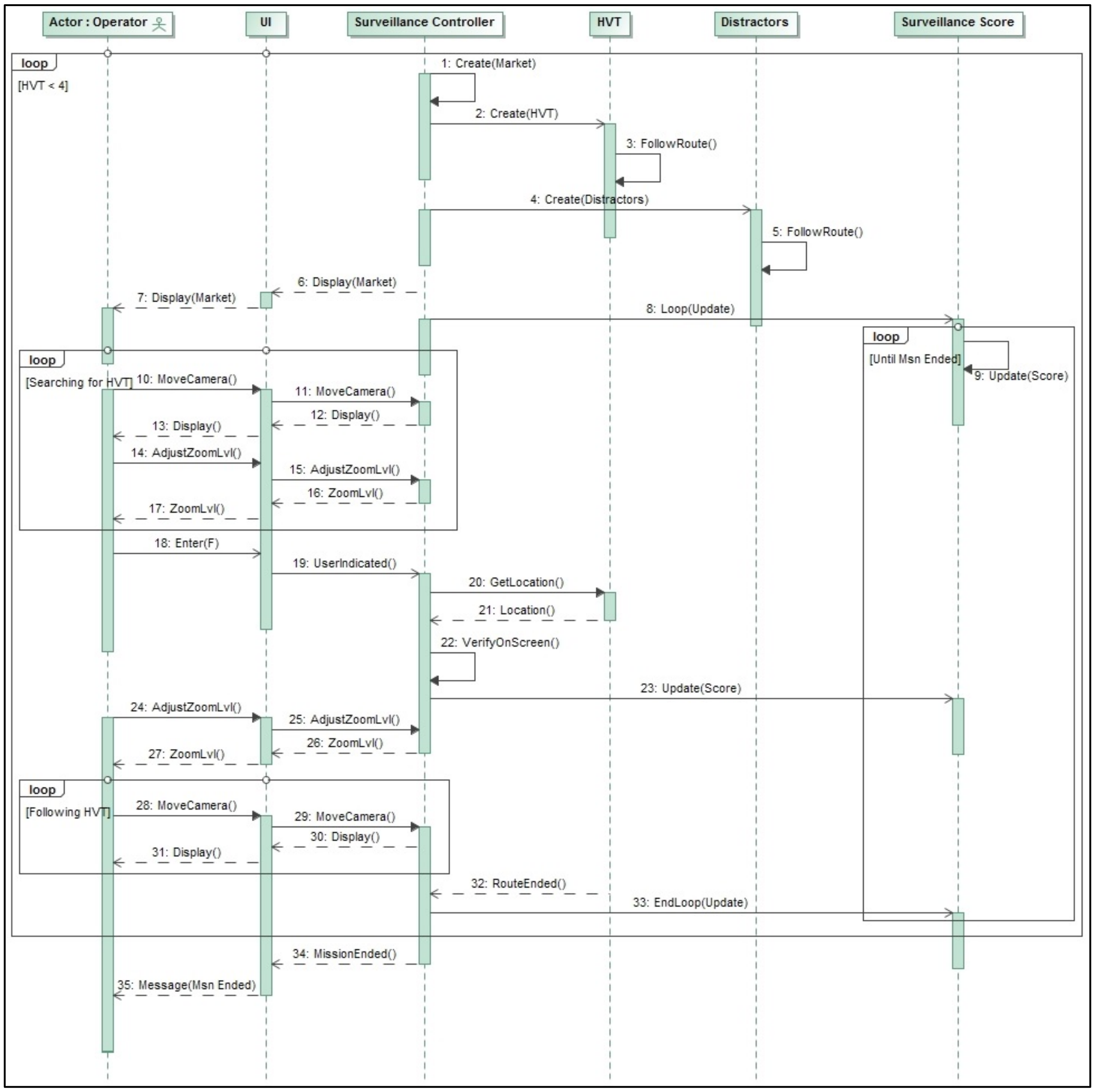

Figure 6: Sequence Diagram of System-Centered Surveillance Task

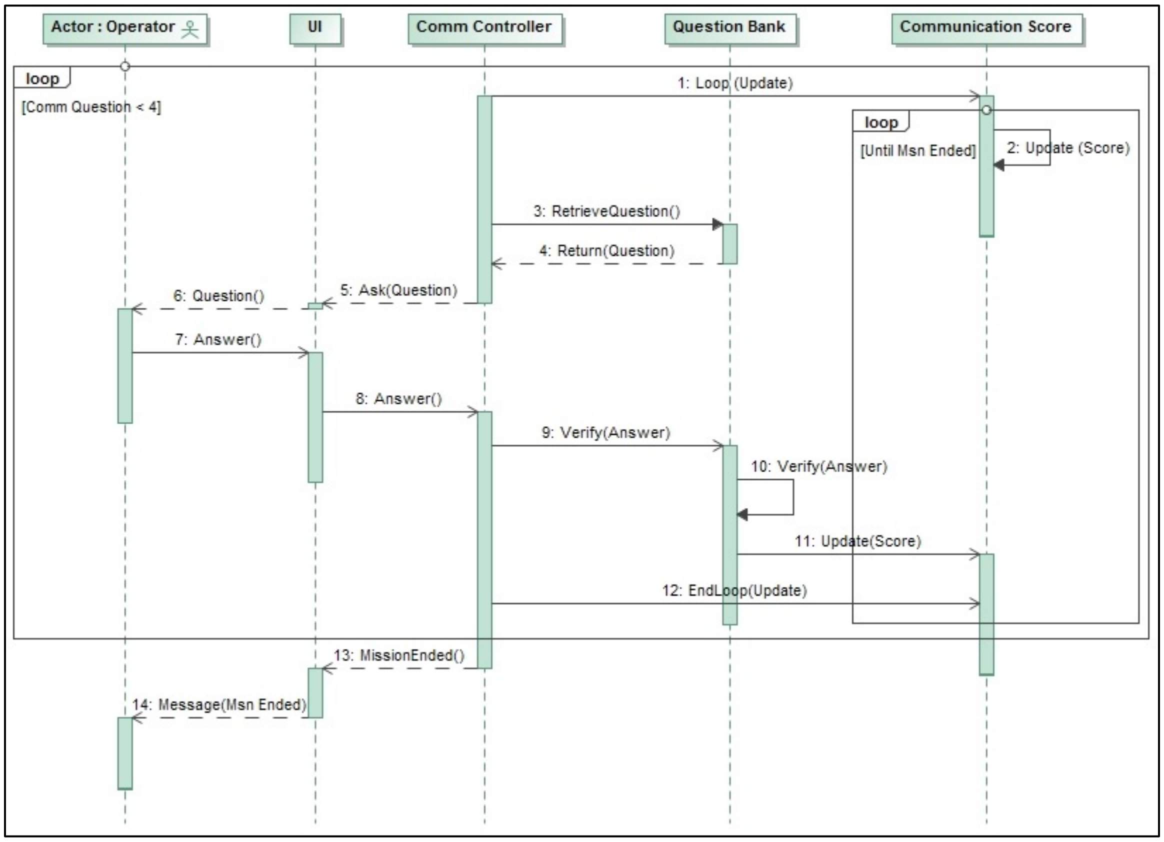

Figure 7: Sequence Diagram of System-Centered Communication Task

In the surveillance diagram in Figure 6, the system is divided into five abstract subcomponents: the user interface (UI), controller, target, distractors, and score. The use of a UI and controller are common when depicting software-based systems. It is important to note that even though these are system-centered diagrams, the human is still represented to a degree. The operator, represented by a single lifeline on the leftmost side of the diagram, interacts solely with the UI. The controller manages the system's activities and timing, creates and manipulates objects such as the HVT and distractors, and delegates tasks such as continuously updating the score until the mission has ended.

The sequence diagram generally depicts the same task flow as the activity diagram, but with more details of what is specifically involved with each task. For example, searching for the HVT consists of the operator continuously sending commands to the controller via the UI to move the RPA camera and adjust the zoom level until the operator finds the HVT. By contrast, the activity diagram in Figure 5 represents searching for the HVT simply by a single action node. The communication diagram in Figure 7 is similarly-focused, with the system divided into four subcomponents: the UI, controller, question bank, and score.

Because the sequence diagrams were built from a systems engineer's point of view, less emphasis is placed on the user. To rebuild the diagrams from a human factors engineer's perspective instead, we analyzed in more detail the ways in which the user specifically interacts with the system. In the same manner that the system was split into subcomponents, the user can likewise be represented not just as a single entity, but composed of several "subsystems" or resources, where each performs specific tasks. For example, listening tasks such as hearing the communication question can only be performed by the human's auditory system. Likewise, response tasks such as indicating the HVT as found or answering the question are performed by the human's motor systems. Though Vigilant Spirit is an existing system, if the system had not yet been designed, the human factors engineer would have options for the implementation of certain tasks. For example, responding to the communication question could occur orally through headset or manually through keyboard. These options could have implications not only for system design but for overall system performance.

As these diagrams are purely human-centered, less emphasis is placed on inter-system events and instead the system is abstracted to just focus on its interaction with the user. The re-designed, human-centered sequence diagrams are shown in Figure 8 and Figure 9, in which the human is divided into its visual, auditory, cognitive, and psychomotor components (highlighted in blue) and the system's UI is further divided into three subcomponents with which the user interacts: the computer keyboard, mouse, and monitor (highlighted in green).

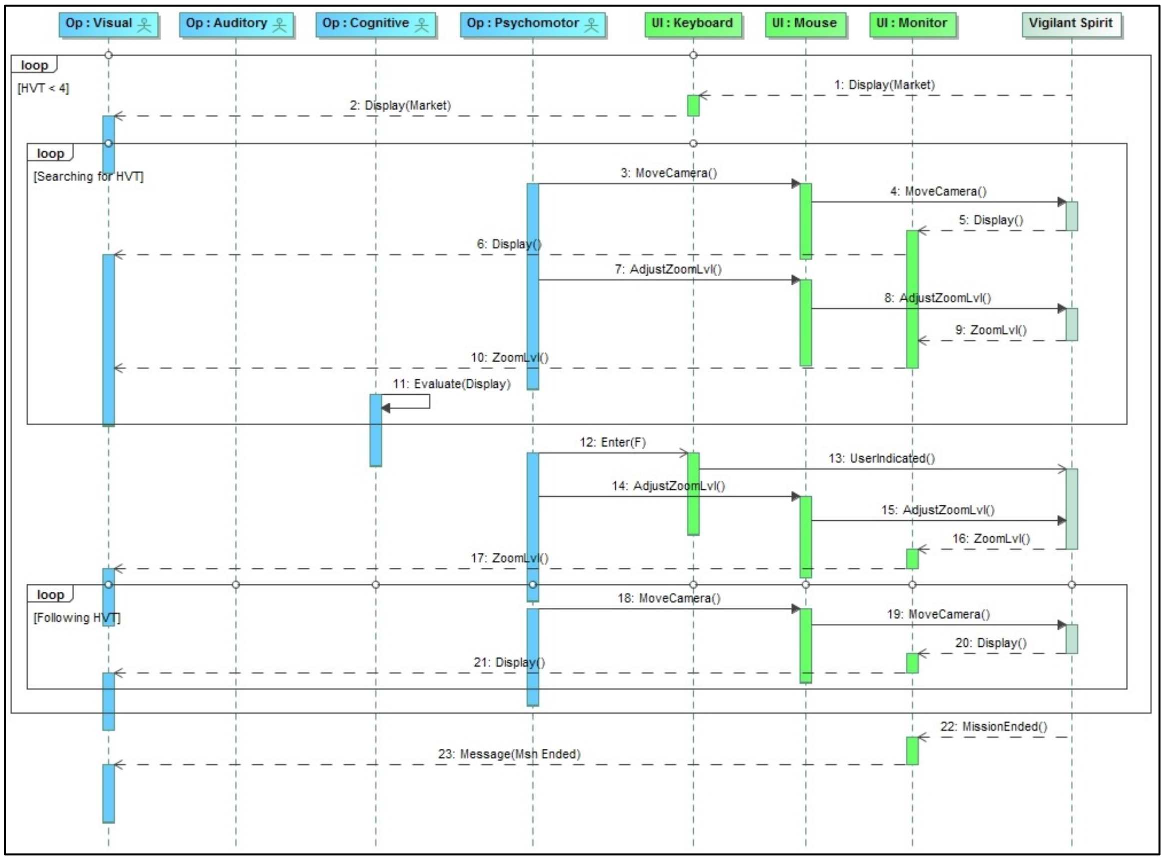

Figure 8: Sequence Diagram of Human-Centered Surveillance Task

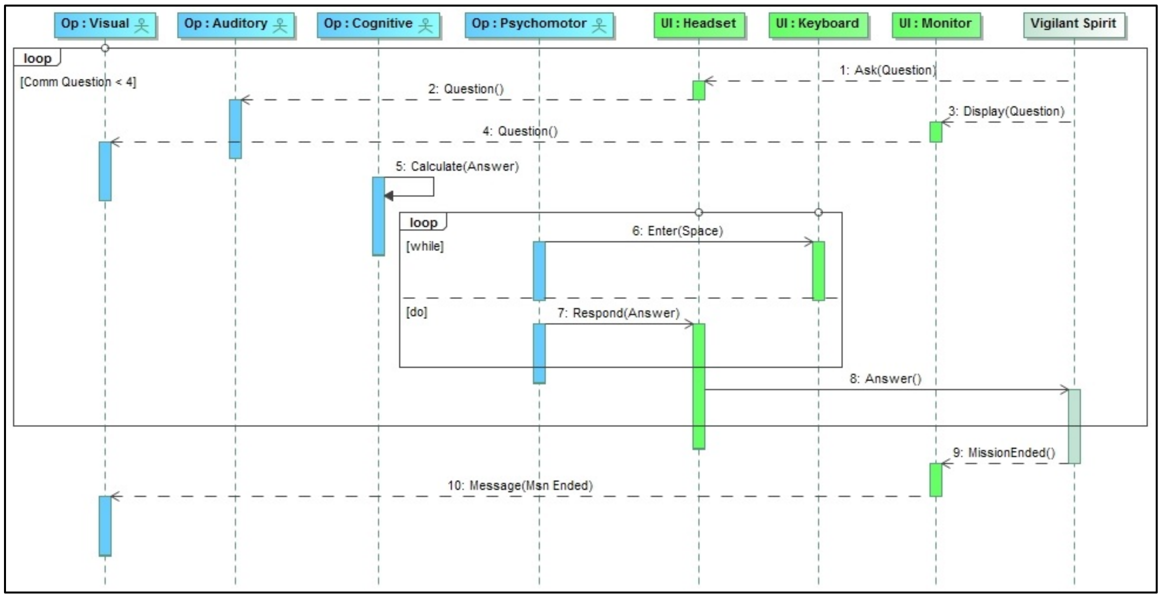

Figure 9: Sequence Diagram of Human-Centered Communication Task

Having sets of diagrams from both the system's and human's perspectives, we qualitatively compared the diagrams to find similarities and differences. The system-centered sequence diagrams contained detailed depictions of Vigilant Spirit's subsystems and its inter-system communication while treating the user as a "black box." Conversely, the human-centered sequence diagrams focused upon the human's subcomponents and the ways in which they interact with the system while treating the system as a "black box." However, each set of diagrams' narrow focus is also its unique strength, providing system and human insights into Vigilant Spirit that demonstrate the benefit of creating an integrated set. Because the medium with which the user and system communicate to each other is the UI, this served as the bridge to connect the two diagram sets. The integrated diagrams are shown in Figure 10 and Figure 11 with the UI’s subcomponents highlighted in green.

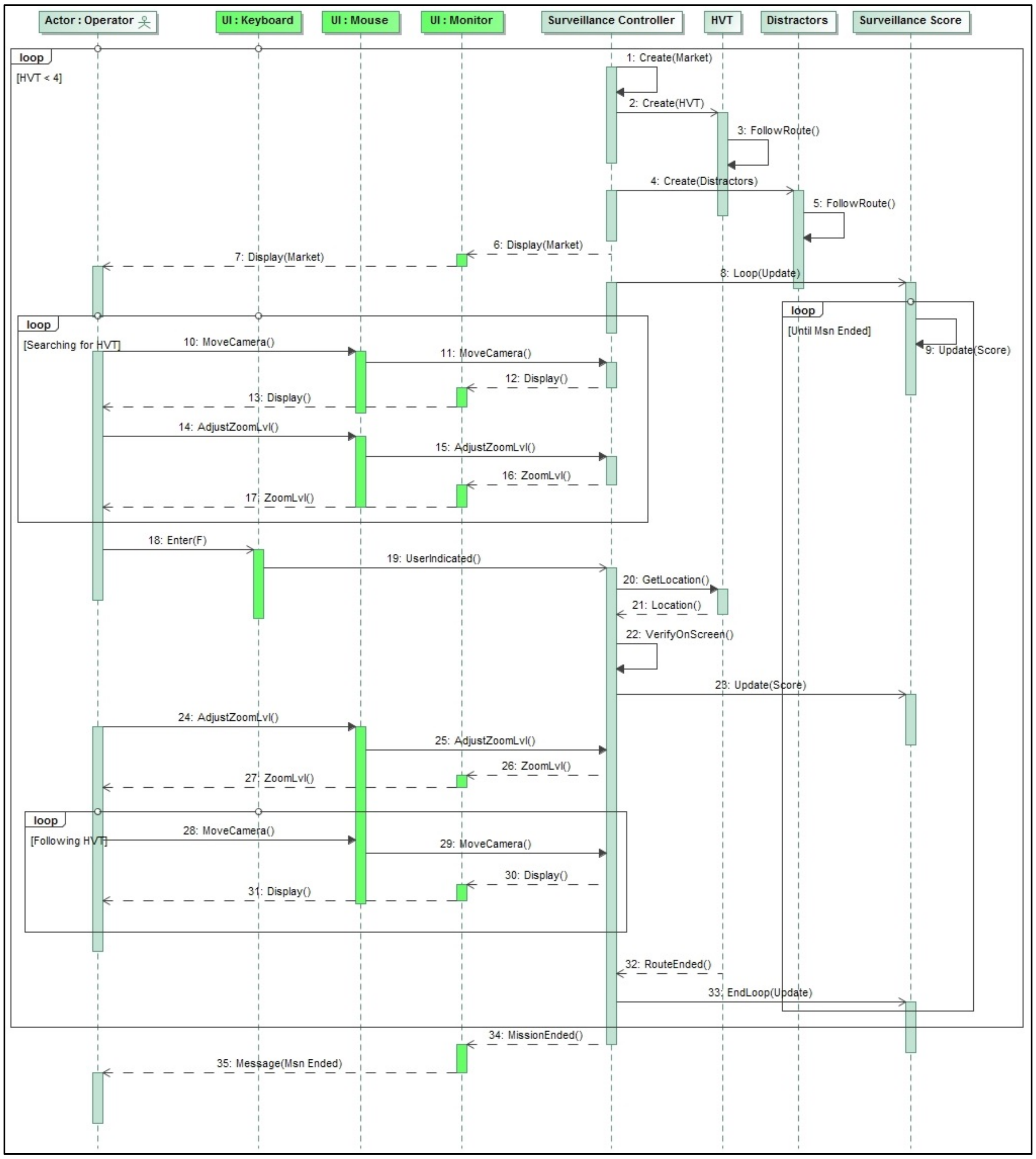

Figure 10: Sequence Diagram of Integrated Surveillance Task

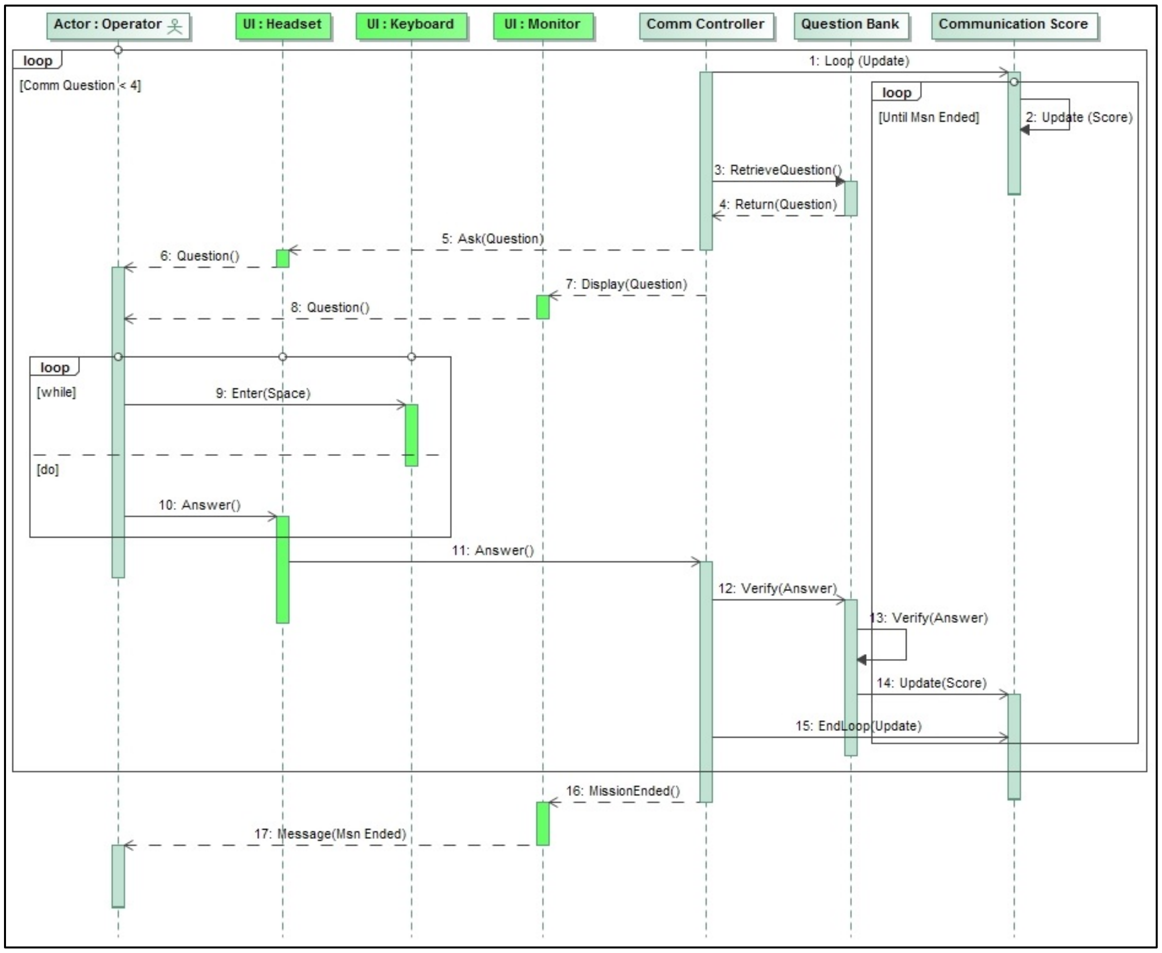

Figure 11: Sequence Diagram of Integrated Communication Task

Because the UI is the anchor-point between the system and user, incorporating the UI’s subcomponents into the integrated diagrams provides the systems engineer insight into human-system interaction without also needing to include the human’s resources. Thus, the diagrams allow for the necessary amount of detail for a systems engineer’s area of interest in a modeling language with which the systems engineer is familiar. It is mainly important that the human is considered and included in the system diagrams; its resources are implied by the UI breakout and sufficiently represented therein.

The benefit of creating these integrated diagrams is the ability to gain additional insight into the human processes and interfaces involved while the user is interacting with Vigilant Spirit. For instance, by specifically depicting the types of interfaces with which the user interacts and when the user must use them, the potential for imbalance of resource allocation may be more easily identified. An example of an imbalance would be if the user were required to answer the question by typing the answer while still needing to search for and indicate the HVT, thus requiring the use of the same keyboard/mouse interface for concurrent tasks. This benefit comes without needing to sacrifice detailed models of Vigilant Spirit and its subsystems.

Discussion and Conclusions

As with the specific Vigilant Spirit scenario, creating a set of SysML diagrams integrated with UCD concepts provides advantages for systems in other contexts as well. Systems engineers understand the benefit of dividing the system into its subcomponents, thus allowing the system designer to look at system aspects in detail and the interaction thereof. Dividing the human into its “subcomponents” provides the same benefit. A result of this representation is the realization that if the human’s subsystems are tasked to do multiple tasks simultaneously, conflicts may occur. The human-centered diagrams represented the human’s visual, auditory, cognitive, and psychomotor subcomponents as resources of attention. If one of those resources is allocated to a specific function, it may not be available for another function. The integrated diagrams capture this concept by depicting how the human interacts with the system interfaces, thus allowing the systems engineer to determine the human's attention capabilities. For example, if two tasks are designed to use a keyboard and monitor, the user may have to disengage from one task in order to perform the other. Whereas, if one task were an auditory/oral task and the other a visual/manual task, then the user could potentially multitask and perform both simultaneously. By revising system diagrams to include the human, systems engineers can gain insight into the possibility for the human to perform all or some of its allocated tasks, and the potential for conflicts. Accounting for resource allocation helps mitigate creating unrealistic expectations of the human's naturally limited resources of attention. This consideration would not necessarily be otherwise accounted for by normal SE practices.

Future work in this area of study will focus on further bolstering SE-HSI process integration using human performance modeling. A discrete event simulation software tool called the Improved Performance Research Integration Tool (IMPRINT) will be used to fully capture the dynamic aspects of the human's performance. By utilizing IMPRINT, we will be able to analyze the interaction between the system and human across a range of dynamic activities occurring in the Vigilant Spirit environment. Additionally, this proposed integration method need not be limited to the specific MBSE and HSI tools this research used. There may be other modeling tools besides SysML which would yield the same benefits from integrating human considerations. Likewise, though this paper focused on the human factors engineering domain to integrate, a future goal is to expand integration efforts across the rest of the nine HSI domains.

To better integrate human considerations into system designs, it is necessary for systems engineers to first acknowledge and consider the human as an important part of the system. However, mere acknowledgment is not enough if the human is not integrated sufficiently into the SE process. Similarly, human factors engineers need to be a part of the SE process in order to ensure sufficient human integration. Integration needs to be sufficiently scoped and at a level of detail that is able to capture the important aspects of the human as well as implications for human-based system performance effects. At the tools level, MBSE's SysML is a vehicle by which this may be accomplished, and this study's proposed approach provides an avenue to achieve that goal. Localizing UCD early in the process allows for a reduction in total system cost while still achieving effective user interfaces. By performing a system task analysis, creating and analyzing system- and human-centered diagrams, then creating an integrated set of diagrams to inform system design, both systems and human factors engineers will be able to effectively account for the human as a crucial component of system development.

Chapter Summary

This chapter addressed the first two investigative questions of the research. The following is a discussion each question:

1. What information should be captured in human-centric and system-centric models to enable effective integration?

To answer this question, the type of information needed depends upon the purpose of the integration effort. Since the purpose of this article was to integrate human- and system-centric models through interface design, the behavioral interactions between the system and human were analyzed. This purpose also drove the use of behavioral SysML diagrams. Activity and sequence diagrams were specifically chosen with which to perform integration. Activity diagrams offer a high-level look at the actions the system and human must perform during the mission, in terms of inputs and outputs through a flow of activities. Sequence diagrams look at these activities in more detail, with a focus on processes internal to the system and human in terms of messages sent between subcomponents. Because of the behavioral nature of these diagrams, relevant information about the human could be more easily identified, such as the human's actions and resources. Note that the other behavioral SysML diagrams, state machine and use case, might also be successfully used as well.

With the types of SysML diagrams chosen, the question's focus shifts to building these diagrams. SysML diagrams are traditionally understood as being system-centered. However, this article used a unique approach where SysML was used to build both system- and human-centric models. In this manner, we stayed within the SE toolset of SysML and folded the human perspective inside of it. For the system-centric models, it is necessary to identify what systems and subsystems there are, how the systems communicate with each other, and what information is exchanged. For the human-centric models, it is necessary to identify how the human communicates with the system, which of the human's resources are used to communicate, and with what interfaces that communication occurs. The method of building both types of models is similar, but with a shift in perspective. The activity diagram in the article allowed an initial look at Vigilant Spirit and served as a stepping stone to building the sequence diagrams, which were then used as the diagrams with which to integrate the two perspectives.

2. What considerations and decisions must be made when integrating between human-centric and system-centric models?

The previous investigative question addressed what types of SysML diagrams to integrate and what information is needed to build system- and human-centered SysML diagrams. This question addresses what is involved during the actual integration of the two types of models. As the purpose of the final model is to offer systems engineers an integrated human- and system-centered view of Vigilant Spirit, it is important to consider what information is necessary to include in these integrated diagrams. However, it is important to not merely try and include all the information from both models, as doing so would result in an overabundance of information and unnecessarily cumbersome models. Some of this information may be superfluous and irrelevant to the systems engineer, thus degrading the quality of the final integrated model. Therefore, a balance between the two types of models must be achieved and a decision must be made on what information to include and not include in the integrated model. When performing a combined modeling methodology of staying within a particular toolset, such as this article's use of SysML, it is important to include only the information relevant to that particular toolset. To aid in making that determination, this article suggested the use of similarities as anchor-points between models. These anchor-points can be used to direct the focus of the integrated model.

III. Performing System Tradeoff Analyses Using Human Performance Modeling

Chapter Overview

This chapter continues the concepts presented in the previous chapter by revisiting the first two investigative questions, in addition to addressing the final two investigative questions. Specifically, the new investigative questions addressed are what information can currently be passed between IMPRINT and SysML models, and what information do SysML models actually need to inform tradeoff analyses. The chapter discusses a dynamic method of integration, in which human factors engineers can use IMPRINT's human performance modeling functionality in order to perform tradeoff analyses that inform system design. This methodology maintains the SE and HSI disciplines' individual toolsets while enabling cross-communication. The work presented in this chapter is an adaptation of a journal article in submission.

Abstract

The human is a critical aspect of the system, but there is generally a failure to properly account for human capabilities and involvement during system design. This research presents an approach for human factors engineers to integrate the human into system models using human performance modeling. Starting with a set of system-centered Systems Modeling Language (SysML) diagrams, a task analysis is performed to understand the user's tasks and to create a baseline model in the Improved Performance Research Integration Tool (IMPRINT). An alternative IMPRINT model is created with varying design parameters and utilized to perform a tradeoff analysis between system trades. Through the tradeoff analysis, constraints and assumptions placed on the human

are verified and the results applied to create a human-system integrated set of SysML models. These new system models account for the human, allowing systems engineers to make more informed system design decisions.

Introduction

The human is a crucial aspect of nearly every system, making it vital to consider during system development. Designers can effectively account for the human during systems engineering (SE) by practicing Human Systems Integration (HSI) (U.S. Air Force, 2010). However, research indicates that HSI is not often applied during the SE process, especially early, resulting in the human being inadequately considered (Orellana & Madni, 2014). Failure to properly account for the human during early system design decisions, when tradeoffs are being made, may lead to unreasonable expectations of overall system performance. The human's performance also needs to be captured to truly understand the ramifications of design decisions.

Not understanding the interaction between the human and system can lead to significant system failures. For example, investigations involving the Global Hawk unmanned aerial vehicle and Patriot air and missile defense (AMD) systems have attributed some serious issues to improper system design. In both cases, during the implementation of automation, system designers undervalued the human's role. This led to Global Hawk and Patriot operators over-relying on their system's automation and losing situation awareness. The final result was mission degradation and, in the case of the Patriot AMD system, two fratricide incidents when the system misclassified friendly forces as threats (Hopcroft, Burchat, & Vince, 2006; Hawley, 2011).

The previous examples highlight the importance of accounting for the human's capabilities and role in system design. Failure to do so may result in a system design that makes unrealistic assumptions about the human, leading to an overestimation of the human's capabilities and thus the system's capabilities. To fully understand the system's performance, the HSI process must be better integrated into the SE process. This paper proposes a method of integration that allows human factors engineers to use human performance simulation to verify and update system models. Using these updated models, systems engineers can make design decisions that also account for the human.

Background

Human Systems Integration

HSI is a process that allows for effective consideration of the human during system design. The U.S. Air Force divides HSI into nine domains, and considers human factors engineering as the primary domain with which to focus integration (2010). One of the methods of practicing human factors engineering is through human performance modeling, where the human is modeled mainly via simulation (Allender, 2000). One such modeling tool is called the Improved Performance Research Integration Tool (IMPRINT). Developed by the Army Research Laboratory to support HSI efforts, IMPRINT is used to analyze the interaction between the system and humans. IMPRINT allows the analyst to first represent a mission in terms of a series of functions and tasks performed by both the system and human, then run a discrete event simulation (DES) of the system and human accomplishing the mission. In this manner, the analyst can observe effects on performance and cognitive workload (Mitchell, Agan, & Samms, 2011). For a more detailed review of the definitions relating to HSI, see Appendix A.

Systems Engineering

The data gained from HSI analyses, such as those using IMPRINT, can be used to inform the SE process. SE offers a unique approach to system development by integrating all the components and disciplines of the system throughout its lifecycle (International Council on Systems Engineering, 2015). Model-based systems engineering (MBSE) is a method of practicing SE, where the system is represented through descriptive models and analytical simulation (Friedenthal, Moore, & Steiner, 2014). The Systems Modeling Language (SysML) provides a means of visualizing MBSE models through diagrams (Delligatti, 2014). For a more detailed review of the definitions relating to SE, see Appendix A.

Although it is important to consider HSI during the early stages of the system's lifecycle, this is not often accomplished. This lack of integration contributes to higher lifecycle costs and decreased compatibility (Orellana & Madni, 2014; Hardman & Colombi, 2012; Mitchell, Agan, & Samms, 2011), as well as more serious consequences, as previously illustrated.

This paper explores integration of the HSI process with SE by creating system models with MBSE and conducting a theoretical trade study using IMPRINT. To make this trade study representative of those conducted in the real world, the sections below review the types of tradeoffs considered using MBSE and IMPRINT and the methods with which such trade studies are performed.

MBSE Tradeoffs and Methods

Systems engineers, now using MBSE practices, perform tradeoff analyses involving several factors such as cost, mission effectiveness, size (weight and volume), performance, and the “-ilities.” Cost is a common factor among system tradeoffs (Crane & Brownlow, 2015; Do, Cook, & Lay, 2014; Russell, 2012). It is often traded between factors influencing mission effectiveness such as supportability (Russell, 2012) and performance (Crane & Brownlow, 2015). System designers commonly have to make decisions regarding increasing the performance of a system at the expense of also increasing system cost. A balance must be reached between the level of performance desired by the system stakeholders and an acceptable total cost.

Mission effectiveness is a broad factor that includes system tradeoffs such as mobility, survivability, supportability, and performance (Cloutier, Sauser, Bone, & Taylor, 2015; Crane & Brownlow, 2015; Kaslow, Soremekun, Kim, & Spangelo, 2014; Russell, 2012). These individual factors may be traded between themselves or other tradeoff factors such as equipment weight and volume (Cloutier, Sauser, Bone, & Taylor, 2015; Crane & Brownlow, 2015; Kaslow, Soremekun, Kim, & Spangelo, 2014). For example, decreasing a system’s weight and volume may increase mobility, which affects the system’s survivability and overall mission effectiveness (Cloutier, Sauser, Bone, & Taylor, 2015). However, increasing the system’s volume may allow for more armor or ammunition, thus increasing the system’s lethality and again affecting survivability. Equipment weight and volume may also be traded with cost, such as within the context of a satellite constellation when considering orbital altitude and constellation size (Crane & Brownlow, 2015). At a higher altitude, fewer satellites are needed to cover an area, but at the cost of needing better-quality sensors. Conversely, spacecraft are cheaper at lower altitudes due to size and weight reductions, but more are needed to cover the same area.

When faced with similar alternatives, adding other tradeoff factors may aid in deciding on a solution. For instance, in satellite constellation design, other factors that could be considered are disaggregation, resiliency, and lower costs (Thompson, Colombi, Black, & Ayres, 2015).

MBSE tradeoff analyses are performed using both qualitative and quantitative methods. The primary qualitative method used to perform these analyses involves visualization of the system via SysML diagrams (Cloutier, Sauser, Bone, & Taylor, 2015; Russell, 2012). These SysML diagrams are used to analyze tradeoffs and enable system design decisions. Activity diagrams and use case diagrams provide the MBSE practitioner a way to graphically highlight dependencies between components within the system.

Quantitative methods mainly involve the use of simulations (Crane & Brownlow, 2015; Kaslow, Soremekun, Kim, & Spangelo, 2014). In these methods, MBSE parametric diagrams are commonly created to establish relationships between the system's requirements and design constraints, which then feed into simulation models. Based on the parameter inputs, the modeler can see the outputs' impact on mission performance and determine if requirements are being met. Aside from simulations, quantitative analyses may also be uniquely developed to suit the system (Do, Cook, & Lay, 2014). For example, Do et al. (2014) self-developed a tradeoff analysis method by first assigning a series of weight and value functions to the system tradeoffs, then evaluating those functions to quantitatively determine a system design solution.

IMPRINT Tradeoffs and Methods

One tool human factors engineers use is IMPRINT. Using this tool, human factors engineers perform tradeoff analyses involving several factors, which include manning, performance, workload, equipment design, and task allocation. These factors are different than those relating to MBSE because they focus specifically on the human instead of the broader system. However, they still indirectly relate to and affect some MBSE factors, such as usability and system performance. Manning is a key factor in many human-related system tradeoff studies (Allender, 2000; Mitchell D. K., Samms, Henthorn, & Wojciechowski, 2003; Mitchell, Samms, & Wojcik, 2006; Mitchell D. K., 2008). Even the U.S. Navy-developed predecessor to IMPRINT, called HARDMAN (Hardware vs. Manpower), was created with the intention of analyzing tradeoffs between hardware and manpower (Dickason, Sargent, & Bagnall, 2009).

Many studies examine the impact of a reduction in manning on the other tradeoff factors mentioned (Allender, 2000; Mitchell D. K., Samms, Henthorn, & Wojciechowski, 2003; Mitchell, Samms, & Wojcik, 2006). For example, Allender (2000) describes a trade study in which the manning on a U.S. Navy destroyer bridge was reduced with the expectation of maintaining the same operational performance. Various IMPRINT models were built to measure the variation in the crew’s workload and determine the feasibility of this plan. Results showed that the reduction in manning caused an unsustainable workload for the reduced crew. In addition to workload, the influence of manning reductions is also studied on equipment design (Allender, 2000) and performance, where performance may be defined in terms of mission performance (e.g. time taken to complete the mission) or in terms of human performance (e.g. the number of errors committed) (Allender, 2000; Mitchell, Samms, & Wojcik, 2006).

System automation is sometimes used to offset tradeoff factors such as manning (Mitchell D. K., 2003; Mitchell D. K., Samms, Henthorn, & Wojciechowski, 2003; Allender, 2000) and task allocation (Colombi, et al., 2011; Mitchell & McDowell, 2008; Wickens, Bagnall, Gosakan, & Walters, 2012). In the Navy bridge crew trade study, a proposed solution to offset the manning reduction was to supplement the bridge crew with automation (Allender, 2000). A similar solution was proposed in a trade study performed on task allocation for remotely-piloted aircraft (RPA) operators, in which task automation was suggested as a way to offload some of the operator's tasks and balance workload (Wickens, Bagnall, Gosakan, & Walters, 2012). However, Colombi et al. (2011) recommend the strategic implementation of automation, warning that simply automating the "easiest" functions could actually have a negative effect on workload.

Workload is a focus of many human-related trade studies performed using IMPRINT. The assessment of workload is usually placed in the larger context of evaluating other tradeoff factors like manning requirements or operator performance. Aside from manning, a modeler may wish to determine which crewmember could assume additional tasks with the least amount of added workload while maintaining performance (Mitchell & Chen, 2006; Mitchell & McDowell, 2008), or to determine the task allocation for the entire crew (Mitchell D. K., 2003). The type of study in which IMPRINT is used to evaluate the effect on workload from changing another factor, is common throughout the U.S. Army (Allender, 2000; Mitchell, Samms, & Wojcik, 2006; Mitchell & Chen, 2006; Mitchell & McDowell, 2008; Mitchell D. K., 2008; Cassenti, Kelley, Colle, & McGregor, 2011), Navy (Allender, 2000), Air Force (Colombi, et al., 2011; Wickens, Bagnall, Gosakan, & Walters, 2012), and academia (Harriott, Zhang, & Adams, 2013; Rusnock & Geiger, 2014).

Equipment or system designs may drive performance analyses, where performance is measured through IMPRINT workload modeling. For instance, Rusnock and Geiger (2014) performed a trade study which analyzed the effect on performance due to varying workload levels for each of four different system designs. While most workload studies deal with the human's cognitive workload, Harriott, Zhang, and Adams (2013) uniquely studied the effect on physical workload from a human-robot partnership system design.

These tradeoff factors may vary and even interchange as independent, dependent, and controlled variables, depending on the particular trade study's objectives. For instance, while equipment design was previously described as being dependent on manning, it could, conversely, influence manning. The number of crewmembers may need to be reduced to accommodate a smaller vehicle (Mitchell D. K., 2008), or the manning required to operate a system may need to be re-assessed due to an equipment re-design (Allender, 2000).

Tradeoff analyses using IMPRINT are performed using similar methods as in Allender's (2000) trade study of the Navy destroyer. A series of baseline and alternative models may be built in IMPRINT either as a feasibility study or to determine the tradeoff effect of one factor on another. IMPRINT trade studies may be implemented by simulating human workload or performance, where performance could be measured by factors such as task time, accuracy, or completion rates.

Integration Efforts via IMPRINT Modeling

There have been efforts to integrate the human into system design using IMPRINT, with all approaching integration in various ways. Mitchell, Agan, and Samms (2011) expanded upon IMPRINT's pre-conceived utility by modeling the system in addition to the human. They emphasize that deconstructing the system and human are essential to system development, but these processes should not be conducted independently of each other. If so, each side misses key variables that could have been otherwise accounted. Mitchell et al. (2011) used IMPRINT to model both the system capabilities and the human functions of a conceptual system in order to identify areas in which both sides can be accounted to improve success.

Smillie and Handley (2009) used IMPRINT to augment a human-focused architectural framework view called the Human View. While the Human View's purpose is to provide system developers a means to focus on the human, Smillie and Handley sought to use the dynamic nature of IMPRINT's DES capabilities to expand upon the Human View. First utilizing the Human View to define a model of a system in a sample case study, they then translated various components of the Human View into IMPRINT. For example, the Human View's Roles, Tasks, and Constraints subviews translated into IMPRINT inputs such as operators, assignments, and moderators. Finally, the IMPRINT model's outputs were analyzed to evaluate the system's impact on the human's performance and workload.

Both Mitchell (2005) and Colombi et al. (2011) used established system models to inform IMPRINT in order to perform system analyses. Colombi et al. used SysML diagrams representing the system's operational concept as a basis for defining the human’s tasks, which in turn fed the creation of a workload model. By analyzing the human’s envisioned tasks and resultant workload, IMPRINT is able to act as a method of assessing system feasibility early in development.

Although Mitchell (2005) used SysML’s predecessor, the Unified Modeling Language (UML), to build the diagrams in her study, the concept is similar to Colombi et al.’s (2011). Mitchell states that while the UML and IMPRINT are effective for developing system and human requirements, respectively, they are not affected by the other’s constraints as they should. For example, an activity diagram alone cannot properly depict human performance impacts on the system. Mitchell used a pilot study to develop an approach to link the two modeling methods. An activity diagram depicting the system was used to populate an IMPRINT model representing the human-system interaction, which was then run and the resulting workload outputs analyzed. Through this manual translation from UML to IMPRINT, the analyst is able to see the feasibility of the constraints placed on the human. However, Mitchell admits that a limitation to the study is the absence of a translation from the IMPRINT analysis back to UML, which would help ensure that the human is properly represented by the system.

Research Gap

There have been relatively few efforts to integrate the HSI process with SE using human performance tools like IMPRINT. These efforts largely utilize IMPRINT to assess human performance and workload with regard to the system, but lack an integration plan to inform system-level models following human-performance analysis. Further, SysML has been integrated with a variety of software tools (Rashid, Anwar, & Khan, 2015), but there is a noticeable lack of integration with HSI tools. SysML diagrams are indeed useful for informing IMPRINT models on which tasks, activities, and interactions need to be captured in a human-performance simulation. However, following IMPRINT analysis, results should feed back into SysML to allow the system model to capture a system-human integrated perspective. Any constraints on human-performance identified through the HSI tools may be used to update the system's requirements and constraints. This consideration enables effective attention to all aspects of the human in system design.

The purpose of this paper is to present an approach for human factors engineers to integrate human considerations into system design. Starting with a set of system-centered SysML diagrams, a DES model is built in IMPRINT in order to verify and update the system diagrams' constraints and assumptions placed on the human. The results are then used to inform the SysML diagrams. These new system models account for the human and may be used by systems engineers to enhance system trade studies, allowing for more informed design decisions.

Methodology

This study's methodology is similar to common human-centered design processes (Rogers, Sharp, & Preece, 2011), but adapted to include system elements and modeling. Using an example system as a case study, SysML diagrams are generated to serve as a pre-existing, system-centered basis. A task analysis is performed to understand the user's tasks, then a baseline model is created in IMPRINT and validated. An alternative model is next created with varying design parameters and used to perform a sample trade study between several system tradeoffs, the results of which are evaluated and applied to update the system diagrams.

Obtain or Create System Diagrams

For this study, SysML diagrams of the system are created to serve as a pre-existing basis with which to later integrate results. Realistically, these diagrams would either be obtained from the systems engineer or created from a systems engineer's perspective. As such, the emphasis in these diagrams is centered on the system. The focus of these diagrams could include the structural layout of the system, performance-related requirements, and performance- or specification-related constraints on the system. As in the previous chapter, this study's integration method is demonstrated using Vigilant Spirit as the example system. For an extended review of Vigilant Spirit and the 711th Human Performance Wing's (HPW) experiment, see Chapter II.

Perform Task Analysis

A functional decomposition/task analysis is performed on the system to identify the relevant task flows and interactions between the system and human. This task analysis is accomplished through observation of Vigilant Spirit and analysis of the 711th HPW's human subjects' data, where relevant activities and tasks performed by both the system and participants are identified and represented by task networks.

Of note for this study is that participating subjects are given scores for how well they perform the Vigilant Spirit surveillance and communication tasks. The score for the surveillance task is based on how long they can follow an identified high value target (HVT), while subjects are scored for the communication task based on how quickly they correctly answer the questions. The surveillance and communication scores together comprise a total performance score. A sample screenshot of the Vigilant Spirit camera feed displaying the HVT and marketplace is shown in Figure 12.

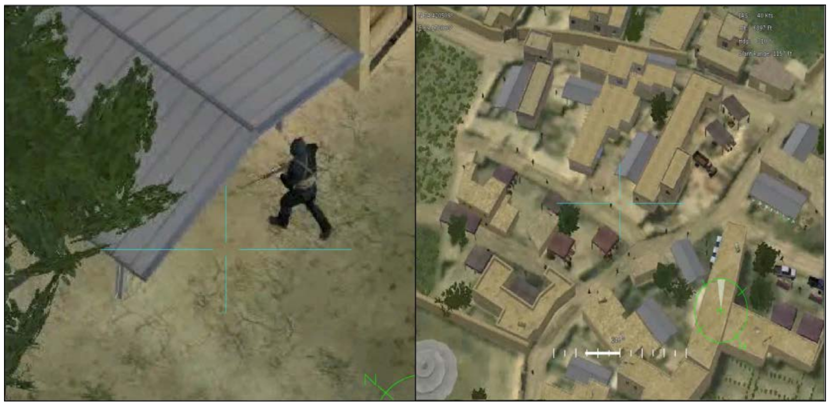

Figure 12: Screenshot of Vigilant Spirit Camera Feed

Create Baseline Model

The results of the task analysis are used to build a baseline model of the system in IMPRINT. Through the process of the task analysis, a conceptual model of the system is developed that translates into the IMPRINT task network. This task network captures the human and system tasks involved in performing a mission, the tasks' path flow, and the timing, accuracy, and associated workload of each task. Once the baseline model is built, it is verified against the conceptual model to ensure it performs as intended and its outputs make sense. Ways in which the model is verified include analyzing the task durations, the number of task repetitions, the model's overall clock timing, the values of variables, and trends in workload levels. Additionally, the model is peer- and subject-matter expert- (SME) reviewed to verify task sequencing, model assumptions, and task workload value assignments. After the baseline is verified, it is validated against Vigilant

Spirit's real-world data. The model's performance output data is statistically compared with real-world data. However, the model's workload output data cannot be directly compared with the real-world data, so validation occurs through SME review. Once the baseline IMPRINT model is validated, it can reliably be used to simulate the system in order to perform the tradeoff study.

Perform Trade Study

To perform the trade study, relevant system tradeoffs are identified. For the Vigilant Spirit tradeoff scenario, six different scan algorithms are selected as trades. These trades are driven by conflicting parameters such that one tradeoff is not clearly better than another. The six algorithms vary on independent variables of scan accuracy and speed. IMPRINT is the tool used to assess these trades in relation to the dependent variables of performance and workload. An alternative model is created in IMPRINT that varies the tradeoff parameters, the outputs of which are compared against the baseline to determine the effects on the system. From this trade study, the preferred alternative is identified.

Integrate Results into System Diagrams

Once the trade study identifies the preferred alternative, the results are integrated back into existing system diagrams. Originally system-focused, SysML diagrams like block definition, requirements, and parametric diagrams are updated to reflect the results of the tradeoff analysis. The result is a set of diagrams that accounts for both the system and human, which may then be used to better inform system design.

Results and Analysis

The intention of performing the trade study on Vigilant Spirit is to demonstrate the value of accounting for human-performance when making system design decisions based on total system performance. By considering a human-focused viewpoint of the system, potential HSI-related insights may be gained and integrated back into existing SysML diagrams of the system. An example scenario was created to serve as an impetus for system analysis. From the review of literature concerning tradeoffs, we see that automation is sometimes used to offset human workload as an HSI-related tradeoff factor (Allender, 2000; Colombi, et al., 2011; Mitchell & McDowell, 2008; Wickens, Bagnall, Gosakan, & Walters, 2012). Additionally, automation could also be used to potentially improve system performance – a key MBSE-related tradeoff factor. Thus, in our scenario, the system developer seeks to redesign Vigilant Spirit by incorporating system automation to improve HVT identification, and thus overall score. While the operator searches for the HVT, the system automation also scans the potential targets on-screen and notifies the operator if the system assesses that it successfully identified the HVT.

However, from a SE perspective, one might wish to specify the required performance of the algorithm. Two typical tradeoff factors associated with this system automation which might be evaluated are the speed and accuracy of the scanning algorithm. In this analysis, we assume that these two factors have inverse (but not necessarily linear) relationships, in that the faster the scanning algorithm's speed, the lower its accuracy and vice-versa. The scanning algorithm uses a number of features from the environment to identify the HVT, such as on-screen individuals' movements, behaviors, and equipment. As such, to increase the algorithm's scanning speed, fewer environmental criteria are used to assess potential targets, thus decreasing accuracy.

Six algorithm settings were chosen to analyze as trades, each with varying levels of speed and accuracy, listed in Table 1. Accuracy is defined in terms of the percentage of time a suggested target identified by the automation is actually the true HVT. Speed is defined in terms of the number of seconds it takes the automation to suggest a HVT, and is a Weibull probability distribution with a threshold of four and defined shape and scale parameters. The Weibull distribution was chosen because its versatility allows modeling of many types of characteristics, to include the search times for this scenario, and because of its similarity to the 711th HPW's human subjects search data.

Table 1: Scanning Algorithm Accuracy and Speed Settings

	Accuracy	Speed (4+Weibull)			
		Shape	Scale	Mean (M)	Variance (V)
Trade 1	90%	8.5	25.5	28.084	11.399
Trade 2		6.4	23.5	25.879	15.955
Trade 3	80%	6.7	20	22.666	10.669
Trade 4		4	17	19.409	18.687
Trade 5	70%	3.2	10	12.957	9.438
Trade 6		1.7	7	10.246	14.301

Obtain or Create System Diagrams

First, we created sample SysML diagrams depicting the Vigilant Spirit system with the addition of automation. These diagrams are modeled from a systems engineer's perspective. Because they are system-focused, the diagrams place less emphasis on the human. Figure 13 through Figure 15 show the SysML block definition, requirements, and parametric diagrams of the system, respectively.

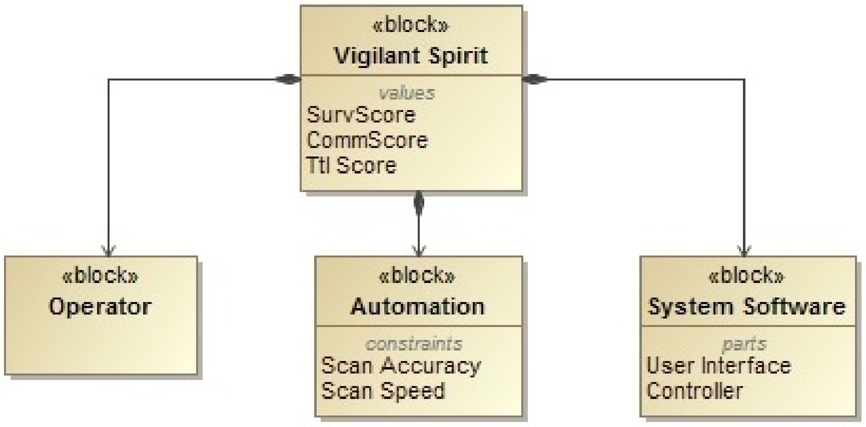

Figure 13: System Block Definition Diagram

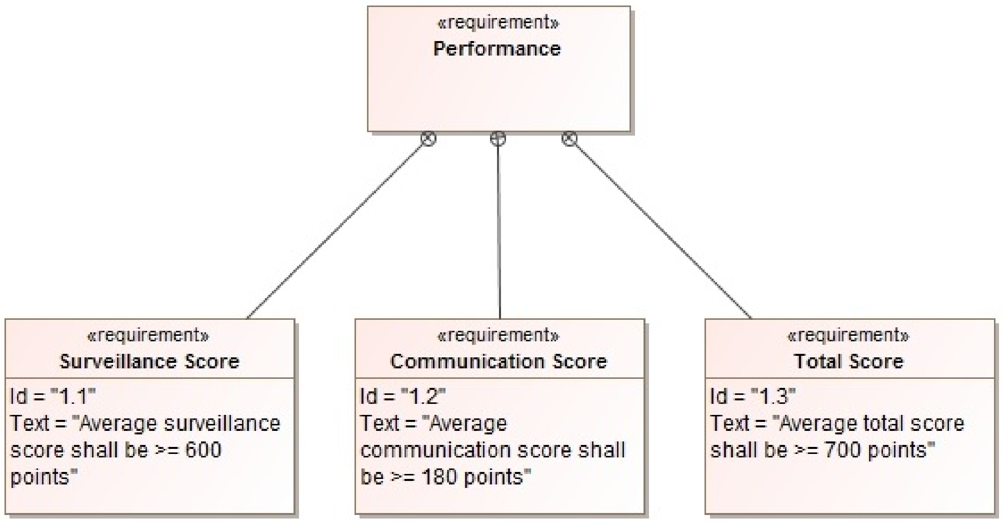
«requirement»
Performance
«requirement»
Surveillance Score
Id = "1.1"
Text = "Average surveillance score shall be >= 600 points"
«requirement»
Communication Score
Id = "1.2"
Text = "Average communication score shall be >= 180 points"
«requirement»
Total Score
Id = "1.3"
Text = "Average total score shall be >= 700 points"

Figure 14: System Requirements Diagram

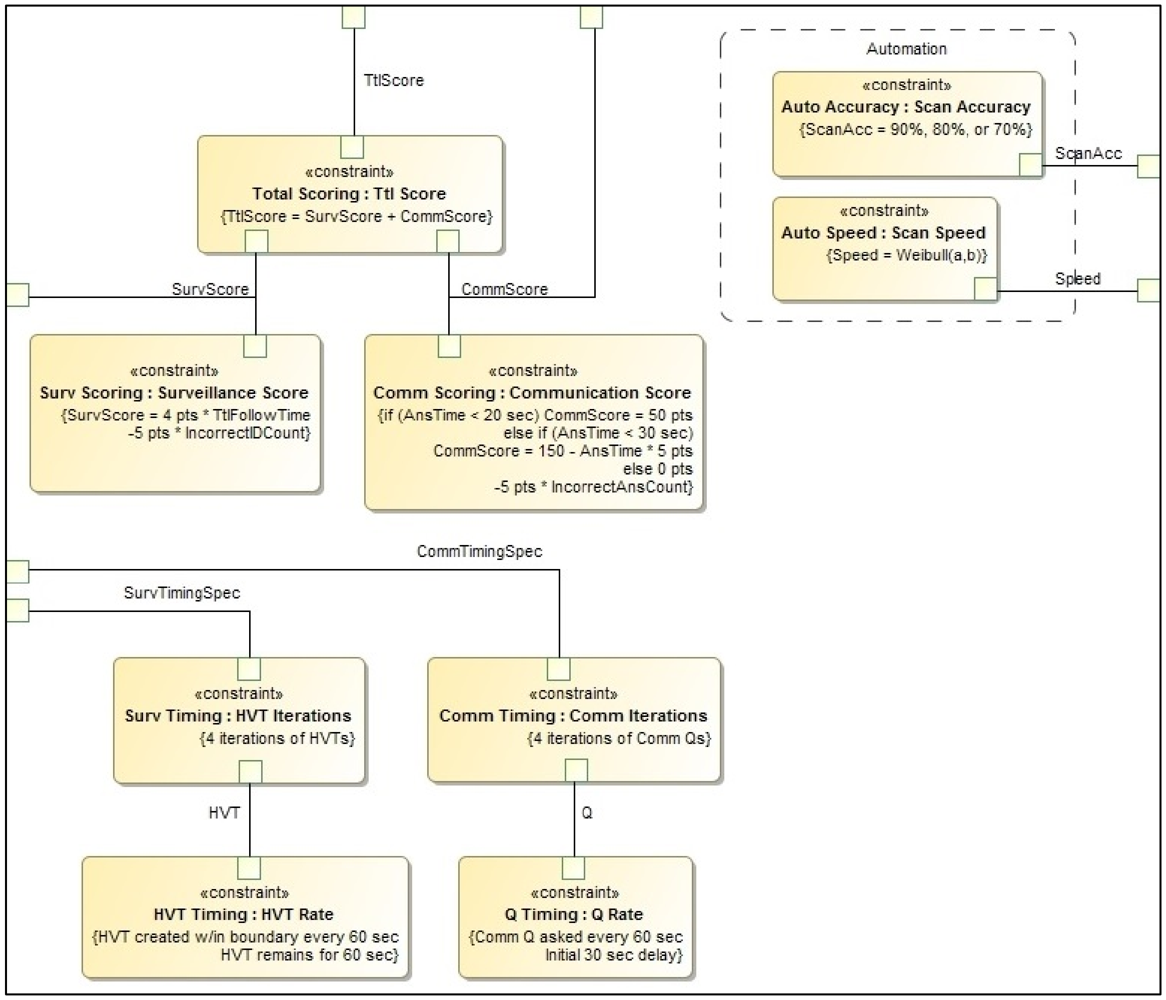
TtlScore
«constraint»
Total Scoring : Ttl Score
{TtlScore = SurvScore + CommScore}
SurvScore
CommScore
«constraint»
Surv Scoring : Surveillance Score
{SurvScore = 4 pts * TtlFollowTime
-5 pts * IncorrectIDCount}
«constraint»
Comm Scoring : Communication Score
{if (AnsTime < 20 sec) CommScore = 50 pts
else if (AnsTime < 30 sec)
CommScore = 150 - AnsTime * 5 pts
else 0 pts
-5 pts * IncorrectAnsCount}
Automation
«constraint»
Auto Accuracy : Scan Accuracy
{ScanAcc = 90%, 80%, or 70%}
ScanAcc
«constraint»
Auto Speed : Scan Speed
{Speed = Weibull(a,b)}
Speed
CommTimingSpec
SurvTimingSpec
«constraint»
Surv Timing : HVT Iterations
{4 iterations of HVTs}
«constraint»
Comm Timing : Comm Iterations
{4 iterations of Comm Qs}
HVT
Q
«constraint»
HVT Timing : HVT Rate
{HVT created w/in boundary every 60 sec
HVT remains for 60 sec}
«constraint»
Q Timing : Q Rate
{Comm Q asked every 60 sec
Initial 30 sec delay}

Figure 15: System Parametric Diagram

The block definition diagram portrays the structural aspect of the Vigilant Spirit system, to include the operator, automation, and system software. Note that the human is included as an actor in the block definition diagram, albeit from a generalized viewpoint. Three performance requirements are highlighted in the requirements diagram upon which to focus. However, additional system requirements would realistically exist. The performance requirements are defined in terms of score, where there is a separate requirement each for the surveillance, communication, and total scores. Generally, the operator without automation can achieve a surveillance score of 600 points by finding the HVT in 23 seconds and continuing to follow without losing it, or by finding the HVT in less than 23 seconds with minimal lost-HVT time. Similar reasoning applies to the communication requirement, as well. The parametric diagram further defines the performance and specification requirements from the requirements diagram, as well as the automation's scanning settings from the block definition diagram.

It is unclear from studying either the SysML diagrams or Table 1 which automation setting will produce the best results, or if any of the settings would even meet the performance requirements. The existing system models are insufficient by themselves to evaluate these trades. Fortunately, IMPRINT can be used to perform a trade study to determine the effects of each of the automation options on overall system performance.

Perform Task Analysis

To inform our IMPRINT model, our task analysis of Vigilant Spirit examined the tasks required of both the system and operator throughout the simulated mission (without any automated search algorithms). The analysis showed two independent sets of activities performed by the system and operator for each of the surveillance and communication tasks. During each iteration of the surveillance task, the system spawns a HVT, for which the operator searches, indicates if found, and follows. Likewise, during each iteration of the communications task, the system asks a question, prompting calculations and answers from the operator. A SysML activity diagram was selected to represent the results of the task analysis, shown in Figure 16. Activity diagrams are well-suited for representing task analyses because of their ability to visually depict the actors, decisions, and task flows involved in activities at a high level.

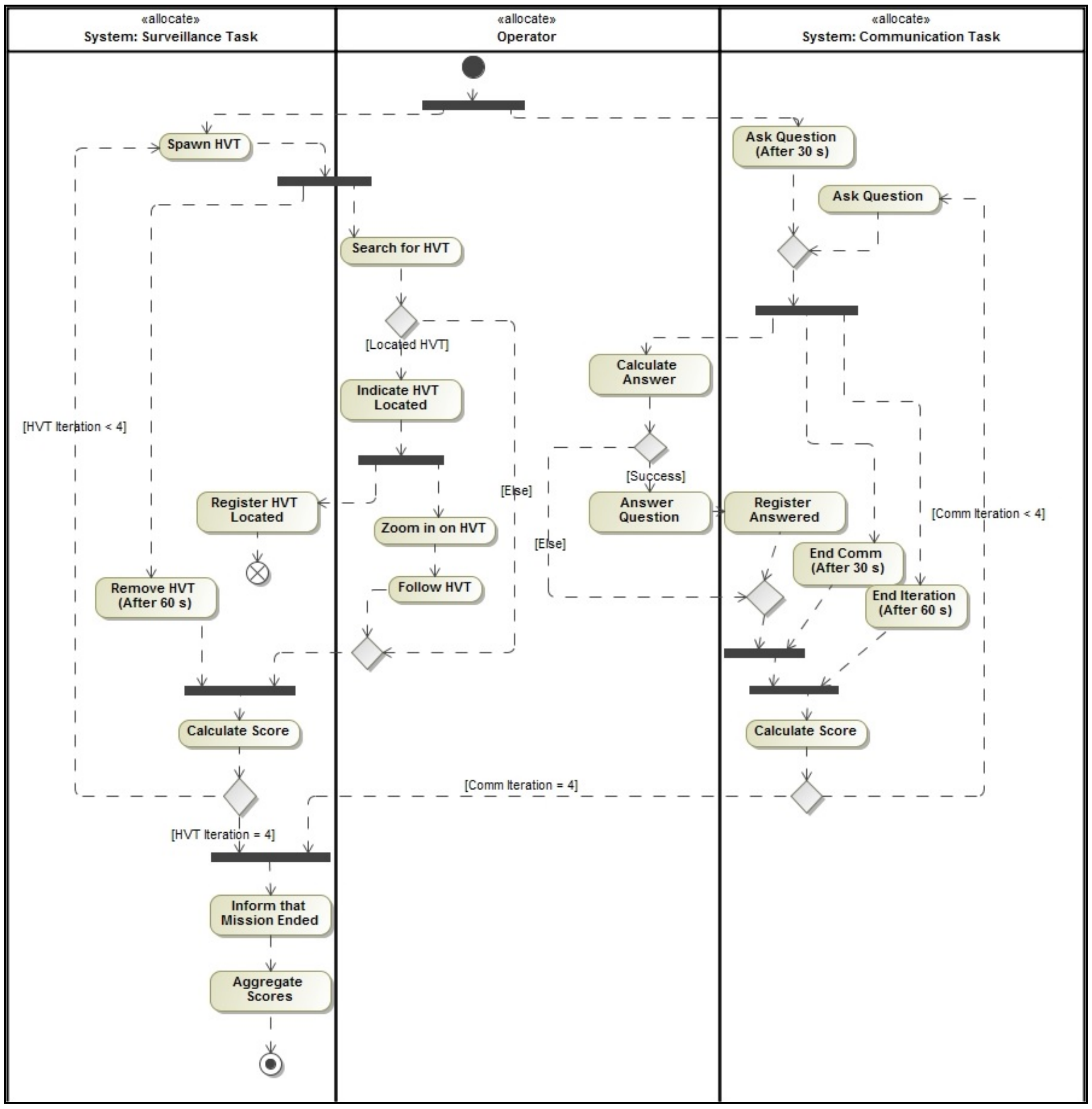

Figure 16: Activity Diagram of Vigilant Spirit Tasks

Create Baseline Model

Starting from the above activity diagram as a basis, a task network was built in IMPRINT to represent Vigilant Spirit. The baseline task network is shown in Figure 17.

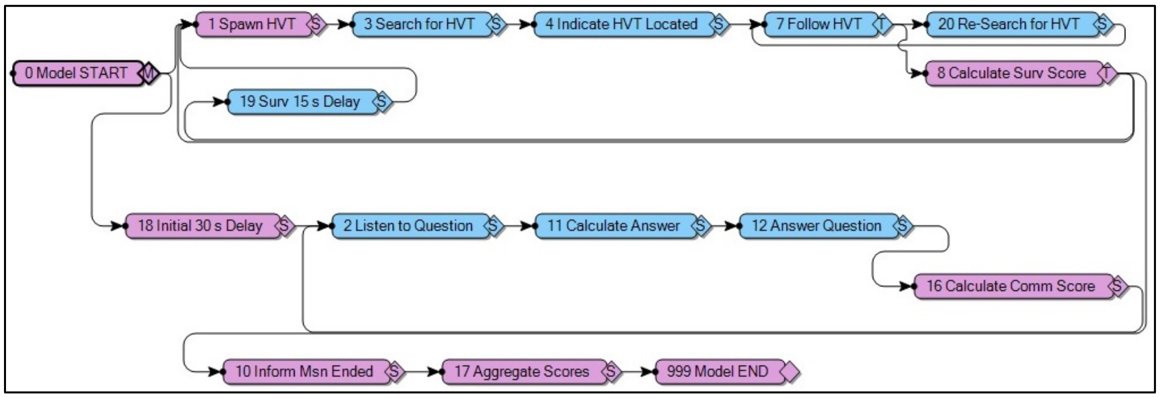

Figure 17: Baseline IMPRINT Task Network

There is a clear resemblance between the activity diagram and task network in Figure 16 and Figure 17, respectively. While building the IMPRINT model, the task flow was largely kept the same as in the activity diagram. Each task network node contains coding for task effects, timing, path logic, and workload demand. The system's tasks are shown in purple, and the human's tasks in blue. When the model starts, two separate task flows occur: the surveillance task in the top half of the network and the communication task in the bottom half of the network. At the conclusion of the mission, both flows converge to aggregate the operator's performance scores and end the model. For a detailed description of the baseline model, see Appendix B.

The baseline model was validated against the real-world Vigilant Spirit simulation with regard to two of the model's outputs: operator performance score and workload. An independent-samples t-test was conducted to compare the performance scores for the model and the human subjects experiment data collected by the 711th HPW. There was not a significant difference in the scores for the model (M=487.286, SD=141.088) and real-world (M=480.984, SD=152.858) conditions; t(75)=0.19246, p=0.84791. These results suggest that there is insufficient evidence that the baseline model differs from the real-world, thus resulting in a successful validation. A visual evaluation of the confidence intervals for the real-world and model data corroborates this result, as the two data sets almost completely overlap, shown in Figure 18.

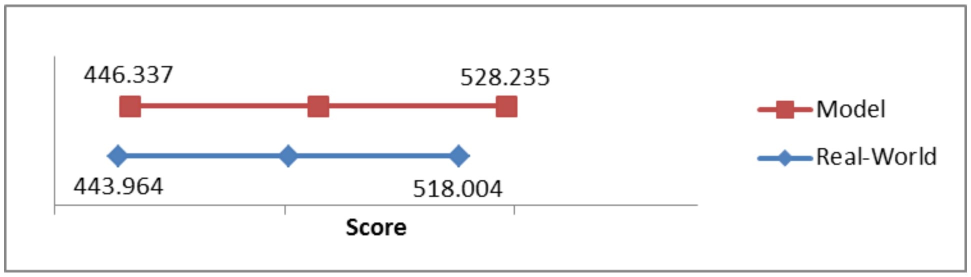

Figure 18: Confidence Intervals for Real-World and Baseline Model Score Data

Unlike the score data, the model's workload outputs could not be directly compared against the real-world. The human subjects in the 711th's study completed NASA-TLX (Hart & Staveland, 1988) forms upon completion of the experiment. By contrast, IMPRINT measures workload using the visual, auditory, cognitive, psychomotor (VACP) method (Bierbaum, Szabo, & Aldrich, 1989). These two differently formatted workload measurements prevented statistical comparison between the two data sets. Therefore, the baseline model's workload outputs were validated using peer and SME reviews. For a detailed description of how the baseline model was validated for both score and workload, see Appendix C.

Perform Trade Study

Once the baseline model was successfully validated, an alternative model incorporating the automation settings was built in order to conduct the trade study. The alternative model's task network is shown in Figure 19.

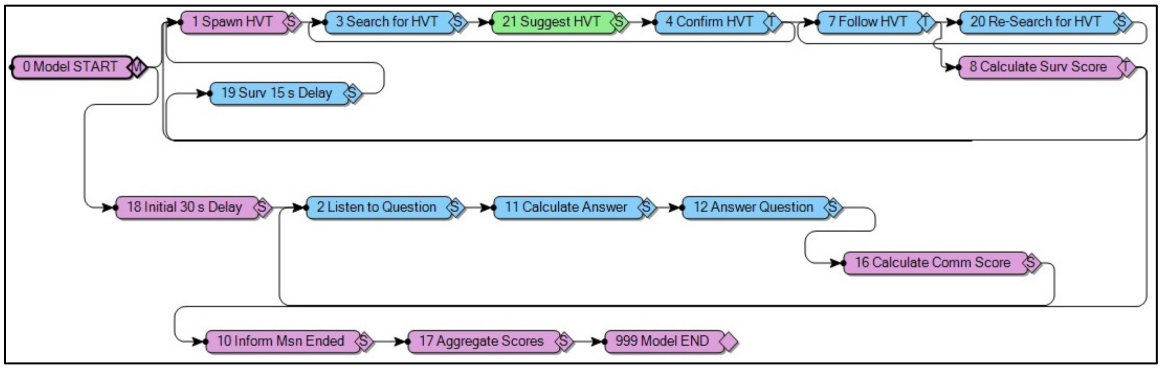

Figure 19: Alternative IMPRINT Task Network

The alternative IMPRINT model looks largely the same as the baseline, but includes some key differences. The "Search for HVT" task node incorporates both the automation and operator searching for the HVT. The "Suggest HVT" node is highlighted in green as a system-automated task, wherein the system automation suggests the HVT to the operator if it identifies the HVT first. The "Confirm HVT" node encompasses the operator either confirming the suggested HVT as correct or incorrect if the automation identified the HVT first, or self-identifying the HVT if the operator is faster than the automation. It should also be noted that if the system automation incorrectly identifies a

HVT, then both the operator and automation revert back to searching for the HVT. For a detailed description of the alternative model, see Appendix D.

The alternative model was run for each of the six algorithm setting trades shown in Table 1, with automation accuracy and speed parameters varying per trade. Each trade's performance score and workload data were collected and statistically evaluated against the baseline's. A one-way analysis of variance (ANOVA) was conducted to compare the effect of scan accuracy and speed on score in the baseline and each of the six trade conditions. There was a significant effect of scan accuracy and speed on score at the $p<0.05$ level for the baseline and six trade conditions [$F(6, 231) = 41.93$, $p = 0.000$]. Post hoc comparisons using Tukey's honest significant difference (HSD) test indicated that the mean scores for all six trades were significantly different than the baseline. However, it should be noted that although all six trades were statistically better than the baseline, they were not necessarily statistically different from each other. The mean, standard deviation, and relative groupings of scores for the baseline and six trades are shown in Table 2, where means that do not share a grouping letter are significantly different.

Table 2: Tukey's HSD Results for Baseline/Trades for Score

	Mean (M)	Standard Deviation (SD)	Grouping			
Baseline	487.3	141.1	A			
Trade 1	669.0	71.3		B		
Trade 2	678.6	71.4		B		
Trade 3	675.3	82.3		B		
Trade 4	714.2	95.3		B	C	
Trade 5	779.1	80.5			C	D
Trade 6	822.1	108.4				D

These results suggest that the automation accuracy and speed settings in all six trades significantly improved performance scores from the baseline. This can also be clearly observed in Figure 20, where the vertical red line indicates the original score requirement.

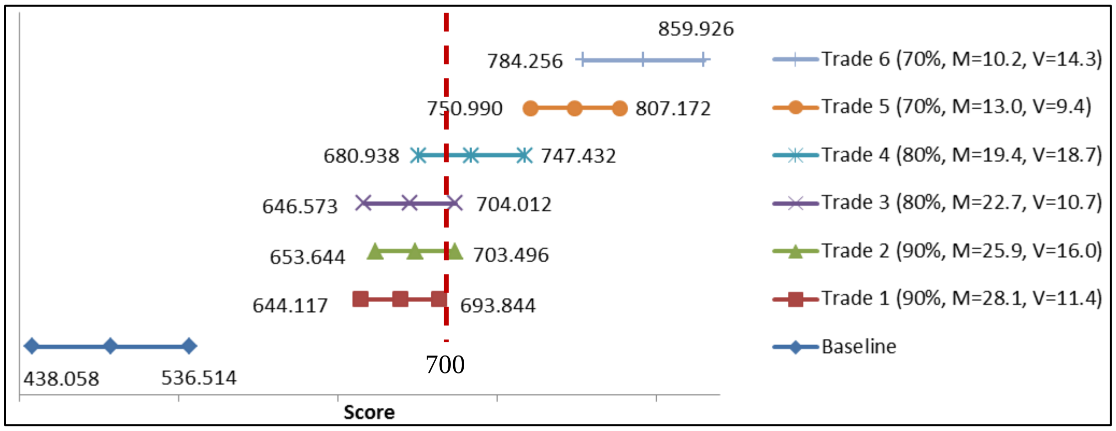

Figure 20: Confidence Intervals for Baseline/Tradeoff Models for Score

A one-way ANOVA was also conducted to compare the effect of scan accuracy and speed on workload in the baseline and each of the six trade conditions. There was a significant effect of scan accuracy and speed on workload at the p<0.05 level for the baseline and six trade conditions [F(6, 231) = 3.48, p = 0.003]. Post hoc comparisons using Tukey's HSD test indicated that the mean workload for Trades 5 and 6 were significantly different than the baseline. However, Trades 1-4 did not significantly differ from the baseline or each other. The mean, standard deviation, and relative groupings of workload for the baseline and six trades are shown in Table 3, where means that do not share a grouping letter are significantly different.

Table 3: Tukey's HSD Results for Baseline/Trades for Workload

	Mean (M)	Standard Deviation (SD)	Grouping	
Baseline	20.116	0.806	A	
Trade 1	19.645	0.598	A	B
Trade 2	19.693	0.668	A	B
Trade 3	19.827	0.777	A	B
Trade 4	19.751	0.833	A	B
Trade 5	19.483	0.704		B
Trade 6	19.425	0.604		B

These results suggest that the automation accuracy and speed settings in Trades 5 and 6 were the only two alternatives to significantly improve workload from the baseline. Figure 21 shows that even though workload is decreased for all alternatives, the data sets from Trades 5 and 6 are the only ones that do not overlap the baseline. For a detailed description of the alternative model's score and workload output analysis, see Appendix E.

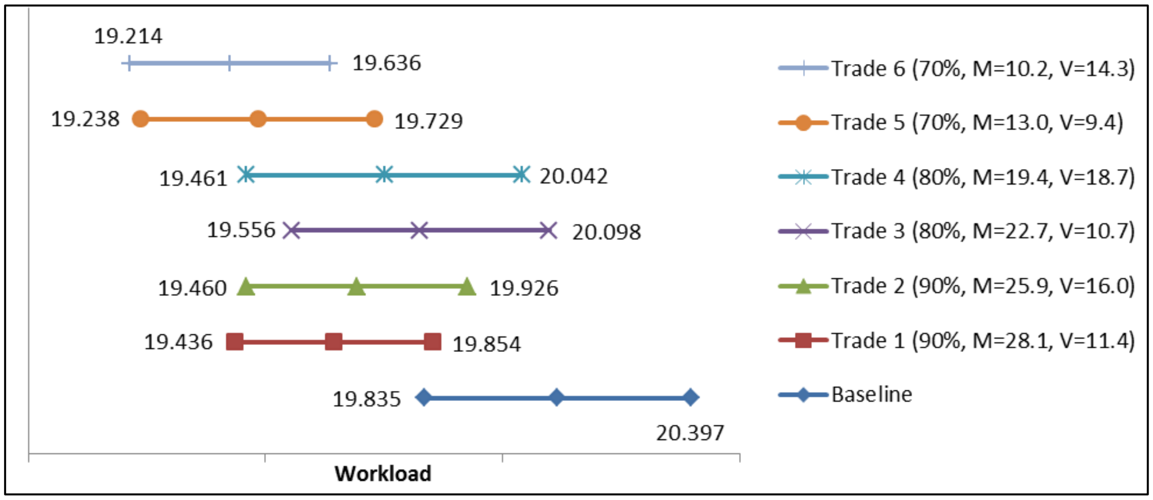

Figure 21: Confidence Intervals for Baseline/Tradeoff Models for Workload

From analysis of the alternative model's varying score and workload outputs, Trades 5 and 6 offer the best automation settings with which to aid the operator and improve performance. These two trades are when the automation has a higher scanning speed, but at the expense of having only 70% accuracy. Therefore, upon performing the tradeoff analysis, it can be concluded that Vigilant Spirit's system automation speed is more important than its accuracy for achieving increased performance score. Surprisingly, the best scores are produced when the automation has its highest speed but lowest accuracy. This observation is not something we could have predicted without running the human-performance analysis in IMPRINT. However, it should also be noted that even the lowest scores with automation are still higher than the human working alone, showing the general advantage of this particular automation implementation for the improvement of system performance, regardless of settings. Because Trades 5 and 6 are statistically the same, the decision of which to implement may be based on other factors, such as cost or complexity. For the purposes of this example, Trade 6 was assumed to be the preferred automation setting to implement based on these secondary factors.

Integrate Results into System Diagrams

The results of the trade study were then integrated back into the existing SysML diagrams of Vigilant Spirit. The idea of the operator and automation performing as a single, cohesive team was formed by the human focus of the trade study. This concept was incorporated into the integrated diagrams. The original requirements diagram shown previously in Figure 14 stated requirements of achieving a surveillance score of 600, communication score of 180, and a total score of 700. By conducting the trade study, we see that the surveillance and total score requirements can only be met with the system automation aiding the human. However, the addition of the automation has no impact on the communication score, and we observed that even with the human-automation team, the communication requirement cannot be achieved. Therefore, a decision would need to be made on whether to accept the risk, modify the requirement, or adjust the system's design.

Additionally, the preferred automation settings identified by the trade study may be used to create automation performance requirements. Adding these new requirements to the diagram will help ensure the automation settings are maintained. Note that, in addition to automation, new human requirements could also be created with regard to accuracy and speed. These human performance requirements would translate into training requirements to ensure system operators can perform at the level necessary to achieve the score requirements. An updated requirements diagram is shown in Figure 22, with the automation requirements added and met and unmet requirements outlined in green and red, respectively.

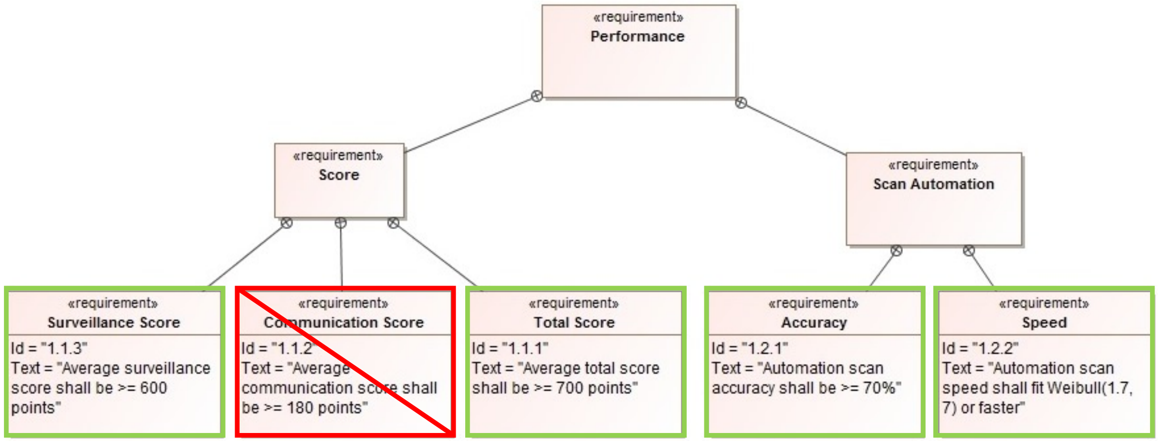
«requirement»
Performance
«requirement»
Score
«requirement»
Scan Automation
«requirement»
Surveillance Score
Id = "1.1.3"
Text = "Average surveillance score shall be >= 600 points"
«requirement»
Communication Score
Id = "1.1.2"
Text = "Average communication score shall be >= 180 points"
«requirement»
Total Score
Id = "1.1.1"
Text = "Average total score shall be >= 700 points"
«requirement»
Accuracy
Id = "1.2.1"
Text = "Automation scan accuracy shall be >= 70%"
«requirement»
Speed
Id = "1.2.2"
Text = "Automation scan speed shall fit Weibull(1.7, 7) or faster"

Figure 22: Integrated Requirements Diagram

Additional insight can also be gained by examining the process itself leading to the trade study. By interpreting the 711th HPW's experimental data, building the IMPRINT models, and performing the trade study, limitations of the human were uncovered. For example, we learned that the operator has an initial rate of losing the HVT of 43.8%, but that rate decreases to 34% if the HVT has already been lost in that iteration. Additionally, the operator has a very high rate (93.8%) of correctly answering the communication question. Concerning the operator and automation working together to perform the mission, the team had a combined HVT identification accuracy of 74%. Previously unaccounted in the system model, these human constraints allow for a more accurate picture of the system's constraints. This information was used to update the system's parametric diagram. The integrated parametric diagram in Figure 23 shows these new constraints highlighted in green.

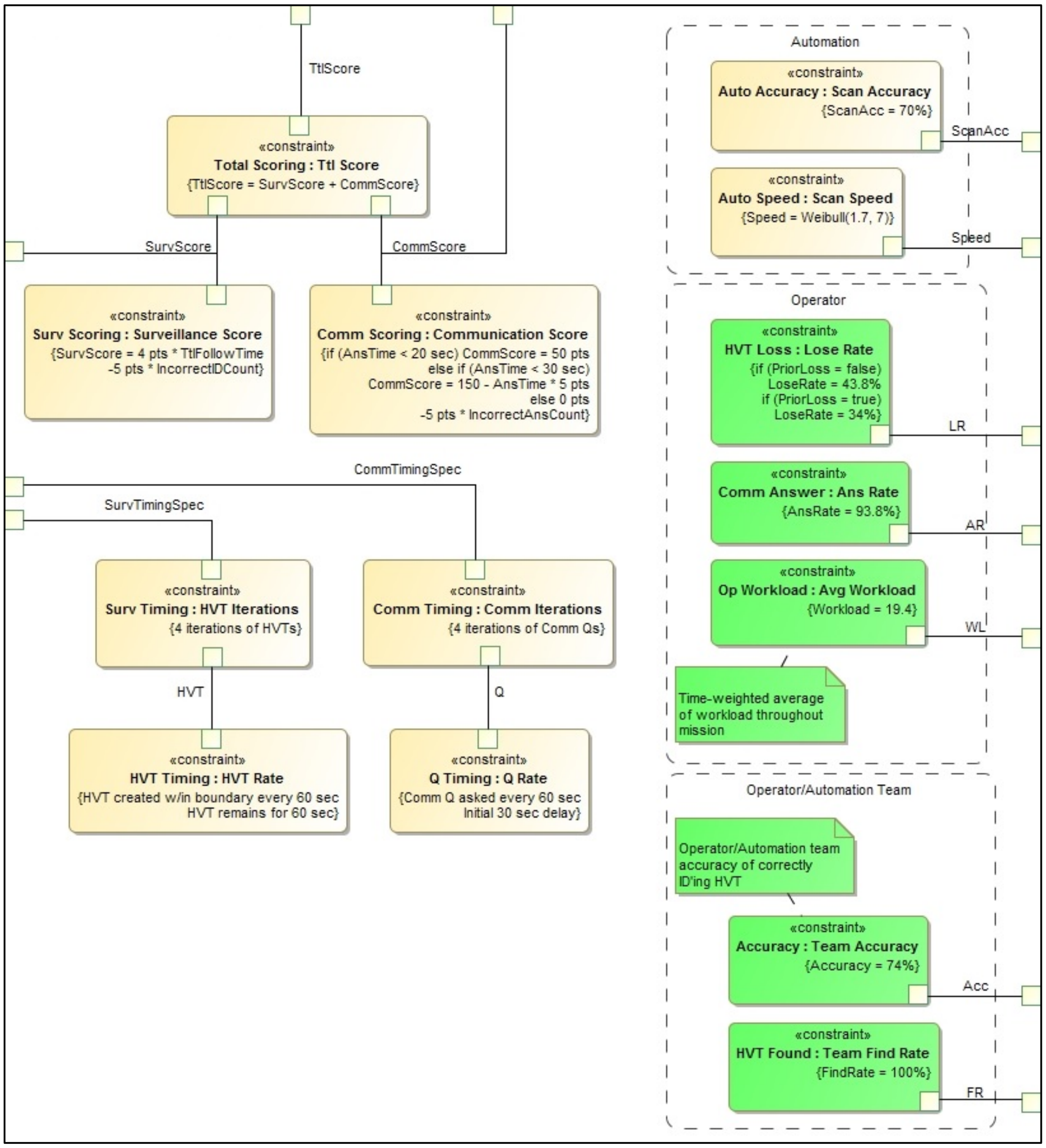
TtlScore
«constraint» Total Scoring : Ttl Score {TtlScore = SurvScore + CommScore}
SurvScore
CommScore
«constraint» Surv Scoring : Surveillance Score {SurvScore = 4 pts * TtlFollowTime -5 pts * IncorrectIDCount}
«constraint» Comm Scoring : Communication Score {if (AnsTime < 20 sec) CommScore = 50 pts else if (AnsTime < 30 sec) CommScore = 150 - AnsTime * 5 pts else 0 pts -5 pts * IncorrectAnsCount}
CommTimingSpec
SurvTimingSpec
«constraint» Surv Timing : HVT Iterations {4 iterations of HVTs}
«constraint» Comm Timing : Comm Iterations {4 iterations of Comm Qs}
HVT
Q
«constraint» HVT Timing : HVT Rate {HVT created w/in boundary every 60 sec HVT remains for 60 sec}
«constraint» Q Timing : Q Rate {Comm Q asked every 60 sec Initial 30 sec delay}
Automation
«constraint» Auto Accuracy : Scan Accuracy {ScanAcc = 70%}
ScanAcc
«constraint» Auto Speed : Scan Speed {Speed = Weibull(1.7, 7)}
Speed
Operator
«constraint» HVT Loss : Lose Rate {if (PriorLoss = false) LoseRate = 43.8% if (PriorLoss = true) LoseRate = 34%}
LR
«constraint» Comm Answer : Ans Rate {AnsRate = 93.8%}
AR
«constraint» Op Workload : Avg Workload {Workload = 19.4}
WL
Time-weighted average of workload throughout mission
Operator/Automation Team
Operator/Automation team accuracy of correctly ID'ing HVT
«constraint» Accuracy : Team Accuracy {Accuracy = 74%}
Acc
«constraint» HVT Found : Team Find Rate {FindRate = 100%}
FR

Figure 23: Integrated Parametric Diagram

The constraints uncovered within the parametric diagram also feed into the block definition diagram. Whereas the human was first portrayed in the block definition diagram at a high level, we can now further define the human to include constraint properties that were unknown prior to the analysis. The updated block definition diagram shown in Figure 24 includes both the human's constraints and the human-automation team's constraints.

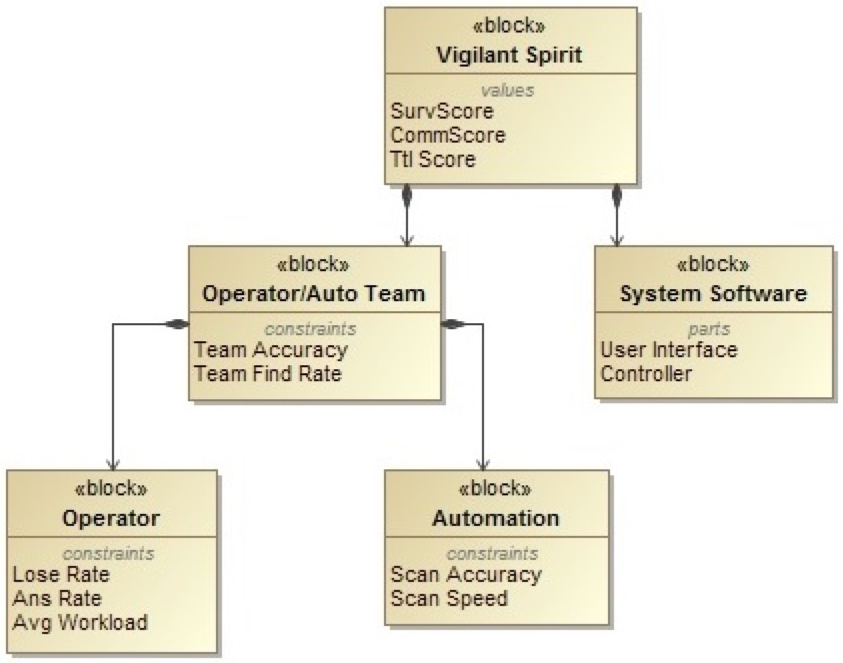

Figure 24: Integrated Block Definition Diagram

By performing this tradeoff analysis, we were able to gain insight into the human's capabilities with regard to Vigilant Spirit. By then further integrating the analysis' results back into existing system models, we saw how these insights affect the overall system. The results of the tradeoff analysis and the observations gained by performing the process of the analysis were used to update the assumptions and limitations placed on the human via the system's block definition, requirements, and parametric diagrams. In this manner, we were able to gain insight into the Vigilant Spirit system's limitations and constraints, verify requirements and even add new system requirements.

Discussion and Conclusions

As with the specific Vigilant Spirit scenario, the benefits gained from the process of integrating human trade study results into system models may also be applied to systems in other contexts. Any system design that does not properly account for the human's involvement may make unrealistic assumptions about the human, overestimating the human's capabilities and thus overestimating the system's capabilities. Therefore, it is essential for human considerations to be integrated into system design and development.

Though this study highlighted the benefit of using IMPRINT to inform system models, the results of the IMPRINT analysis were translated manually back into the SysML diagrams. Future work will focus upon developing a means to automate this information transfer between IMPRINT and MBSE software. Being able to automatically update relevant diagrams of the system with IMPRINT's outputs would reduce potential interpretation errors, further enable human factors engineers to effectively communicate human considerations to systems engineers, and improve efficiency while exploring alternate system designs. Additionally, although this method focused specifically on integrating IMPRINT analyses with SysML, the use of other HSI and SE tools could yield similar or additional benefits. Likewise, integration efforts could also be expanded across the rest of the nine HSI domains besides just human factors engineering.

Using human performance- and workload-modeling tools such as IMPRINT to perform tradeoff analyses, human factors engineers can attain realistic data about the human subsystem. These data may then be integrated back into existing SysML diagrams created by the systems engineers. In so doing, additional insights into the whole system can be gained that would not be possible if human factors engineers and systems engineers worked independently. Thus, the human is effectively incorporated into the system's design and the total system performance may be predicted, allowing for an improved system design.

Chapter Summary

This chapter addressed the final two investigative questions, in addition to revisiting the first two questions. Because this article presented a different method of integration than the previous article, it was necessary to re-address the first two investigative questions. The following is a discussion of each question:

1. What information should be captured in human-centric and system-centric models to enable effective integration?

The purpose of this article was to integrate human- and system-centric models by analyzing how the human's performance affects that of the system. As such, SysML diagrams relating to performance, such as block definition, requirements, and parametric diagrams, were chosen for this integration effort. Requirements diagrams capture system requirements, many of which are measured and evaluated based on performance. Parametric diagrams capture the system's constraints, which can also be performance-related. The nature of these diagrams allows for the inclusion of human performance requirements and constraints. The block definition diagram was used to clarify the requirements and parametric diagrams by defining the structure of the Vigilant Spirit system.

This article recognized that the SE and HSI disciplines have separate and unique toolsets, and capitalized on the benefits of maintaining both toolsets while still having cross-communication. The human-centric models were defined as the IMPRINT models of Vigilant Spirit, and the system-centric models defined as the initial system-focused SysML diagrams of Vigilant Spirit. Because the article made use of traditionally-purposed models to perform integration, the information needed to build the IMPRINT and SysML models was no different than would be required if they were built separately by engineers from their respective disciplines. The article assumed the system-focused SysML models were already built, and instead focused primarily on building the IMPRINT model. A task analysis of the system is necessary before creating the IMPRINT model, as the analysis will uncover the mission task flow and interactions between the system and human. Because of this study's performance focus, Vigilant Spirit's scoring algorithm was coded into the IMPRINT model in order to measure the human's simulated performance.

2. What considerations and decisions must be made when integrating between human-centric and system-centric models?

Because requirements and parametric diagrams tend to be more data-focused, the integration of the results from the IMPRINT analysis is straightforward: Once the IMPRINT-enabled trade study is complete and the preferred trade is selected for implementation, the initial SysML models of the system can be updated to include this new data. A compromise in information is unnecessary when using a modeling methodology that keeps both disciplines' toolsets and allows communication between the two. Therefore, updates to the SysML diagrams in this study could be made without compromise because relevant human information could more directly flow into them. However, note that updating the block definition diagram is more subjective and based on any enlightening information gained while performing the IMPRINT analysis.

3. What information can currently be passed from IMPRINT to SysML models?

Because of the versatility and range of available diagrams within SysML, virtually all output data gained from performing IMPRINT analyses can be passed to SysML models. IMPRINT has the ability to output many types of information in report or chart form, to include mission performance, task sequence, and human workload data. Additionally, myriad other types of data can be output through the use of customizable "snapshots," which capture specific variable data. This data can all be readily passed to SysML models in order to validate system requirements or add/update existing system constraints. However, higher-level information such as task flows, allocations, or relationships between the system and human cannot as readily be passed. This is because such information is more abstract and does not afford a direct data transfer to SysML models, as is the case for hierarchical system diagrams. Thus, more effort must be applied to synthesize, translate, and pass this information to SysML models.

4. What information do SysML models need from IMPRINT to effectively inform tradeoff analyses?

SysML diagrams are uniquely-purposed to convey specific types of information; thus, each diagram uses information from IMPRINT in a different way in order to be effective. Parametric diagrams are primarily data-focused, so they need the data gained from IMPRINT’s output reports, discussed in the previous question. Additionally, information gained through the task analysis as part of the integration process itself may be helpful, such as user accuracy and success/failure rates. This data can be used to either add or update existing constraint blocks.

The task analysis can also be used to inform block definition diagrams, as it may uncover parts of the system that previously were not considered, such as the idea of the human-automation team from Vigilant Spirit. Since the information from the task analysis is used to build the IMPRINT model, IMPRINT should be capable of conveying any new human-system interactions so that the block definition diagram can capture those relationships.

Similar to parametric diagrams, requirements diagrams can use IMPRINT’s output data to verify if requirements are able to be met or to create new requirements. This is easier with performance-related requirements, as they are usually quantifiable. The IMPRINT model should output meaningful data that can be used to verify requirements. For example, a variable “snapshot” was created in Vigilant Spirit’s model to specifically capture the operator’s score data to be compared with the defined requirements. Note that some information may be useful to other SysML diagrams in different ways. The score data used to verify requirements could have also been used to inform a parametric constraint.

These concepts may also be applied to other SysML diagrams besides just the ones discussed. With each SysML diagram uniquely utilizing IMPRINT's output and task analysis information, systems engineers can be better equipped with the necessary information to make effective system trades and informed design decisions.

IV. Conclusions and Recommendations

Chapter Overview

This chapter begins by providing an overview of this research, to include the current problem and motivation behind the research, previous efforts to address the problem, research gap, and objectives of the research. The chapter then discusses how the four investigative questions were answered by the research, provides recommendations for future work in this area, and concludes by summarizing the significance of the research.

Research Overview

Systems engineering (SE) is an important part of the lifecycle management of systems. As a part of the SE process, it is imperative to consider the human as an integral component of the system. The process by which the human can be effectively accounted during system development is called Human Systems Integration (HSI) (U.S. Air Force, 2010). Failure to consider HSI during the SE process can lead to serious consequences, such as Global Hawk's mission degradation, the Patriot air and missile defense system's fratricide incidents, and Three Mile Island's partial nuclear meltdown (Hopcroft, Burchat, & Vince, 2006; Hawley, 2011; U.S. Nuclear Regulatory Commission, 2014).

The Department of Defense (DoD) recognizes the necessity of HSI during system development, and has even mandated its implementation during the system's lifecycle (Department of Defense, 2015). However, there is currently a failure to integrate the HSI process into the SE process at the correct level of detail and developmental phase (Handley & Knapp, 2014; Orellana & Madni, 2014; Hardman & Colombi, 2012), resulting in a system with higher lifecycle costs and less compatibility (Mitchell, Agan, & Samms, 2011; Hardman & Colombi, 2012). There have been several efforts to integrate the HSI and SE processes, but few have focused on development phases past concept development. Additionally, few efforts have focused at the tools level of scope, and those at the tools level lack an integration plan to re-inform system models. Therefore, this research focused on integration during the preliminary or detailed system design phases, and at their respective tools levels of scope.

This research had three objectives that were met. The first two objectives were to determine first if it is possible to integrate the HSI process into SE practices, then how to effectively do so. These objectives were both completed by conducting an extensive review of literature and identifying the research gap. The final objective was to demonstrate the value of integration, which was evinced by the methods prescribed in Chapters II and III. These objectives were guided by the following research question: ***How can HSI models be integrated with SE models in order to perform system design tradeoffs?***

Investigative Questions

To aid in answering the research question and meeting the research objectives, four investigative questions were formulated. These questions were addressed by the articles in Chapters II and III using static and dynamic methods, respectively.

The first question asked **what information should be captured in human-centric and system-centric models to enable effective integration**. This information is uncovered by analyzing the places where relevant interactions occur between the human and system, with regard to the purpose of integration. If the purpose is system interface design, as was the case for the static method, then those interactions occur at the interface level. Therefore, it is necessary to know the different system and human components and how they communicate. As such, behavioral SysML diagrams like activity and sequence diagrams are conducive to capturing this information. If the purpose is instead performance-related, as was the case for the dynamic method, then relevant performance interactions and task flows between the human and system should be captured. Requirements diagrams, along with structural SysML diagrams such as parametric and block definition diagrams, are conducive to the integration of human information when focusing on the system's and human's performance.

The second question asked **what considerations and decisions must be made when integrating between human-centric and system-centric models**. When integrating within one discipline's toolset, a balance must be achieved between keeping relevant and helpful information in the integrated model and eliminating irrelevant or superfluous information. This was evinced during the static method, when SysML was used to create both human- and system-centric models. However, if each disciplines' toolsets are maintained during integration, this compromise is unnecessary because the human data may directly feed into the other toolset. For example, during the dynamic method, the use of SysML requirements and parametric diagrams allowed for direct input of the information gained from the IMPRINT analysis.

The third question asked **what information can currently be passed from IMPRINT to SysML models**. Relating to the second question, because of the versatility of SysML diagrams, all output data gained from IMPRINT analysis can be passed to the

SysML models by updating existing diagram components or creating new ones. However, more abstract information about task flows and general relationships cannot be readily passed without manual synthesis and translation.

The final question hones the previous by asking, although most information can be passed, **what information do SysML models need from IMPRINT to effectively inform tradeoff analyses**. Of the SysML diagrams discussed, the most beneficial data comes from IMPRINT's output reports and variable "snapshot" data. This information is needed by parametric diagrams to update constraints or add new ones and by requirements diagrams to verify performance-related requirements. Additionally, the less data-focused and more design-focused information gained from task analyses may also be used by parametric diagrams, and is needed to update system-human relationships depicted in block definition diagrams.

Recommendations for Future Research

While this work successfully answered the research question by providing methods of integrating the HSI process into SE, there are still areas of this research that can be expanded upon and further improved.

Automate the IMPRINT-SysML Data Transfer

Although the approach presented in Chapter III offered an integration plan for re-informing previously system-centered SysML diagrams following IMPRINT analysis, this currently involves a manual transfer of information. The hazard of a manual transfer is that it opens the opportunity for translation mistakes and interpretation errors to occur. Such errors could be mitigated if system diagrams were able to be automatically updated with IMPRINT's outputs. An automatic data transfer would further enable effective communication between human factors engineers and systems engineers, and ultimately ease analysis for systems engineers.

If this transfer were automated, what data should be passed to the SysML software? Ideally, variables within the IMPRINT model would store the necessary output data following analysis. These variables would then be passed to the SysML requirements and parametric diagrams and would either update existing SysML diagram blocks or create new blocks with the new information gained.

Similarly, automating the IMPRINT tradeoff analysis would also provide benefits. If the process to run the tradeoff analysis in IMPRINT could be automated so that the preferred alternative is chosen and used to update system models, errors would be further reduced and efficiency increased. Additionally, the data from the unsuccessful trades could be documented and stored for later reference, as needed.

Translate Sequence Diagrams into IMPRINT

The sequence diagrams presented in Chapter II could also be used to inform IMPRINT models. For example, the diagrams' different user interface (UI) lifelines could be translated into corresponding keyboard, mouse, monitor, and headset interfaces in IMPRINT.

Additionally, greater definition between the operator-UI interaction in the sequence diagram could provide further benefits. While messages that pass between the system subcomponents are highly defined, messages between the UI and operator generally are more abstract. Therefore, increasing the definition of this interaction could provide added value to the integration process. One method of accomplishing such definition could be through visual, auditory, cognitive, and psychomotor (VACP) measures, as was used in Chapter II. Using VACP for this approach allows for a direct translation of the sequence diagrams' VACP components into corresponding workload assignments in IMPRINT.

Generate Human Training Requirements

The trade study conducted in Chapter III allowed for the creation of new system requirements, baselining the automation's required accuracy and speed in order to meet score requirements. Similar requirements could also be generated for the human. By conducting human performance modeling, traits about the human's accuracy and speed are also obtained. For example, the operator has a percentage of time and a speed at which he or she finds the target. Using this data, training requirements could be created as a similar baseline for the operator.

Apply Integration Methods in Different Contexts

The integration approaches presented in Chapters II and III offer a way of integrating SE and HSI processes in a specific context; however, this context could be expanded in several areas. The first area is in the example Vigilant Spirit case scenario. While this case scenario was meant to be broad in order to expand its applicability, demonstrating these same methods of integration in a different scenario could uncover unique benefits or additional challenges.

The next area to expand is by integrating different MBSE tools besides SysML and different HSI methods and tools besides user-centered design and IMPRINT. The use of other tools could yield similar or additional benefits. Even within SysML, other diagrams besides those used in this research could also provide integration benefits.

Similarly, while this research focused on the human factors engineering domain, further research could address integration of the remaining eight HSI domains.

Additional research could also focus on implementing human considerations into system design in other lifecycle phases. Past efforts integrated primarily in the concept phase, while this research integrated further in the preliminary and detailed design phases. Future work could address integration at even later lifecycle phases.

Significance of Research

This research showed that integration of the HSI process into the SE process is achievable through various methods, including both static and dynamic modeling. The static method is a unique approach that uses user-centered design to break out both the system and user into their different components in order to determine the functional allocation of tasks. This approach provides the benefit of identifying potential conflicts if any of these subsystems are asked to perform simultaneous tasks. Therefore, this static approach allows systems engineers to see whether it is possible for the human to perform some of these tasks, and the potential for conflicts.

The dynamic method approaches integration from a performance standpoint. By utilizing the human performance simulation tool IMPRINT, this method provides human factors professionals with a mechanism for conveying important human considerations to systems engineers, and does so by using SysML to convey that information. The benefit of this approach is the added ability to combine both system and human considerations into a single performance measure, allowing system tradeoff analyses to be more effectively performed.

By applying either of these integration methods early in the system’s lifecycle, systems engineers can recognize that humans are indeed critical components of the system, and gain additional ways to effectively account for the human during system design and development. Thus, system design is improved through the integration of human systems and systems engineering models.

Appendix A – Literature Review

Overview

Systems engineering (SE) approaches development from a holistic perspective, dividing the system into its components. As one of these components, the human is an important part of nearly every system. However, currently Human Systems Integration (HSI) is not being implemented during the SE process, resulting in higher lifecycle costs and decreased user compatibility (Mitchell, Agan, & Samms, 2011; Hardman & Colombi, 2012). To address this problem, the HSI process needs to be integrated with SE.

This appendix begins by providing a background on SE and HSI. To enhance future integration efforts, it is helpful to know what tradeoffs are considered during system design. Thus, the appendix reviews the types of Model-Based Systems Engineering (MBSE)-related tradeoffs and tradeoffs using the Improved Performance Research Integration Tool (IMPRINT), and how such analysis methods are being performed. It then discusses various integration methods at the process, methods, and tools levels of scope, and continues at the tools level by discussing integration efforts using IMPRINT modeling. The appendix concludes by highlighting the research gap in integration efforts thus far.

Systems Engineering

SE is a process that has become an increasingly important part of the overall lifecycle management of Department of Defense (DoD) systems, to the point of becoming an institutionalized disciplinary approach to the development of defense acquisition programs (Department of Defense, 2015). The International Council on Systems Engineering (INCOSE) (2015) defines a system as "an integrated set of elements, subsystems, or assemblies that accomplish a defined objective," whereas these elements could not otherwise produce the same results by themselves. Elements may include hardware, software, people, information, and facilities. SE offers a holistic approach to developing these systems by integrating the many disciplines involved and thereby accounting for factors such as requirements, cost, and schedule early in the system's lifecycle and continuing through development, operation, and eventually disposal. To support this consideration throughout the lifecycle, the SE process is composed of 14 technical sub-processes ranging from stakeholder requirements definition to system disposal (International Council on Systems Engineering, 2015).

There are many different methods of practicing SE. Model-based systems engineering (MBSE) is an emerging method with which to perform SE. Whereas the traditional document-based method is driven by the development of a set of disjointed documents, each separately detailing system-related information such as requirements or design specifications, MBSE allows for the development of the same information through a series of interrelated models that together form a complete system model (Friedenthal, Moore, & Steiner, 2014). The MBSE method results in improved team communication, increased quality of the system's specification and design, and the ability to reuse the model throughout the system's lifecycle (Friedenthal, Moore, & Steiner, 2014).

If MBSE is a method of practicing SE, then the Systems Modeling Language (SysML) is a tool with which to implement MBSE. There are several graphical modeling languages available for SE applications, SysML being one of them (Delligatti, 2014).

SysML provides a means of communicating system information via a selection of uniquely-purposed diagrams. These diagrams allow the modeler to represent requirements as well as behavioral and structural aspects of the system, as shown in Figure 25 (Delligatti, 2014).

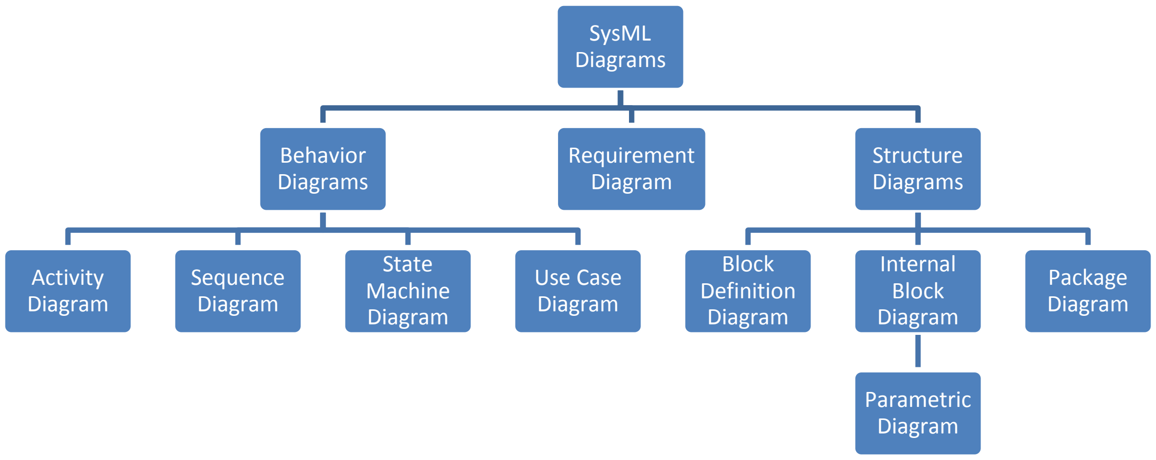

Figure 25: SysML Diagram Taxonomy – adapted from (Delligatti, 2014)

Human Systems Integration

The human should also be a critical consideration during system development and the SE process in general. The Air Force HSI Handbook defines HSI as the "process by which to design and develop systems that effectively and affordably integrate human capabilities and limitations" (U.S. Air Force, 2010). This approach is necessary because humans who operate, maintain, and support the system are an integral part of the total system itself (Department of Defense, 2013). HSI is divided into nine domains: manpower, personnel, training, human factors engineering, environment, safety, occupational health, survivability, and habitability (U.S. Air Force, 2010).

Human factors engineering is the primary HSI domain with which to focus on integration (U.S. Air Force, 2010). Human factors engineering, also called ergonomics, is the study of the interactions between the human and system, and the efficiency of those interactions (International Ergonomics Association, 2016).

There are various methods of practicing human factors engineering. One method is through user-centered design (UCD). UCD is the idea of designing a system with a focus primarily on the user and involving the user in the design process. By focusing on the user's goals, preferences, tools needed, and tasks performed, the goal is that the end-system will be best suited for what the user needs (Norman & Draper, 1986).

Another method of practicing human factors engineering is through human performance modeling, where the human is modeled mainly via simulation (Allender, 2000). One such modeling tool is called the Improved Performance Research Integration Tool (IMPRINT). Developed by the Army Research Laboratory to support HSI efforts, IMPRINT is used to analyze the interaction between the system and humans. IMPRINT allows the analyst to first represent a mission in terms of a series of functions and tasks performed by both the system and human, then run a discrete event simulation (DES) of the system and human accomplishing the mission. In this manner, the analyst can observe effects on performance and cognitive workload (Mitchell, Agan, & Samms, 2011).

MBSE Tradeoffs and Methods

Systems engineers, now using MBSE practices, perform tradeoff analyses involving several factors such as cost, mission effectiveness, size (weight and volume), performance, and the "-ilities." Cost is a common factor among system tradeoffs (Crane & Brownlow, 2015; Do, Cook, & Lay, 2014; Russell, 2012). It is often traded between factors influencing mission effectiveness such as supportability (Russell, 2012) and performance (Crane & Brownlow, 2015). System designers commonly have to make decisions regarding increasing the performance of a system at the expense of also increasing system cost. A balance must be reached between the level of performance desired by the system stakeholders and an acceptable total cost.

Mission effectiveness is a broad factor that includes system tradeoffs such as mobility, survivability, supportability, and performance (Cloutier, Sauser, Bone, & Taylor, 2015; Crane & Brownlow, 2015; Kaslow, Soremekun, Kim, & Spangelo, 2014; Russell, 2012). These individual factors may be traded between themselves or other tradeoff factors such as equipment weight and volume (Cloutier, Sauser, Bone, & Taylor, 2015; Crane & Brownlow, 2015; Kaslow, Soremekun, Kim, & Spangelo, 2014). For example, decreasing a system's weight and volume may increase mobility, which affects the system's survivability and overall mission effectiveness (Cloutier, Sauser, Bone, & Taylor, 2015). However, increasing the system's volume may allow for more armor or ammunition, thus increasing the system's lethality and again affecting survivability. Equipment weight and volume may also be traded with cost, such as within the context of a satellite constellation when considering orbital altitude and constellation size (Crane & Brownlow, 2015). At a higher altitude, fewer satellites are needed to cover an area, but at the cost of needing better-quality sensors. Conversely, spacecraft are cheaper at lower altitudes due to size and weight reductions, but more are needed to cover the same area.

When faced with similar alternatives, adding other tradeoff factors may aid in deciding on a solution. For instance, in satellite constellation design, other factors that could be considered are disaggregation, resiliency, and lower costs (Thompson, Colombi, Black, & Ayres, 2015).

MBSE tradeoff analyses are performed using both qualitative and quantitative methods. The primary qualitative method used to perform these analyses involves visualization of the system via SysML diagrams (Cloutier, Sauser, Bone, & Taylor, 2015; Russell, 2012). These SysML diagrams are used to analyze tradeoffs and enable system design decisions. Activity diagrams and use case diagrams provide the MBSE practitioner a way to graphically highlight dependencies between components within the system.

Quantitative methods mainly involve the use of simulations (Crane & Brownlow, 2015; Kaslow, Soremekun, Kim, & Spangelo, 2014). In these methods, MBSE parametric diagrams are commonly created to establish relationships between the system's requirements and design constraints, which then feed into simulation models. Based on the parameter inputs, the modeler can see the outputs' impact on mission performance and determine if requirements are being met. Aside from simulations, quantitative analyses may also be uniquely developed to suit the system (Do, Cook, & Lay, 2014). For example, Do et al. (2014) self-developed a tradeoff analysis method by first assigning a series of weight and value functions to the system tradeoffs, then evaluating those functions to quantitatively determine a system design solution.

IMPRINT Tradeoffs and Methods

One tool human factors engineers use is IMPRINT. Using this tool, human factors engineers perform tradeoff analyses involving several factors, which include manning, performance, workload, equipment design, and task allocation. These factors are different than those relating to MBSE because they focus specifically on the human instead of the broader system. However, they still indirectly relate to and affect some MBSE factors, such as usability and system performance. Manning is a key factor in many human-related system tradeoff studies (Allender, 2000; Mitchell D. K., Samms, Henthorn, & Wojciechowski, 2003; Mitchell, Samms, & Wojcik, 2006; Mitchell D. K., 2008). Even the U.S. Navy-developed predecessor to IMPRINT, called HARDMAN (Hardware vs. Manpower), was created with the intention of analyzing tradeoffs between hardware and manpower (Dickason, Sargent, & Bagnall, 2009).

Many studies examine the impact of a reduction in manning on the other tradeoff factors mentioned (Allender, 2000; Mitchell D. K., Samms, Henthorn, & Wojciechowski, 2003; Mitchell, Samms, & Wojcik, 2006). For example, Allender (2000) describes a trade study in which the manning on a U.S. Navy destroyer bridge was reduced with the expectation of maintaining the same operational performance. Various IMPRINT models were built to measure the variation in the crew's workload and determine the feasibility of this plan. Results showed that the reduction in manning caused an unsustainable workload for the reduced crew. In addition to workload, the influence of manning reductions is also studied on equipment design (Allender, 2000) and performance, where performance may be defined in terms of mission performance (e.g. time taken to complete the mission) or in terms of human performance (e.g. the number of errors committed) (Allender, 2000; Mitchell, Samms, & Wojcik, 2006).

System automation is sometimes used to offset tradeoff factors such as manning (Mitchell D. K., 2003; Mitchell D. K., Samms, Henthorn, & Wojciechowski, 2003; Allender, 2000) and task allocation (Colombi, et al., 2011; Mitchell & McDowell, 2008; Wickens, Bagnall, Gosakan, & Walters, 2012). In the Navy bridge crew trade study, a proposed solution to offset the manning reduction was to supplement the bridge crew with automation (Allender, 2000). A similar solution was proposed in a trade study performed on task allocation for remotely-piloted aircraft (RPA) operators, in which task automation was suggested as a way to offload some of the operator's tasks and balance workload (Wickens, Bagnall, Gosakan, & Walters, 2012). However, Colombi et al. (2011) recommend the strategic implementation of automation, warning that simply automating the "easiest" functions could actually have a negative effect on workload.

Workload is a focus of many human-related trade studies performed using IMPRINT. The assessment of workload is usually placed in the larger context of evaluating other tradeoff factors like manning requirements or operator performance. Aside from manning, a modeler may wish to determine which crewmember could assume additional tasks with the least amount of added workload while maintaining performance (Mitchell & Chen, 2006; Mitchell & McDowell, 2008), or to determine the task allocation for the entire crew (Mitchell D. K., 2003). The type of study in which IMPRINT is used to evaluate the effect on workload from changing another factor, is common throughout the U.S. Army (Allender, 2000; Mitchell, Samms, & Wojcik, 2006; Mitchell & Chen, 2006; Mitchell & McDowell, 2008; Mitchell D. K., 2008; Cassenti, Kelley, Colle, & McGregor, 2011), Navy (Allender, 2000), Air Force (Colombi, et al., 2011; Wickens, Bagnall, Gosakan, & Walters, 2012), and academia (Harriott, Zhang, & Adams, 2013; Rusnock & Geiger, 2014).

Equipment or system designs may drive performance analyses, where performance is measured through IMPRINT workload modeling. For instance, Rusnock and Geiger (2014) performed a trade study which analyzed the effect on performance due to varying workload levels for each of four different system designs. While most workload studies deal with the human's cognitive workload, Harriott, Zhang, and Adams (2013) uniquely studied the effect on physical workload from a human-robot partnership system design.

These tradeoff factors may vary and even interchange as independent, dependent, and controlled variables, depending on the particular trade study's objectives. For instance, while equipment design was previously described as being dependent on manning, it could, conversely, influence manning. The number of crewmembers may need to be reduced to accommodate a smaller vehicle (Mitchell D. K., 2008), or the manning required to operate a system may need to be re-assessed due to an equipment re-design (Allender, 2000).

Tradeoff analyses using IMPRINT are performed using similar methods as in Allender's (2000) trade study of the Navy destroyer. A series of baseline and alternative models may be built in IMPRINT either as a feasibility study or to determine the tradeoff effect of one factor on another. IMPRINT trade studies may be implemented by simulating human workload or performance, where performance could be measured by factors such as task time, accuracy, or completion rates.

Processes, Methods, and Tools

Before discussing previous integration efforts at the process, methods, and tools levels, it is necessary to first define these three levels of scope. For the purposes of this research, a process is defined as a philosophical approach defining *what* activities should be accomplished to achieve an objective. Methods support processes by defining in greater detail *how* to accomplish those activities. Tools are the *enabling* mechanisms that facilitate and enhance the implementation of a given method (Martin, 1996). There may be more than one tool capable of supporting a particular method, and there likewise could be multiple methods capable of supporting a process.

Process-Level Integration

Integration efforts at the process level strive to fundamentally change or augment the SE and/or HSI process itself. Chua and Feigh (2011) offer various ways in which human factors may be generally included in early system development. They organize their ideas according to four system design stages: requirements acquisition, concept generation, preliminary, and detailed. Admittedly at a high level of detail, Chua and Feigh provide general suggestions in an effort to encourage communication between systems engineers and human factors engineers, and to promote awareness of human factors during system design.

Hardman and Colombi (2012) extend the idea of augmenting the SE process by highlighting the necessity for quantitative methods of expressing HSI requirements in order to be properly considered by program management during system development. As such, Hardman and Colombi outline areas in which to emphasize HSI throughout the early requirements analysis, function allocation, and design stages of systems development, and further suggest the usage of empirical measures such as safety and human subjects data to minimize subjectivity.

Another process-level idea is to standardize the terminology between SE and HSI. Hardman, Colombi, Jacques, and Miller (2008) clarify the HSI terminology across the DoD and HSI communities. There are inconsistencies between numerous DoD and HSI publications, such as between the DoDAF, Defense Acquisition Guide, and INCOSE's handbook. The idea of standardization may be extended from the DoD to the entire SE community (Madni, 2009; Orellana & Madni, 2014). Orellana and Madni (2014) argue that the reason why there is a lack of integration between the SE and HSI processes is because differences in terminology prevent systems engineers and those untrained in HSI from communicating with those who are trained. A proposed solution is to build a common HSI ontology to connect the semantics of the two fields, thus providing a means to address HSI concerns during system design (Madni, 2009; Orellana & Madni, 2014). Bruseberg (2008) corroborates Orellana and Madni's claim, citing several examples of differences between HSI and SE's interpretations of terminology. For instance, whereas the term "activity" has a high-level connotation to systems engineers, its scope is more low-level and detailed to human factors engineers.

Methods-Level Integration

Efforts at the methods level strive to enhance integration by improving one of the existing SE design or analysis methods, or by proposing a new method. Crisp, Hoang, Karangelen, and Britton (2000) do the latter. Continuing the ideas put forth by Hardman et al. (2008), Orellana and Madni (2014), and Bruseberg (2008), once a common language between SE and HSI is established, Crisp et al. propose a way to further establish an effective integration. Due to the need for systems engineers to synchronize multiple disciplines, a central software interchange could implement this common language as a data schema in order to translate information between disciplines' software tools and allow communication.

Hardman et al. (2008) propose an augmentation to the DoDAF to improve integration. They examine how each of the nine HSI domains can be addressed in the existing DoDAF products. Each HSI domain lends itself to a DoDAF capability. For example, since the manpower and personnel domains deal with the numbers of users and associated knowledge and skills needed to operate the system, these domains may be addressed by the DoDAF's Operational or Services Views. A properly developed use case can also address manpower in addition to addressing the training domain. Human factors engineering is a key domain to address in system development since it addresses system limitations as a result of human involvement. As such, there are many DoDAF products that may be used to identify problem areas or tradeoff opportunities, such as the Systems Interface Description (SV-1), Systems-Systems Matrix (SV-3), and the Systems Functionality Description (SV-4).

Piaszczyk (2011) proposes a method of integration similar to Hardman et al.'s (2008) DoDAF augmentation. However, Piaszczyk uses a MBSE approach instead, focusing on the DoDAF's graphical products to represent the human. He describes how to factor the human into existing DoDAF views in order to derive human-related requirements and drive system design throughout the acquisition lifecycle. These product re-scopes encompass the DoDAF's Operational and System Views. For example, the Operational Architectural Diagram (OV-2) is used to derive system operator requirements and the Organizational Relationships Diagram (OV-4) is used to define the human's roles with regard to the system. The methods proposed by Hardman et al. (2008) and Piaszczyk (2011) present ways to include HSI in the DoDAF without developing new products.

Another integration method is to create a new, human-focused product to augment existing architecture frameworks. In 2007, representatives from the United States, United Kingdom, Canada, and the Netherlands convened the North Atlantic Treaty Organization (NATO) Human View Panel in order to examine the current state of Human View presence within architecture frameworks around the world, and to propose a standard Human View that could be adopted by any architecture framework (Handley & Smillie, 2008). The resultant NATO Human View is comprised of eight products:

- HV-A: Concept
- HV-B: Constraints
- HV-C: Tasks
- HV-D: Roles
- HV-E: Human Network
- HV-F: Training
- HV-G: Metrics
- HV-H: Human Dynamics

All of these products are designed to address different human aspects that are important to consider during system design and development. For example, the Concept

(HV-A) offers a high-level look at the human component of the system, while Constraints (HV-B) focuses on weaknesses or limitations the human brings that affect the system. HV-B can be further subdivided into subviews such as Manpower Projection Constraints and Personnel Policy Constraints. Since most of these views are static by nature, Human Dynamics (HV-H) is designed to address the dynamic aspects from each of the other views, to include state changes, conditions, time units, and performance measures. The Human View is intended to force systems architects to consider the human in its own architecture framework view instead of arbitrarily adding human considerations into other views. Another goal of adding a Human View directly into an architecture framework is to enable systems engineers and HSI analysts to collaborate early in system development, thus contributing more effectively to design (Smillie & Handley, 2009).

Furthermore, Handley and Knapp (2014) detail four stages by which to compile the Human View products, with each stage focusing on certain sets of models at a time. Moving to the next stage shifts focus to another model, while still reiterating through previous models in order to ensure a complete product is formed. Figure 26 shows the completed Human View development (Handley & Knapp, 2014).

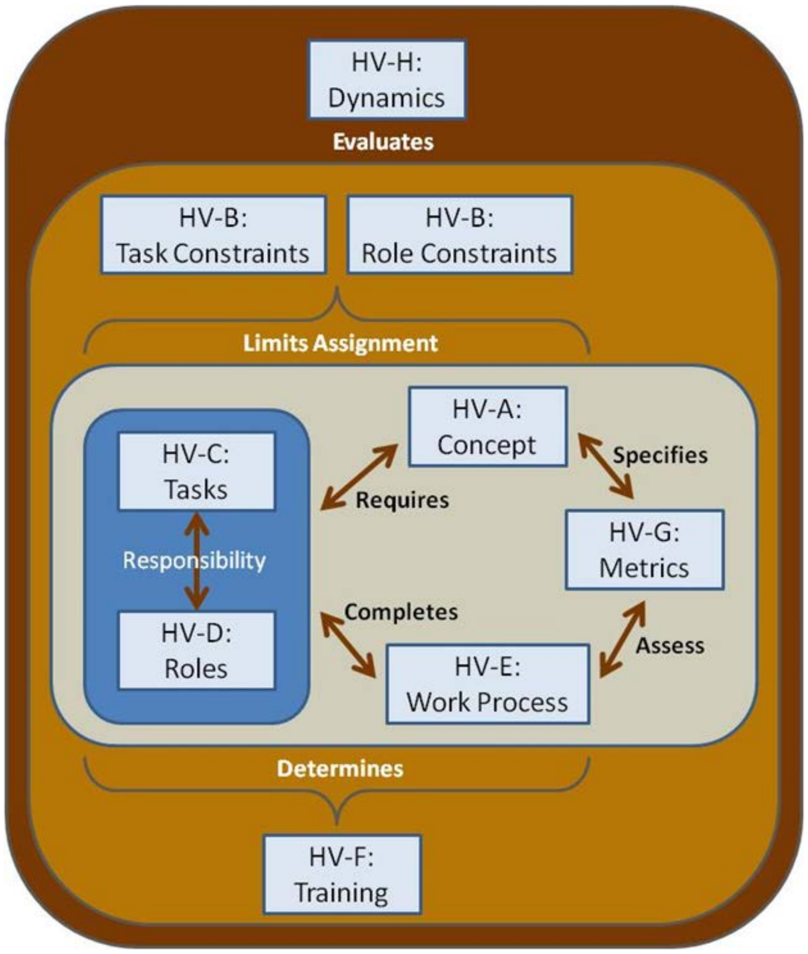

Figure 26: Human View Development – adapted from (Handley & Knapp, 2014)

Handley (2011) made an effort to further adapt the NATO Human View specifically to the DoDAF. The DoDAF 2.0, released in 2009, allows for easier integration of human-centered information within the framework, mainly due to the inclusion of the DoDAF 2.0 Meta Model (DM2). Since the DM2 allows the system architect to create "Fit for Purpose" views to augment the existing architecture framework, Handley claims that the NATO Human View may be mapped to the DM2 more easily than in previous DoDAFs.

Similarly, Bruseberg (2008) proposed a Human View specifically for the British Ministry of Defence Architecture Framework (MODAF). Listing several of the same human-related shortcomings in the MODAF as does Handley (2014) for the DoDAF, Bruseberg (2008) details ways in which her Human View can improve the MODAF's representation of the human during system development. She argues that human views aid in modeling the "soft systems" human side of system development, thus bridging the communication gap between systems engineers and human factors engineers. The MODAF Human View is comprised of seven products, HV-A through G. These products largely parallel the NATO Human View's eight products. For example, the MODAF Human View also has products capturing human functions and tasks (HV-E), roles and competencies (HV-F), and dynamic aspects of human behavior (HV-G). Though similar to the DoDAF-centered Human View, development of the MODAF Human View predates Handley's work and even the NATO Human View.

Sharples (2014) put the NATO Human View into practice to solve a real-world problem for German-based Airbus Defence and Space. Sharples integrated the Human View with Airbus's existing architecture for a remotely-piloted aircraft (RPA) system in order to identify human-related deficiencies and refine the architecture. By taking the Human View's separate products and augmenting the operational and system views from the existing RPA architecture, Sharples was able to identify system gaps such as the absence of several roles from the original model.

Tools-Level Integration

The most in-depth, narrowly-scoped way to integrate the HSI and SE processes is to approach integration at a tools level. Efforts at this level focus on improving the way in which tools such as SysML can be used to incorporate the human into SE. While some researchers advocate the use of modeling and simulation in general to consider HSI (Boy & Narkevicius, 2013), some efforts have specifically used MBSE modeling to accomplish this task. Bodenhamer (2012) states that to understand the human's interaction with the system, the human must first be deconstructed into the functional components necessary to operate the system. These components include sensory channels, cognitive processing, psychomotor capabilities, and physical interfaces. The system itself must also be deconstructed into its components, treating the user as one of these components. Using a landmine detector system as a case study, Bodenhamer created a high-level architectural concept of the system to demonstrate this concept. He modeled the behavioral aspects of the system by creating activity and sequence diagrams. These diagrams visually highlight the human-system interaction that is necessary for mission success. By doing so, Bodenhamer claims that the modeler can identify HSI-related problems that could affect system performance or mission success.

Ramos, Ferreira, and Barceló (2013) address human integration from the process, methods, and tools levels. As part of their larger effort to enhance the overall SE process they amalgamate aspects from a variety of methodologies in order to present a revised, more agile MBSE methodology. However, their main focus is at the tools level. HSI is considered as a part of the overall methodology, in which Ramos et al. advocate a systems engineer-focused implementation of HSI via SysML diagrams such as activity and internal block diagrams.

Orellana and Madni (2014) also address integration from multiple levels of scope. After proposing their process-level HSI ontology, they narrow to the tools level. Orellana and Madni's ontology is influenced by defining the human in terms of SysML diagrams. The goal of the ontology is to "bridge the gap" between systems engineers and human factors engineers by allowing systems engineers to define the human using their own MBSE modeling methods. Orellana and Madni provide a high-level description of ways in which the human can generally be represented through SysML diagrams. Ahram and Karwowski (2009) also recommend a common language by incorporating a HSI framework into systems engineers' SysML modeling practices.

Integration Efforts via IMPRINT Modeling

One of the methods of integration at the tools level is through IMPRINT modeling. There have been efforts to integrate the human into system design using IMPRINT, with all approaching integration in various ways. Mitchell, Agan, and Samms (2011) expanded upon IMPRINT's pre-conceived utility by modeling the system in addition to the human. They emphasize that deconstructing the system and human are essential to system development, but these processes should not be conducted independently of each other. If so, each side misses key variables that could have been otherwise accounted. Mitchell et al. (2011) used IMPRINT to model both the system capabilities and the human functions of a conceptual system in order to identify areas in which both sides can be accounted to improve success.

Smillie and Handley (2009) used IMPRINT to augment a human-focused architectural framework view called the Human View. While the Human View's purpose is to provide system developers a means to focus on the human, Smillie and Handley sought to use the dynamic nature of IMPRINT's DES capabilities to expand upon the Human View. First utilizing the Human View to define a model of a system in a sample case study, they then translated various components of the Human View into IMPRINT. For example, the Human View's Roles, Tasks, and Constraints subviews translated into IMPRINT inputs such as operators, assignments, and moderators. Finally, the IMPRINT model's outputs were analyzed to evaluate the system's impact on the human's performance and workload.

Both Mitchell (2005) and Colombi et al. (2011) used established system models to inform IMPRINT in order to perform system analyses. Colombi et al. used SysML diagrams representing the system's operational concept as a basis for defining the human's tasks, which in turn fed the creation of a workload model. By analyzing the human's envisioned tasks and resultant workload, IMPRINT is able to act as a method of assessing system feasibility early in development.

Although Mitchell (2005) used SysML's predecessor, the Unified Modeling Language (UML), to build the diagrams in her study, the concept is similar to Colombi et al.'s (2011). Mitchell states that while the UML and IMPRINT are effective for developing system and human requirements, respectively, they are not affected by the other's constraints as they should. For example, an activity diagram alone cannot properly depict human performance impacts on the system. Mitchell used a pilot study to develop an approach to link the two modeling methods. An activity diagram depicting the system was used to populate an IMPRINT model representing the human-system interaction, which was then run and the resulting workload outputs analyzed. Through this manual translation from UML to IMPRINT, the analyst is able to see the feasibility of the constraints placed on the human. However, Mitchell admits that a limitation to the study is the absence of a translation from the IMPRINT analysis back to UML, which would help ensure that the human is properly represented by the system.

Research Gap

There have been several previous efforts to integrate HSI methods into systems engineering practices. These efforts have addressed the integration problem from various standpoints: the process level, methods level, and tools level. Numerous processes and methods have been proposed, but most efforts have focused on integration only at the early concept phase of the system's lifecycle. Additionally, few efforts have tried to integrate by addressing SE at the tools level, especially using human performance tools like IMPRINT. These efforts largely utilize IMPRINT to assess human performance and workload with regard to the system, but lack an integration plan to inform system-level models following human-performance analysis. SysML has been integrated with a variety of software tools (Rashid, Anwar, & Khan, 2015), but there is a noticeable lack of integration with HSI tools. SysML and HSI tools are currently disjointed from each other with no clear path on how to integrate them.

A solution to the lack of integration between the HSI and SE processes is to address integration later in the design phases of the system's lifecycle and at the tools level of scope. Focusing at the tools level, system models can incorporate information from human models and UCD at a lower level of system detail. This detail is enabled by integrating later in the system's design phases, such as preliminary or detailed design, when more information about the system and human are known. HSI methods like UCD enable a focus on system design from a user perspective. Similarly, HSI tools like IMPRINT model human performance, the results of which can be used to update system models. The result is system models that better attend to human considerations, thus improving system design.

Appendix B – Baseline Model Description

Overview

This appendix provides a detailed description of the Improved Performance Research Integration Tool (IMPRINT) baseline model of Vigilant Spirit, the assumptions made when creating the model, and how the model was verified.

Introduction

A baseline model was established to accurately represent the 711th Human Performance Wing's (HPW) human subjects experiment. The baseline model was developed to replicate scenario four of the experiment, in which there are a high level of distractors (48) and visual static noise imposed over the camera feed. Figure 27 shows the baseline model that was built using IMPRINT. The model assumes the Operator has been trained on how to perform the mission. There are two preconditions for the mission: that the Operator is sitting at the remotely piloted aircraft (RPA) simulation station with the experiment's equipment operational and running, and that the experiment administrator has started the program for the Operator and the mission has begun. Upon Model Start, two separate task flows initialize and run concurrently. The primary task flow is the high value target (HVT) surveillance task. The secondary task flow is the communication task. Purple nodes shown in Figure 27 are tasks performed by the system, and blue nodes are tasks performed by the Operator.

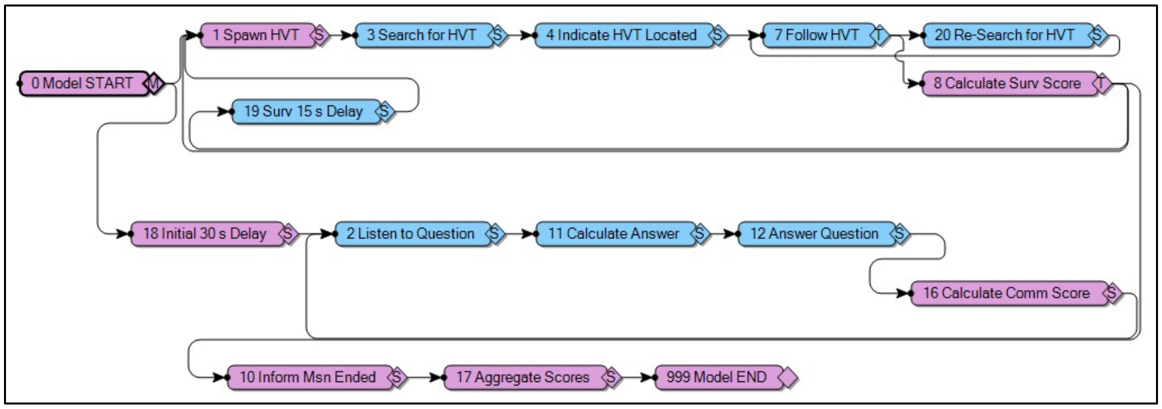

Figure 27: Baseline Vigilant Spirit Model in IMPRINT

Surveillance Task

The following is a detailed description of the surveillance portion of the model, with a step-by-step walkthrough of each task node within the network.

Node 1: Spawn HVT

This task is performed by the system. When the surveillance entity arrives at the Spawn HVT node, the integer global variable *HVTIteration* is immediately incremented up by one. *HVTIteration* keeps track of how many HVT iterations there have been, therefore this signifies that the first, or next, HVT has been spawned. Additionally, the Boolean global variable *LoseHVT* is set to false and the floating point global variable *TtlFollowTime* is set to 0, which serve as variable "resets" for each iteration. These two variables will be further explained in later nodes. This task takes 0 seconds to complete, so as not to detract from the 59 seconds that the Operator has to search for the HVT.

Note that each of the four HVT iterations are spawned at 0, 60, 120, and 195 seconds, corresponding to the global *Clock* variable.

Node 3: Search for HVT

In this node, the Operator searches for the HVT and either succeeds in locating it within 59 seconds or fails to find the HVT. To determine success or failure, a local variable named *random* is created and set to a randomly-generated floating point number between 0 and 1. If *random* is less than 0.63, then the Operator found the HVT. 0.63 is a probability derived from analyzing the 711th HPW's experiment data. Subjects found the HVT 121 times out of 192 iterations of searching, yielding 0.63 as the probability.

Three global variables are used in this task node. *FoundHVT* tracks whether or not the Operator found the HVT in an iteration. It is a Boolean variable with an initial value of false. *SearchTime* is the time it takes the Operator to search for the HVT. It is a floating point variable. *SearchTime*, along with all other floating point variables in the model, has an initial value of 0. *SurvTimeLeft* is the time remaining in the iteration after the Operator searches for the HVT and either finds the HVT or does not. It is also a floating point variable.

If the HVT is found, *FoundHVT* is set to true and *SearchTime* is set to a value determined by a Weibull distribution with a threshold of 4 and with shape and scale parameters of 1.74 and 25.7, respectively. The Weibull distribution was determined by inputting the experiment subjects' 121 successful search times into Arena's Input Analyzer software and choosing a best fit distribution to model the data. The Weibull distribution had the least square error, with a value of 0.007866. Operators have 59 seconds to search for the HVT. Thus, because the Weibull distribution has no upper bound, any values generated that are greater than 58.8 seconds are rounded to equal 59 seconds. The 0.2 second difference is because if the Operator finds the HVT but does not have at least 0.2 seconds to indicate the HVT is found (described in the next task node), then the search time takes the entire 59 seconds. *SurvTimeLeft* is set to *SearchTime* subtracted from the originally allotted 59 seconds.

If the Operator never finds a HVT in an iteration, *SearchTime* is set to 60 seconds and all subsequent task nodes in the iteration are set to 0 seconds to provide a seamless transition for the Operator to continue searching into the next HVT iteration without a break in workload. Additionally, *FoundHVT* is set to false and *SurvTimeLeft* is set to 0 seconds.

The task time for this node is the *SearchTime*, however long it took the Operator to search for the HVT.

Node 4: Indicate HVT Located

As determined by IMPRINT's micromodel for single finger keying rate, it takes the Operator 0.2 seconds to press the F key on the keyboard if the HVT is found. If *SurvTimeLeft* is at least 0.2 seconds, then the task time is simply 0.2 seconds and *SurvTimeLeft*'s value is updated to subtract 0.2. Otherwise, *SurvTimeLeft* is 0 seconds. This means the Operator either spent the entire 59 seconds searching but never found the HVT, or found the HVT but did not have enough time to indicate so. If *SurvTimeLeft* equals 0, then this node's task time is also 0 seconds.

Node 7: Follow HVT

In this node, the Operator either follows the HVT for the remaining time in the iteration, or follows for some time then loses track of the HVT. This node uses the Boolean *LoseHVT*, floating point *FollowTime*, and floating point *TtlFollowTime* global variables. *LoseHVT* tracks whether or not the Operator lost the HVT once found in an iteration. *FollowTime* is the time it takes the Operator to follow the HVT in this task node. *TtlFollowTime* is the total amount of time the Operator has followed the HVT in the entire iteration.

Upon entering the node, if *SurvTimeLeft* is greater than 0 and *LoseHVT* is false, then the Operator found the HVT for the first time in this iteration. To determine if the Operator loses the HVT, a local variable named *random* is created and set to a randomly-generated floating point number between 0 and 1. If *random* is less than 0.562, then the Operator follows the HVT for the remaining time without losing it. 0.562 is a probability derived from analyzing the 711th HPW's data, where subjects did not lose the HVT 68 times out of 121 iterations of following, yielding 0.562 as the probability. If the Operator does not lose the HVT, then *LoseHVT* is set to false, *FollowTime* is set to equal *SurvTimeLeft*, and the task time for this node is the value of *SurvTimeLeft*.

If the Operator loses the HVT, then *FollowTime* is set to a value determined by a Beta distribution with a threshold of 42 and shape parameters of 0.588 and 1.23. The Beta distribution was determined by inputting the experiment subjects' 53 follow times (if lost HVT) into Arena's Input Analyzer and choosing a best fit distribution to model the data. The Beta distribution had the least square error, with a value of 0.007394. Because the time spent following the HVT cannot be longer than the time remaining in the iteration, if the *FollowTime* value generated is greater than *SurvTimeLeft*, then it is assumed that the Operator successfully followed the HVT for the remaining time without losing it. Therefore, *LoseHVT* is set to false and *FollowTime* is set to equal *SurvTimeLeft*. Else, *LoseHVT* is set to true. In either case, *SurvTimeLeft* is updated to subtract the *FollowTime*, and the task time for this node is the value of *FollowTime*.

Alternatively, upon entering the node, if *SurvTimeLeft* is greater than 0 and *LoseHVT* is true, then the Operator had previously found the HVT in this iteration, but lost it and had to re-search and find the HVT again. The logic is the same for this task path as in the previous two paragraphs, with the exceptions being a different probability and probability distribution. The reason for this change is because Operators have a higher probability of successfully following the HVT if they have already previously lost and found it again. Therefore, to determine if the Operator loses the HVT again, a 0.66 probability is used. Out of 53 times of originally losing the HVT, 711th HPW subjects never lost the HVT again 35 times, yielding 0.66.

If the Operator loses the HVT again, the *FollowTime* is set to an Exponential distribution with a mean of 12.2, determined by inputting the subjects' 78 re-follow times (after losing the HVT, re-searching, and finding again) into Arena's Input Analyzer and choosing a best fit distribution. The Exponential distribution had the least square error, with a value of 0.007821.

If *SurvTimeLeft* is 0 seconds, then the Operator either never found the HVT or did not have enough time to indicate and start following, thus this task is "skipped." *LoseHVT* is set to false, *FollowTime* is set to equal *SurvTimeLeft*, and the node's task time is 0 seconds.

Regardless of the task time, the node has an ending effect that updates the *TtlFollowTime* variable, which is calculated by summing the original *TtlFollowTime* with *FollowTime* from this task. If this is the first time the Operator followed the HVT, then the original *TtlFollowTime* is 0, yielding a new *TtlFollowTime* equal to this task's

FollowTime. But if the Operator has previously followed the HVT in this iteration and lost it, then the original *TtlFollowTime* is any time spent previously following the HVT.

This node has tactical path logic. If *LoseHVT* is false, then the Operator did not lose the HVT and follows the HVT for the remainder of the iteration, so the entity moves to Node 8: Calculate Surveillance Score, described later. If *LoseHVT* is true, then the Operator lost the HVT and searches for it again in Node 20: Re-Search for HVT, described next.

Node 20: Re-Search for HVT

In this node, the Operator re-searches for the HVT after having lost it, and either locates it in the time remaining or fails to find the HVT again in the iteration. To determine success or failure, a local variable named *random* is created and set to a randomly-generated floating point number between 0 and 1. If *random* is less than 0.929, then the Operator found the HVT again. 711th HPW subjects found the HVT 78 times out of 84 iterations of re-searching, yielding 0.929 as the probability.

This node uses the floating point global variable *ReSearchTime*, which is the time it takes the Operator to search for the HVT again after losing it.

If the HVT is found again, *ReSearchTime* is set to a value determined by a Lognormal distribution with a mean and standard deviation of 3.03 and 3.61, respectively. The Lognormal distribution was determined by inputting the experiment subjects' 84 re-search times into Arena's Input Analyzer and choosing a best fit distribution. The Lognormal distribution had the least square error, with a value of 0.004792. Because the time re-searching cannot be longer than the time left in the iteration, any *ReSearchTime* values generated that are greater than *SurvTimeLeft* are set to equal *SurvTimeLeft*, meaning the Operator re-searches for the remainder of the iteration. *SurvTimeLeft* is updated to subtract the *ReSearchTime*.

If the Operator never finds the HVT again in the iteration, *ReSearchTime* is set to equal *SurvTimeLeft*, meaning the Operator re-searches for the remainder of the iteration. Additionally, *SurvTimeLeft* is set to equal 0.

The task time for this node is the *ReSearchTime*. Upon completion of this node, the entity loops back to Node 7: Follow HVT for re-evaluation.

Node 8: Calculate Surveillance Score

In this node, the system calculates the Operator's surveillance score for the iteration completed. The Operator receives four points per second for the total time spent following the HVT after it was found and indicated as found. The value of *TtlFollowTime* is the time used to calculate the score because it is the total time the Operator spent following the HVT, disregarding any time spent re-searching for the HVT if the Operator lost it. The score is saved in the global variable *SurvScore*, which is a floating point variable. Each subsequent iteration adds the score for that iteration to the previous *SurvScore* value for a cumulative surveillance score.

The time for this task is dependent on if the Operator found the HVT. If *FoundHVT* is set to true, meaning the Operator found the HVT, then the task time is 1 second. This is because each HVT is spawned every 60 seconds, so there is a 1 second pause between when an iteration's HVT is removed and the next iteration's HVT appears. Note that *FoundHVT* can be true even if the Operator did not have enough time to indicate the HVT as found. In such a case, there is still a break in searching because the Operator did actually still find the HVT, albeit with however little time left. If

FoundHVT is set to false, meaning the Operator never found the HVT and spent the entire iteration searching, then the task time is 0 seconds. This is to represent that since the Operator spent the entire iteration searching, he/she will continue searching into the next iteration without a break, as the removal and re-spawning of the HVT is transparent to the Operator. Recall that, in this case, *SearchTime* is the full 60 second iteration length and all other variable times are 0 seconds.

This node has tactical path logic. If *HVTIteration* is less than three, then the surveillance entity will directly return to Node 1: Spawn HVT, thus beginning the next iteration and repeating the previously described task nodes. If *HVTIteration* equals three, then the entity will move to Node 19: Surv 15 s Delay, described next. If *HVTIteration* equals four, then the mission is complete and the entity moves to Node 10: Inform Msn Ended, described later.

Node 19: Surv 15 s Delay

After the third iteration, the Operator searches for the HVT for 15 seconds before the system actually spawns the fourth and final HVT. This is to represent that in the 711th HPW's experiment, in between the third and fourth iterations a bio marker was implanted in each subject's mouth, thus a 15 second buffer was built into the timing so as not to give the subjects a disadvantage at searching due to the interruption. In the model, the task is 15 seconds and the Operator experiences the same workload as in Node 3: Search for HVT. Upon completion, the entity returns to Node 1: Spawn HVT where the final iteration is started. Recall that Node 1: Spawn HVT task time is 0 seconds, so the flow from Node 19: Surv 15 s Delay to Node 3: Search for HVT is transparent to the Operator.

Communication Task

While the primary surveillance task flow is being accomplished, the secondary communication task flow concurrently runs. The following is a description of the communication path, with descriptions of each task node.

Node 18: Initial 30 s Delay

After Model Start, the system waits 30 seconds before asking the first communication question. This is because each of the four communication questions is asked halfway through each corresponding HVT iteration.

Node 2: Listen to Question

In this node, the Operator listens to a communication question being asked by the system. When the communication entity arrives at Node 2: Listen to Question, the integer global variable *CommIteration* is immediately incremented up by one. *CommIteration* keeps track of how many iterations of communication questions there have been, therefore this signifies that the first, or next, question has been asked.

There are a variety of communication questions the system may ask, with length ranging from 15 to 23 words. As determined by IMPRINT's micromodel for speech, it takes the system 5.17 seconds to speak 15 words and 7.93 seconds to speak 23 words. The floating point global variable *AskTime* represents the time it takes the system to ask the communication question, and is set to a value determined by a Rectangular distribution with a minimum of 5.17 and mean of 6.55, where the mean is determined by averaging 5.17 and 7.93 seconds. A Rectangular distribution was chosen because there are equal chances of the system asking a question 15 to 23 words long. The task time for this node is the *AskTime*.

The global variable *CommTimeLeft* is the time remaining in the iteration after the system asks the question and, later in the model, after the Operator either calculates and answers the question or does not. It is a floating point variable, and acts as the "countdown timer" for each iteration. For communication iterations one and two, the iteration lasts 60 seconds. Therefore, *CommTimeLeft* is set to *AskTime* subtracted from the original 60 seconds. To account for the bio marker implantation described previously, an extra 15 seconds is added to the third communication iteration for a total length of 75 seconds. Therefore, *CommTimeLeft* is set to *AskTime* subtracted from 75 seconds. The fourth and final iteration is only 30 seconds, thus *CommTimeLeft* is set to *AskTime* subtracted from 30 seconds.

Recall that each question is asked halfway through each HVT iteration, at 30, 90, 150, and 225 seconds according to the *Clock* variable.

Node 11: Calculate Answer

In this node, the Operator attempts to calculate the answer to the communication question and either successfully calculates the answer or fails to do so. Success is defined as calculating the answer within 30 seconds, which will be elaborated upon later. To determine success or failure, a local variable named *random* is created and set to a randomly-generated floating point number between 0 and 1. If *random* is less than 0.938, then the Operator successfully calculated the answer. 0.938 is a probability derived from analyzing the 711th HPW's data. Out of 192 questions asked, subjects answered within 30 seconds 180 times, yielding 0.938 as the probability.

The global variable *CalcTime* is the time it takes the Operator to calculate the answer to the question, and is a floating variable. If the answer is successfully calculated, *CalcTime* is set to a value determined by a Lognormal distribution offset by 6 and with a mean and standard deviation of 6.75 and 4.24, respectively. The Lognormal distribution was determined by inputting the experiment subjects' 180 successful calculation times into Arena's Input Analyzer and choosing a best fit distribution. The Lognormal distribution had the least square error, with a value of 0.000477. Operators only have the remaining iteration time after the system asks the question. Thus, because the Lognormal distribution has no upper bound, any *CalcTime* values generated that are greater than the value of *CommTimeLeft* are rounded to equal the value of *CommTimeLeft*.

If the Operator fails to calculate an answer, *CalcTime* is set to equal the value of *CommTimeLeft* to represent that the Operator spent the entire communication iteration attempting to calculate an answer. As a result, all subsequent task node times in the iteration are 0 seconds so that the Operator continues to calculate an answer until asked another question.

Additionally, *CommTimeLeft* is updated to subtract out the *CalcTime*. If the Operator failed to answer the question, then this value results in 0 seconds. The task time for this node is the *CalcTime*: however long it took the Operator to calculate an answer.

Node 12: Answer Question

In this node, the Operator answers the question if he/she was able to successfully calculate it. The length of an answer may range from 5 to 8 words. As determined by IMPRINT's micromodel for speech, it takes the Operator 1.72 seconds to speak 5 words and 2.76 seconds to speak 8 words. The floating point global variable *AnsTime* is the time it takes the Operator to answer the communication question, and is set to a value determined by a Rectangular distribution with a minimum of 1.72 and mean of 2.24, where the mean is determined by averaging 1.72 and 2.76 seconds. A Rectangular distribution was chosen because there are equal chances of the Operator's answer being 5 to 8 words long.

The task time for this node is the *AnsTime*. If *CommTimeLeft* is at least equal to *AnsTime*, then *CommTimeLeft*'s value is updated to subtract out the *AnsTime*. Otherwise, two updates occur. First, *AnsTime* is set to equal *CommTimeLeft*. This means that the Operator either never calculated an answer or calculated an answer but did not have enough time to fully answer, thus the Operator talked for whatever time remained in the iteration. Note that this is still considered a failure. Second, *CommTimeLeft* is set to 0 seconds. If *AnsTime* is 0 seconds, this means the Operator spent the entire iteration calculating an answer, thus this task is "skipped."

Node 16: Calculate Comm Score

In this node, the system calculates the Operator's communication score for the iteration completed. The Operator may earn a maximum of 50 points per question. The Operator receives full points if the question is answered in 20 seconds or less, then loses five points for every second taken afterwards, until the Operator finally receives no points if answered in 30 seconds or greater. The total time of *CalcTime* added to *AnsTime* is the value used to calculate the score because it is the time the Operator spent both calculating and answering the question. Note that the Operator receives all 50 points in the fourth iteration, regardless of the time. This is due to the Operator's impaired speaking ability resulting from the implanted bio marker. The score is saved in the global variable *CommScore*, which is a floating point variable. Each subsequent iteration adds the score for that iteration to the previous *CommScore* value for a cumulative communication score.

The time for this task is simply the time remaining in the iteration, as determined by *CommTimeLeft*. If the Operator answered the question relatively quickly, then this time will be several seconds long. If the Operator spent the entire iteration calculating the answer, then this task time is 0 seconds.

This node has path logic associated with it. If *CommIteration* is less than four, then the communication entity will return to Node 2: Listen to Question, thus beginning the next iteration and repeating the previously described task nodes. Otherwise, *CommIteration* equals four and the task flow stops because the mission has ended.

Post-HVT/Communications Iterations

After both the surveillance and communication task flows are complete, the model ends by performing the following two tasks:

Node 10: Inform Msn Ended

This is a system node modeled to convey that the system informs the Operator that the mission has ended. It has no effects or task time.

Node 17: Aggregate Scores

This is also a system node, where the system combines the Operator's surveillance and communication scores to form a total score. The cumulative scores from all four iterations, recorded in *SurvScore* and *CommScore*, are added together. The resulting value is saved in the global variable *TotalScore*, which is a floating point variable. This node has 0 seconds task time.

After the previous task node is complete, the Model Ends. The model takes 255 seconds to complete, corresponding to the *Clock* variable.

Assumptions

The baseline model was created with several underlying assumptions. These assumptions are described in Table 4 below.

Table 4: Baseline Model Assumptions

Assumption	Rationale
The data in the "Surveillance Scenario 4 – Reverse Engineer.xls" file are correct.	The human subjects' various times were extracted from this file and used to fit several probability distributions for the model's HVT search and follow times. Since the file is used as the basis for the model and will be what the model results are compared to, it is assumed that the data within the file is correct.
The data in the "HUMAN Formal Study 1 Key Press Data.xls" file are correct.	The times spent calculating the answers to the communications questions were extracted from this file and used to fit a probability distribution for the model's answer times. It is assumed that the data within the file is correct.
The HVT is visible for exactly 59 seconds once it spawns.	The actual visible time in the experiment varied between 54-56 seconds, as the HVT is obscured while walking in or out of the tent. 59 seconds was modeled to standardize the model's timing.
It always takes the Operator 0.2 seconds to press the F key to indicate the HVT as found.	IMPRINT's micromodel for single finger keying rate (to press the F key) was chosen in order to simplify the model, though the actual time would vary per individual.

A false alarm of hitting the F key when no HVT is on-screen has a 0% probability of occurring.	Since a false alarm occurred rarely to never in the experiment, the Operator was modeled as never incorrectly hitting the F key in order to simplify the model.
The Operator has already zoomed the camera in enough for maximum points when he/she finds the HVT and starts following.	The mouse data from the experiment was unavailable. However, upon visual inspection of the score data, it appeared that most or all of the experiment subjects were already zoomed in when they started following the HVT.
If the Operator finds the HVT but loses and re-finds again, he/she does not have to re-indicate the HVT as found.	As in the experiment, the Operator's initial indication of finding the HVT alerts the system to begin scoring, thus the system is henceforth aware of whether the HVT is on-screen and does not need to be re-alerted.
If the Operator finds the HVT in an iteration, it takes the system 1 second to calculate the surveillance score.	This task's timing was modeled in this manner to ensure each iteration lasted 60 seconds. Each HVT is visible for 59 seconds, then re-spawns 1 second thereafter.
Answering a communication question incorrectly has a 0% probability of occurring.	Since answering a communication question incorrectly occurred rarely to never in the experiment, the Operator was modeled as never incorrectly answering the question in order to simplify the model.
The Operator already has a hand positioned on the respective keyboard keys when indicating that he/she has found the HVT and when answering a communication question.	This assumption affects the timing and cognitive workload of associated tasks in the model. Although it is unknown where the subjects actually placed their hands, it is probable that their hands were positioned in this way during the experiment, as opposed to re-positioning every time. Thus, it is modeled as such in order to simplify the model.

For the timing of the Calculate Comm Score task, it takes the system whatever the difference is between the total time allotted for that iteration and how long it took the Operator to calculate the answer.	As with the Calculate Surv Score task, this task's timing was modeled to maintain the experiment's timing. Each new question is asked at certain times during the experiment, so this task makes up the timing difference.
The VACP workload values assigned to the Operator are representative of workload experienced during the experiment.	Without the knowledge of each subject's personal strategy or what they were actually thinking, workload values are assigned in the model to reflect how the experiment was intended to be completed. For example, that the Operator is actively searching for the HVTs and trying to calculate the answer to the questions, as opposed to not trying.
The Operator completes each task modeled without deviating from the model.	As with the previous assumption, each subject's personal strategy or what they were actually thinking is unknown. Thus, the Operator was modeled as following each task in order to reflect how the experiment was intended to be completed.
The workload experienced while performing the secondary communication task does not affect the Operator's workload for the primary surveillance task.	Without the knowledge of each subject's personal strategy or what they were thinking, workload was modeled in this way in order to simplify the model.

Model Verification

To verify that the baseline model's results were as expected, several factors regarding task performance and workload were evaluated while running the IMPRINT model.

Task Performance

The number of times tasks were performed: The appropriate surveillance task nodes (Spawn HVT, Search for HVT, Indicate HVT Located, and Calculate Surv Score) and communication task nodes (Listen to Question, Calculate Answer, Answer Question, and Calculate Comm Score) were each performed four times, corresponding to the four iterations. Recurring nodes (Follow HVT and Re-Search for HVT) were performed multiple times during iterations, corresponding to the Operator losing and re-finding the HVT. Other nodes (Initial 30 s Delay, Surv 15 s Delay, Inform Msn Ended, and Aggregate Scores) were only performed once because they only occurred at single points in the model.

The task durations: All nodes with deterministic times (Spawn HVT, Initial 30 s Delay, Surv 15 s Delay, Inform Msn Ended, and Aggregate Scores) were unchanged. The other nodes that had stochastic or tactical task times exhibited times consistent with the model's coding and values within parameters. Additionally, the model's total run time was indeed 255 seconds.

Workload

The task start times: All tasks that were supposed to occur at certain times performed as expected. Each new HVT was spawned every 60 seconds and each new communication question was asked 30 seconds into each HVT iteration.

Congruence of the Operator's workload throughout the model: As seen in the sample workload graph in Figure 28, most HVT iterations generally start with the Operator searching for the HVT with a VACP workload of 17.4. A slightly lower score shows the Operator found the HVT and has indicated so, followed by an even lower score as the Operator progresses into "follow" mode. This decreased workload makes sense, as the Operator is not as actively engaged and merely must follow the HVT. When a communication question is asked, this naturally increases the Operator's workload. Note that calculating an answer has a higher workload than listening to the question. This is because of the combination of looking up the necessary value on the monitor to calculate the answer, and the cognitive process of actually calculating the answer. The workload value when answering the question is in between the workload values of the two other communication tasks due to the high number of channels used but relatively lower cognitive demand. The sharp drop-offs in workload occur when the Operator is following an HVT and the HVT is removed at the end of the iteration. For that one second, the Operator has nothing to do before starting to search for the next HVT.

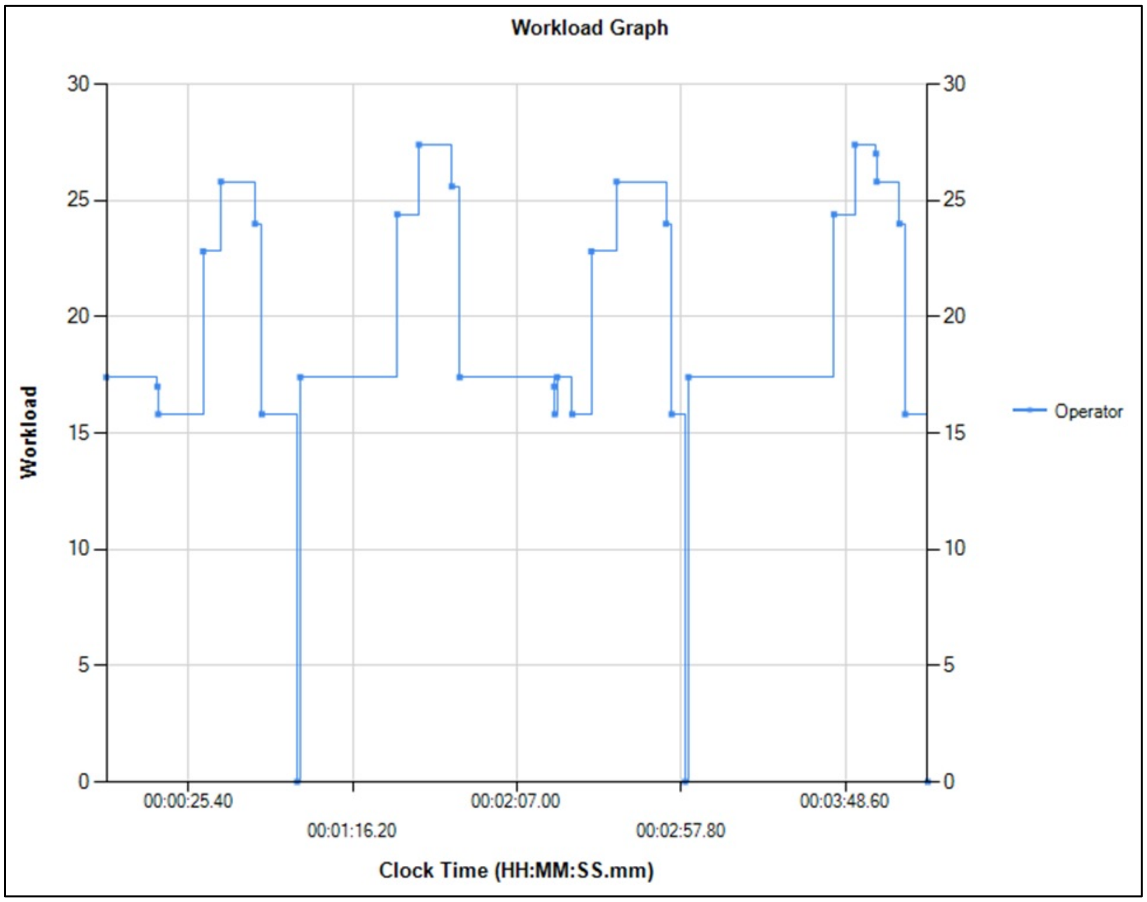

Figure 28: Sample Operator Workload Graph

All of the results presented in the IMPRINT reports were as expected and within parameters. Thus the baseline model was successfully verified to match the conceptual model and was consistent with how the model was intended to perform.

Appendix C – Baseline Model Validation

Overview

This appendix provides a detailed description of how the Vigilant Spirit baseline Improved Performance Research Integration Tool (IMPRINT) model was validated.

Model Response Variables

Two response variables were measured in the model and validated. The first variable is the operator's performance score. This is the total score resulting from aggregating the individual surveillance and communication scores from the primary and secondary tasks during the mission. The surveillance score measures how quickly the operator located the four HVTs and how well he/she was able to follow the HVTs once located. The communication score measures how quickly and accurately the operator was able to answer four questions asked during the mission. The two scores combined measure the operator's total performance during the mission.

The second variable is the operator's workload. IMPRINT models workload using the visual, auditory, cognitive, psychomotor (VACP) method (Bierbaum, Szabo, & Aldrich, 1989). In this method, each of the operator's tasks are assigned a workload value corresponding to the auditory, cognitive, fine motor, gross motor, speech, tactile, and visual demands associated with that task.

Real-World Response Variables

The real-world data to validate against was collected during a study conducted by the 711th Human Performance Wing (HPW). In this study, operator performance scores and workload were also measured. The performance score is described in the same way as the model. Since the study involved subjects undergoing a computer-simulated mission, the score data was collected directly via the simulation software.

The workload data collected is different than the model's data. After the 711th HPW's subjects completed the mission, they filled out a NASA-TLX form, in which they rated the mental demand, physical demand, temporal demand, performance, effort, and frustration they experienced during the entire mission (Hart & Staveland, 1988). NASA-TLX workload measurements are subjective and empirical in nature. By contrast, the VACP method is more objective and analytical. The dichotomy between these two methods precludes a statistical validation of the model's workload with the real-world.

Validating Performance Score

The model's performance score was assessed via two methods: visual and statistical evaluation.

First, in order to determine the number of model replications needed to validate the data, a desired confidence interval was determined. To aid in doing so, the descriptive statistics for the real-world data were calculated, shown in Table 5. This included calculating the corresponding interval half-widths for various confidence levels to get an idea of the data spread.

Table 5: Real-World Data Statistics for Score

n	48	Confidence Level	.95	.90	.80	.70
Mean (M)	480.984					
Std Deviation (SD)	152.858					
Variance	23365.617	Half-Width	44.385	37.020	28.678	23.122
Min	200					
Max	851.452					
Range	651.452					

The 37.020 half-width for a 90% confidence level corresponded to a 15.4% range of the full data set centered on the mean. This 15.4% range provided a tight interval of (443.964, 518.004) while also keeping a high confidence level. Therefore, the 90% confidence level with a half-width of 37.020 (highlighted in Table 5) was chosen as the desired half-width for the model data.

Next, an initial set of ten pilot runs were performed by the model to determine the number of replications needed to achieve the desired half-width. A 90% confidence level on the pilot runs yielded a half-width of 68.203. Thus, it was calculated that 34 replications were needed to achieve the closest half-width to 37.020. These replications were run, with the descriptive statistics shown in Table 6.

Table 6: Model Data Statistics for Score

n	34
Mean	487.286
Std Deviation	141.088
Variance	19905.85
Min	122.773
Max	704.569
Range	581.796
Confidence Level	.90
Half-Width	40.949

The model's 40.949 half-width for a 90% confidence level corresponded to a 16.8% range of the full data set centered on the mean. This range also provided a tight interval of (446.337, 528.235), which is only 1.5% different than the spread of the real-world data. The confidence intervals for the real-world and model data were plotted and are shown in Figure 29.

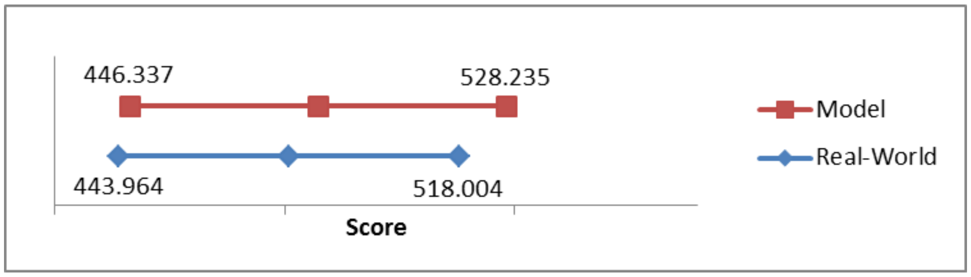

Figure 29: Confidence Intervals for Real-World and Baseline Model Score Data

Upon visual evaluation of the confidence intervals, there is a 72 point overlap between the two data sets. This overlap is significant, considering it is greater than the half-widths of either sample. This overlap provided an indication that the model's data was similar enough to the real-world data that the model may be valid.

Next, a statistical evaluation of the data was conducted by performing an independent-samples t-test between the two data sets. The null hypothesis H_0 was defined such that there was no difference between the real-world and model data, while the alternative hypothesis H_A stated that there was a difference. The t-test results are displayed in Table 7.

Table 7: Independent-Samples T-Test Results

T-Value	0.19246
degrees of freedom (df)	75
Critical Value ($t_{.90, 75}$)	1.6654
P-Value	0.84791

There was not a significant difference in the scores for the model (M=487.286, SD=141.088) and real-world (M=480.984, SD=152.858) conditions; t(75)=0.19246, p=0.84791. The critical region in which H_0 should be rejected is t>1.6654. Since t=0.19246, we failed to reject H_0, meaning there was insufficient evidence the model differed from the real-world. Further, using the p-value approach, the fact that p=0.84791 was significantly greater than the chosen critical value of p=0.10 indicated that there was no significant difference between the two sets.

These results suggested that there is insufficient evidence the baseline model differs from the real-world, thus resulting in a successful validation of score.

Validating Workload

As previously mentioned, the contrast between the model's VACP method and the 711th HPW's NASA-TLX workload assessment prevented a direct comparison between the two data sets. However, the model's workload was still assessed via peer and subject matter expert (SME) review.

First, peer reviews were conducted with two colleagues, during which each of the model's human tasks were explained to them and they provided their opinions on the associated workload levels for each task. An analysis of their assigned workloads revealed the same overall trend in surveillance workload patterns throughout the mission as in the model. The Search for HVT node has a higher workload than Follow HVT, while the Indicate HVT Located workload values are similar to Search for HVT. There was some disagreement between whether Indicate HVT Located should be a higher or lower value than Search for HVT. For the communication tasks, there was no general agreement on an overall workload pattern between Listen to Question, Calculate Answer, and Answer Question. However, the demand values between the model and peers only varied by one or two points, and were all between 7 and 11. For all task nodes, when both peers assigned workload values that were consistently higher or lower than the model's, adjusting the corresponding value within the model was considered. Otherwise, personal judgment was used to assess whether the model's workload values needed to be changed. Following analysis of the peer reviews, the cognitive workload values for Search for HVT, Indicate HVT Located, and Follow HVT were increased. Overall, the workload values from the peer reviews closely correlated with those of the model, and little adjustment was necessary.

For the SME review, a member of the 711th HPW that was involved in conducting the experiment was interviewed. The following questions were asked in order to aid the SME in assessing his opinion on task workload levels in relation to each other:

1. Do you think workload is higher (or the same) when the Operator is searching for the HVT, or following the HVT?
2. Do you think the workload for indicating the HVT is located is higher or lower than either or both of the other two tasks?
3. Do you think workload is higher (or the same) when the Operator is listening to the question or answering the question?
4. Do you think the workload for calculating the answer is higher or lower than either or both of the other two tasks?

For the surveillance tasks, the SME's responses correlated with the trend in the model's workload patterns. Search for HVT is unequivocally the task with which Operators experience the highest demand. Indicate HVT Located follows, which the SME stated was slightly more difficult than Follow HVT. For the communication tasks, the SME agreed that Calculate Answer had the highest workload. However, the SME's opinion on the two remaining tasks slightly differed from the model's workload. The SME thought that Listen to Question had a higher demand than Answer Question because all the latter involves is stating the answer since the Operator already calculated it. In contrast, the model places a higher workload value on Answer Question than Listen to Question. However, the SME pointed out uncertainty in his assessment of Listen to Question being higher demand, stating that the two tasks are very close. This agrees with the model, as the VACP point difference between the two tasks is only 1.2. Therefore, it was decided that the model's current workload values were acceptable as is. The reason for this decision is that a high-level discussion of workload fails to take into account the intricacies of the different channels used in a particular task. Most people do not consider the fine motor and tactile activities that are part of answering the question, which are the key factors that make Answer Question a slightly higher demand value than Listen to Question in the model.

Following peer and SME review, it was concluded that the model's workload response variable was validated. The demand values assessed by peers and the opinions discussed by the SME either directly agreed with the model or differed slightly enough as to yield no significant changes to the model. The only change to workload values in the model was an increase in cognitive demand for the surveillance tasks.

Appendix D – Alternative Model Description

Overview

This appendix provides a detailed description of the Improved Performance Research Integration Tool (IMPRINT) alternative model of Vigilant Spirit, to include changes from the baseline, and the assumptions made when creating the model.

Introduction

The alternative design of the remotely piloted aircraft (RPA) surveillance mission incorporates a theoretical system automation in an effort to aid the operator's performance and workload. The system automation uses an algorithm to scan for the high value target (HVT) within the camera's field of view while the operator also searches the market. The scanning algorithm uses a number of features from the environment to identify the HVT, such as on-screen individuals' movements, behaviors, and equipment. If the system determines the HVT is in view, it notifies the operator and requests confirmation. If the potential target is indeed the HVT, then the operator confirms by pressing the F key and proceeds to follow the HVT. If the operator denies the potential HVT, the system begins scanning for the HVT again while the operator continues searching. Figure 30 shows the IMPRINT task network for the alternative model, with the system automated task in green.

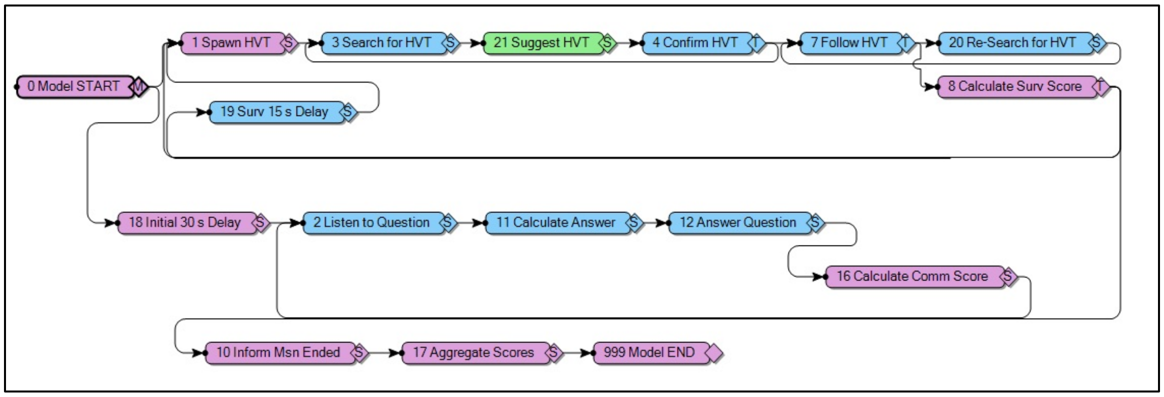

Figure 30: Alternative Vigilant Spirit Model in IMPRINT

Changes from Baseline Model

The following is a detailed description of all of the alternative model's changes from the baseline.

Node 1: Spawn HVT

The only change with this node is the addition of another variable setting, where *SurvTimeLeft* is set to 59 seconds. The reason for this change is because of the increased complexity and recurrent nature of subsequent nodes.

Node 3: Search for HVT

This task is now performed by both the Operator and Automation instead of just the Operator, and has several changes. The key difference is that both the Operator and Automation search for the HVT independently of each other. The Operator's probability of finding the HVT and probability distribution of search time are unchanged, but the data is now captured in the human-specific variables *HmnFoundHVT* and *HmnSearchTime* instead of *FoundHVT* and *SearchTime*. The Automation's search time is captured by the system-specific variable *SysSearchTime*, and is determined by a Weibull distribution with a threshold of 4 and varying shape and scale parameters. For an explanation of the Weibull shape and scale parameters used for the Automation, see Appendix E. Any Weibull values generated that are greater than 58.56 seconds are rounded to equal 60 seconds. This is because if the Automation finds the HVT but the Operator does not have 0.44 seconds to confirm (discussed later in Node 4: Confirm HVT), or if the Automation never finds the HVT in an iteration, then the Automation continues searching into the next iteration. In that case, the system-specific variable *SysFoundHVT* is set to false. If the Automation finds the HVT, then *SysFoundHVT* is set to true. *SearchTime* and *FoundHVT* correspond to whoever's search time is the shortest: the Operator or Automation. If the Automation finds the HVT faster than the Operator, then *SearchTime* equals *SysSearchTime*, *FoundHVT* equals *SysFoundHVT*, and *HmnFaster* is set to false, where *HmnFaster* is a Boolean variable that tracks whether or not the Operator found the HVT before the Automation. Otherwise, *SearchTime*, *FoundHVT*, and *HmnFaster* are set accordingly to the human-specific values.

If *FoundHVT* is true, meaning if either the Operator or Automation found the HVT, then *SurvTimeLeft* is updated to subtract *SearchTime*. If the value of *SearchTime* is greater than *SurvTimeLeft*, then *SearchTime* is set to equal *SurvTimeLeft* and *SurvTimeLeft* is set to 0 seconds. If the HVT was not found, then *SearchTime* is set to equal *SurvTimeLeft* plus 1 second so that the Operator and Automation continue searching into the next iteration, and *SurvTimeLeft* is set to 0 seconds.

Node 21: Suggest HVT

This is a new task node and is performed by the Automation. It is modeled to convey that the Automation indicates to the Operator a suggested HVT, requesting confirmation. This happens virtually instantaneously, so the node has no task time or effects.

Node 4: Confirm HVT

This task node is modified from the baseline's Node 4: Indicate HVT Located. The first thing that happens in this node is a determination of whether or not the Automation correctly identified the HVT. This data is captured by the integer variable *SysWrong* with an initial value of 0. If there is no time left in the iteration (*SurvTimeLeft* = 0 seconds) or if the Operator found the HVT first (*HmnFaster* = true), then *SysWrong* is set to 0 (i.e. not applicable). If the Automation found the HVT first, its accuracy is set by a local random variable named *random* that creates a randomly-generated floating point number between 0 and 1. If *random* is less than a varying number (0.9, 0.8, or 0.7), then the Automation accurately identified the HVT and *SysWrong* is set to 2 (i.e. false). Else, the Automation was incorrect and *SysWrong* is set to 1 (i.e. true). For an explanation of the accuracy parameters used for the Automation, see Appendix E.

If the Operator found the HVT first, then the task time for this node is unchanged from the baseline. If the Automation found the HVT first, then 0.44 seconds is the value used instead of the baseline's 0.2 seconds. This is determined by IMPRINT's micromodel for choice reaction time, where it takes 0.24 seconds to choose between two possible alternatives. Once the Operator decides, it still takes 0.2 seconds to press a keyboard key to confirm or deny, adding to 0.44 total seconds.

The node now has tactical path logic. If there is no time left in the iteration (*SurvTimeLeft* = 0), the Operator found the HVT first (*HmnFaster* = true), or if the Automation accurately identified the HVT (*SysWrong* = 2), then the entity continues to Node 7: Follow HVT. Otherwise, the Operator and Automation must continue searching for the HVT, so the entity loops back to Node 3: Search for HVT.

Assumptions

As with the baseline model, the alternative model was created with several assumptions. These assumptions are described in Table 8 below.

Table 8: Alternative Model Assumptions

Assumption	Rationale
It takes the Automation 0 seconds to indicate to the Operator if it thinks it found the HVT.	In reality it would take a small amount of time for the Automation to process the information and send the command to indicate to the Operator. However, this time is assumed to be 0 seconds to simplify the model.
It always takes the Operator 0.24 seconds to decide whether the System found the HVT.	IMPRINT's micromodel for choice reaction time between two choices (confirm or deny) was chosen in order to simplify the model, though the actual time would vary per individual.
It always takes the Operator 0.2 seconds to press the F key to either indicate the HVT as found or to confirm whether the Automation found the HVT.	IMPRINT's micromodel for single finger keying rate (to press the F key) was chosen in order to simplify the model, though the actual time would vary per individual.
If the Automation suggests a HVT and the Operator denies it as correct, then both begin searching again using the same probability distributions for Search Times.	In reality the Operator's search efforts might be affected, consciously or subconsciously, by the distraction of the Automation's incorrect suggestion. However, the Operator is assumed to be unaffected in order to simplify the model.
The addition of Automation does not affect the probability distribution for the Operator's Calculate Answer time for the communications questions.	The probability distribution for the task timing is kept the same as the baseline's to avoid introducing subjectivity into the model by creating an arbitrary distribution.

Appendix E – Alternative Model Output Analysis

Overview

This appendix provides a detailed description of how the Improved Performance Research Integration Tool (IMPRINT) alternative model's output analysis was compared to the baseline model.

Introduction

Recall that the alternative model implemented an automated scanning algorithm designed to aid the operator in searching for the high value target. Six algorithm settings were chosen to analyze as trades, with automation accuracy and speed parameters varying per trade, listed in Table 9.

Table 9: Scanning Algorithm Accuracy and Speed Settings

	Accuracy	**Speed** (4+Weibull)			
		Shape	**Scale**	**Mean**	**Variance**
Trade 1	90%	8.5	25.5	28.084	11.399
Trade 2		6.4	23.5	25.879	15.955
Trade 3	80%	6.7	20	22.666	10.669
Trade 4		4	17	19.409	18.687
Trade 5	70%	3.2	10	12.957	9.438
Trade 6		1.7	7	10.246	14.301

The alternative model was run for each of these six tradeoffs and the outputs compared against the baseline. The goal was to successfully validate that the alternative tradeoff models provided significant increases in performance and decreases in user workload from the baseline. In order to test at a 90% confidence level if the alternative improved upon the baseline in both performance and workload, a Bonferonni Correction yielded the requirement to test each response variable individually at a 95% confidence level.

Performance Score

The tradeoff alternative models' performance scores were assessed via two methods: visual and statistical evaluation. As previously determined from the baseline model validation, the IMPRINT simulation was run 34 times for each of the six tradeoff alternative trials to collect the correct data sample size to compare with the baseline. The descriptive statistics for the tradeoff data were calculated, shown in Table 10.

Table 10: Tradeoff Data Statistics for Score

Trade 1: 90%, 4+Weibull(8.5, 25.5)		**Trade 2:** 90%, 4+Weibull(6.4, 23.5)	
n	34	n	34
Mean	668.981	Mean	678.57
Std Deviation	71.259	Std Deviation	71.439
Variance	5077.899	Variance	5103.560
Min	518.073	Min	489.533
Max	801.996	Max	817.605
Range	283.922	Range	328.072
Confidence Level	.95	Confidence Level	.95
Half-Width	24.864	Half-Width	24.926
Trade 3: 80%, 4+Weibull(6.7, 20)		**Trade 4:** 80%, 4+Weibull(4, 17)	
n	34	n	34
Mean	675.292	Mean	714.185
Std Deviation	82.309	Std Deviation	95.286
Variance	6774.828	Variance	9079.450
Min	528.215	Min	511.860
Max	796.587	Max	875.007
Range	268.372	Range	363.147
Confidence Level	.95	Confidence Level	.95
Half-Width	28.719	Half-Width	33.247
Trade 5: 70%, 4+Weibull(3.2, 10)		**Trade 6:** 70%, 4+Weibull(1.7, 7)	
n	34	n	34
Mean	779.081	Mean	822.092
Std Deviation	80.510	Std Deviation	108.436
Variance	6481.904	Variance	11758.288
Min	621.604	Min	544.017
Max	924.039	Max	972.681
Range	302.435	Range	428.664
Confidence Level	.95	Confidence Level	.95
Half-Width	28.091	Half-Width	37.835

The 95% confidence intervals for the baseline and tradeoff data were plotted and are displayed in Figure 31.

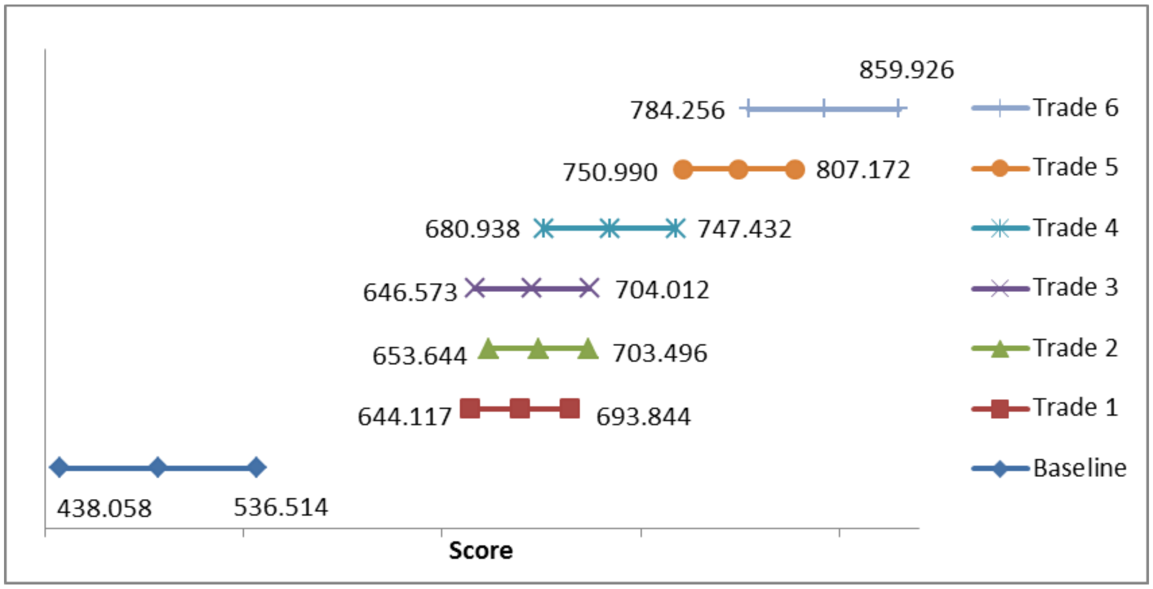

Figure 31: Confidence Intervals for Baseline/Tradeoff Models for Score

Upon visual evaluation of the confidence intervals, all six trades appeared to be significantly improved from the baseline. The goal was to have no point overlap between the baseline and alternative data sets, which all alternatives achieved. Trade 1 contained the lowest scores out of the six alternatives, and its interval's minimum value was still 108 points higher than the baseline's maximum value. Thus, it appeared that all tradeoff alternatives improved performance enough to be significantly different than the baseline.

Next, each trade's score data were statistically evaluated against the baseline. A one-way analysis of variance (ANOVA) was conducted to compare the effect of scan accuracy and speed on score in the baseline and each of the six trade conditions. The null hypothesis H_0 was defined such that there was no difference between the baseline and alternative data, while the alternative hypothesis H_A stated that there was a difference. The ANOVA found a significant effect of scan accuracy and speed on score at the $p<0.05$

level for the baseline and six trade conditions [F(6, 231) = 41.93, p = 0.000]. Thus H_0 was rejected, meaning there was sufficient evidence that at least one of the alternatives differed from the baseline. The results of the ANOVA are displayed in Table 11.

Table 11: One-Way ANOVA Data for Score

Source	DF	SS	MS	F	P
Base, Trades 1-6	6	2306807	384468	41.93	0.000
Error	231	2117999	9169		
Total	237	4424806			

Post hoc comparisons using Tukey's honest significant difference (HSD) test indicated that the mean scores for all six trades were significantly different than the baseline. These statistical observations confirm what was visually observed in Figure 31. However, it should be noted that although all six trades were statistically better than the baseline, they were not necessarily statistically different from each other. The mean, standard deviation, and relative groupings of scores for the baseline and six trades are shown in Table 12, where means that do not share a grouping letter are significantly different. Similarly, the Tukey differences of means for simultaneous 95% confidence intervals of the baseline and six trades are displayed in Figure 32, where mean intervals that do not contain zero are significantly different.

Table 12: Tukey's HSD Results for Baseline/Trades for Score

	Mean	Standard Deviation	Grouping			
Baseline	487.3	141.1	A			
Trade 1	669.0	71.3		B		
Trade 2	678.6	71.4		B		
Trade 3	675.3	82.3		B		
Trade 4	714.2	95.3		B	C	
Trade 5	779.1	80.5			C	D
Trade 6	822.1	108.4				D

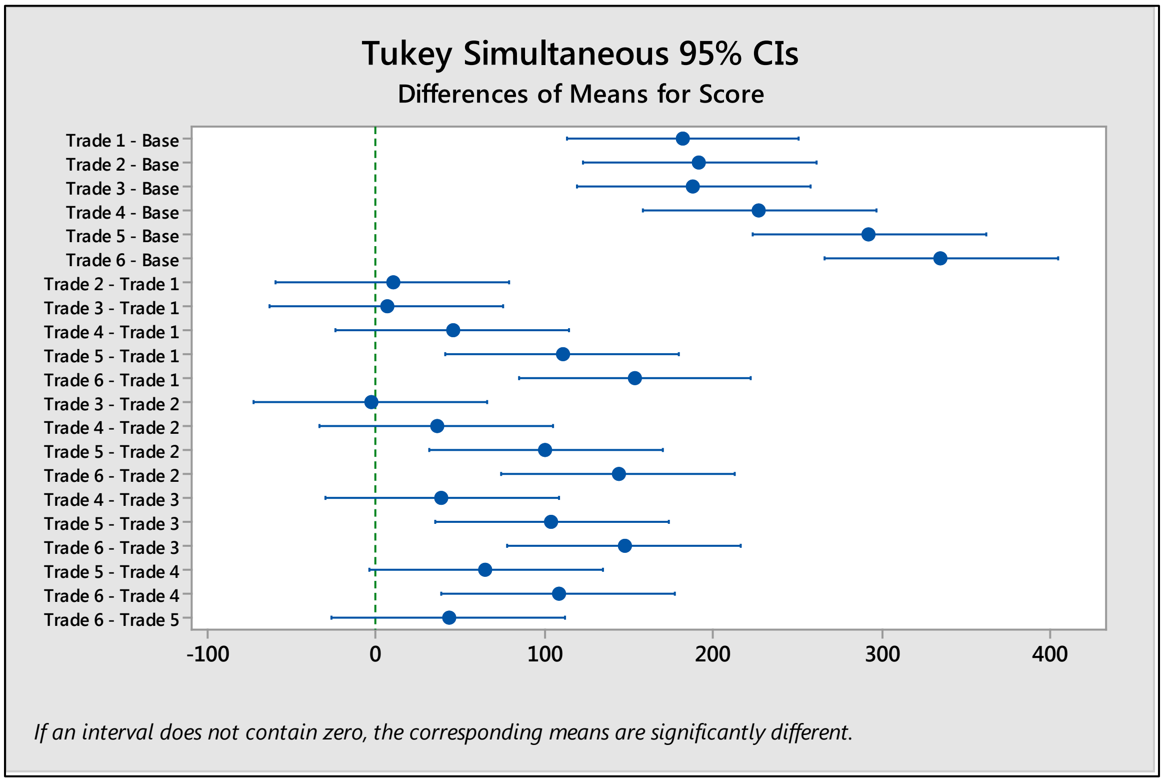

Figure 32: Tukey Differences of Means for 95% CI for Score

Workload

The alternative tradeoff models' workload values were also assessed visually and statistically. To do this, the IMPRINT baseline and six alternative trials were each run 34 times and a time-weighted average workload was derived for each run. The data sample size of 34 was chosen for consistency with the performance score data. From this data, the descriptive statistics were calculated and are shown in Table 13.

Table 13: Baseline/Tradeoff Data Statistics for Workload

Baseline	
n	34
Mean	20.116
Std Deviation	0.8062
Variance	0.6499
Min	18.973
Max	22.399
Range	3.426
Confidence Level	.95
Half-Width	0.2813

Trade 1: 90%, 4+Weibull(8.5, 25.5)		**Trade 2:** 90%, 4+Weibull(6.4, 23.5)	
n	34	n	34
Mean	19.645	Mean	19.693
Std Deviation	0.5985	Std Deviation	0.6677
Variance	0.3582	Variance	0.4459
Min	18.984	Min	18.960
Max	21.120	Max	21.545
Range	2.136	Range	2.585
Confidence Level	.95	Confidence Level	.95
Half-Width	0.2088	Half-Width	0.2330
Trade 3: 80%, 4+Weibull(6.7, 20)		**Trade 4:** 80%, 4+Weibull(4, 17)	
n	34	n	34
Mean	19.827	Mean	19.751
Std Deviation	0.7775	Std Deviation	0.8328
Variance	0.6045	Variance	0.6936
Min	18.826	Min	18.743
Max	21.549	Max	21.504
Range	2.723	Range	2.761
Confidence Level	.95	Confidence Level	.95
Half-Width	0.2713	Half-Width	0.2906
Trade 5: 70%, 4+Weibull(3.2, 10)		**Trade 6:** 70%, 4+Weibull(1.7, 7)	
n	34	n	34
Mean	19.483	Mean	19.425
Std Deviation	0.7042	Std Deviation	0.6043
Variance	0.4959	Variance	0.3652
Min	18.577	Min	18.503
Max	21.468	Max	21.211
Range	2.891	Range	2.708
Confidence Level	.95	Confidence Level	.95
Half-Width	0.2457	Half-Width	0.2108

The 95% confidence intervals for the baseline and alternative tradeoff data were plotted and are displayed in Figure 33.

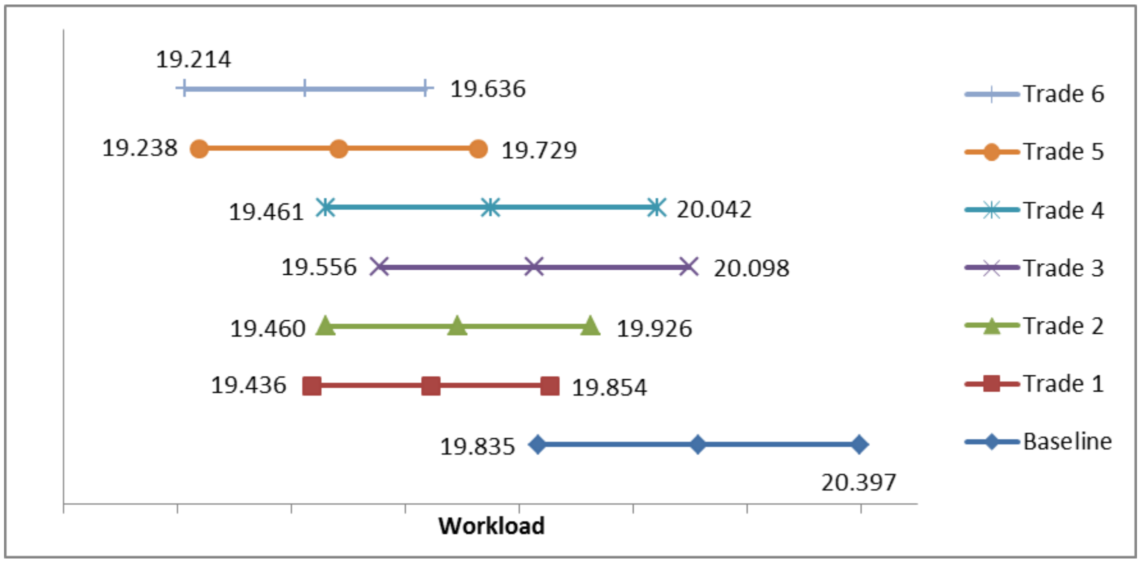

Figure 33: Confidence Intervals for Baseline/Tradeoff Models for Workload

Upon visual evaluation of the confidence intervals, it was observed that all the tradeoffs' interval values were less than the baseline's, showing that workload was decreased. However, Trades 1-4's values all overlapped the baseline by varying amounts. Only Trades 5 and 6 had intervals that did not overlap the baseline. Thus, it appeared that Trades 5 and 6 were the only two alternatives to have significantly improved workload values.

As with the performance score data, each trade's workload data were statistically evaluated against the baseline. A one-way ANOVA was also conducted to compare the effect of scan accuracy and speed on workload in the baseline and each of the six trade conditions. The null hypothesis H_0 was defined such that there is no difference between the baseline and alternative data, while the alternative hypothesis H_A stated that there was a difference. The ANOVA found a significant effect of scan accuracy and speed on workload at the $p<0.05$ level for the baseline and six trade conditions [F(6, 231) = 3.48, p = 0.003]. Thus H_0 was rejected, meaning there was sufficient evidence that at least one of the alternative models differed from the baseline. The results of the ANOVA are displayed in Table 14.

Table 14: One-Way ANOVA Data for Workload

Source	DF	SS	MS	F	P
Base, Trades 1-6	6	10.79	1.7984	3.48	0.003
Error	231	119.24	0.5162		
Total	237	130.03			

Post hoc comparisons using Tukey's HSD test indicated that the mean workload for Trades 5 and 6 were significantly different than the baseline. However, Trades 1-4 did not significantly differ from the baseline or each other. These statistical observations corroborate what was visually observed in Figure 33, showing that Trades 5-6 are the only two alternatives to be statistically different than the baseline. The mean, standard deviation, and relative groupings of workload for the baseline and six trades are shown in Table 15, where means that do not share a grouping letter are significantly different. The Tukey differences of means for simultaneous 95% confidence intervals of the baseline and six trades are displayed in Figure 34, where mean intervals that do not contain zero are significantly different.

Table 15: Tukey's HSD Results for Baseline/Trades for Workload

	Mean	Standard Deviation	Grouping	
Baseline	20.116	0.806	A	
Trade 1	19.645	0.598	A	B
Trade 2	19.693	0.668	A	B
Trade 3	19.827	0.777	A	B
Trade 4	19.751	0.833	A	B
Trade 5	19.483	0.704		B
Trade 6	19.425	0.604		B

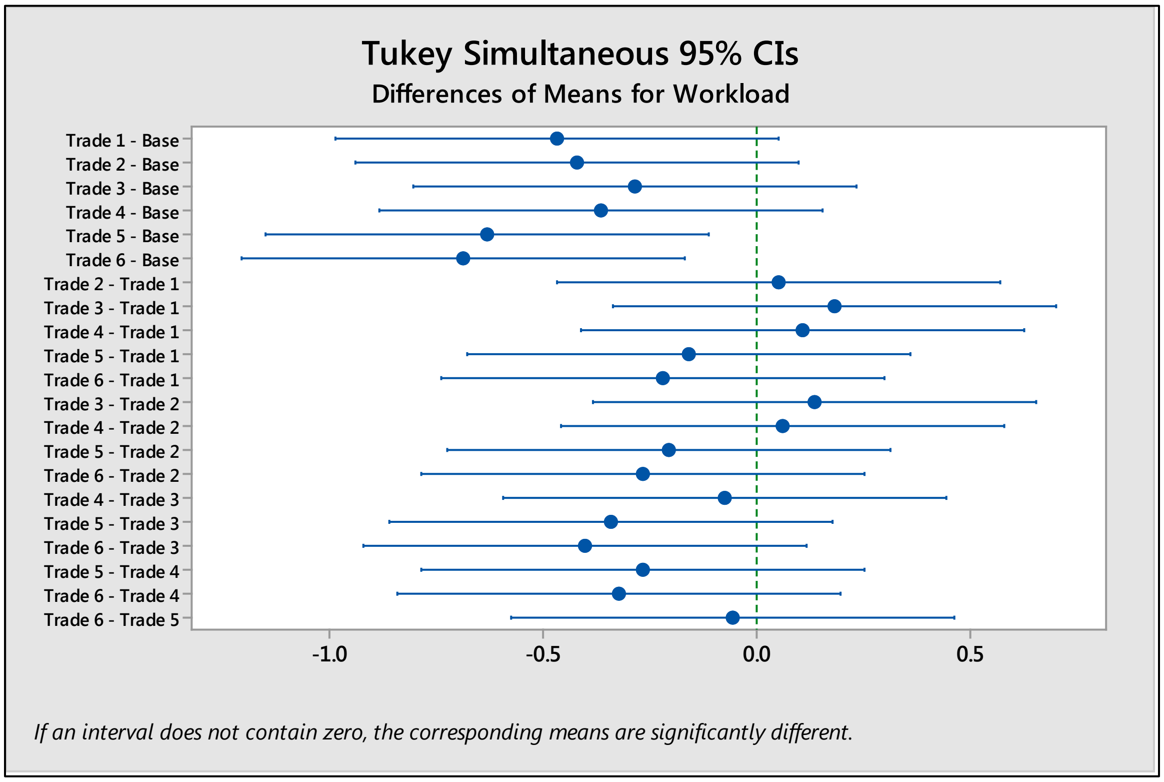

Figure 34: Tukey Differences of Means for 95% CI for Workload

Bibliography

Ahram, T., & Karwowski, W. (2009). Human Systems Integration Modeling Using Systems Modeling Language. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (pp. 1849-1853). San Antonio: SAGE.

Allender, L. (2000). Modeling Human Performance: Impacting System Design, Performance, and Cost. *Proceedings of the Military, Government and Aerospace Simulation Symposium, 2000 Advanced Simulation Technologies Conference*, (pp. 139-144). Washington D.C.

Bierbaum, C. R., Szabo, S. M., & Aldrich, T. B. (1989). *Task Analysis of the UH-60 Mission and Decision Rules for Developing a UH-60 Workload Prediction Model, Volume 1: Summary Report.* Fort Rucker: U.S. Army Research Institute.

Bodenhamer, A. (2012). Adaptations in the US Army MANPRINT process to utilize HSI-inclusive systems architectures. *Procedia Computer Science 8*, 249-254.

Boy, G. A., & Narkevicius, J. M. (2013). Unifying Human Centered Design and Systems Engineering for Human Systems Integration. *Proceedings of the 4th International Conference on Complex Systems Design and Management* (pp. 151-162). Paris: Springer.

Bruseberg, A. (2008). Human Views for MODAF as a Bridge Between Human Factors Integration and Systems Engineering. *Journal of Cognitive Engineering and Decision Making*, 220-248.

Cassenti, D. N., Kelley, T. D., Colle, H. A., & McGregor, E. A. (2011). Modeling Performance Measures and Self-Ratings of Workload in a Visual Scanning Task. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (pp. 870-874). Las Vegas: SAGE.

Chua, Z. K., & Feigh, K. M. (2011). Integrating Human Factors Principles into Systems Engineering. *Digital Avionics Systems Conference Proceedings* (pp. 6A11-6A111). Seattle: IEEE.

Cloutier, R., Sauser, B., Bone, M., & Taylor, A. (2015). Transitioning Systems Thinking to Model-Based Systems Engineering: Systemigrams to SysML Models. *IEEE Transactions on Systems, Man, and Cybernetics*, 662-674.

Colombi, J. M., Miller, M. E., Schneider, M., McGrogan, J., Long, D. S., & Plaga, J. (2011). Predictive Mental Workload Modeling for Semiautonomous System Design: Implications for Systems of Systems. *Systems Engineering*, 448-460.

Crane, J., & Brownlow, L. (2015). Optimization of Multi-Satellite Systems Using Integrated Model Based System Engineering (MBSE) Techniques. *IEEE International Systems Conference* (pp. 206-211). Vancouver: IEEE.

Crisp, H. E., Hoang, N. T., Karangelen, N. T., & Britton, D. A. (2000). An Integrated Tools Environment for Human Centered Design of Complex Systems. *Proceedings of SPIE - The International Society for Optical Engineering* (pp. 155-163). San Diego: SPIE.

Delligatti, L. (2014). *SysML Distilled: A Brief Guide to the Systems Modeling Language.* Upper Saddle River: Addison-Wesley.

Department of Defense. (2009). *DoDAF V2.0, Vol 2: Architectural Data and Models.* Washington, DC.

Department of Defense. (2013). *Defense Acquisition Guidebook.* Washington, DC.

Department of Defense. (2015). *DoD Instruction 5000.02: Operation of the Defense Acquisition System.* Washington, DC.

Dickason, D., Sargent, B., & Bagnall, T. (2009). *Investigating the Incorporation of Personality Constructs into IMPRINT.* Millington: Navy Personnel Research, Studies, and Technology.

Do, Q., Cook, S., & Lay, M. (2014). An investigation of MBSE practices across the contractual boundary. *Conference on Systems Engineering Research* (pp. 692-701). Redondo Beach: Elsevier B.V.

Forsberg, K., Mooz, H., & Cotterman, H. (2005). *Visualizing Project Management, 3rd ed.* Hoboken: John Wiley & Sons, Inc.

Friedenthal, S., Moore, A., & Steiner, R. (2014). *A Practical Guide to SysML: The Systems Modeling Language, 3rd ed.* Waltham: Morgan Kaufmann OMG Press.

Handley, H. A. (2011). Incorporating the NATO Human View in the DoDAF 2.0 Meta Model. *Systems Engineering*, 108-117.

Handley, H. A., & Smillie, R. J. (2008). Architecture Framework Human View: The NATO Approach. *Systems Engineering,* 156-164.

Handley, H. A., & Smillie, R. J. (2009). Human View Dynamics - The NATO Approach. *Systems Engineering*, 72-79.

Handley, H., & Knapp, B. (2014). Where are the People? The Human Viewpoint Approach for Architecting and Acquisition. *Defense Acquisition Research Journal*, 852-874.

Hardman, N., & Colombi, J. (2012). An Empirical Methodology for Human Integration in the SE Technical Processes. *Systems Engineering*, 172-190.

Hardman, N., Colombi, J., Jacques, D., & Miller, J. (2008). Human Systems Integration within the DoD Architecture Framework. *IIE Annual Conference and Expo* (pp. 840-845). Vancouver: IIE.

Harriott, C. E., Zhang, T., & Adams, J. A. (2013). Assessing Physical Workload for Human-Robot Peer-Based Teams. *International Journal of Human Computer Studies,* 821-837.

Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In P. A. Hancock, & N. Meshkati (Eds.), *Human Mental Workload* (pp. 139-183). Amsterdam: Elsevier.

Hawley, J. K. (2011). Not By Widgets Alone: The Human Challenge of Technology-Intensive Military Systems. *Armed Forces Journal*, 24-28, 41.

Hoepf, M., Middendorf, M., Epling, S., & Galster, S. (2015). Physiological Indicators of Workload in a Remotely Piloted Aircraft Simulation. *18th International Symposium on Aviation Psychology* (pp. 428-433). Dayton: Curran Associates, Inc.

Hopcroft, R., Burchat, E., & Vince, J. (2006). *Unmanned Aerial Vehicles for Maritime Patrol: Human Factors Issues.* Victoria: Defence Science and Technology Organisation.

International Council on Systems Engineering. (2015). *Systems Engineering Handbook.* Hoboken: John Wiley & Sons, Inc.

International Ergonomics Association. (2016). *Definition and Domains of Ergonomics*. Retrieved from International Ergonomics Association: http://www.iea.cc/whats/

Kaslow, D., Soremekun, G., Kim, H., & Spangelo, S. (2014). Integrated Model-Based Systems Engineering (MBSE) Applied to the Simulation of a CubeSat Mission. *IEEE Aerospace Conference* (pp. 1-14). Big Sky: IEEE Computer Society.

Madni, A. M. (2009). Integrating Humans with Software and Systems: Technical Challenges and a Research Agenda. *Systems Engineering*, 232-245.

Malik, T. (2015, July 28). *Deadly SpaceShipTwo Crash Caused by Co-Pilot Error: NTSB*. Retrieved from Space.com: http://www.space.com/30073-virgin-galactic-spaceshiptwo-crash-pilot-error.html

Martin, J. N. (1996). *Systems Engineering Guidebook: A Process for Developing Systems and Products.* Boca Raton: CRC Press.

Mitchell, D. K. (2003). *Advanced Improved Performance Research Integration Tool (IMPRINT) Vetronics Technology Test Bed Model Development.* Aberdeen Proving Ground: Army Research Laboratory Human Research & Engineering Directorate.

Mitchell, D. K. (2005). Enhancing System Design by Modeling IMPRINT Task Workload Analysis Results in the Unified Modeling Language (UML). *Proceedings of the ASME Human Systems Integration Symposium.* Arlington.

Mitchell, D. K. (2008). United We Stand: Using Multiple Tools to Solve a Multi-dimensional Problem. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (pp. 1939-1943). New York: SAGE.

Mitchell, D. K., & Chen, J. Y. (2006). Impacting System Design with Human Performance Modeling and Experiment: Another Success Story. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (pp. 2477-2481). San Francisco: SAGE.

Mitchell, D. K., & McDowell, K. (2008). Using Modeling as a Lens to Focus Testing. *International Symposium on Collaborative Technologies and Systems* (pp. 477-482). Irvine: IEEE.

Mitchell, D. K., Agan, K., & Samms, C. (2011). Both Sides of the Coin: Technique for Integrating Human Factors and Systems Engineering in System Development. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (pp. 2025-2029). Las Vegas: SAGE.

Mitchell, D. K., Samms, C. L., Henthorn, T., & Wojciechowski, J. Q. (2003). *Trade Study: A Two- Versus Three-Soldier Crew for the Mounted Combat System (MCS) and Other Future Combat Systems Platforms.* Aberdeen Proving Ground: Army Research Laboratory, Human Research & Engineering Directorate.

Mitchell, D. K., Samms, C., & Wojcik, T. M. (2006). System-of-systems Modeling: The Evolution of an Approach for True Human System Integration. *15th Conference on Behavior Representation in Modeling and Simulation* (pp. 67-74). Baltimore: SISO.

Norman, D. A., & Draper, S. W. (1986). *User Centered System Design: New Perspectives on Human-Computer Interaction.* Hillsdale: Lawrence Erlbaum Associates.

Orellana, D. W., & Madni, A. M. (2014). Human System Integration Ontology: Enhancing Model Based Systems Engineering to Evaluate Human-System Performance. *Procedia Computer Science 28,* 19-25.

Piaszczyk, C. (2011). Model Based Systems Engineering with Department of Defense Architectural Framework. *Systems Engineering*, 305-326.

Ramos, A. L., Ferreira, J. V., & Barcelo, J. (2013). LITHE: An Agile Methodology for Human-Centric Model-Based Systems Engineering. *IEEE Transactions on Systems, Man, and Cybernetics: Systems and Humans*, 504-521.

Rashid, M., Anwar, M. W., & Khan, A. M. (2015). Toward the Tools Selection in Model Based System Engineering for Embedded Systems - A Systematic Literature Review. *The Journal of Systems and Software,* 150-163.

Rogers, Y., Sharp, H., & Preece, J. (2011). *Interaction Design: Beyond Human-Computer Interaction, 3rd ed.* Chichester: John Wiley & Sons Ltd.

Rusnock, C. F., & Geiger, C. D. (2014). Simulation-based Assessment of Performance-Workload Tradeoffs for System Design Evaluation. *IIE Annual Conference and Expo* (pp. 394-403). Montreal: Institute of Industrial Engineers.

Russell, M. (2012). Using MBSE to Enhance System Design Decision Making. *Conference on Systems Engineering Research* (pp. 188-193). St Louis: Elsevier Ltd.

Sharples, R. A. (2014). Introduction of Human Views into Operational Capability Development within an Architectural Framework. *IEEE International Systems Conference* (pp. 325-329). Ottawa: IEEE Computer Society.

Smillie, R., & Handley, H. (2009). Human-Centered Design Focus in Systems Engineering Analysis: Human View Methodology. *Proceedings of the International Conference on Contemporary Ergonomics* (pp. 310-319). London: Taylor & Francis.

Thompson, R. E., Colombi, J. M., Black, J., & Ayres, B. J. (2015). Disaggregated Space System Concept Optimization: Model-Based Conceptual Design Methods. *Systems Engineering*, 549-567.

U.S. Air Force. (2010). *Air Force Human Systems Integration Handbook.*

U.S. Nuclear Regulatory Commission. (2014, December 12). *Backgrounder on the Three Mile Island Accident*. Retrieved from United States Nuclear Regulatory Commission: http://www.nrc.gov/reading-rm/doc-collections/fact-sheets/3mile-isle.html

Wickens, C., Bagnall, T., Gosakan, M., & Walters, B. (2012). A Cognitive Model of the Control of Unmanned Aerial Vehicles. *International Symposium on Aviation Psychology* (pp. 535-540). Dayton: Curran Associates, Inc.

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 24-03-2016	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From – To)* October 2014 – March 2016
TITLE AND SUBTITLE Improving System Design Through the Integration of Human Systems and Systems Engineering Models		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Watson, Michael E., Captain, USAF		**5d. PROJECT NUMBER**
		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 Way, Building 640 WPAFB OH 45433-7765		**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENV-MS-16-M-190
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Human Research and Engineering Directorate Army Research Laboratory Christopher Best 2800 Powder Mill Road Adelphi, MD 20783-1138 christopher.j.best17.civ@mail.mil		**10. SPONSOR/MONITOR'S ACRONYM(S)** ARL/HRED
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

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

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

14. ABSTRACT
The human is a critical aspect of many systems, but frequently there is a failure to properly account for human capabilities and involvement during system design. This inattention results in systems with higher lifecycle costs, decreased user compatibility, and the potential to produce disastrous consequences. This research presents an approach to integrating the human into system models by using two methods: static and dynamic modeling. The static method uses a user-centered design framework to create system- and human-centered models that deconstruct the system and user into their respective components. These models are integrated to create system models that include relevant information about the human and highlight potentially conflicting tasks. The dynamic method uses a human performance modeling tool to create a discrete event simulation (DES) of the system. This DES model is used to perform an analysis between system trades, by which constraints and assumptions placed on the human are verified. Data gained from the analysis are integrated back into system models in order to reflect true system performance. By applying these two integration methods early in the system's lifecycle, system models can more effectively account for the human as a critical component of the system, thus improving system design.

15. SUBJECT TERMS
SE, MBSE, SysML, HSI, human factors, UCD, human performance modeling, IMPRINT, tradeoffs

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Maj Christina F. Rusnock, AFIT/ENV
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	175	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636 x4611 (christina.rusnock@afit.edu)

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