2006

A Holistic Usability Framework For Distributed Simulation Systems

Jeffrey Dawson
University of Central Florida

A HOLISTIC USABILITY FRAMEWORK FOR DISTRIBUTED SIMULATON SYSTEMS

by

JEFFREY WATSON DAWSON
B.S.I.E., University of Tennessee, 1982
M.B.A., University of Central Florida, 1994
M.S.I.E., University of Central Florida, 2003

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Industrial Engineering and Management Systems
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2006

Major Professor: Luis C. Rabelo

ABSTRACT

This dissertation develops a holistic usability framework for distributed simulation systems (DSSs). The framework is developed considering relevant research in human-computer interaction, computer science, technical writing, engineering, management, and psychology. The methodology used consists of three steps: (1) framework development, (2) surveys of users to validate and refine the framework, and to determine attribute weights, and (3) application of the framework to two real-world systems. The concept of a holistic usability framework for DSSs arose during a project to improve the usability of the Virtual Test Bed, a prototypical DSS, and the framework is partly a result of that project. In addition, DSSs at Ames Research Center were studied for additional insights. The framework has six dimensions: end user needs, end user interface(s), programming, installation, training, and documentation. The categories of participants in this study include managers, researchers, programmers, end users, trainers, and trainees. The first survey was used to obtain qualitative and quantitative data to validate and refine the framework. Attributes that failed the validation test were dropped from the framework. A second survey was used to obtain attribute weights. The refined framework was used to evaluate two existing DSSs, measuring their holistic usabilities.

Ensuring that the needs of the variety of types of users who interact with the system during design, development, and use are met is important to launch a successful system. Adequate consideration of system usability along the several dimensions in the framework will not only ensure system success but also increase productivity, lower life cycle costs, and result in a more pleasurable working experience for people who work with the system.

ACKNOWLEDGMENTS

I thank my committee members: Dr. Malone, for honing my statistical skills; Dr. Rabelo, for insisting on high quality; Dr. Remington, for his helpful insights; Dr. Resnick, for providing valuable guidance on usability; Dr. Sepulveda, for teaching me advanced simulation; and Dr. Wang, for his advice. This work could not have been accomplished without the help of a large number of people. I thank my colleagues who worked with me on the VTB project: Ping Chen, Fred Gruber, Yanshen Gui, Ethling Hernandez, Mario Marin, and Serge Sala-Diakanda, all of whom I hold in high regard. I am grateful to all the experts around the world who took my surveys and provided useful commentary, to the many kind people at NASA Ames Research Center, who provided a warm welcome and insightful forum for my research, to the many people at Embry-Riddle University who overwhelmed me with their willingness to help and allowed me to do whatever was necessary to accomplish my work, to the industry practitioners and vendors who took time out of their busy schedules to provide feedback, and to many people unnamed. I am grateful for the kind help of the staff in the Industrial Engineering and Management Systems department. I thank my parents for teaching me to think independently. I especially thank Dawn, for her support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

ANSI	American National Standards Institute
API	application programming interface
ATC	Air Traffic Control
ARTT	Aviation Research Training Tool
CIF	Common Industry Format
Dod	Department of Defense
DSS	Distributed Simulation System
DVA	data visualization and analysis
GUI	Graphical User Interface
HCI	human-computer interaction
HE	heuristic evaluation
HIP	human information processing
HLA	High Level Architecture
I	infrastructure
IDE	integrated development environment
IRB	institutional review board
ISO	International Standards Organization
JRD3C	Joint Rapid Distributed Database Development Capability
K-LM	keystroke-level model
M&S	modeling and simulation
MOT	metaphors of thinking

MUS	Master Usability Scaling
MUSiC	Metrics for Usability Standards in Computing
N/A	not applicable
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
OMG	Object Management Group
PCI	programming, configuration, and installation
PDS	parallel and distributed simulation
QUIS	Questionnaire for User Interaction Satisfaction
RTI	Run Time Infrastructure
RUP	Rational Unified Process
SUMI	Software Usability Measurement Inventory
SUS	System Usability Scale
T	training
UCID	User-Centered Information Design
UCF	University of Central Florida
UID	user interface design
UME	Usability Magnitude Estimation
UML	Unified Modeling Language
UN	user needs
VAST	Virtual Airspace Simulation Technology
VLAB	Virtual Laboratory
VMS	Virtual Motion Laboratory

VTB	Virtual Test Bed
VV&A	validation, verification, and accreditation
XMSF	Extensible Modeling and Simulation Framework

CHAPTER ONE: INTRODUCTION

Research Problem

Designing and building a distributed simulation system (DSS) is a major undertaking requiring much work from experts in a variety of disciplines. The ultimate quality of the system depends on how well the system meets the needs of the users and how easy the system is to use. Ease of system use is often forgotten in the design and implementation phases. A framework for the usability of DSSs is needed in order to help ensure good usability. While this framework will have the obvious benefit of making the system easier and more pleasurable to use, consideration of usability in the system design will also have a significant impact in reducing system lifecycle cost. The application of a framework for the usability of DSSs has the benefits of (1) improved user experience and productivity, (2) a higher probability of system success, and (3) lowered system lifecycle costs for design, development, operation, and maintenance.

The viewpoint taken for this framework is from a high level looking at all the aspects of the system that affect numerous types of users who work with the system. From this holistic viewpoint many issues are observed that, if gone into detail, would result in separate research studies for each item.

Ensuring that the needs of the various types of users who interact with the system during design, development, and use are met is important to launch a successful system. The framework's research contributions include the development of the holistic usability framework for DSSs, which is a new approach to usability; the development of a methodology to measure

holistic usability given the framework; and the synthesis of the knowledge of various fields needed in creating the framework.

Usability

Usability is the art and science of making systems and products that are easy to use and that people like to use. While much of the effort and literature devoted to usability has been focused on human-computer interfaces, a considerable effort has also been devoted to products in general. Indeed, every product or system that a person uses has aspects that are usability related, from the instruction manual to the ergonomics to the cognitive load put on the person's mind to use the system. Success or failure of a product, in the marketplace or on the battlefield under stress, depends on its usability.

For systems that are already designed, a standardized format for usability evaluation has been developed which defines usability as a system's effectiveness, efficiency, and level of user satisfaction. Effectiveness is defined as whether or not the user of a system can successfully accomplish desired tasks. Efficiency can be defined as a system's learnability and memorability. The time to accomplish a specified goal is often used as a measure of efficiency. Learnability is how easy it is to learn how to use a system. Memorability is how easy it is to remember how to use a system. User satisfaction is a measure of how much a system's features and interface please users. While these basic measures provide quantifiable yardsticks, there is much more to usability than these three items.

Usability analysis is sometimes performed as an afterthought after a system is designed, but it is best performed in a concurrent engineering or product development environment, as an

integral part of the design cycle. The money spent on usability engineering usually pays returns many times more than the investment required for the analysis.

Usability needs to be considered in the design of any system. As the cost and complexity of a system increases, so does the risk of failures in the deployment of the system and in its user interface. Aside from the possibility of total project failure, there are costs associated with poor usability; these costs occur during system design, deployment, and over the life cycle of the system. For example, Mayhew and Mantei (1994) give a detailed example of where a project to improve the usability of a workplace application costs $132,185, but results in a savings due to usability improvement of $209,490; in addition, the benefits of improved usability accrue yearly. A holistic usability framework that can be used both as an aid to DSS designers and as a means to evaluate existing systems is an important contribution to the toolbox we have to design and use DSSs. The development, refinement, and application of this framework is the subject of this dissertation.

Distributed Simulation

Distributed simulation is simulation that takes place using more than one program running simultaneously and to a certain extent independently, usually on more than one computer. The computers can be in the same room or geographically dispersed. While the rationale for using more than one computer will at times be to take advantage of the power (and potential cost reduction) distributed computing offers, the ability to interoperate and reuse a variety of simulation systems—new and legacy—is also important. A large number of configurations and purposes of DSSs exist, ranging from human factors research, virtual

environments, entertainment, military research, marketing studies, business studies, and financial analysis. DSSs may involve discrete, real-time, human-in-the-loop, continuous, and/or hybrid simulation. They are usually complex and require high levels of expertise to use; however, if simplicity of user interfaces were a design goal for DSSs, ease of use could be improved.

As an example of a DSS, consider the Vertical Motion Simulator (VMS) at the NASA Ames Research Center. This system can be configured to provide a virtual simulation of any aircraft cockpit, complete with real hardware, head-up displays, instruments with simulated data, video and audio streams, and motion. Its most important use is in training astronauts whose job is to fly the space shuttle. While the astronauts in the cockpit experience a realistic training experience, people watch them in the Virtual Lab (VLAB).

The VLAB has a room full of equipment, computer monitors and interfaces, and chart recorders. Different aspects of the simulation are displayed on separate monitors. In this room, researchers interface with the pilots flying the simulator and monitor, record, and study data. A three-dimensional, nonimmersive, virtual view of this room is transmitted to other NASA centers. Researchers at other NASA centers have the capability to zoom around the virtual VLAB using a two- or three-dimensional cursor via joystick or keyboard commands, view the cockpit mockup and the simulated view the pilot is seeing, select a number of displays to enlarge into windows on the screen, and travel in the Virtual Lab going to places such as a white board, where messages are shared between centers. This example shows distributed simulation used to monitor subjects in a simulator, while also providing a virtual view to remote researchers.

The simulations in DSSs can be synchronized to the same real time clock, although even when each is simulating a system in the same time period they can lead or lag each other; these deviations from absolute synchronicity are handled by the software infrastructure used to link

them together. Data flows between the independent simulations via the exchange of messages that may be synchronous or asynchronous.

A typical use of DSSs is in decision making. By studying how different processes interact, and looking at alternative scenarios, one can obtain information to help make difficult decisions. One can also study how humans interact with systems to determine how best to design the systems for use with people (e.g., air traffic control [ATC] systems). In such a case actual people may be in the loop, performing their roles as system users; as an alternative, hardware can be used to simulate the actions people would make if they were in the loop.

The Usability of Distributed Simulation Systems

When looking at the usability of DSSs, one becomes aware that a large team effort is underway. Each member of the team can be considered a user in some way. Indeed, from a managerial perspective, the usability may depend on the resources required to maintain the team who uses the system and how well the team can work with the system to accomplish stated goals. From a researcher's perspective, usability may be how easy it is to obtain the desired data and how good the data are. From a maintainer's perspective, usability may reflect how easy the system is to maintain.

The usability of a DSS is not simply the usability of its user interfaces. Given a system, these attributes are important:

- Design of its components' user interfaces (graphical and/or command line)
- Methodology of starting each component and the overall simulation

- Usability for each user; researchers, who use the simulation to obtain data; programmers who may be needed to integrate various simulations with different interfaces; operators, who are required to start, stop, and offer real time troubleshooting for the simulation runs; human participants, who may be required to play a support role during the simulation (e.g., acting as pilots during an ATC simulation), in addition to users who may be experimental subjects during a simulation or trainees in a simulated environment
- Usability, from a programming and project management viewpoint, of the software construction, methods, platforms and programming language(s) used to create the simulation system/models
- Usability for maintenance and administration of the system
- Ease of upgrading the hardware
- Ease of interconnectivity of the network infrastructure(s) required to run the simulation
- Ease of integrating legacy systems
- Troubleshooting support for when things go wrong
- Ease of maintenance timing, data synchronicity, or pseudosynchronicity
- Training requirements for those who use and support the system

This list goes beyond an analysis of traditional graphical user interface (GUI) usability as usability of a product. However, if one considers the accepted usability measures: efficiency, effectiveness, and user satisfaction, all of the above list affects the usability of the DSS. A DSS has multiple users at several levels of system interaction. Figure 1 below shows the human users of a DSS and their unified modeling language (UML) use cases (users on the left are termed "initiating actors"; users on the right are termed "receiving actors"). This use case diagram is not

exhaustive; there are many types and purposes of DSSs, each of which has a variety of configurational options. In any given system, who performs each role may be clear, although in an ideal system interface flexibility would allow for dynamic roles.

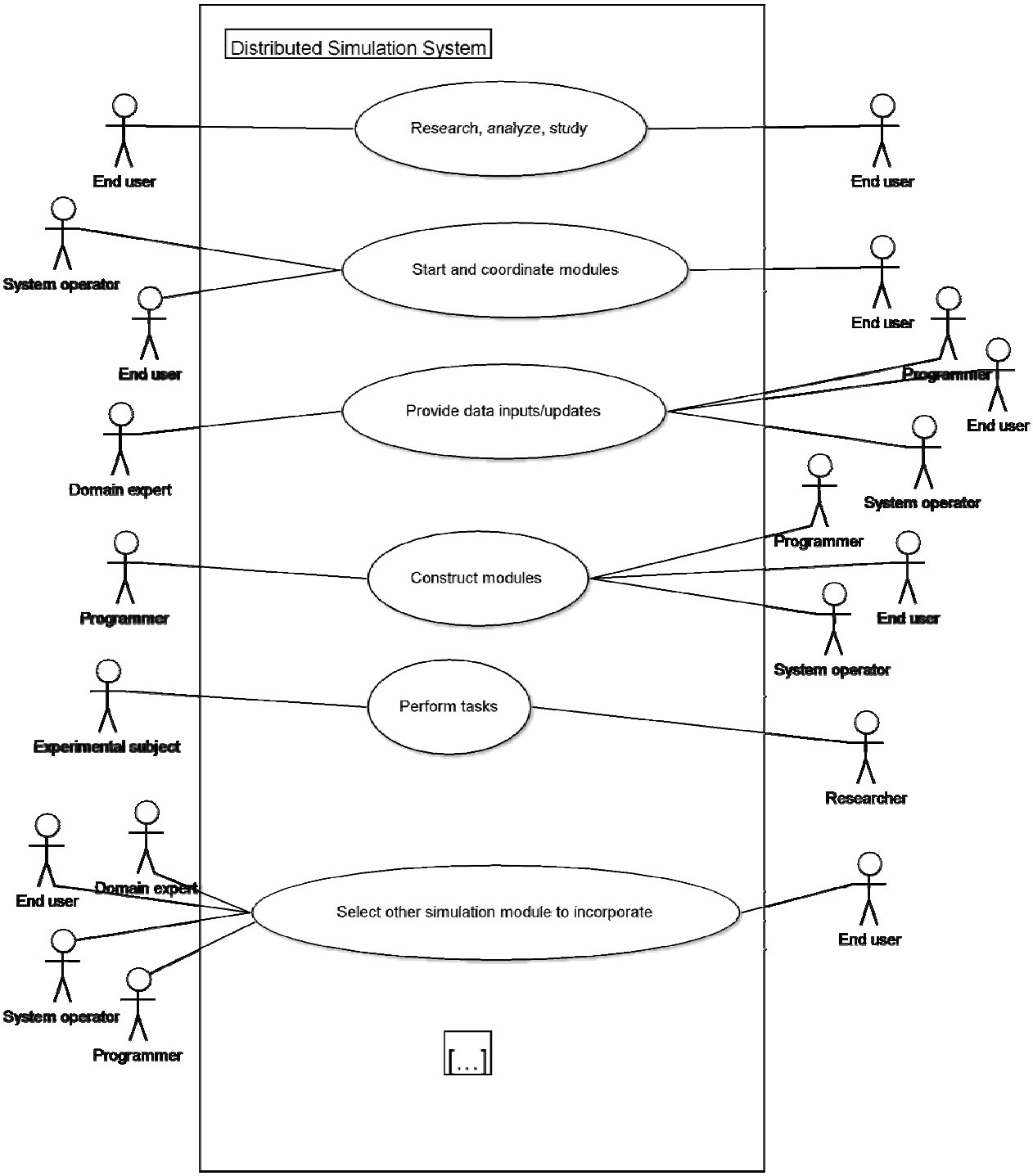

Figure 1. Top-level Use Case Diagram for a Distributed Simulation System

In a concurrent engineering environment, the application of usability principles throughout iterative design cycles will enhance the usefulness of the system and reduce its cost of operation. Developers of DSSs sometimes "get it right" due to decades of cumulative experience, iterative design, and high levels of expertise. There is no one, best way to design DSSs, which are often constructed for ad hoc purposes. A set of usability guidelines, however, will offer a reference and starting point for system developers and a means of assessing the overall usability of a DSS.

This research contributes to the existing body of knowledge in several ways. Taking a holistic viewpoint of the usability of a large, complex system, considering not just the end users but the designers, builders, and maintainers, is a new approach. Although a DSS is the subject, it could also be a paperless engineering design system, a global e-commerce infrastructure, or a stock exchange's information technology system. The framework can be used as a design tool that will help ensure that key holistic usability areas are considered; designers often suffer from "tunnel vision" as they focus on their subtask, pressured by schedules, narrow requirements, and a lack of knowledge of key user needs. Another area where the framework can be used is in performing comparative design studies, assessing various system configurations. The ability to measure holistic usability along several dimensions to make an overall assessment provides a quantitative measure of a DSS's holistic usability. This research contribution will allow for the measurement of not only an already-designed system's holistic usability but can also be applied to a system as it is being designed, so that improvements can be made. The use of the framework as a design tool will lead to lower life cycle costs, higher productivity, and increased user satisfaction for all people who interface with the system in its design, development, maintenance, and usage.

Chapter Organization

This dissertation is arranged as follows. The general topic is introduced in this introduction, as are general concepts of usability, simulation, and distributed simulation. Chapter 2 provides a literature review of relevant aspects of usability and simulation. Chapter 3 discusses the methodology used and introduces the holistic usability framework for distributed simulation. Chapter 4 discusses a survey of distributed simulation experts that is used to obtain feedback to validate and refine the framework. Chapter 5 shows the application of the framework to two existing systems and a technique to measure the usability of DSSs. Chapter 6 discusses the results from a second survey, independent of the first, that determines attribute weights. These weights are applied to the systems analyzed in chapter 5. Chapter 7 gives the results of feedback from HLA-RTI vendors and DSS practitioners. Chapter 8 gives the final framework, contributions to the body of knowledge, suggestions for future research, and conclusion.

CHAPTER TWO: LITERATURE REVIEW

Simulation

Simulation is the art and science of making models of entities and systems in the real world in order to study them. These models are mathematical creations that use computers to emulate the most salient features relevant to a problem about a system or situation. Given a problem statement, one studies a phenomenon and collects data that are used to model the phenomenon, often using probability distributions or generating modeling behavior from actual historical probability distributions. By creating a software model of a system, random behavior can be studied over many runs. Also, alternative design scenarios can be studied without actually needing to build a system.

When creating a simulation model, first the real-world system is studied, then a conceptual design is made of the situation. A common tool used in the conceptual stage is a set of UML diagrams of the relevant system aspects if the simulation is object oriented. In performing a simulation study, one first designs and creates the simulation model, then performs an analysis of the simulation results. One looks at a real-world system, creates a conceptual model at the abstract level, then creates a simulation model at the software level (Garrido, 2001). Figure 2 below shows a diagram of the levels involved in a simulation study.

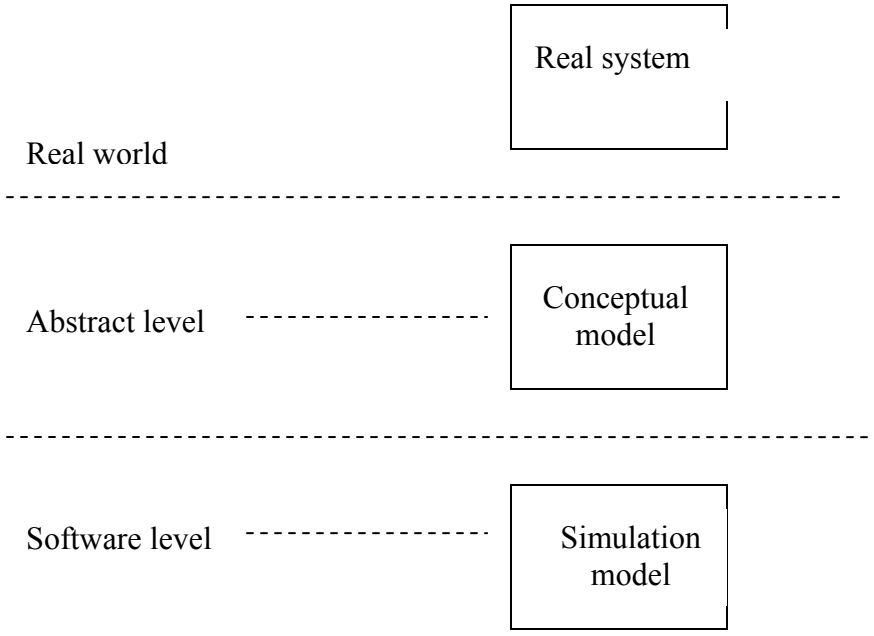

Figure 2. Conceptual Model and a Simulation Model (from Garrido, 2001, p. 7)

There are numerous simulation software packages available that people who do not know a programming language can use to create simulation. An example of this is Arena, a product of Rockwell Software. These packages typically include at least some provision to integrate more advanced capability through an Application Programming Interface (API) or a means of interfacing with a language such as Microsoft's Visual Basic. Most advanced simulation work, however, will involve software implementation that requires writing code. The ease with which the programming can be done is dependent on the language(s) used and the infrastructure for the simulation.

Software programming languages have progressed from procedural (e.g., FORTRAN), to structured (e.g., C), to object-oriented (e.g., C++ and Java). For some basic simulation problems, a structured language may be adequate, although most software development performed today

uses an object-oriented programming approach. In fact, the first object-oriented language, SIMULA, was created for the purpose of constructing simulation models. The strength and power of object-oriented software often make it the tool of choice for large-scale simulations, including distributed simulations. In the coming decades, Model Driven Architecture (MDA) may replace object-oriented design as the basic paradigm. The Object Management Group (OMG), a prime developer of the theory of object-oriented software, "decided to raise the level of abstraction and focus on describing how the system should be integrated" with MDA (Harmon, 2005).

It is not the intention of this dissertation to provide a detailed study of the software aspects of various simulation approaches. However, an understanding of software approaches and the programming challenges involved is necessary to evaluate the software components in the holistic usability framework developed herein. For the designers and programmers (and the manager who must oversee the budgets and manpower schedules), the software approach taken and in particular the challenges encountered and the ease of use of the software development tools used, has a profound effect on the difficulty level of their jobs and their productivity, and hence the relevant aspects of the usability framework.

Simulation is appropriate when a closed form mathematical solution to a study is not possible and the cost of the simulation effort is justified. Advantageous uses include the study of new procedures and designs, the testing of hypotheses, when time compression or expansion is needed, studying interaction and the importance of variables, effective bottleneck analysis, understanding how a system actually operates rather than the intuitive understanding of the system, and for alternative design studies (Pegden et al., 1995; Banks et al., 1996). Banks et al. discuss the disadvantages of simulation, then give offsetting factors that help mitigate the

disadvantages. The first three of these disadvantages relate directly to issues in the holistic usability framework developed herein: (1) Modeling requires training and skill. A novice cannot simply create good simulation models without training. This is offset somewhat with software packages that offer templates for basic simulations. (2) Results of simulations can be challenging to interpret. This can be offset by improving the usability of data analysis and visualization capabilities in the system. (3) Creating and analyzing simulation models are often expensive and time intensive. This is offset somewhat by improved simulation software and templates for basic models (Banks et al., 1996, pp. 5-6).

In sum, simulation is a tool that helps us analyze and study situations and solve problems. Taking the concept of simulation to the next level, distributed simulation is discussed in the following section.

Distributed Simulation Systems

Distributed simulation systems are used for increasingly sophisticated purposes, such as the creation of full scale virtual or mixed-reality environments, training, entertainment, real-time data analysis and optimization of complex systems, and business situations (e.g., world-wide financial market "what ifs"). One of the current methods for distributed simulation is the High-Level Architecture Run Time Infrastructure (HLA-RTI). The US Department of Defense (DoD) approved the HLA as the standard for all DoD simulations in 1996. The OMG adopted the HLA as the Facility for Distributed Simulation Systems in 1998. In 2000 the Institute of Electrical and Electronic Engineers approved the HLA as an open standard (Defense Modeling and Simulation

Office, 2004). The HLA is an architecture; the RTI is the software that provides the needed infrastructure for the interlinkage of simulations.

In a distributed simulation environment, system performance is an important factor in usability. Belleman and Shulakov (2002) note that usability increases as the update time for the simulation decreases. The minimization of delays is important for interactive simulation. Permulla (2002) found that the Extensible Modeling and Simulation Framework (XMSF) can increase usability in a dynamic, distributed simulation environment as compared to HLA-RTI, but at a higher cost. He noted that HLA RTI research has focused mainly on static configurations of federates.

In looking at continuous distributed simulation systems for the military, Ceranowicz et al. (2003) noted that "Even if we overcome the limitations of scope and scalability, ease of use will remain a roadblock to making M&S ubiquitous in the concept of development process." The authors mention that it would be preferred if a user could call up scenarios and run the simulation from his or her interface, but that currently the user must coordinate with several other people at various distributed computers in order to run the simulation. The goal is for a single user to be capable of controlling all the computers used in a distributed simulation, without needing assistance at the remote sites. Combining the control and monitoring capabilities for several systems into a single interface also allows for a user to better understand the overall system.

A set of taxonomies for computer-based simulations was provided by Sulistio, Yeo, and Buyya (2004). Their paper lists taxonomies for simulation tools, parallel and distributed simulations (PDSs), usage, simulation, and design (with design taxonomies for the simulation engine, modeling framework, programming framework, design environment, user interface, and system support). They focused on research-based simulation tools rather than simulation tools

made by private companies, because the former's design characteristics are available, while the latter's are generally not disclosed. Although their simulation interest is geared towards computer infrastructure and network details, their taxonomies are imminently applicable to the evaluation of any simulation system. They classify distributed simulation systems into five groups: "Internet, intranet, mobile systems, embedded systems or telephony systems." Interestingly, their simulation taxonomy not only includes discrete and continuous, and deterministic and probabilistic systems, but also, relative to time, static and dynamic systems. Their user interface taxonomy includes nonvisual and visual systems. (Although they note that a visual system is preferred, in certain situations an expert may prefer a command-line interface for speed and directness.) The visual user interface part of the taxonomy is broken down into design (drag and drop versus form), execution (animation versus graph), and integrated environment. The integrated environment they discuss is analogous to an integrated development environment (IDE) for program development (such as a Java IDE), although it is for programming simulations. They note that "Most tools have plans to incorporate a visual integrated environment in the future to enable better usability, but implementing a good user interface is not trivial and requires lots of time and effort. This is why most simulation tools are not able to provide a visual interface."

DSSs are also important for business. Chorafas and Steinman (1995) present numerous examples where real-time simulation can be used to analyze and visualize data, noting that the use of a "hypermedia solution space" (p. 139) is a solution to a data processing bottleneck. In other words, having enough information to be able to solve a problem is not enough; the information needs to be analyzed in real time in a way that allows one to see a solution. One of the examples Chorafas and Steinman present is the failure of Metallgesellschaft (a subsidiary of

the Jürgen Schneider Company) due to its speculation in financial derivatives without the requisite real-time visualization tools to understand their strategies. Another is the failure of a company in financing a real estate developer with loans as great as DM 1.15 billion. The company lacked needed "high-fidelity real-time simulation tools" (p. 141) that would have helped the company understand the real estate developer's records. Mention is made of a software package by the Swiss financial company Securum, which allows a virtual view of a building's plans, several layers of changes in drawings, real estate accounting with budgets, an expert system, and a job specification. In complex business where large amounts of information need to be processed and understood in order to make correct decisions, a DSS can be the key to survival. These systems can involve accessing databases and simulations in several locations simultaneously. For distributed simulation systems, much of the literature only mentions the need for a central control and monitoring GUI. Also mentioned is the fact that the usability of DSSs needs improvement. The difficulties of coordination and collaboration in running models together from separate computers in separate locations is often mentioned. From an empirical viewpoint, aside from an evaluation of existing research literature, examination of existing systems, interviews with the researchers who use them, and ethnographic observation of systems being built and in use are informative about the nature of the problem.

As an example of recent research in this area, below is a quote from Fishwick (2004) from an article entitled "Toward an Integrative Multimodeling Interface: A Human-Computer Interface Approach to Interrelating Model Structures" (p. 422-23):

> Despite the wealth of work already done in creating a substantial visual user interface environment on top of a simulation program, the integration of model

> structures and their execution results has only begun. The community needs to focus on more effective ways of interfacing with the human, whose cognitive system is comfortable with metaphors and engaging iconic models and less comfortable with purely symbolic language interfaces…the grand challenge…is *to enable a human-computer interface in which models of different types can be integrated with one another through effective interaction means.*

Although Fishwick also examines issues associated with data visualization, he does speak of the challenge of designing usable interfaces for frameworks with multiple models. In addition to usability, he raises issues of emotion/aesthetics, immersion (related to what is termed "presence"), and customization (which is a needed feature for DSS interfaces). When speaking of a distributed simulation architecture different than the HLA, Frank Sogandares at the Center for Advanced Simulation System Development discussed this requirement in a paper reflecting a decade of research in distributed simulation work in an aviation context. His observations are worth quoting (Sogandares, 2002, p. 126):

> Simulations and simulation clients must be "easy" to execute, configure, pause, and resume. Simulations must startup and shutdown cleanly.
>
> This is an extremely important requirement. A simple to use environment allows developers, analysts and experimenters to execute simulations without the assistance of programmers.

Usability

A usability analysis can be evaluative or formative. Evaluative usability is when a system's usability is evaluated either by heuristic analysis (an expert analyzing a system by applying heuristics), by empirical testing using human participants, or by other methods. Formative usability is when usability analysis is part of the design process; it is designed into the system. Formative usability falls into the provinces of interaction design and other user-centered design approaches.

Although there is a large body of research in human-computer interaction (HCI), there is a relative lack of theoretical literature concerning usability. There is, however, a large body of practitioner literature, which is fragmented, spanning across many disciplines, with the approach varying depending on the viewpoint taken.

A collection of papers in a book edited by Trenner and Bawa, eds., (1998) entitled *The Politics of Usability* discusses topics about how to justify usability, how to navigate company politics while performing studies and design work, and how to effectively deal with international and multicultural issues. Beyond politics, some of the papers in the book offer advice on cost justifying usability and the standardization of usability practice. *Usability Evaluation in Industry* (Jordan, Thomas, Weerdmeester and McClelland, eds., 1996) is another collection of papers that offer mainly a European perspective on usability practice. Numerous chapters are provided on evaluation methods, as well as chapters on field studies, informal methods, and task analysis. Relatively new evaluation methods discussed include feature checklists, co-discovery exploration, and repertory grid theory.

Because of its ubiquity, social, and economic importance, much research has been done on the usability of World Wide Web sites. An example of a book discussing this is *Shaping Web Usability, Interaction Design in Context* (Badre, 2002). The book begins with a discussion of the human-computer interaction (HCI) information processing approach to usability, then discusses a variety of topics germane to Web design, such as demographics, genres, and aesthetics. While much Web design can be for entertainment or business purposes, since most distributed computing applications are Web enabled—able to run over an intranet or an Internet via a Web browser—some Web design guidelines can also be applicable to distributed simulation systems.

Evaluative Usability

Currently, industry practice in usability evaluation is often reported in a standardized format. In the US, the format started as a project at the National Institute of Standards and Technology (NIST); hence, sometimes one hears reference to the "NIST" format (NIST, 1999). After a successful industry collaborative effort with NIST, the maintenance of the format was transferred to the American National Standards Institute (ANSI). At the time of writing, work is underway to modify the format to include formative testing. The ANSI format and the NIST format for usability reports are very similar. The ANSI format, ANSI NCITS 354-2001, is called "Common Industry Format for Usability Test Reports" (ANSI, 2001). Many companies prefer this format for reporting. Three measures are stated for measuring usability in the ANSI format: effectiveness, efficiency, and satisfaction; metrics for these items are required. Of particular note, in the definitions section, one finds "context of use," the "user group," "goal," and "task"; the definition of these terms is self-explanatory. For effectiveness, measures can include completion

rate—the percentage of tasks successfully completed, errors, and assists—when the participant asked for help. Efficiency is usually taken to be mean time on task, with standard deviation and range also noted. Satisfaction is often measured using semantic differential or Likert scales. A variety of standard, validated surveys are used to measure satisfaction, or the analysis team can construct its own. (The same measures are also given in ISO 9241-11, *Guidance on Usability*—a European standard [International Organization for Standardization, 2003].) Annex A of the ANSI document is a checklist for making sure nothing is missed in the Common Industry Format (CIF). A report template is included. The ANSI CIF provides a standardized format that knowledgeable professionals are familiar with; interested readers will turn to the sections that interest them. It is for testing products that already exist, not for development, although it can be used after prototypes are developed. The format can also be taken as a starting point for developing a unique format specialized for a given usability evaluation. A wide variety of usability measures and techniques exist, however, that are not included—and would not necessarily be appropriate to include—in the ANSI format for test reports, which is a good tool to use where appropriate.

Preferably, a validated survey is used when measuring user satisfaction. Shneiderman (1992) created a survey that is often used to measure satisfaction with computer interfaces. It is called the Questionnaire for User Interaction Satisfaction (QUIS) and is currently at version 7.0, available for a fee from the University of Maryland. This validated survey can be used to evaluate the level of satisfaction with a computer interface; it can also be modified as needed for a specific situation. A problem with using long surveys is that respondents, particularly if not sufficiently motivated, may tend to skip questions or not tend to them seriously at the end of the survey. When Digital Equipment Corporation wanted a quick and reliable measure of user

evaluations of their software, they developed the System Usability Scale (SUS), a short ten-question survey using Likert scale questions (Brooke, 1996). The SUS survey is shown below in Figure 3. The scoring of the SUS results in a number ranging from 0 to 100 that is a measure of overall satisfaction. The SUS is an example of how user satisfaction can be measured. One can also count negative and positive comments made aloud by users while they are using the interface during a test, as well as post-test comments, to get other measures of user satisfaction. The QUIS is considerably longer and more detailed.

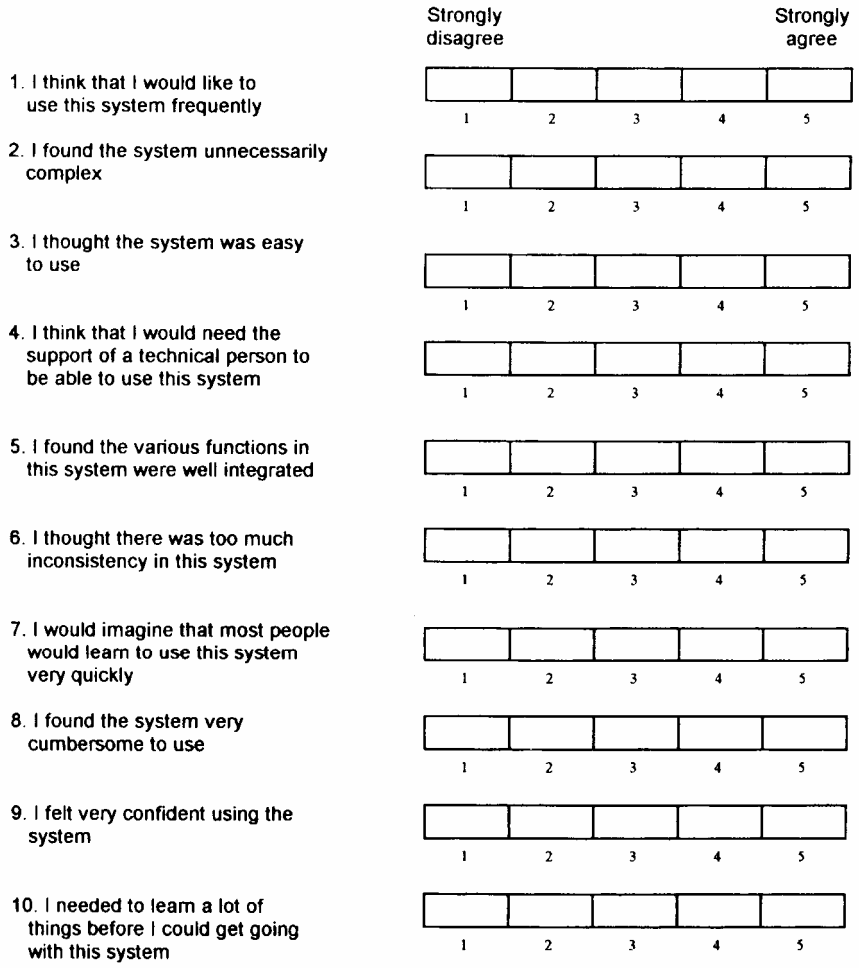

Figure 3. System Usability Scale (source: Digital Equipment Corporation)

An overview of usability measurement methodology was given by Bevan and Macleod (1994); an in-depth discussion of their work, which covers a lot of ground, is warranted. They noted that improved usability reduces cost and increases productivity and user satisfaction. In Europe, the Display Screen Equipment Directive requires the application of software ergonomics and usability principles in creating new software. Objectives for usability measurement include the attainment of a minimum level of product usability, feedback on whether design objectives are being met, and the identification of problems. Numerous European standards are related to usability. For instance: ISO 9241-11, Guidance on specifying and measuring usability; ISO 9241-10, Dialog principles, which includes, the authors note, "suitability for the task, suitability for learning, suitability for individualization, conformity with user expectations, self descriptiveness, controllability, and error tolerance"; and ISO 9241-14, Menu dialog guidelines. Usability is not as easy to specify as some other software attributes because it depends on the context of use. Bevan and Macleod note that guidelines and checklists are useful, but guidelines are sometimes appropriate only for a given type of user and system, can be interpreted differently, are seldom if ever exhaustive, are not always compatible with other guidelines, are applied depending on the skill of the evaluator, are context-dependent, do not ensure a given level of usability is met, can be overgeneralized, can be interpreted differently by different experts, and cannot be turned into measures. Formal measurement methods, however, can make usability predictions early in the design process. A keystroke-level model (K-LM) looks at keystroke and mouse movements. It can predict the amount of time an expert user takes to do a task, but does not consider contextual issues. The K-LM, which is a low-level analysis, can be extremely useful in design evaluation. As Bevan and Macleod state, "…some applications are

being developed which force users to perform cumbersome sequences of mouse actions, to carry out simple operations. A K-LM analysis of such operations can identify the potential advantage in actions and time saved of providing single keystroke alternatives for frequently performed operations." (Programmers often do not think much about usability, in the same way that hardware designers do not think much about reliability and safety—an independent review is needed to help the designer or programmer think about issues other than simply meeting specifications.) Cognitive models to understand the thought processes of the user are a more sophisticated technique to evaluate designs; they require a high level of expertise to develop, however. ("Thinking out loud" is a simpler technique that can determine some of the same information as a cognitive model—what the user is thinking.) A SANE model is a technique that compares and simulates different design models, simulates procedures, and derives usability measures.

Bevan and Macleod note that it is necessary to specify the context of use before the usability of an interface can be evaluated. Stating that "usability is the quality of use in a context," they quote ISO 9241-11's definition: "The effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments." The context of use is specified by four factors: users, task, equipment, and environment. The authors note different views of usability: a product-centered view, a context of use view, and a quality of use view. They note that efficiency can be human efficiency, which ultimately can be translated into economic efficiency. Usability for software maintenance is also mentioned as an important measure; this points out the need for considering not just the end user, but the programmers maintaining the software as well.

The MUSiC (Metrics for Usability Standards in Computing) measures were developed in order to develop usability tools and measurements. Bevan and Macleod provide a summary of MUSiC, which includes detailed tools to measure and specify many aspects of usability. For instance, there is a 50-item questionnaire, the Software Usability Measurement Inventory (SUMI), that is internationally standardized and available in five languages, which provides three measures: "an Overall Assessment, a Usability Profile, and Item Consensual Analysis which gives more detailed information."

An oft-referenced book is *Usability Engineering* by Jakob Nielsen (1993). It was written specifically to address computer interfaces, although it is generalizable to other types of systems. Nielsen defines five main usability attributes: learnability, efficiency, memorability, errors, and satisfaction. Learnability and memorability can be considered to be part of effectiveness (mentioned above in the CIF format). Nielsen defines ten usability principles (heuristics) that can be used to evaluate an existing design (p. 20):

- Simple and natural language
- Speak the user's language.
- Minimize the user's memory load.
- Consistency
- Feedback
- Clearly marked exits
- Shortcuts
- Good error messages
- Prevent errors.

- Help and documentation

Use of these heuristics, of course, requires clear definition of what each means. (This list of heuristics has some similarities to Shneiderman's [1992] "eight golden rules of dialog design": "Strive for consistency; enable frequent users to use shortcuts; offer informative feedback; design dialogs to yield closure; offer simple error handling; permit easy reversal of actions; support internal locus of control; reduce short-term memory load" [pp. 72-73].) Much longer lists of heuristics exist, but when applying heuristics it is sometimes useful to use a short list to keep the job manageable. Since no usability analyst finds all the problems, the most cost efficient way to evaluate a system using heuristics is to use three to five expert evaluators; more results in diminishing returns. Nielsen notes three dimensions that differentiate users: level of knowledge of the domain, amount of computer experience, and whether a novice or expert with the system. Designing a system for a type of user, or more than one type (e.g., novice and expert), is important; different features are needed to support different types of users. A detailed discussion of the usability engineering lifecycle is given in Nielsen's book, noting many important issues. For instance, two dimensions of prototyping, horizontal and vertical, are given. A vertical prototype gives complete functionality of a few features, while horizontal prototyping gives many features but little depth of functionality. Design of user interfaces is almost always iterative, if for no other reason than the fact that without user testing, one does not know what users will do with the system.

Master Usability Scaling (MUS) was proposed by McGee (2004) as a usability measure. It is based on an a priori scale-independent rating method used by individual evaluators called Usability Magnitude Estimation (UME). The rationale for the development of MUS is a critique

of current usability metrics. While the measurement methods proposed and tested by McGee are intriguing from both theoretical and practical viewpoints, he notes that the methods are "sophisticated enough to pose a significant barrier to widespread MUS adoption" (p. 342). In spite of the complexity of MUS and UME, they offer the ability to take measurements that may offer more statistical validity in certain experiments.

Heuristic evaluation (HE) is a technique whereby expert evaluators inspect an interface or device to find usability problems, using a set of heuristics as a guideline. Although this widely-used technique has many strong points, one of its weaknesses is that evaluators have a tendency to find cosmetic problems that are of minor importance in addition to serious usability problems. Hornbeck and Frokjaer (2004) propose a new usability inspection technique, metaphors of thinking (MOT). There are five key metaphors in this technique: (1) "habit formation is like a landscape eroded by water," (2) "thinking as a stream of thought," (3) "awareness as a jumping octopus in a pile of rags," (4) "utterances as splashes over water," (5) "knowing as a building site is in progress" (pp. 359-361). After learning the concepts of the five metaphors, an evaluator then becomes familiar with the application, considers three typical user tasks, then follows an iterative procedure involving evaluating tasks in light of each of the metaphors. The authors performed an experiment comparing HE to MOT in evaluating a system. They found that MOT tends to find less cosmetic problems than HE, but finds deeper, more serious usability problems. Further research is needed, but MOT is a promising technique to supplement HE and other inspection methods.

Scandinavian countries are known to be in the vanguard of the implementation of human-centered design and work techniques. A recent survey of the usability profession in Sweden was reported by Gulliksen et al. (2004). Several software development models were in use by the

respondents. One of the most widely-used software development models mentioned in the survey is the Rational Unified Process (RUP). The RUP was noted by most who commented on it as being woefully inadequate from a usability perspective; some companies have modified it to include usability by using a usability plug-in. Of usability methods in use in the software development process, "plug-ins, general frameworks, generic HCI methods, process-oriented methods, heuristic methods, and no model" were listed (p. 211). An interesting chart generated from survey data shows 25 "methods and techniques" (p. 212) rated using a five-degree scale; the chart shows the percentage of responses of each rating for each technique. Following are examples of the many readings one could take off of this chart: More than 50% of the respondents rated check lists, personas, and benchmarking as neither good nor bad, fairly bad, or very bad. Approximately 80% or more rated thinking aloud with users, prototyping, low-fidelity prototyping, evaluations, scenarios, interviews, and field studies as very good or fairly good. Some other techniques received high, but not as high, ratings. Even though Sweden is known to be at the forefront of human-centered development, the authors noted that serious usability problems still exist in most software, and that the country has a long way to go in implementing good usability practices in industry.

An extensive look at usability measurements in a large number of research papers was undertaken by Hornbaek (2006). He noted that usability is defined by how it is measured, but it can only be measured indirectly. In reviewing research studies, it was found that the methods by which effectiveness, efficiency, and user satisfaction are measured vary greatly, are inconsistent, and are difficult to compare. Subjective measures for usability include users' perceptions and attitudes, while objective measures are taken from observation and analysis independent of

users' opinions. Hornbaek notes that it is important not to confuse subjective and objective measures.

The *Handbook of Usability Testing* (Rubin, 1992) discusses key issues related to usability testing. Testing is a key component of usability evaluation if the budget permits, particularly for the usability of interfaces. Every human being will respond to a system in a unique way. A wide variety of responses is almost always found in a test. People are unpredictable, and unexpected responses will occur during a usability test. While expert evaluators can check to see that good usability heuristics are followed in design, only real users can provide needed empirical feedback.

Software development techniques will increasingly be using modular components. Brinkman et al. (2001) investigated how to measure the usability of user interface components and also brought up the subject of how the usability of one module may affect the usability of others. They considered the user interaction to be an exchange of messages between a component and the user. Taking this as a measure of interaction, they tested the hypothesis that, looking at different components performing the same function, the component that received the fewest messages was the most efficient one. Their preliminary results substantiated this hypothesis.

Formative Usability

The usability engineering lifecycle is discussed at length in Mayhew's (1999) book of that title. Like Nielsen's book, the emphasis is on computer systems. The approach used is to first perform a requirements analysis that determines a user profile, perform a task analysis, look at

the hardware and general design principles, then use all of these to determine usability goals, ultimately creating a style guide for the user interface. Following this procedure (as shown on a detailed flowchart for the usability engineering lifecycle), there are three levels of design, testing, and development—all with detailed tasks and an iterative loop for each, then finally system installation with provisions for user feedback and enhancements. There is a section on organizational issues in the book, including sections on project planning and cost justification (among others).

The design of software can be considered from many angles; the multidisciplinary arena offers a growing number of viewpoints, as specialists from various fields offer their inputs. Henry (1998) has approached usability from the viewpoint of communications and as a technical writer. In a process he terms User-Centered Information Design (UCID), he considers usability to be largely a problem of communicating with users via UCID. He sees software usability as being determined by labels (text or icons), messages, online support elements (e.g., help screens), and printed support elements (manuals or help cards). Often the communication issue is given limited treatment in software designs. Henry noted that, while there are pluses and minuses to having paper or Web-based documentation, on-line help screens should be written separately from the system manuals, while drawing from the same information pool. Furthermore, he notes that the writing used to communicate with the users should be professionally written, rather than left solely to the programming team. The writing—both for the online communication and any paper or Web-based manuals—should be considered part of the framework for usability of DSSs (it is considered in the documentation dimension). Indeed, for teaching installers, users, and operators to work with the system, documentation is an important factor in usability.

Another practitioner's perspective on usability is offered by Snyder (2003) in her book *Paper Prototyping.* Typical of the field, emphasis is placed on quick prototyping using user feedback to make changes to iterative designs. Empirical testing and iterative design are required in order to incorporate information from user evaluations. Not only will users inform designers of things they never considered in the design, the great variety in human behavior—even among people with the same background—will ensure considerable variation in how the users approach a system. The approach offered by Snyder involves using a task analysis of many low-level, detailed tasks that a user needs to perform with the system to design a GUI. By focusing on specific low-level tasks, then evaluating rapid prototypes to see if users can successfully complete the tasks and if they will use the interface as expected, a viable interface can be designed before the start of software coding.

From an interaction design viewpoint, formative and evaluative approaches blend into an integrated analysis (often, however, budget constraints preclude the ideal approach). The formative approach results in much value added from usability analysis. As noted by Card, Moran and Newell (1983), "Design is where the action is in the human-computer interface" (p. 11).

Holistic Usability

The concept of holistic usability has been raised by a few practitioners. For example, the Innovation North Faculty of Information and Technology of Leeds Metropolitan University made this statement:

> The International Standards Organisation (ISO) defines usability as “the effectiveness, efficiency and satisfaction" with which a specified set of users can achieve a specified set of tasks in a particular environment”. In our usability studies, we expand on this definition to consider a broad range of factors within a holistic model.

The holistic model of which they speak is shown in the figure below, taken from their Web site (Leeds University, 2005).

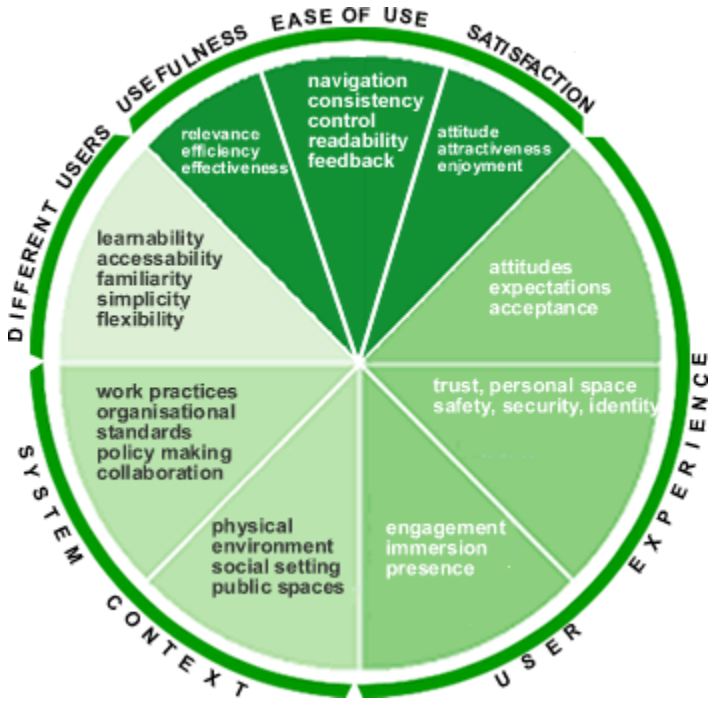

Figure 4. Holistic Usability Model (source: Innovation North Faculty of Information and Technology of Leeds Metropolitan University)

The main groupings for this holistic usability model are designing for different users, usefulness, ease of use, satisfaction, system context, and user experience. This framework is meant to be used to analyze a general system and shows one approach to a holistic usability model (or framework).

Another holistic approach to usability was used by researchers at Microsoft while considering design issues for Internet Web pages (Agarwal and Venkatesh, 2005). In these holistic usability guidelines for Web pages, the key groupings are content, ease of use, promotion, made-for-the-medium, and emotion. It is possible to quibble about whether or not every item in the above framework can be considered usability, but as the definition of usability is expanded some blurring of territories or definitional ambiguities will occur. This example helps to show the validity of a holistic approach to usability.

These two examples of holistic usability models/frameworks are similar to the concept developed in this dissertation. The first example given above presents a framework that is generic and applicable to any system. The second presents a framework for Web pages only. The framework proposed herein takes a system-wide view of distributed simulation; it is similar to these examples in that the lens that views usability is broadened to look at items other than just the user interface.

Human-computer Interaction Viewpoint

In the field of human-computer interaction (HCI), of which usability is a subarea, cognitive psychology is of paramount importance. Human information processing, theories of how human memory works, and the user's conceptualization of mental models are important

topics. Working with computers, however, has much in common with working with any tool. Green (1990) noted that "the real aim…is to explain the General Theory of the Artifact" (p. 22). He also noted that when theorizing about HCI, there was a need to set boundaries on the HCI theories, lest HCI theories attempt to encompass the whole of cognitive psychology. When working in usability, principles of cognitive psychology must be kept in mind when developing guidelines for usability.

In usability, human information processing (HIP) plays a role in the development of the user's mental model of the system. The development of the user's mental model, and its maintenance and changes as learning takes place, are a key part of effective usability. The development of mental models is discussed in Wickens 2002, Ackemann and Tauber (eds., 1990), Badre (2002), Eberts (1994), Gentner and Stevens (eds., 1983), Johnson-Laird (1983), Preece, Rogers and Sharp (2002), Oakhill and Garnham (eds., 1996), and Norman (1986). The intended way a system is designed to be used is called the designer's conceptual model. (The actual workings of a complex system are not necessarily part of this model, but will be addressed with other, lower-level models, which the end user does not need to know.) In order for a system to be used as intended, the designer's conceptual model must be conveyed to the user in the form of a mental model that the user learns. An interface must be designed to aid and guide the user in the formation of the desired mental model. Eberts' 1994 book *User Interface Design* diagrams this process (p. 140). A long list of human factors tools and concepts aids in effective interface design. These tools and concepts are supported by cognitive psychological theories and empirical evidence.

A variety of types of mental models were discussed by Young (1983). The types of models he discussed are: "strong analogy, surrogate, mapping, coherence, vocabulary, problem

space, psychological grammar, commonality" (p. 38). A description of all of these types will not be given here, but the variety of possible types of mental models—and combinations of them and introductions of new categorizations or types—leads one to keep in mind that for any given artifact different users may use different types of mental models, and that it is possible for a user to hold more than one type of mental model in mind simultaneously when thinking about the same system. For example, a user might think of a form in a word processor as being like a paper form (a "strong analogy" model), while when typing data into the form use a "mapping" or "task/action mapping" model to know what his or her actions do when using the computer.

A large number of HIP models have been constructed over the past sixty years (e.g., McCormick, 1976, p. 35; Welford, 1965, p. 6; Card, Moran and Newell, 1983, p. 26; Badre, 2002, p. 46). Numerous models of human memory have also been constructed, concomitant with the HIP models or separately (e.g., Norman, 1969, p. 152, Wickens, 1992, Chapter 6). Research into perception and attentional processes is also relevant to studying memory and HIP. HIP, human memory, and attentional processes are all active fields of study and theoretical development in cognitive psychology; a number of competing theories exist. One can begin with William James' 1890 work the *Principles of Psychology* and follow the development of these theories to their present state.

Industrial engineers have, along with psychologists, looked at applying human engineering knowledge to problems in an applied setting. Research using human participants to help determine how best to design systems has long been a field of study in engineering (a good overview is given in *Research Techniques in Human Engineering,* Chapanis, 1959). The HCI field has been an impetus for much research in cognitive psychology and practical design advice. HCI is a subset of human factors engineering (one can speak of human-systems interaction,

interaction design, or many nearly synonymous terms). When constructing a framework for the usability of DSSs, HCI research must be considered, along with general usability information and knowledge about simulation.

Reviewing existing literature and research in usability, simulation, and HCI reveals that there has been no work done on the usability of DSSs, although their ubiquity is increasing. This dissertation research will help fill this void, as a starting point for not just the practical application of usability in DSSs, but also for the theoretical development of usability in simulation environments. A framework is needed to aid designers of distributed simulation systems. While general interface guidelines are available, there are none for DSSs.

The breadth of the subject matter needed to understand the usability of distributed simulation includes fields related to distributed computing, simulation in all its guises, and usability. These fields include psychology, business analysis, programming, engineering, computer science, statistics, design, cognitive engineering, technical writing, graphical design, interface design, human-systems interaction, and human factors. In order to develop the framework herein, all these fields have been studied to the extent necessary to understand their needed place in the framework's dimensions.

Summary and Objective

Simulation, as discussed, involves making a model of a real world system. Distributed simulation involves multiple computers interlinking multiple simulations, so that knowledge and insight are gained from the combined simulations. While this allows for solutions to previously-

intractable problems, making it work is often a difficult challenge, as can be interpreting the results.

The field of usability engineering is focused on making systems easier to use. We have considered evaluative usability—measuring the usability of an existing system, formative usability—the task of designing in ease of use, and holistic usability—a new concept that looks beyond just the user interface to all aspects of the system, from the installer to the end user.

The objective of this dissertation is to develop a framework for the usability of distributed simulation systems that can be used as an aid for designers in the formative usability stage and as a tool for assessing the holistic usability of a system, providing metrics. The holistic approach taken expands the common definition of usability to include all people associated with the system. Taking a holistic viewpoint is especially helpful with a large, complex system, in that the benefits of improved usability can be multiplied by looking not only at the user interface, but also at various aspects of system design, installation, maintenance, and use.

CHAPTER THREE: METHODOLOGY

The methodology for this dissertation consists of a three-step approach: (1) the development of a holistic usability framework for DSSs, (2) surveys of users to validate and refine the framework, and to determine attribute weights, and (3) the application of this framework to two existing systems, including the development of a technique to measure holistic usability. The flowchart below shows the steps taken.

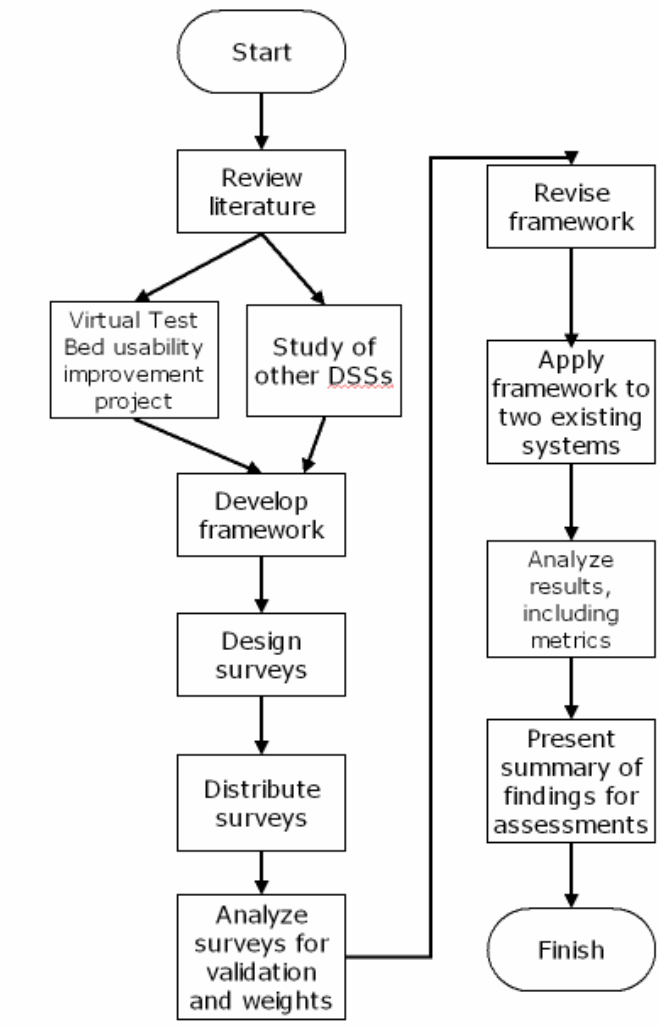

Figure 5. Flowchart of Methodology

Holistic Usability Framework Development

The Virtual Test Bed (VTB) was developed to test and develop distributed simulation concepts for NASA. While working to improve the usability of the VTB, it became apparent that a systematic approach was needed when designing DSSs, in order to ensure not only that an efficient and effective system interface was developed, but that the various types of users and workers who interfaced with developing and using the system over its lifecycle did not lose time bogged down in problems that could be avoided by good usability practice during system design and development. The first part of this methodology—the development of a holistic usability framework for DSSs—grew in part out of this research.

DSSs at NASA Ames Research Center were also studied. While the systems at NASA Ames are larger and more complex than the VTB, similar usability issues arose. For instance, while using the HLA-RTI in the Virtual Airspace Simulation Technology (VAST) system, it was necessary for people starting different simulation models to coordinate starting and stopping the simulations by voice communication over speakerphones. The need for a central control and monitoring GUI was evident.

A study was made of the fields related to usability vis-à-vis simulation in order to understand the domain knowledge required for a holistic approach to improving the usability of DSSs. For a given user base, the tasks, needs, goals, and characteristics of the users must be considered. In order to design a DSS, the required resources (money, time, technology, talent, and knowledge) are brought together in a multidisciplinary teamwork environment to create a system to meet user needs. Keeping this domain knowledge, the user base, and typical resources available in mind, a framework for the design of DSSs was developed.

This framework looks at the various dimensions of the interaction of different types of users of the system. Each dimension represents a particular aspect of the system design relative to usability. The total holistic usability of a system is the ease of all human interaction with the system along multiple dimensions. The figure below shows the usability dimensions of the framework.

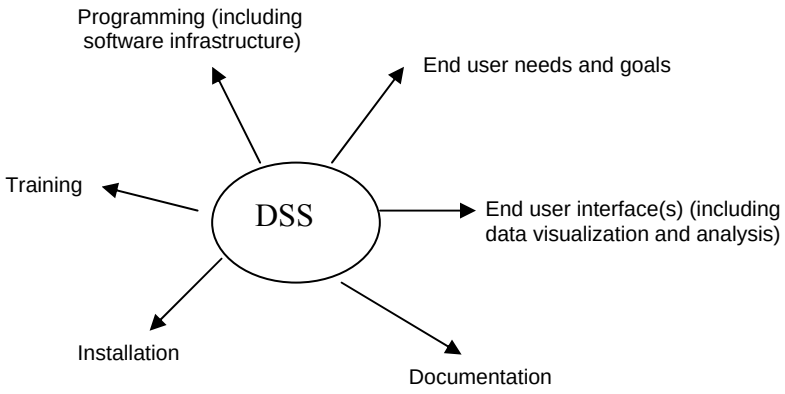

Figure 6. Holistic Usability Dimensions for a Distributed Simulation System

The dimensions of the holistic usability framework are:

- End user needs and goals
- End user interface(s) (including data visualization and analysis)
- Programming (including software infrastructure)
- Training
- Installation
- Documentation

Each of the dimensions is discussed below.

End User Needs and Goals

The End User Needs and Goals dimension considers how well the end users' purposes for the system are served. Goals may vary from end user to end user. A list should be made of the needs of the end users (from a high-level, global perspective). The list should be checked against the system's actual design. The range of goal fulfillment is "not at all" to "completely."

End user needs also include the tracking of lessons learned for future improvement, system reliability, and, if a vendor-provided system, vendor support.

End User Interface(s)

The End User Interface for a DSS ideally allows for system-wide control of all simulations in the system. A number of the attributes for this dimension relate to that need. If no central control GUI exists, then the system interfaces will need to be assessed, with examination of coordination and communication capabilities between the GUIs.

The assessment of the end user interface dimension does not attempt to go into a standard usability assessment of a GUI. The purpose of the end user interface assessment is to assess mainly those aspects of the interface that are germane to distributed simulation. The general quality of the interface from a high-level usability perspective, however, is assessed. If multiple end user interfaces are used in the system, they should all be assessed and their metrics averaged.

Programming

The Programming dimension concerns the ease of use for programmers working with the system. Programmers must create the software that translates the design vision of the system into

reality. A number of factors affect how easy it is for programmers to work with the system. From a usability perspective, ideally distributed simulations could be created using a graphical programming interface, so that linking up individual simulations would be a relatively simple task. Although that may be possible in the future, at present programming a typical DSS is a nontrivial task.

Proprietary simulation packages usually do not offer a direct interface to the simulation infrastructure, such as the HLA-RTI. Typically, a software "wrapper" must be used—a program layer that goes between the proprietary simulation package and the DSS infrastructure. A wrapper will offer a limited function set that allows synchronization and data exchange between the program and the infrastructure. The loss in efficiency due to this is that the programmer must not only know the infrastructure, but learn the wrapper as well, and some details of the propriety simulation package. In addition, remotely controlling start and stop of a proprietary simulation can sometimes be accomplished by using a vendor-provided API—such as the Visual Basic for Applications interface for the Arena software package. A DSS can consist of any number of simulations linked together. Simulations with open code—such as those in Java—are easier to work with than simulations requiring a wrapper to connect to proprietary code; thus simulations in open code increase the usability for programmers.

Training

Adequate training is important to all people who use the system. End users need training on how to use the system interface and in how the system works conceptually. Installers need training to learn how to efficiently install the system. Training programmers to understand the

system software design and also the DSS infrastructure is important to make the most effective use of the programmers' time. Training materials, both on-line and written hard copy, help to ensure adequate training.

Installation

The Installation dimension measures how easy the DSS is to install. The installation scenario will depend on the level of expertise of the installers, how complex the system is, and how well the installation process has been documented and managed. Troubleshooting capability is also important to the installation process. Due to system complexities, if good troubleshooting capability is not available, the installer will need to spend a lot of time researching the problem and typically require the help of programmers to determine the problem.

Documentation

The assessment of documentation for each of the dimensions could have been included in the assessments of those dimensions. Documentation has been broken out as a separate dimension due to its importance and to provide the ability to assess the usability of the system documentation as an integrated whole. For any system of the complexity of a typical DSS, documentation is crucial to the ability of each user to do his or her job. While the usability of the documentation itself is an important issue, good documentation for each dimension improves the usability for that dimension.

For the Programming dimension, good documentation is important for the overall system design, whether it is UML or some other high-level modeling methodology. As one programmer

noted, “Programmers need something to refer to when the muse is running.” For code-level documentation, enough detail in some form is needed so that new programmers can take over where other programmers left off or perform maintenance. Also, communication among project team members is facilitated by good code documentation. The assessment of programming documentation will be based on whatever information can be obtained—interviews with programmers, code inspection, or other means.

Documentation for the End User Needs and Goals dimension should state the overall system concept, the users’ needs, and how those needs are met by the system. While this documentation should be concise, its length will depend in part on the complexity and size of the DSS under consideration.

The ability to access on-line detailed help and system information is a requirement of good software design. The end user interface should have both a help system and on-line system documentation. In addition, written materials should be provided as needed. Ideally, help system content and off-line documentation for end users should be written by professional writers who are conversant with the technology and system.

Documentation for the Installation dimension should include sufficient instructions for installers to install and hook up all hardware and software, including troubleshooting instructions. Writing must be targeted to the people with the requisite level of expertise for system installation. If, during installation, days (or weeks) of troubleshooting are needed, with phone calls to programmers, this would be considered a failure. A log should be kept of everything done to the system during and after installation. A well-documented log file will make troubleshooting and maintenance easier.

Documentation for Training is important primarily for end users, installers, and programmers. End users need training documentation to teach them how to use the system (which may be supplemented with a trainer). Although in certain situations the interface design with its built-in help may be adequate, professional trainers may be needed. Installers may need training, although the installation documentation may obviate the need for their training. Programmers may need training when they are new to a system.

Framework Attributes

The following list specifies the attributes in the first version of the holistic usability framework for a DSS. Each of these attributes can be linked directly to one or more of the three key usability measures: effectiveness (ability to successfully do the job), efficiency (speed, which includes learnability and memorability), and user satisfaction. A validation methodology, used to refine the framework, is discussed in Chapter 4. Attributes in the list below that passed a validation test were kept in the framework; attributes that failed the validation test were removed from the framework.

End User Needs and Goals

1. The end users' needs and goals with the system should be fully supported.
2. The end users should be satisfied with the system.
3. Lessons learned should be tracked for future improvements.
4. The system hardware and software should be reliable.

5. If a vendor-provided system, vendor support should be adequate. (Note: This attribute is included for future reference. It would only be assessed during private consulting, due to the sensitive nature of this measure.)

End User Interface(s)

1. The end users should be satisfied with the interface(s).

2. The overall quality of the interface(s) should be good.

control features

3. There should be a central control and monitoring point.

4. One should be able to change parameters in individual simulations from a central interface.

5. One should be able to start and stop individual simulations from a central control interface.

6. One should be able to see others who are logged into the system and communicate with them.

7. Exception handling should be adequate. This means that when something goes wrong (e.g., the system freezes), the operator or user should have system support to locate, recover from, and identify the problem.

data visualization and analysis

8. The display should be able to show the relevant variables in all simulations running simultaneously.

9. It should be possible to review data from several simulation scenarios simultaneously.

10. One should be able to save, analyze, and export statistics.

11. Data visualization capability should be good.

12. Information from various simulations should be able to be combined in a way that allows good understanding of interrelationships and results.

Programming

1. The programming environment's complexity for the system should be as low as possible. Care should be taken to make software choices that minimize complexity.

2. The number of simulations written with proprietary simulation packages should be minimized. The inability to see source code and have direct interfaces to proprietary simulations reduces the ease of constructing the system and limits options.

3. The number of software "wrappers" required around individual simulations should be minimized.

4. The DSS infrastructure should be as easy to use as possible. For example, later versions of the HLA-RTI have a higher usability than earlier versions, due partly to better naming conventions and fewer bugs.

5. For a given number of simulations, the amount of coding required should be minimized.

6. The lower the level of expertise needed for coding, the more productive the programming team will be.

7. Programmers should be satisfied with the programming environment.

8. The shorter the time to get programmers on board and up to speed in productive work, the more efficient their time will be. The amount of time it takes a new programmer to become productive should be minimized.

9. The system should be designed to be as easy to program as practicable.

10. The software infrastructure that allows the greatest ease of connecting simulations to it should be chosen.

11. The data formats between simulations should be compatible.

12. Configuration control between simulations should be adequate.

Installation

1. A detailed log should be kept of all installation details, including troubleshooting, problems encountered, and their solutions.

2. Personnel of average ability, but taught the job, should be able to install the system components.

3. The skills needed for the installation team should be specified.

4. The installation process should be as easy as practicable.

5. The skill level required to install the system should not be too high.

6. The number of people required to install the system should be minimized.

7. Effective troubleshooting capability should be provided for the system installers.

8. The time required to install the system should be minimized.

9. Installers should be satisfied with the installation scenario.

Training

1. The training should be effective, i.e., it should prepare the trainee for what he or she needs to do.

2. The training should be efficient, i.e., it should accomplish its goals in an optimal amount of time.

3. Written materials should be available to support the training.

4. On-line materials should be available to support the training.

5. The training should be geared to the knowledge/skill level of the audience.

6. Trainers should be satisfied with the training scenario.

7. Trainees should be satisfied with the training.

8. The overall quality of the installation training should be good.

9. The overall quality of the end user interface training should be good.

10. The overall quality of the programmer training should be good.

Documentation

programming

1. The quality of programming code-level documentation should be sufficient.

2. Software design characteristics should be clearly specified in a conceptual design (e.g., using the Unified Modeling Language), independent of any programming language.

e*nd user needs and goals*

3. The end user needs and goals should be clearly documented.

training

4. The quality of training documentation should be sufficient.

installation

5. The quality of installation documentation should be sufficient.

end user interface

6. The quality of written end user interface documentation should be sufficient.

7. The quality of on-line help and support for end users should be sufficient.

Measurement of Attributes of Framework Dimensions

Below is a table that shows measurement details for attributes of each of the dimensions. The measurements taken will be used in summation equations to obtain summary measurements for each of the dimensions' usabilities.

Table 1. Holistic Usability Framework Measurements

DIMENSIONS	MEASURES
End User Needs and Goals	
The end users should be satisfied with the system.	1 to 7
The end users' needs and goals with the system should be fully supported.	1 to 5
Lessons learned should be tracked for future improvements.	Y/N (5/0)
The system hardware and software should be reliable.	1 to 5
If a vendor-provided system, vendor support should be adequate.	1 to 5
End User Interface(s)	
The end users should be satisfied with the interface(s).	1 to 7
The overall quality of the interface(s) should be good.	1 to 5
control features	
There should be a central control and monitoring point.	Y/N (5/0)

DIMENSIONS	MEASURES
One should be able to change parameters in individual simulations from a central interface.	Y/N (5/0)
One should be able to start and stop individual simulations from a central control interface.	Y/N (5/0)
One should be able to see others who are logged into system and communicate with them.	Y/N (5/0)
Exception handling should be adequate.	1 to 5
data visualization and analysis	
The display should be able to show the relevant variables in all simulations running simultaneously.	Y/N (5/0)
It should be possible to review data from several simulation scenarios simultaneously.	Y/N (5/0)
One should be able to save, analyze, and export statistics.	Y/N (5/0)
Data visualization capability should be good.	1 to 5
Information from various simulations should be able to be combined in a way that allows good understanding of interrelationships and results.	1 to 5
Programming	
The programming environment's complexity for the system should be as low as possible.	1 to 5
The number of simulations written with proprietary simulation packages should be minimized. (Proprietary simulations are those with closed code that cannot be seen or modified.)	Integer
The number of software "wrappers" required around individual simulations should be minimized.	Integer
The DSS infrastructure should be as easy to use as possible.	1 to 5
For a given number of simulations, the amount of coding required should be minimized.	lines of code
The lower the level of expertise needed for coding, the more productive the programming team will be.	1 to 5
Programmers should be satisfied with the programming environment.	1 to 7
The amount of time it takes a new programmer to become productive should be minimized.	Weeks
The system should be designed to be as easy to program as practicable.	1 to 5
The software infrastructure that allows the greatest ease of connecting simulations to it should be chosen.	1 to 5
The data formats between simulations should be compatible.	1 to 5
Configuration control between simulations should be adequate.	1 to 5
Installation	
A detailed log should be kept of all installation details, including troubleshooting, problems encountered, and their solutions.	Y/N (5/0)
Personnel of average ability, but taught the job, should be able to install the system components.	Y/N (5/0)
The skills needed for the installation team should be specified.	Y/N (5/0)
The number of people required to install the system should be minimized.	Integer
Effective troubleshooting capability should be provided for the system installers.	1 to 5
The time required to install the system should be minimized.	weeks or days
The installation process should be as easy as practicable.	1 to 5
The skill level required to install the system should not be too high.	1 to 5
Installers should be satisfied with the installation scenario.	1 to 7
Training	

DIMENSIONS	MEASURES
The training should be effective, i.e., it should prepare the trainee for what he or she needs to do.	1 to 5
The training should be efficient, i.e., it should accomplish its goals in an optimal amount of time.	1 to 5
The trainees should be satisfied with the training.	1 to 7
The trainers should be satisfied with the training scenario.	1 to 7
Written materials should be available to support the training.	Y/N (5/0)
On-line materials should be available to support the training.	Y/N (5/0)
The training should be geared to the knowledge/skill level of the audience.	Y/N (5/0)
The overall quality of the installation training should be good.	1 to 5
The overall quality of the end user interface training should be good.	1 to 5
The overall quality of the programmer training should be good.	1 to 5
Documentation	
The quality of programming code-level documentation should be sufficient.	1 to 5
Software design characteristics should be clearly specified in a conceptual design.	1 to 5
The end user needs and goals should be clearly documented.	1 to 5
The quality of training documentation should be sufficient.	1 to 5
The quality of installation documentation should be sufficient.	1 to 5
The quality of written end user interface documentation should be sufficient.	1 to 5
The quality of on-line help and support for end users should be sufficient.	1 to 5

Improvement in the measurements used to assess the individual dimensions indicates an improvement of the usability for that dimension. The improvement of usability for a dimension means that the effectiveness, efficiency, and/or user satisfaction for the user types associated with that dimension will improve. For example, a difficult-to-use software infrastructure for distributed simulation will result in less efficient programming work and dissatisfied programmers. (Distributed simulation infrastructures are inherently complex, partly because they must control the timing of different simulation models simultaneously.) An improvement in the ease of use of the software infrastructure will result in an improvement in programmer efficiency and satisfaction. While this is an obvious relationship, stated in formal terms: it is assumed that increasing the usability of the measures used for each dimension will increase the effectiveness, efficiency, and/or satisfaction for users of that dimension. In the development of the framework,

problems were observed in systems due to less-than-optimal characteristics of the framework dimensions.

As discussed previously, traditionally usability is measured by three metrics: efficiency, effectiveness, and satisfaction. As stated by ANSI (2001), "The choice of measures depends on the goals of a particular study, characteristics of the users, the specific tasks, and context-dependent features…" (p. 9). In this holistic usability framework, satisfaction is being measured directly by asking the users. In this initial framework and in the two system assessments, it is measured on a scale of 1 to 7; in the final framework this is changed to 1 to 5 based on input from a reviewer (discussed in lessons learned). However, because satisfaction is a separate construct its weight in an overall usability metric can vary depending on its relative importance (Nielsen, 2001). For instance for a game, satisfaction might be deemed more important than for a military battle simulation. Out of 55 metrics in the final framework, six are satisfaction measures. I initially chose a scale of 1 to 7 to give the satisfaction metric slightly more weight than the other attribute metrics.

The next step in this methodology, discussed in the next chapter, was to survey professionals in order to obtain their feedback in order to validate and improve the framework.

CHAPTER FOUR: VALIDATION AND FRAMEWORK REFINEMENT

Validation Survey

Often in human factors work, empirical results from experiments with humans in the laboratory are validated to ensure those results are generalizable to the reference situation in the field. Validation is often straightforward for studies of details of human-system interaction, such as the particulars of using a mouse. For a system-wide validation for aspects of a large system, validation may require a combination of common sense, expert evaluation, and field observation of factors.

Several DSSs were studied while developing the framework, and the dimensions of the framework and their effects on usability were observed. In addition, discussions were held with vendors of distributed simulation infrastructures, during which the importance of some of the attributes were mentioned. Insofar as the framework is based on actual system study and observation, the results are inherently valid for those systems. Further validation was obtained by using a survey to gain feedback about the framework from users of DSSs. Although a few open-ended questions in the survey were asked to gather information that may be used to improve the framework, the main objective of the survey was to validate the framework. This survey is shown in Appendix A.

The survey was presented on-line via a Web site. This allowed for fast data collection and easy survey availability to participants. Questions consider how the attributes of the framework's dimensions affect standard usability measures of efficiency, effectiveness, and user satisfaction. The opinions and perceptions of people who work with distributed simulation—experts in the

field—were obtained via the survey. Thus, expert opinion was gathered about the measures used in the framework.

A pilot survey was administered to a small group of participants in order to discover any problems with the survey design before it was administered to a larger sample of participants. Based on the pilot survey, some changes were made to improve the questions.

The survey included Likert scale, multiple choice, and open-ended questions as needed to ascertain the desired data from the respondents. Care was taken to make the survey as short as possible while attempting to gather the needed data.

Tables 2 and 3 below show the number of different types of users among the participants and the number of types of DSSs these participants have experience with. Because multiple responses were allowed, the totals are greater than the number of participants.

Table 2. Types of Users Surveyed

Type of User	Percent	Number of Participants
manager	29.8	18
researcher	71.9	46
end user	24.6	14
programmer	36.8	23
designer	49.1	30
trainer	8.8	6
installer	1.8	1
other	1.8	2

Table 3. Types of Distributed Simulation Systems Participants Have Experience With

Type of DSS	Percent	Number of Participants
military	40.4	25
entertainment	7.0	5
aerospace	47.4	28
business	19.3	13
engineering	36.8	25
medical	3.5	2
pharmaceutical	3.5	2
other	7.0	7

The average number of years of distributed simulation experience the participants had was 9.6 years, with a median of 9.0, and a standard deviation of 6.35. The range was one to thirty years. There were 63 participants.

The variety of organizations from whom anonymous participants responded, as reported in the survey, includes Vrije Universiteit Amsterdam, SPARTA, ACM, British Ergonomics Society, SIE (Società Italiana di Ergonomia), Fraunhofer Institute, a "semiconductor company," Brazier Motti, NASA, "academic research," University of Texas Southwestern Medical Center, San Diego State University, General Dynamics UK, ORNL, USN NAVAIR, Ericsson, DoN, US Army, Bucknell University, SICS, DLR, University of Delaware, "reinsurance brokerage firm," Navy, "independent," Georgia Tech, UTA, ONERA, Empirix, MDA, DMSO, United Space Alliance, SJSUF, AFAMS, NAVAIR ORLANDO TSD, UCF, Walt Disney World, FAA, DoD, Convergys, and SAIC.

The survey participants were first given an introduction to the concept of the holistic usability framework for DSSs. Then a series of questions was asked that address attributes of each dimension in the framework.

The concept of the survey was to validate framework attributes by asking experts if they agree with a statement. Each statement is linked to a usability attribute for a dimension; if a participant agreed with a statement, they validated the attribute associated with the statement. There are were two negative answers (“strongly disagree,” “disagree”), one neutral answer (“neutral”), and two positive answers (“agree,” “strongly agree”). Answer responses were transformed into dichotomous variables: the two negative and neutral answers were combined to "no"; the agree and strongly agree answers were combined to "yes."

The validation technique used is as follows. If more than 50 percent of the participants answered "yes" to the question linked to an attribute, this validated that attribute. The rationale for using 50 percent as the decision point is that if greater than 50 percent of the participants answer "yes" this means that the majority of the experts agree with the statement. The dichotomous answers were analyzed using an hypothesis test for proportions. In addition to the hypothesis test, a 95 percent confidence interval was calculated for each tested attribute.

The hypothesis for testing the validation of each attribute, using the associated survey question, is:

H_0: $p = 0.5$

H_1: $p > 0.5$

The analysis for testing proportions is performed using binomial probabilities. A normal approximation to the binomial can be used when p "is not extremely close to 0 or 1" (Walpole and Myers, 1978, p. 262). It is necessary for the assumptions $np > 4$ and $nq > 4$ to be true for the normal approximation to the binomial to hold; this is usually the case for large-sample dichotomous survey questions, but was tested for each question. (For any question for which these assumptions do not hold, an exact binomial calculation can be used for the hypothesis test.)

The equation to calculate the normal z value is $z = \frac{p - 0.5}{\sqrt{(p)(q)/n}}$.

The probability that $Z \leq z$ is read from a two-tail normal probability distribution table. Subtracting this value from one yields the significance of the test. The significance level (or critical area) chosen for determining whether or not an attribute was validated is 0.05. If the significance level is 0.05 or smaller, we reject the null hypothesis and conclude that greater than 50 percent of the survey population agrees with the statement.

Using this procedure, 34 attributes were validated, and nine attributes were shown to be invalid (as shown in Table 4 below). Attributes that were not validated were dropped from the framework. The spreadsheet used for the calculations for the hypothesis tests is shown in Appendix B.

A 95 percent confidence interval for the proportion of respondents answering "yes" in the survey was also calculated. Referring again to Walpole and Myers (1978), when the sample size $n \geq 30$, "a $(1 - \alpha)100$ percent confidence interval for the binomial parameter p is approximately

$$\hat{p} - z_{\alpha/2}\sqrt{\frac{\hat{p}\hat{q}}{n}} < p < \hat{p} + z_{\alpha/2}\sqrt{\frac{\hat{p}\hat{q}}{n}}$$

where $\hat{p}$ is the proportion of successes in a random sample of size n, $\hat{q} = 1 - \hat{p}$, and $z_{\alpha/2}$ is the value of the standard normal curve leaving an area of $\alpha/2$ to the right " (p. 210). In the present case, the 95 percent confidence interval is give by

$$\hat{p} \pm 1.96\sqrt{\frac{\hat{p}\hat{q}}{n}}.$$

As seen from the results in Table 4, the significance for the attributes that were validated is usually much greater than 0.05.

Refer to table 4 below. Column one shows the attributes by dimension. Column two shows the measures. Columns three through five indicate which usability factor(s) the validation survey questions test. If the attribute is one of the eleven not tested in the survey, columns three through five indicate the usability factors affected by that attribute. Columns six through ten show the numbers of the corresponding validation survey question, the p values of the validation hypothesis tests, upper and lower 95 percent confidence interval values, and whether or not an attribute was validated.

Table 4. Validation Survey Results

Dimensions and Attributes	Measures	Effectiveness	Efficiency	Satisfaction	Survey Question	p value	C.I. lower limit	C.I. upper limit	Validated?
End User Needs and Goals									
The end users should be satisfied with the system.	1 to 7			X	N/A	N/A	N/A	N/A	N/A
The end users' needs and goals with the system should be fully supported.	1 to 5	X			N/A	N/A	N/A	N/A	N/A
Lessons learned should be tracked for future improvements.	Y/N	X	X	X	N/A	N/A	N/A	N/A	N/A
The system hardware and software should be reliable.	1 to 5	X		X	N/A	N/A	N/A	N/A	N/A
If a vendor-provided system, vendor support should be adequate.	1 to 5	X	X	X	N/A	N/A	N/A	N/A	N/A
End User Interface(s)									
The end users should be satisfied with the interface(s).	1 to 7			X	N/A	N/A	N/A	N/A	N/A
The overall quality of the interface(s) should be good.	1 to 5	X	X	X	N/A	N/A	N/A	N/A	N/A
control features									
There should be a central control and monitoring point.	Y/N		X		6	0.000	0.854	0.987	Yes

Dimensions and Attributes	Measures	Effectiveness	Efficiency	Satisfaction	Survey Question	p value	C.I. lower limit	C.I. upper limit	Validated?
One should be able to change parameters in individual simulations from a central interface.	Y/N		X		7	0.000	0.767	0.943	Yes
One should be able to start and stop individual simulations from a central control interface.	Y/N		X		8	0.000	0.732	0.919	Yes
One should be able to see others who are logged into system and communicate with them.	Y/N		X		9	0.000	0.621	0.840	Yes
Exception handling should be adequate.	1 to 5		X		10	0.000	0.811	0.966	Yes
data visualization and analysis									
The display should be able to show the relevant variables in all simulations running simultaneously.	Y/N		X		11	0.184	0.433	0.682	No
It should be possible to review data from several simulation scenarios simultaneously.	Y/N		X		12	0.000	0.675	0.880	Yes
One should be able to save, analyze, and export statistics.	Y/N		X		13	0.000	0.727	0.918	Yes
Data visualization capability should be good.	1 to 5	X			14	0.000	0.854	0.987	Yes
Information from various simulations should be able to be combined in a way that allows good understanding of interrelationships and results.	1 to 5			X	15	0.000	0.675	0.880	Yes
Programming									
The programming environment's complexity for the system should be as low as possible.	1 to 5		X		16	0.940	0.281	0.525	No

Dimensions and Attributes	Measures	Effectiveness	Efficiency	Satisfaction	Survey Question	p value	C.I. lower limit	C.I. upper limit	Validated?
The number of simulations written with proprietary simulation packages should be minimized. (Proprietary simulations are those with closed code that cannot be seen or modified.)	integer		X		17	0.001	0.572	0.805	Yes
The number of software "wrappers" required around individual simulations should be minimized.	integer		X		18	0.008	0.526	0.764	Yes
The DSS infrastructure should be as easy to use as possible.	1 to 5		X		19	0.000	0.767	0.943	Yes
For a given number of simulations, the amount of coding required should be minimized.	lines of code		X		22	0.302	0.407	0.660	no
The lower the level of expertise needed for coding, the more productive the programming team will be.	1 to 5		X		23	0.874	0.306	0.551	no
Programmers should be satisfied with the programming environment.	1 to 7			X	N/A	N/A	N/A	N/A	N/A
The amount of time it takes a new programmer to become productive should be minimized.	Weeks		X		20	0.000	0.675	0.880	yes
The system should be designed to be as easy to program as practicable.	1 to 5			X	21	0.695	0.344	0.592	no
The software infrastructure that allows the greatest ease of connecting simulations to it should be chosen.	1 to 5		X		24	0.000	0.876	0.997	yes
The data formats between simulations should be compatible.	1 to 5		X		25	0.000	0.854	0.987	yes
Configuration control between simulations should be adequate.	1 to 5		X		53	0.000	0.761	0.976	yes

Dimensions and Attributes	Measures	Effectiveness	Efficiency	Satisfaction	Survey Question	p value	C.I. lower limit	C.I. upper limit	Validated?
Installation									
A detailed log should be kept of all installation details, including troubleshooting, problems encountered, and their solutions.	Y/N		X		26	0.000	0.832	0.977	yes
Personnel of average ability, but taught the job, should be able to install the system components.	Y/N		X		27	0.034	0.492	0.734	yes
The skills needed for the installation team should be specified.	Y/N		X		28	0.000	0.585	0.812	yes
The number of people required to install the system should be minimized.	integer		X		29	0.000	0.597	0.823	yes
Effective troubleshooting capability should be provided for the system installers.	1 to 5	X			30	0.000	0.791	0.955	yes
The time required to install the system should be minimized.	weeks or days		X		31	0.996	0.221	0.457	no
The installation process should be as easy as practicable.	1 to 5		X		32	0.901	0.297	0.542	no
The skill level required to install the system should not be too high.	1 to 5		X		33	1.000	0.152	0.373	no
Installers should be satisfied with the installation scenario.	1 to 7			X	N/A	N/A	N/A	N/A	N/A
Training									
The training should be effective, i.e., it should prepare the trainee for what he or she needs to do.	1 to 5		X		34	0.000	0.805	0.965	yes
The training should be efficient, i.e., it should accomplish its goals in an optimal amount of time.	1 to 5		X		35	0.153	0.441	0.688	no
Are the trainees satisfied with the training?	1 to 7			X	N/A	N/A	N/A	N/A	N/A

Dimensions and Attributes	Measures	Effectiveness	Efficiency	Satisfaction	Survey Question	p value	C.I. lower limit	C.I. upper limit	Validated?
Are the trainers satisfied with the training scenario?	1 to 7			X	N/A	N/A	N/A	N/A	N/A
Written materials should be available to support the training.	Y/N		X		36	0.008	0.526	0.764	yes
On-line materials should be available to support the training.	Y/N			X	37	0.034	0.492	0.734	yes
The training should be geared to the knowledge/skill level of the audience.	Y/N		X		38	0.000	0.849	0.987	yes
The overall quality of the installation training should be good.	1 to 5	X			39	0.000	0.763	0.941	yes
The overall quality of the end user interface training should be good.	1 to 5		X		40	0.000	0.824	0.976	yes
The overall quality of the programmer training should be good.	1 to 5		X		41	0.000	0.824	0.976	yes
Documentation									
The quality of programming code-level documentation should be sufficient.	1 to 5		X		42	0.000	0.773	0.951	yes
Software design characteristics should be clearly specified in a conceptual design.	1 to 5		X		43	0.000	0.847	0.987	yes
The end user needs and goals should be clearly documented.	1 to 5	X			44	0.000	0.844	0.986	yes
The quality of training documentation should be sufficient.	1 to 5	X			45	0.000	0.824	0.976	yes
The quality of installation documentation should be sufficient.	1 to 5		X		46	0.000	0.714	0.913	yes
The quality of written end user interface documentation should be sufficient.	1 to 5		X	X	47, 48	0.000, 0.000	0.714, 0.735	0.913, 0.926	yes
The quality of on-line help and support for end users should be sufficient.	1 to 5			X	49	0.000	0.603	0.831	yes

Validation of the Other Attributes

Eleven attributes were not put through the survey validation process. Six of these are user satisfaction, a generally accepted usability measure (and a requirement per both ISO and ANSI usability standards as previously discussed). One is overall quality of the interface, which represents a basic usability evaluation of an interface. One is a measure of how well the user's goals are achieved with the system. This is similar to utility and is validated by the fact that without this goal being at least partially met, the system is useless to the user. Also, this is a key goal of any simulation (or product). This is a unique approach taken in the holistic framework, which expands the concept of usability. Utility is not usually included in a usability assessment. The attribute that tracks lessons learned for future improvements was suggested by a committee member. This attribute is validated by the fact that there is a need for continuous improvement. System reliability is validated by observation. With any DSS, it seems to come up during on-site inspections. Vendor support is also validated by observation and goes with reliability. Due to commercial and ethical concerns, it would only be reported in private consulting.

The next chapter discusses the application of the framework to real-world systems.

CHAPTER FIVE: APPLICATION OF FRAMEWORK TO TWO DISTRIBUTED SIMULATION SYSTEMS

Approach

Real-world application is desirable in order to demonstrate the utility of the framework and to obtain feedback and knowledge for future improvements. The evaluation of a system requires studying the dimensions of the system's holistic usability, while keeping in mind:

- the need for qualitative improvements noticed by observation
- system strengths and weaknesses as evaluated by measurement

The amount of time required to assess the holistic usability of a DSS will vary with the complexity of the system and the level of depth desired in the assessment. It is recommended that the person performing the assessment be a person competent in the field of usability and knowledgeable about simulation technology. As a rule of thumb—which will very depending on the situation—it is suggested that eight days be planned for the assessment. One day will be needed to meet personnel involved, obtain appropriate management approvals, and become familiar with the system. Six days can be used to assess the dimensions, at an average rate of one dimension per day. The eighth day will be used to finish writing the report and present the findings.

When the attributes are measured singly, with one measurement, this is a one-level measurement. When attributes are measured with multiple measurements, this is a multi-level measurement. Multi-level measurements will be combined into single measurements using summation equations. Thus if there were one user interface in the system rated as a 4, that would

be the value of the end user interface quality attribute. If there were three different end user interfaces, rated 3, 4, and 5, the value of the end user interface attribute would be

$$\frac{\sum_{i=1}^{n} x}{n} = \frac{3+4+5}{3} = 4.$$

In this manner, attribute measurements can have as many levels as necessary to provide the amount of detail required for their measurement.

Each dimension was measured using a summation of metrics of the dimension's attributes. The weights of each attribute of each dimension were chosen based on an evaluation of the attribute's importance. A variable was assigned to each dimension:

UN ≡ user needs and goals

EUI ≡ end user interface

I ≡ installation

P ≡ programming

T ≡ training

D ≡ documentation

The ideal score for each dimension is 100 percent of the possible points from a perfect score summing the dimension's attributes.

The assessment will be reported as six individual scores. Determining a composite score of the overall holistic usability was considered, but it was decided that the most value in the assessment is to look at each dimensional assessment separately, comparing the dimensions and seeing which dimensions need the most improvement. This facilitates the effective allocation of resources to target the components most in need of usability enhancement.

In order to combine metrics of the different attributes of the dimensions into dimensional scores, some assumptions and careful judgment of the importance of each attribute is needed. Obtaining maximum values for all attributes would result in a perfect score for that dimension. Some attributes will be rated on a scale of one to five. User satisfaction metrics will be measured on a scale of one to seven. Dichotomous variables, such as the yes/no evaluations, shall be assumed to take on two possible values, typically 0 and a positive or negative integer, depending on whether the variable's presence or absence in the dimension results in a positive, neutral, or negative effect on the system.

Two systems were evaluated. The evaluations include quantitative measurement of all of the attributes in the holistic usability framework for distributed simulation and a discussion of findings that includes qualitative aspects that were noted during the system inspection. The first system analyzed was the prototypical Virtual Test Bed that was developed at UCF for NASA. The second system was the Aviation Research Training Tool (ARTT) Radar at Embry-Riddle Aeronautical University.

The procedure for assessing the holistic usability of a DSS is as follows. The evaluator will become familiar with the salient aspects of the system. A sample of key user types will be given a survey concerning user satisfaction. These attributes require user feedback to measure. System documentation, interface(s), design, and programming/infrastructure aspects will be evaluated from study, observation, and discussion with personnel. A concise report will be generated using inputs from the above process that

- summarizes the metrics of dimensional attributes
- lists the strengths and weaknesses
- makes recommendations for improvements

Each of the system assessments is discussed in four sections: (1) system description, (2) assessment details and observations, (3) summary of results, and (4) strengths, weaknesses, and recommendations.

Who Does What in an Assessment

The holistic usability assessment of a system may be performed by one person or with a team approach. The person leading the assessment should be a usability expert, preferably also having expertise in the type of DSS or the domain the system is in. If evaluating a large system and the resources are available, a team of two to five usability experts could be used to lead the assessment. Users and managers will be recruited, surveyed, observed working with the system, and/or interviewed as required.

The six satisfaction metrics in the framework will be measured by asking the users their level of satisfaction with the system, either verbally or with a survey instrument.

End User Needs and Goals

Users will be asked if their goals with the system are fully supported and to rate this attribute. As an alternative, a list can be made of all end user goals and a check made to determine if they are met. Management should be asked if lessons learned are tracked. Reliability can be assessed by talking to users and managers. The reliability attribute is intended to be a rough estimate of how reliable the system is relative to the user's needs. If desired by the client, a formal reliability assessment can be used. Vendor support would be rated by the owners/users of the system.

End User Interfaces(s)

For determining a quick look at the overall quality of the end user interface, an expert in usability is required. In order to evaluate the usability of an interface by any of several evaluation techniques—whether "quick and dirty," user testing, field studies, or predictive, someone is needed with the requisite background in usability (Preece, Rogers, & Sharp, 2002, p. 343-344.) Usability evaluation requires training and developed expertise; it could not be left to the users, most of whom will not know basic usability principles (Mayhew, 1999; Nielsen, 1993; Jordan, 1998). Their input should be sought by all practicable means, however, since their interface in working with the system is the subject at hand. Expert evaluations are more effective when the usability specialist is also an expert in the technology in use (Rubin, 1999). Although usability expertise is needed to evaluate an interface, it is possible that with training in heuristic analysis, nonexperts can find usability problems (Nielsen, 1994). However, the attribute measuring overall quality of the end user interface is a quick look at the interface, not an in-depth usability analysis of it. For the purposes of the assessment, it is best to have an expert make a quick examination of the interfaces. If possible, observing users use the interface in an informal field study also gives useful information; if problems are observed, they should be noted.

Most of the attributes under the *control features* section can be assessed by quick inspection, except for exception handling. For this attribute, discussion with users should reveal if exception handling is adequate.

Some attributes under the data visualization and analysis section can be determined by inspection or documentation (e.g., the ability to save statistics). Depending on the situation, the adequacy of data visualization and the ability to understand interrelationships and results may require user input.

Programming

Even if the person in charge of the assessment knows the programming languages used in the simulations and is knowledgeable about software conceptual design processes, assessing the programming dimension will require talking to the system programmers and reviewing relevant documentation. While the programming dimension represents the area where the most resources are usually spent in simulation and also the area of highest complexity, programmers are usually easily able to clearly articulate where their problems are and what is needed to alleviate them, and more than willing to share this information.

Installation

The installation dimension consists of six attributes, one of which is a satisfaction measure. The other five can be assessed by either talking to the installers or giving them a survey. Other than the satisfaction metric, which must be measured by asking the users, a usability specialist is not needed to obtain these measurements.

Training

Two of the six training dimension attributes are satisfaction measures, one for the trainers, the other for the trainees. Assessment of the training dimension can be performed by the usability specialist doing the overall assessment or any other knowledgeable person. Observation of training in progress is recommended, as is talking to both trainers and trainees.

Documentation

Assessment of the documentation dimension requires inspecting the documentation. While every word does not need to be read, a thorough look at on-line, help, and written documentation is required. Asking various users for their opinions of the documentation will

help to reveal any problems. Programmers will need to be asked the quality of the code-level documentation and the quality of the conceptual software design documentation.

Virtual Test Bed Assessment

VTB System Description

The VTB consists of five HLA-RTI federates configured to simulate a virtual spaceport: the Virtual Range, Launch Pad, Control Room, Monte Carlo, and Weather Expert System (WES). Four of the federates are programmed in Arena and interface with the RTI through an adapter that was developed by the National Institute of Standards and Technology (NIST). The NIST-developed distributed manufacturing adapter, written in C++, was developed to allow commercial software packages to interface with the HLA-RTI (McLean and Riddick, 2000). WES is a simulation-supporting live participant rather than a simulation in itself; its adapter is written in Java.

The five federates operate as follows. The Launch Pad model simulates the flow of the space shuttle as it arrives at Kennedy Space Center, is processed through the Orbital Processing Facility and the Vehicle Assembly Building, and travels to the launch pad. Upon arrival at the pad, a message is sent to the Control Room informing it that the shuttle is ready for launch. If conditions are good for a launch, authorization is given, after which the Launch Pad shows the shuttle circling the earth and eventually landing, if the flight is successful. The Control Room checks for failures in four systems and queries the Weather Expert System. If conditions are good, it sends the go ahead to the Launch Pad. The Weather Expert System collects weather

information from several Web sites and uses it to determine if conditions are good for a launch. When a launch occurs, the Monte Carlo model determines if a failure occurs that would cause a disaster. If a failure occurs, the Virtual Range model determines the location of the accident in space and the amount of contaminants released into the atmosphere. A CALPUFF air quality model uses the Weather Expert System-provided weather information to determine contaminant concentrations around the accident site. Then ArcView is used to create a map showing where contaminant concentrations exceed safe limits. SpatialAnalyst shows the population exposed on the ArcView-generated map, obtaining the population data from LandScan. The Virtual Range displays the number of people exposed on a map of the affected area.

Initially, the VTB required a person to operate each computer a model was running on separately. A prototypical GUI was designed in a NASA-funded project to improve the VTB's usability. In this GUI design, the five federates connect to both the HLA-RTI and WebLogic Server. The GUI communicates with the federation models through WebLogic Server and also contains a control federate that communicates with the federation via the RTI. This allows for control of the individual simulations through WebLogic Server, while the HLA-RTI Control Federate allows starting and stopping the distributed simulation. A help module, written in Java, was incorporated into the GUI to provide the capability to offer on-line help and explanatory information. The figure below shows the configuration of the VTB and its control GUI.

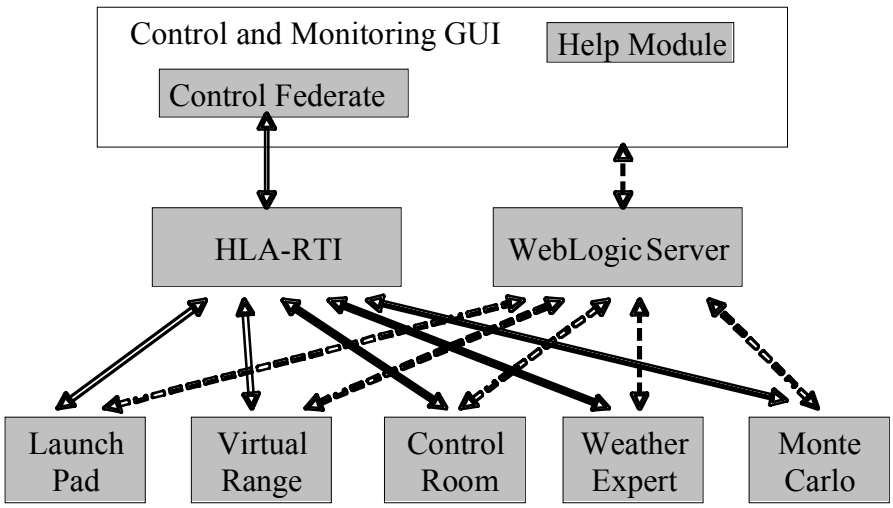

Figure 7. Virtual Test Bed GUI Design Approach

VTB Assessment Details and Observations

The end user goals of the VTB, from a NASA customer viewpoint, were to create a test bed for virtual spaceport simulations, integrating several simulations in a prototype that demonstrates the feasibility of developing a virtual space port, and to develop capabilities in distributed simulation. In addition, a usability improvement project for the VTB developed the prototypical GUI. NASA was completely pleased with this effort, showing that these end user goals were accomplished. The lessons learned are being tracked for future improvements.

System reliability has been weak, partly due to unknown configuration changes on some workstation computers. Since the VTB is a prototypical system that has not undergone extensive development, low scores would be expected for some of the attributes and dimensions.

The prototypical GUI provides a central control and monitoring point. Future efforts with the VTB should include work to expand the capabilities of the GUI for control, configuration, and data visualization. The ability to change parameters in individual simulations from the GUI

is needed. The ability to review data from different simulation runs as well as to save and export statistics is needed. On individual computers, the data visualization is good for the individual simulations. The ability to see this information on remote computers—especially one running the control GUI—would be helpful to the end user.

While the team worked to developed the control GUI, a number of issues concerning the programming dimension arose. The lack of good code-level documentation meant that programmers new to the project had to spend time learning how the system was programmed, rather than reading a clear explanation. While some documentation exists, it is not adequate for a programmer to develop a full understanding of how components work and interact. In addition to learning how to access HLA-RTI functions in the wrappers for the individual simulations, new programmers needed to learn how to use the HLA-RTI version 1.3. Learning how to program the HLA-RTI and access individual simulation wrappers took the programmers about twelve weeks. This time could have been shortened if they had received a training course in how to use the HLA-RTI. Better documentation would have also made them more productive faster.

Installation of the system, although done in a systematic fashion, has historically required extensive troubleshooting and phone calls to experts no longer associated with the project. A graduate student (who is an expert programmer) complained that the lack of a troubleshooting guide for the Distributed Manufacturing Adapter was a major problem, although there were detailed installation instructions for the adapter. One installation problem, associated with a dynamic link library issue in Microsoft Windows XP (the operating system on which the adapter and its associated simulation package was being installed), had cost a total of several days of work (counting all personnel involved), and still was not resolved. A similar installation problem, which occurred a year earlier, had taken weeks to resolve. While such problems can be

expected in a development environment, this presents an excellent opportunity for improving the usability for the installers and programmers.

VTB Summary of Results

The calculations for the values of each dimension are the summations of the individual attributes for each dimension. The dimensional scores are the total number of points divided by the maximum possible number of points. For this system, the results of the calculations are shown in the table below.

Table 5. Assessment Metrics for the Virtual Test Bed

Dimensions/Attributes	*Measurement details*	*Measurements*	*Max possible score*	*Dimensional scores*	*Notes*
End User Need and Goals			22	0.91	
Are the end users satisfied with the system?	1 to 7	7	7		NASA is pleased with the research effort.
Are the users' goals with the system fully supported?	1 to 5	5	5		
Are lessons learned tracked for future improvements?	Y/N (5/0)	5	5		Lessons learned thus far in the VTB will be used for future system improvements.
Reliability	1 to 5	3	5		In its current prototypical state, reliability needs improvement. Problems with Dynamic Link Library changes are one issue.
End User Interface(s)			57	0.46	

Dimensions/Attributes	*Measurement details*	*Measurements*	*Max possible score*	*Dimensional scores*	*Notes*
Are the end users satisfied with the interfaces?	1 to 7	5	7		Average of two people who have used the interface while working with the VTB.
What is the overall quality of interface(s)?	1 to 5	3	5		While capabilities are innovative and have potential, more refinement is needed to make the interface(s) easy to use for the average user.
control features					
Is there a central control and monitoring point?	Y/N (5/0)	5	5		
Can one change parameters in individual simulations from a central interface?	Y/N (5/0)	0	5		Changes require working from the local computer.
Can one start and stop simulations from a central control interface?	Y/N (5/0)	5	5		Yes. This is the result of an innovative research effort.
Can users see others who are logged into system and communicate with them?	Y/N (5/0)	0	5		Currently this is a one-user system, although designing for multiple users is possible with the current GUI design.
Is there good exception handling?	1 to 5	0	5		There is no exception handling in the prototype.
data visualization and analysis					
Is it possible to review data from several simulation scenarios simultaneously?	Y/N (5/0)	0	5		

Dimensions/Attri-butes	*Measurement details*	*Measurements*	*Max possible score*	*Dimensional scores*	*Notes*
Can statistics be saved and analyzed or exported?	Y/N (5/0)	0	5		
How good is the data visualization capability?	1 to 5	3	5		Development is needed in displaying data in the central GUI.
Can information from various simulations be combined in a way that allows good understanding of interrelationships and results?	1 to 5	5	5		The overall concept of the simulation combines and displays the data in an easy-to-understand fashion. The information should be integrated into one display.
Programming			37	0.54	
number of proprietary simulations	Integer 4 out of 5 simulations are based on proprietary code	1	5		This metric is calculated as the percentage of open-coded simulations times the total number of possible points.
number of software wrappers needed?	Integer, 4 out of 5 simulations require wrappers	1	5		A wrapper is needed for each proprietary simulation.
infrastructure ease of use	1 to 5	3	5		RTI 1.3, integrated with the adapters, is difficult to work with compared to the IEEE 1516 RTI.
Are programmers satisfied with the programming environment?	1 to 7	5	7		
time to get programmers on board, up to speed	Weeks		N/A		This attribute was not used. A basis of comparison is needed. The current estimate is 12 weeks.

Dimensions/Attributes	*Measurement details*	*Measurements*	*Max possible score*	*Dimensional scores*	*Notes*
ease of connecting individual simulation to the infrastructure	1 to 5	2	5		Integrating a new simulation is a major task.
data format compatibility between simulations	1 to 5	5	5		Data formats are compatible.
configuration control between simulations	1 to 5	3	5		A formal configuration control is suggested.
Installation			27	0.33	
Is a detailed log kept of all installation details, including troubleshooting, problems encountered and their solutions?	Y/N (5/0)	5	5		A log is kept in a three-ring binder.
Can personnel of average ability, but taught the job, install the system components?	Y/N (5/0)	0	5		Experience has shown that issues will arise during each installation, such as problems with Dynamic Link Libraries.
Are the different skills needed for the installation team specified?	Y/N (5/0)	0	5		This has not been documented.
number of people required	Integer		N/A		This attribute is not currently used.
troubleshooting capability	1 to 5	1	5		Installers have spent much time troubleshooting and complained about the difficulty of it.
Are installers satisfied with the installation scenario?	1 to 7	3	7		
Training			44	0.59	

Dimensions/Attributes	*Measurement details*	*Measurements*	*Max possible score*	*Dimensional scores*	*Notes*
Is the training effective? That is, does it prepare the trainee for what he or she needs to do?	1 to 5	3	5		
Are the trainees satisfied with the training?	1 to 7	4.5	7		
Are the trainers satisfied with the training scenario?	1 to 7	4.5	7		
Are written materials available to support the training?	Y/N (5/0)	5	5		
Are on-line materials available to support the training?	Y/N (5/0)	0	5		
Is the training geared to the knowledge/skill level of the audience?	Y/N (5/0)	5	5		
Overall quality of installation training.	1 to 5	2	5		
Overall quality of end user interface training.	1 to 5				N/A There is no end user interface training at the current time.
Overall quality of programmer training.	1 to 5	2	5		
Documentation			35	0.54	
Programming: Is the code level documentation good?	1 to 5	2	5		

Dimensions/Attributes	*Measurement details*	*Measurements*	*Max possible score*	*Dimensional scores*	*Notes*
Programming: Is the software design clearly defined in a modeling language to aid programmers in their work?	1 to 5	3	5		
Are the end user needs and goals well documented?	1 to 5	5	5		
Quality of training documentation.	1 to 5	3	5		
Quality of installation documentation.	1 to 5	3	5		
Quality of written end user interface documentation.	1 to 5	1	5		Little effort has been spent documenting the end user interfaces.
Quality of on-line help and support for end users.	1 to 5	2	5		The on-line help module needs content added.

The dimensional holistic usability scores are summarized in the table below.

Table 6. Assessment Summary for the Virtual Test Bed

Dimension	Metric
End User Needs and Goals	0.91
End User Interface	0.53
Programming	0.54
Installation	0.33
Training	0.59
Documentation	0.54

VTB Strengths, Weaknesses, and Recommendations

The VTB is an innovative project that has demonstrated the ability to integrate diverse simulations to create a virtual spaceport. More development is needed in this project to build on what has already been accomplished and to create a more production-oriented product.

The VTB's holistic usability strengths are:

- successful integration of existing simulations with live federate participation to create a virtual spaceport, whose data integration presents a coherent, easy-to-understand simulation of space shuttle operations
- Choosing the HLA-RTI as the infrastructure allows for unlimited growth potential in the size of the VTB and represents the best choice in architecture from a programmer's usability perspective.

- good use of data from various simulations to create a situational visualization and analysis, namely launching and processing the space shuttle

The VTB's weaknesses are:

- Training for people new to the project, especially programmers and installers, is inadequate.
- incomplete documentation of programming and installation details
- the need for a more developed user interface for a typical end user

The recommendations regarding the VTB's holistic usability are:

- Take steps, by improving documentation and training, to reduce the amount of time it takes programmers new to the project to become productive.
- Thoroughly document the installation process, in particular troubleshooting issues with the Distributed Manufacturing Adapter.
- Continue the project, with emphasis on three things:
    - the integration of a variety of selectable, distributed software modules/simulations that enhance the utility of the system to NASA end users
    - Focus on making the system oriented more to day-to-day use, rather than a prototype.
    - continual improvement of the GUI, with the focus on capabilities for simulation module selection, integration and visualization of data from various simulations, and ease of use for potential end users of the system

Aviation Research Training Tool Radar Assessment

Aviation Research Training Tool Radar System Description

The second system evaluated is the Aviation Research Training Tool (ARTT) Radar, which is used at Embry-Riddle Aeronautical University to train students in radar use for ATC. Its intended use is to train students in air traffic control for the approach-departure/terminal operations at an airport and also for training students in en-route air traffic control. At the time of writing it is used for three courses, teaching Daytona approach and departure, the Orlando airspace, and the Jacksonville-Ocala sector.

The ARTT is distributed across thirty workstations and one server in two rooms. The ATC radar room contains the main server and fifteen radar workstations, configured so that students may work in pairs at them, with one student being the radar operator and remotely talking to pilots and the other keeping track of pilot strips (strips are paper records kept of aircraft positions). The pseudo pilot room contains fifteen workstations for pseudo pilots; each pseudo pilot workstation controls one or more aircraft. The simulation scenarios are created in advance by the course professor. Each pseudo pilot workstation is associated with an ATC radar room workstation during a scenario. (The system can be configured so that any number of pseudo pilot workstations can interface with any number of ATC radar room workstations, but this capability is not needed for the courses.) While the scenarios are underway, the pseudo pilots communicate with the ATC radar room personnel using the Voice Communication Simulator (VCS), which is shown on displays in the radar and pseudo pilot rooms. The VCS interface consists of a headset and a touch screen control panel. The VCS simulates actual radio usage,

with selectable frequencies. The Gate Keeper application allows different groups of participants to communicate via the VCS using multicast data distribution for voice communication. This application allows for remote startup and shutdown of VCS endpoints. The system includes voice recognition capability, although it is not used for the courses taught using the system. The software package also includes a graphics editor. Below is a sketch of the ATTR layout.

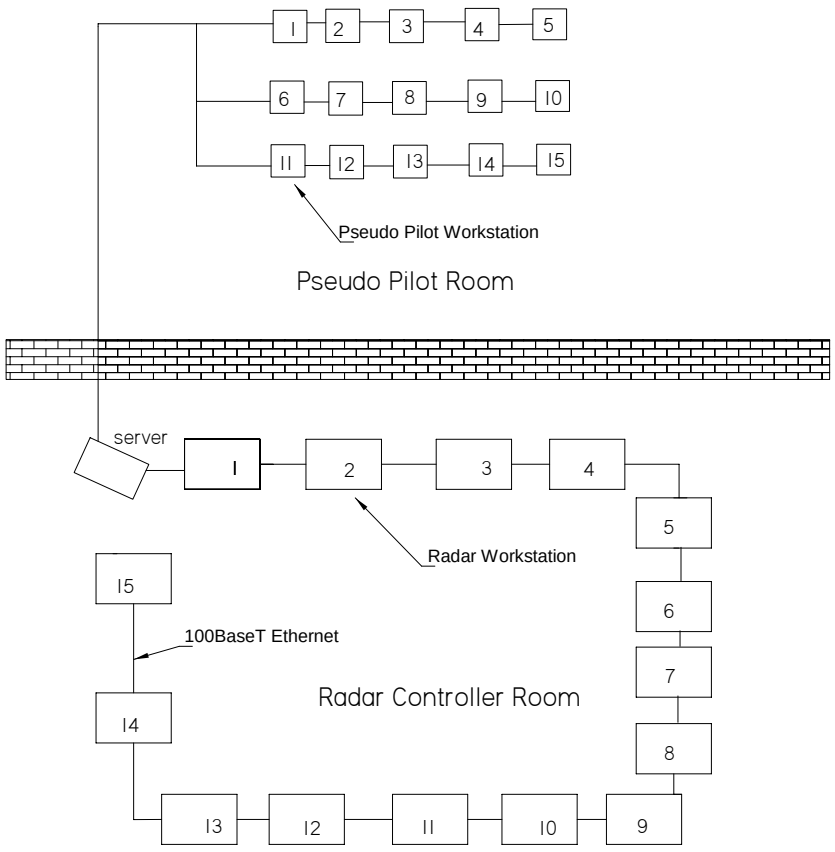

Figure 8. Schematic of the Layout of the ATTR Radar

A scenario must be created to use the system. The Whiteboard is the application used to create simulation scenarios. The professor creates the scenarios to simulate real-world situations chosen to meet the training needs of the students. Scenarios can be recorded and played back. At the start of a simulation, scenarios are loaded off of the main server in the ATC radar room into the workstation groups as needed. Students working as technicians open the scenarios for students before the class starts. During the simulations, pseudo pilots communicate with the ATC radar operators. The pseudo pilots control the planes, while the ATC radar operators give them flight instructions. Although all students in a class period will start off with the same scenario, each simulation will be different over time due to variations in pseudo pilot maneuvers and in the ATC instructions given to their associated pseudo pilots.

When students are using the ATC radar workstations, their data input is via either an ARTS-III keyboard or a mouse. The ARTS-III keyboard is a special keyboard that is used for ATC work. The use of this keyboard and its special command key sequences is mandated by the Federal Aviation Administration (FAA). The intent of the system is to simulate an FAA ATC radar room with high fidelity to the real world. The use of FAA-required ARTS-III keyboard commands is part of the requirements. Students are given ten hours of instruction in learning these keyboard commands at the start of the semester. Alternatively, students can use the mouse to enter commands.

The ARTT workstations and server are connected to each other via a dedicated, hardwired 100BaseT Ethernet. Following are photographs of the system.

Figure 9. Two ATC Radar Room Workstations

Figure 10. ATC Radar Display During Simulation of Airspace

Figure 11. ARTS-III Keyboard for ATC and Mouse at a Radar Room Workstation

Figure 12. Pseudo Pilot Workstations

(Note: Keystroke reminders are permanently displayed on the projector screen.)

ARTT Assessment Details and Observations

The ARTT assessment summary is given in the table in the next section. Some salient aspects of the assessment are noted in this section. I spent five days studying and assessing the system during students' practice labs, when classes were in session (with simulations running), and when classes were not in session (when I could inspect the system interfaces). In addition, I interviewed professors, network administrators, and technicians, and inspected system documentation. Only four of the six dimensions of the framework could be assessed: End user Needs and Goals, End user Interface(s), Training, and Documentation. The Programming and Installation dimensions are proprietary information of the software vendor, Adacel. Although contacted, the vendor declined to share proprietary information.

A brief, anonymous user satisfaction survey (using a seven-point Likert scale) given to the students and technicians indicated that most were very satisfied with the system and the interfaces. All were very satisfied with the training in using the system they had received, both written and oral. During simulations during the classes, the students appeared to be enjoying using the system and learning. Technicians training the students were also satisfied with the training they gave.

Inspection of the documentation and system interfaces revealed that they were both of good quality. Although a detailed, in-depth usability evaluation of the interfaces was not performed—the intention is mainly to assess the overall quality and DSS aspects, I did thoroughly inspect them and found them to be well developed. One observed weakness was that some menus, once chosen, could only be closed with a mouse command; alternative keyboard commands to close a menu, such as pressing the ESC key or using CNTL-W, would indicate a

more highly-developed interface, but given that the course required using the ARTS-III key commands, this was not an issue. Also, as noted, the students rated the interfaces highly in terms of user satisfaction.

The overall system assessment of the dimensions that could be assessed (End user Needs and Goals, End user Interface, Training, and Documentation) would appear to be very high if one only assessed the viewpoint of the students, who are one type of system end user: trainees. The needs of the network administrators and professors, however, who must keep the system running and see that it meets the intent of the training, must also be considered.

The intention of the assessment was to assess the system *as is.* Thus, past reliability problems may indicate poor system reliability, but they may have been resolved by recent updates. Historically, reliability has been a problem with the system; in particular, the VCS has been troublesome; the reliability of the VCS has improved recently due to a server upgrade and software changes. On the last day of the assessment, four VCS stations and two radar workstations were down. It was not known whether this was due to vendor software problems, hardware problems, or operating system instability issues. Based on observation, the reliability of the system is rated as 3 out of 5 and is in need of improvement.

The end user training need of teaching students in a realistic experiential environment was not fully met. Of the two noted training goals—approach-departure/terminal operations and en-route air traffic control—the first was best met. Discussions with two professors who use the system revealed the following.

The ATC terminal operations class professor was satisfied with the system and had no notable concerns. He noted, however, that it was inadequate for en-route training.

Several concerns relative to the end user needs and goals were noted by the en-route ATC class professor. The keystroke emulations using the ARTS-III keyboard are not all accurate to the real world, thus there was a concern about inaccurate emulation as well as teaching students bad habits. Another concern is that when programming scenarios, it is not possible to obtain a simple print out of the flight plans for each aircraft—other than a screen shot—making the programming of complex scenarios difficult; the ability to print an ASCII printout of flight plans is needed. The en-route emulation used in the system merely increases the range of terminal operations; this is not a valid representation of the en-route tracking used at the FAA's twenty-two en-route tracking centers. Concerns were also noted in the pseudo pilot interface, in that one cannot enter multiple commands, but must enter them one at a time. The above issues are reflected in the End User Needs and Goals score.

The user satisfaction metrics for the professors who teach classes, students, and technicians are noted in the table below.

Table 7. ARTT Radar Satisfaction Metrics

Question	*Professor 1*	*Professor 2*	*Professor average*	*Student and technician average*	*Number of students and technicians*
From an end user viewpoint, how satisfied are you with the system? (1 to 7)	5	3	4.00	6.08	13
From an end user viewpoint, how satisfied are you with the interfaces(s)? (1 to 7)	5	1	3.00	6.00	12
How satisfied are you with the training scenario? (1 to 7)	6	1 (en-route) 4 (terminal usage) 2.50 average	4.25		
Are your goals with the system fully supported? (1 to 5)	5	2.5	3.75		
How satisfied are you with the training? (1 to 7)				6.50	12

Below is the summary of results table for the holistic usability assessment of the ARTT Radar. The dimensional scores are calculated by taking the sum of the measurements of the attributes and dividing it by the maximum possible score for the dimension, so that 1.00 is a perfect rating.

Table 8. Assessment Metrics for the Aviation Research Training Tool Radar

Dimensions	*Measurement details*	*Measurements*	*Max possible score*	*Dimensional scores*	*Notes*
End User Needs and Goals			22	0.75	
Are the end users satisfied with the system?	1 to 7	4.70	7		Because this is an instructional system, professor end user satisfaction is weighted more heavily (0.67) for this rating than student satisfaction scores (0.33).
Are the users' goals with the system fully supported?	1 to 5	3.75	5		The goal of training students for air traffic control radar in the terminal area is well supported. Improvement is needed for en-route simulation.
Are lessons learned tracked for future improvements?	Y/N (5/0)	5	5		Lessons learned have been tracked and will be considered in any future systems.
Reliability (hardware and software)	1 to 5	3	5		
Vendor support					Not assessed for reasons of client confidentiality.
End User Interface(s)			52	0.89	
Are the end users satisfied with the interface?	1 to 7	3.99	7		
What is the overall quality of interface(s)?	1 to 5	4	5		Three interfaces were inspected: the Pseudo Pilot, the ATC Radar, and the Voice Communication System
control features					
Is there a central control and monitoring point?	Y/N (5/0)	5	5		Each user group has control of a simulation.
Can one change parameters in individual simulations from a central interface?	Y/N (5/0)	5	5		Yes.

Dimensions	*Measurement details*	*Measurements*	*Max possible score*	*Dimensional scores*	*Notes*
Can one start and stop simulations from a central control interface?	Y/N (5/0)	5	5		Yes.
Can users see others who are logged into system and communicate with them?	Y/N (5/0)	5	5		ATC radar users are often in constant communication with pseudo pilot interface users in another room.
Is there good exception handling?	1 to 5	4	5		The only exceptions noted by users are freezes, which are fixed by stopping/restarting the process.
data visualization and analysis					
Is it possible to review data from several simulation scenarios simultaneously?	Y/N (5/0)	5	5		Simulations can be recorded and replayed. For the purposes of this system, this meets the intent of this attribute: the ability to review simulations.
Can statistics be saved and analyzed or exported?	Y/N (5/0)				N/A. This is not a simulation to generate results, but to train students.
How good is the data visualization capability?	1 to 5	5	5		Data visualization is excellent, in that this simulation has a high fidelity relative to the real world.
Can information from various simulations be combined in a way that allows good understanding of interrelationships and results?	1 to 5	5	5		The combination of the pseudo pilot data and the ATC simulation is effective.
Programming					Programming details are proprietary. Unable to assess.
Installation					Installation is a proprietary process. Unable to assess.
Training			34	0.89	

Dimensions	*Measurement details*	*Measurements*	*Max possible score*	*Dimensional scores*	*Notes*
Is the training effective? That is, does it prepare the trainee for what he or she needs to do?	1 to 5	4.50	5		
Are the trainees satisfied with the training?	1 to 7	6.50	7		
Are the trainers satisfied with the training scenario?	1 to 7	4.25	7		
Are written materials available to support the training?	Y/N	5	5		
Are on-line materials available to support the training?	Y/N				N/A Course requirements discourage the use of on-line training materials.
Is the training geared to the knowledge/skill level of the audience?	Y/N	5	5		
Overall quality of installation training.	1 to 5				N/A Proprietary
Overall quality of end user interface training.	1 to 5	5	5		
Overall quality of programmer training.	1 to 5				N/A Proprietary
Documentation			15	1.00	
Programming: Is the code level documentation good?	1 to 5				Proprietary

Dimensions	*Measurement details*	*Measurements*	*Max possible score*	*Dimensional scores*	*Notes*
Programming: Is the software design clearly defined in a modeling language to aid programmers in their work?	1 to 5				Proprietary
Are the end user needs and goals well documented?	1 to 5	5	5		
Quality of training documentation.	1 to 5	5	5		
Quality of installation documentation.	1 to 5				Proprietary
Quality of written end user interface documentation.	1 to 5	5	5		Thorough system documention is provided, with instructions for using all the features.
Quality of on-line help and support for end users.	1 to 5				N/A. Although on-line help is a feature of the system, students are instructed to use only documentation supplied as course materials.

ARTT Summary of Results

The assessment metrics for the four dimensions of the ARTT that could be assessed are shown in the table below. As noted, the system rates very well in the End User Interface, Documentation, and Training dimensions. Improvement is needed in the End User Needs and Goals dimension.

Table 9. Assessment Summary for the ATTR Radar

Dimension	Metric
End User Needs and Goals	0.75
End User Interface	0.89
Programming	proprietary
Installation	proprietary
Documentation	1.00
Training	0.89

ARTT Strengths, Weaknesses, and Recommendations

Observing a large number of students using the system during class time, no significant trainee problems with the system were observed, and the trainees enjoyed using the system. In addition, the trainees have noted high levels of satisfaction with both the training they have received on how to use the system and when using the system. The system meets most of the needs for terminal radar training. Reliability is a concern. In summary, strengths are:

- interfaces that most trainers and students find easy to use
- innovative voice communication system emulating real-time radio usage
- fairly accurate emulation of ATC operations
- high average levels of user satisfaction

Weaknesses are:

- poor en-route simulation capabilities

- on-going reliability issues
- incomplete realism in the keystroke emulation of the ARTS-III keyboard
- limited scenario programming capability; unable to add to plane database

The ARTT's holistic usability, based on the four dimensions that could be assessed without gaining access to proprietary information, rates very well in the End User Interface(s), Training, and Documentation dimensions.

The recommendations regarding the ARTT's holistic usability are:

- Either update the system for more realistic en-route simulation or use a different system to teach that course.
- Work to improve reliability of the system. Keep a log over time of all system problems and resolutions, so that trends and troubleshooting details will be available to decision makers.
- Suggest that the vendor make the keystroke emulation accurate and improve the scenario programming capability.

The holistic usability of the ARTT is high enough to warrant its continued usage for terminal simulation training scenarios. As noted, the trainees are generally very pleased with the system. Its use for en-route training is questionable.

Lessons Learned and Framework Strengths and Weaknesses

This framework represents preliminary research, and a number of iterations are needed to refine it. Based partly on what has been seen during these two system assessments, a list of framework strengths and weaknesses follows.

Strengths

A holistic look at the usability of DSSs has the ability to:

- improve the efficiency, effectiveness, and satisfaction for all people who work with the system
- show weak areas that are not readily apparent to decision makers and show opportunities for improvement
- show where resources should be spent
- show what is most important to all types of users (Attribute weights in particular are helpful here.)
- indicate areas where research may be needed
- provide formative holistic usability advice during development
- assess the overall state of the system vis-à-vis all people who work with it.
- integrate many aspects of design and use in a multidisciplinary viewpoint that spans several fields.

Weaknesses

A number of weaknesses exist that indicate the need for further research.

- The attribute weights need study.
- More depth is needed in dimensional assessments, with more attributes and especially the expansion of attributes into levels of subattributes.

- The framework does not take the place of a traditional usability assessment. The attribute that assesses the overall quality of the end user interface(s) would ideally be a full-blown, traditional usability assessment in itself.
- There is a need for more refinement of the measurement methodology.
- Proprietary issues mean that some dimensions may not be able to be assessed from a client's viewpoint.
- Different end users with different objectives will tend to distort the End User Needs and Goals dimension measurements.
- As a new concept, it may be hard to sell to management. (Usability was a hard sell for a long time.)

Lessons learned

Several lessons were learned in the application of the framework that are worth noting.

- When assessing a system provided by a vendor, the inability to assess the Programming and Installation dimensions due to proprietary issues is a weakness of the framework. However, the proprietary dimensions could be assessed if the assessment were performed in-house from the viewpoint of the vendor.
- The value of qualitative observations will at times be more valuable than the measurements; adequate attention should be given to both.
- There will be sensitivity issues related to how measurements are mixed together both within an attribute's metrics and in the dimensional metrics.

- Generally, workers who work with a DSS will be very receptive to the idea of a holistic usability assessment and be willing to share much information. This represents an opportunity for management to get to the root of and become cognizant of important issues related to productivity and effectiveness they may not have been aware of. An example of this is the failure of the ARTT Radar system to meet all key user needs and goals. In this case, this knowledge has value not just to the client but to the vendor as well, which suggests both client and vendor viewpoints would result in a better, well-rounded assessment.
- In the two assessments performed, the satisfaction metric was measured on a different scale than the other attribute metrics. This was changed to the same scale as the other metrics. The reasons are that if a user were rating both satisfaction and another attribute they might have trouble switching scales; adjusting weights is more easily performed when attributes are all measured on the same scale; and if there were a desire to compare averages or standard deviations of attributes they would need to be on the same scale.
- There is a need to be able to adjust attribute weights, because the importance of attributes will vary significantly from system to system.

Weights and Sensitivities

A survey to determine attribute weights based on DSS expert input is discussed in the next chapter, and revised assessment numbers for dimensional metrics of the assessments are also given. There are several issues related to attribute weights and measurements that require further research.

From the standpoint of measuring human-computer interaction, the influence of certain attributes on efficiency and effectiveness would be expected to be generally invariant. For example, the quality of configuration control between simulations will significantly affect the ease of programming regardless of the system type. However, not only are there many types of DSSs, but each DSS of the same type has its own unique characteristics.

A good starting point for the weights is the importance survey data. The system designer or owner may have constructive ideas about which weights would be best. The workers in each dimension would be an excellent source of information about weights. Historical work records and problems encountered and how better attribute characteristics could have helped to avoid problems or inefficiencies would be helpful in a detailed weight analysis.

The weight of satisfaction metrics relative to the weight of other attributes is an open question. People with entertainment systems and games may decide to weight satisfaction more heavily than other attributes. As an alternative, satisfaction metrics could be reported separately.

Sensitivities also need to be considered in future research. The relative weights of trainee and trainer satisfaction metrics within attributes in the ARTT Radar assessment are a case in point. Some of the End User Needs and Goals dimension's attributes were weighted based on the assumption that the professors' needs as trainers were more important than students' perceived needs. Also, the equal weight between the professor-provided metrics resulted in a relatively low end user needs attribute score; a sensitivity study would result in the End User Needs and Goals dimension's score being higher for terminal/arrival-departure training usage, and even lower for the en-route training course. The attribute weights in the End User Needs and Goals dimension warrant particular attention in adjustment considerations. The reliability metric, for instance, is

very important in a production system, but might be deemed less important in a prototypical system.

CHAPTER SIX: DEVELOPMENT AND APPLICATION OF ATTRIBUTE WEIGHTS

In Chapter 4, a survey to validate the attributes of the holistic usability framework was described. In this preliminary research, the attributes were chosen based on what was considered to be the best generalized attribute set that would be applicable to various types of DSSs. As noted, the validation approach was based on a link between one of the three standard usability measures and the attribute's effect on the ease of use or working for one of the types of users. In the two examples given of applying the framework, the attributes were given equal weight in the dimensional attribute summations. A second survey, given to a different group of DSS experts (to avoid statistical dependence between two different types of questions asking about the same attribute), asked about the level of importance of each attribute, with the determination of attribute weights in mind.

One approach to this survey would have been to simply ask respondents to distinguish those attributes that are "most important." The approach taken was to list all the attributes in a dimension, asking the respondents to rate them as "very unimportant," "unimportant," "important," "very important," or "extremely important." These choices were presented as a matrix for each dimension in an on-line survey.

Because both surveys were launched simultaneously, the second survey contains attributes that were not validated (i.e., those that were later removed from the framework due to analysis of the first survey). In addition, three attributes were added to the framework as the survey was in progress, based on information gathered from meetings with three HLA-RTI vendors and their technical personnel and field research. As required to maintain statistical integrity, these questions were added at the end of the survey to preserve the order of the

questions. In addition, two open-ended questions were added to the second survey. Based on the high response rate in the first survey to open-ended questions, these open-ended questions are a good vehicle to gather data for future research as well as information to consider for this dissertation. (Often these responses provide information about DSS issues more in-depth than the task at hand, but also frequently reinforce the current framework's attributes.)

The average number of years of distributed simulation experience the participants had was 6.4 years, with a median of 5.0, and a standard deviation of 4.54. The range was one to twenty years. There were 32 participants.

The variety of organizations from whom anonymous participants responded, as reported in the survey, includes NASA, Georgia Tech, Arizona State University, NCSU, NIST, a NASA Contractor, ARC Seibersdorf Research GmbH, Naval Postgraduate School, a defense contractor, Alion Science and Technology MAAD operation, Decisioneering, Singapore Institute of Manufacturing Technology (SIMTech), Intel, LSIS laboratory, Delft University of Technology, UIUC, Systems Navigator, The University of Jordan, and the FAA.

Given this survey data that represents the preferences of a world-wide, generalized set of DSS experts, the question then is: How do we use the data to determine weights?

There are many ways to assign weights, but the most important criterion is that the result be rationale. I used the following approach. Assign -2 to "very unimportant," -1 to "unimportant," +1 to "important," +2 to "very important," and +3 to "extremely important," then multiply the percentages of each attribute ranking times the corresponding number and sum the results to determine the weight.

This does not mean that we know exact weights with a high level of precision. This practical approach, however, uses the data to derive objective weights empirically. Furthermore,

it works with this generalized attribute set, would work just as well with a specialized DSS attribute set (say, only for engineering design systems), and can be readily updated with incoming empirical data.

Table 9 below shows a summary of the attribute importance ratings from the importance survey. The rows of attributes that were not validated are shown in gray (for information only). The calculated weights are shown in the rightmost column.

Table 10. Survey Results for Determination of Attribute Weights

End User Needs and Goals attributes	*Ranking:*						
	very unimportant	unimportant	important	very important	extremely important	Response Total	*Weight*
tracking lessons learned for future improvement	0	5	13	8	6	32	1.31
system reliability (hardware and software)	0	1	8	8	10	27	1.96
if a vendor-provided system vendor support	0	4	8	12	3	27	1.37
End User Interface attributes	*Ranking:*						
	very unimportant	unimportant	important	very important	extremely important	*Response Total*	*Weight*
a central control and monitoring interface	0	3	10	14	5	31	1.61
the ability to change simulation parameters from a central control interface	0	5	5	15	5	30	1.50
the ability to stop and start individual simulations from a central control interface	0	5	9	11	7	32	1.47
the ability to communicate with users who are logged into the system	1	10	13	4	2	30	0.50
good software exception handling	0	0	8	17	5	30	1.90
showing relevant variables in all simulations running simultaneously	0	6	14	6	5	31	1.13

the ability to review the data from several simulation scenarios simultaneously	0	3	14	8	6	31	1.45
the ability to save and analyze or export statistics	0	0	5	12	15	32	2.31
good data visualization capability	0	2	13	6	11	32	1.75
the ability to combine information from different simulations in a way that shows their interrelationships	0	3	12	11	6	32	1.53
Programming attributes	*Ranking:*						
	very unimportant	unimportant	important	very important	extremely important	Response Total	*Weight*
low complexity in the programming environment	0	8	11	8	3	30	0.93
the avoidance of simulation packages with proprietary code	0	7	11	10	3	31	1.06
the avoidance of needing software "wrappers" for individual simulations	0	8	13	7	1	29	0.76
ease of programming the infrastructure	0	3	18	10	0	31	1.13
minimizing the amount of code that needs to be written	1	6	14	7	3	31	0.94
needing a low level of programming expertise	1	7	12	7	3	30	0.87
training programmers to understand the system	0	6	11	9	4	30	1.17
ease of programming	0	7	10	11	3	31	1.10
an infrastructure designed to make connecting simulations to it easy	0	2	8	15	6	31	1.74
compatible data formats between individual simulations	0	1	7	14	10	31	2.06
good configuration control between distributed simulations	0	0	10	14	4	28	1.79
Installation attributes	*Rankings:*						
	very unimportant	unimportant	important	very important	extremely important	Response Total	*Weight*
keeping a detailed log of installation details	1	7	13	7	3	31	0.87

people of average ability being able to install the system	2	6	13	8	2	31	0.81
specifying different skills needed for installation	2	7	16	4	1	30	0.53
minimizing the number of people needed for installation	2	8	9	10	2	31	0.74
good installation troubleshooting capability	0	4	8	14	4	30	1.47
a quick installation process	1	9	11	6	4	31	0.77
an easy installation process	1	3	17	6	4	31	1.16
a low skill level required for installing the system	0	16	8	4	2	30	0.20
Training attributes	*Rankings:*						
	very unimportant	unimportant	important	very important	extremely important	Response Total	*Weight*
effective end user training	1	1	13	9	7	31	1.58
a quick training process	1	8	14	7	2	31	0.77
written training materials	1	2	8	14	6	31	1.61
on-line training materials	0	2	10	12	7	31	1.71
gearing the training materials to the knowledge level of the audience	0	3	9	10	8	30	1.67
training installers	1	5	16	5	3	30	0.93
good end user interface training	1	3	8	12	7	31	1.55
programmer training to help them learn the software design	1	6	14	5	4	30	0.93
Documentation attributes	*Rankings:*						
	very unimportant	unimportant	important	very important	extremely important	Response Total	*Weight*
good code-level documentation	0	2	11	15	3	31	1.55
a clearly specified software design	0	2	12	12	5	31	1.58
documenting end user needs and goals	0	5	8	12	7	32	1.50
good training documentation	0	1	13	11	7	32	1.72
good installation documentation	0	2	13	8	8	31	1.65
good end user documentation	0	3	9	9	11	32	1.78
good on-line help	0	2	7	13	9	31	1.87

Given that we have assessed the holistic usability of two DSSs without using weighted attributes, how different would the result be if we used weights? The two spreadsheets containing the results of the two DSS assessments were recalculated incorporating the weights. The manner in which the validated attributes were weighted was by multiplying their previous maximum value by the weights, giving a new maximum value for that attribute; likewise, the assessed value for the attribute was also multiplied by the weight to adjust accordingly. The maximum possible score of 1.00 for each dimension was maintained by the calculational structure. The satisfaction ratings were not weighted, but are assumed to have the same weight. Also, the weights for the attributes measuring how well the users' goals are supported and the overall quality of the interface(s) were left at 1.00. These weights can be adjusted as desired. (As noted, vendor support is not assessed in either of the two assessments included herein.)

The tables below show the assessment summaries both with and without attribute weights. As can be seen, the effect of the weights does not greatly affect the overall assessment in these cases, but might in others.

Table 11. Weighted Assessment Summary for the Virtual Test Bed

Dimension	Metric	Weighted Metric
End User Needs and Goals	0.91	0.86
End User Interface	0.46	0.44
Programming	0.54	0.59
Installation	0.33	0.35
Training	0.59	0.60
Documentation	0.54	0.53

Table 12. Weighted Assessment Summary for the ATTR Radar

Dimension	Metric	Weighted Metric
End User Needs and Goals	0.75	0.74
End User Interface	0.90	0.92
Programming	Proprietary	--
Installation	Proprietary	--
Training	0.89	0.83
Documentation	1.00	1.00

CHAPTER SEVEN: VENDOR AND PRACTITIONER FEEDBACK

Industry feedback on the value of the framework was sought from three vendors or practitioners. All participants in this feedback discussion were sent a document that describes the framework, lists all the attributes, explains its application in both formative and evaluative usability, and shows an example assessment spreadsheet. The vendors/practitioners were offered four options for providing feedback: (1) in person meeting, (2) telephone meeting, (3) answering questions in a document, or (4) filling out an on-line survey. The reasons for offering a variety of ways to give feedback were to accommodate the schedules of busy professionals and to increase the chance of obtaining feedback.

Feedback 1

Aegis Technologies is company that is a simulation practitioner who is the sole U.S. representative of Pitch Technologies AB, a Swedish HLA-RTI vendor. Aegis also provides simulation and other consulting services and offers its own products. I contacted a manger at Aegis, who thought the best person to provide feedback was a senior computer scientist who was actually in another city. We arranged a telephone conference. Prior to the telephone conference, the senior computer scientist thoroughly reviewed the description of the framework I had sent, and had a number of items listed he wanted to discuss.

In his initial comments, he said that he thought the formative usability process using the framework was "great," and that he had never seen usability looking at the whole system in an integrated fashion before, only usability looking at end user interfaces.

Concerning attribute 3 in End User Needs and Goals, "Lessons learned should be tracked for future improvements and new systems," he commented that this is similar to what is called a "problem report." He noted that often when a problem occurs it might be noted in a log, but often the solution will stay in the mind of the person who solved the problem, so that he becomes the "expert," and thus is indispensable. He felt this attribute was important.

He said that attribute 4 in End User Needs and Goals, "Vendor support should be adequate," had two components, response time and cost. Response time is absolutely critical in his work. He said that one could have 5,000 to 10,000 soldiers—several brigades—on a range California, and have the command and control run from a workstation simulation in Virginia. If the controlling simulation goes down, that is a serious problem. He said a 24-hour response time from a vendor in that situation would be unacceptable.

About attribute 4 in End User Interface(s), "One should be able to change parameters in individual simulations from a central interface," he said this would be "phenomenal" if they could do it in their simulation work environment. He said that currently the Joint Rapid Distributed Database Development Capability (JRD3C) effort is working to make this possible in large military simulations. He said that currently, large military simulations require people at different control GUIs in several places, and that this requires several experts.

We discussed attribute 1 in Programming, "The number of simulations written with proprietary simulation packages should be minimized." He agree that simulations with closed code make work more difficult for programmers. He said that simulations written with certain proprietary packages for distributed simulation are easier to write (than if one were writing in open code), but more difficult to deploy. A related issue mentioned was that some vendor tools

for distributed simulation require that one use a vendor's license server in the federation. If the license server goes down, none of the vendor's applications can be used.

At one point when discussing the programming dimension, he talked about weighting attributes, so I asked if the ability to weight attributes was important. He replied that it was, that it was good to have a set of expert-determined weights as a starting point, but that what was critically important in one simulation might not be critical in another, so the ability to adjust attribute weights was important.

Concerning attribute 8 in Training, "The overall quality of end user interface training should be good," he said that there are different kinds of end users in his work, trainees being the most critical—because if they don't learn from the training the simulation is worthless—and "pucksters." There are two types of pucksters, the people controlling the simulation and the subject matter experts. The subject matter experts often suggest changes to the simulation controllers to make the simulation more realistic.

Referring to the attribute concerning code-level documentation, he gave examples of unmaintainable code he had encountered and noted that often the maintainer of the code has to be familiar with it. One example he explained showed how often in distributed simulation code will have dependencies on other models in it, which without good documentation require line-by-line searching through the code to find; even when found, the nature of the dependency is not always clear. He noted that "code documentation is crucial."

Referring to the flow chart for formative usability in the framework description document I provided (which is also shown in the Final Framework section of chapter 8), he said that the holistic formative usability process "hit right on the head" and if this were done early on, one would "get the product you expect."

After the comments he had prepared to discuss were covered, I moved on to the list of questions I had sent with the framework description document. Those questions and his answers are shown below.

Is this framework realistic?

He said that is was realistic, and that he has never seen anybody address these issues before and that it was more comprehensive than usability work he's seen before because it goes beyond the user interface. He said that the validation, verification, and accreditation (VV&A) process can help, but VV&A didn't cover what the framework covered.

Does your company have its own proprietary ideas or performance measures for any of the attributes in the framework?

No. The company has no assessment tools. He said Aegis has a product called BattleStorm that is a simulation framework that helps integrate simulations, but is not for usability assessments.

Is this framework useful to you or your customers?

He said that yes, he would actually like to use it.

Are there any cases or occasions when this framework, if available, would have helped (to avoid mistakes, reduce costs, increase customer satisfaction, increases efficiency or effectiveness for users/workers)?

He said that the framework would "help in anything " and that "I can't think of a situation where it would not help; there are always unexpected problems and issues."

Would you be willing to hire an outside consultant to assess the holistic usability of a system or help ensure good holistic usability during a system design?

He said that he does not make those decisions, but "it makes perfect sense." He said there are two reasons why it would make sense from a management point of view: they have no in-house usability expertise and there is a need to be unbiased. He said that Aegis Technologies is an independent Verification and Validation agent for customers who need an unbiased evaluation of distributed simulations.

Finally, this reviewer said that he would like to take the framework and try applying it himself to a project he was currently working on.

Feedback 2

Mäk Technologies is company that is an HLA-RTI vendor who sells simulation tools and consulting services to practitioners. I contacted a manger at Mäk, who forwarded my request to Mäk's engineering group. The engineer who responded chose the alternative of answering a set of questions I sent. Below are his unedited responses.

Is this framework realistic?

A: It is not completely clear what this framework would be applied to. A particular software product, and its value to distributed simulation? An HLA federation as a whole? I'm

also not sure what "realistic" means in this context. Most of the questions it asks, and the metrics it uses seem reasonable to me, but many of them are quite obvious: For example, "The quality of the installation documentation should be good." OK, sure, but I'm not sure I need a "framework" to tell me that that's a good idea. ☺

Does your company have its own proprietary ideas or performance measures for any of the attributes in the framework?

A: As a vendor of commercial tools, we are always looking for feedback from customers on our products along a variety of axes. We tend to do this through informal conversations, meetings, emails, etc., rather than through a formal spreadsheet of questions like this. The reason is that it allows us to tailor the questions and conversation to the relevant problems. We might not be providing a "Training System", so the questions about validity of training do not apply. On the other hand, we might ask many more specific questions about ease of use, User Interface, etc.

Is this framework useful to you or your customers?

A: I would say that while many of the issues you identify are quite relevant, I personally find it more beneficial to get feedback/evaluation in a more customer-tailored way. Also, many of your metrics apply more to complete systems, rather than to specific tools that may fit into those systems.

Are there any cases or occasions when this framework, if available, would have helped (to avoid mistakes, reduce costs, increase customer satisfaction, increases efficiency or effectiveness for users/workers)?

A: Again, I think there are specific metrics or elements of the framework that I think are useful.

Would you be willing to hire an outside consultant to assess the holistic usability of a system or help ensure good holistic usability during a system design?

A: Unlikely. We maintain close relationships with many of our customers, and find that we get the most accurate and useful feedback when we talk with them directly.

Are there any other comments that you would like to share?

Yes. I had questions on two of your metrics:

1. The number of simulations written with proprietary simulation packages should be minimized.

2. The number of software wrappers required around individual simulations, if any, should be minimized.

I guess my opinion of these metrics depends on what is meant by proprietary simulation packages. If you meant "stovepiped" systems that do not conform to industry standards like HLA, then I agree. I think it can be detrimental to a system to use tools that do not interoperate with other elements of the system without spending lots of money and time on wrappers and adapters. On the other hand, if proprietary means "closed source", then I disagree. There are many software products where source code is unavailable, where the products are still quite

"open." Our products, for example, all conform to interoperability standards like HLA, DIS the SISO RPR FOM, etc. In addition, they all have very extensive Toolkit APIs that insure that users can write code to extend or modify the products, even though they do not have source. To avoid ambiguity, I would change "proprietary" to "stovepiped" or "non-interoperable", or "packages that support only proprietary communication architectures".

Feedback 3

The third person to provide feedback on the framework is a manager of simulation projects at a Department of Defense (DoD) facility. Because this person cannot speak on behalf of the DoD, the facility will not be named. This respondent chose to respond via an on-line survey. Unedited responses are given below.

What company or institution do you represent?

I work for the Department of Defense, but cannot speak for DoD in an official capacity.

Does your company have its own proprietary ideas or performance measures for any of the attributes in the framework?

No.

Is this framework useful to you or your customers?

Yes, it shows potential.

Are there any cases or occasions when this framework, if available, would have helped (to avoid mistakes, reduce costs, increase customer satisfaction, or increase efficiency or effectiveness for users/workers)?

Yes.

Would you be willing to hire an outside consultant to assess the holistic usability of a system or help ensure good holistic usability during a system design?

Not at this time.

Are there any other comments that you would like to share?

I believe there are systems under development that could benefit from the approach proposed.

Summary Comments

Two of the feedback respondents indicated that they saw value in the framework. One, a representative of a vendor that sells DSS infrastructure, tools to integrate distributed simulations, and consulting services, was less positive, but saw some value in the framework. A person who works for another vendor told me that his company sells a tool to integrate simulations, but said that it goes against the framework's attribute concerning troubleshooting, because customers want to be able to solve their own problems (a statement that was also made by two survey respondents in response to an open-ended question) and use their own tools. In certain situations, such as large military simulations with many players, a spirit of cooperation is needed for holistic

usability to be fostered; how best to ensure this cooperation with proprietary interests at stake is a challenge. The comments made by the second respondent concerning the need for a deeper look at types of proprietary simulation packages and the terminology used to describe them are constructive and should be addressed in the next version of the framework.

CHAPTER EIGHT: CONCLUSION AND FURTHER RESEARCH

Final Framework

Attributes

The final framework is defined by the attributes below.

Table 13. Final Framework

DIMENSIONS	MEASURES	WEIGHTS
End User Needs and Goals		
The end users should be satisfied with the system.	1 to 5	1.00
The users' goals with the system should be achieved.	1 to 5	1.00
Lessons learned should be tracking for future improvements and new systems.	Y/N (5/0)	1.31
The system hardware and software should be reliable.	1 to 5	1.96
Vendor Support (if a vendor-provided system) should be adequate.	1 to 5	1.37
End User Interface(s)		
The end users should be satisfied with the interface(s).	1 to 5	1.00
The overall quality of the interface(s) should be adequate (this is a brief, traditional usability evaluation).	1 to 5	1.00
control features		
There should be a central control and monitoring point.	Y/N (5/0)	1.61
One should be able to change parameters in individual simulations from a central interface.	Y/N (5/0)	1.50
One should be able to start and stop simulations from a central control interface.	Y/N (5/0)	1.47
One should be able to locate others logged into the system and communicate with them.	Y/N (5/0)	0.50
Exception handling should be adequate.	1 to 5	1.90
data visualization and analysis		
It should be possible to review data from several simulation scenarios simultaneously and/or to record simulation scenarios.	Y/N (5/0)	1.45
One should be able to save, analyze, and export statistics.	Y/N (5/0)	2.31
Data visualization capability should be good.	1 to 5	1.75
Information from various simulations be combined in a way that facilitates the understanding of interrelationships and results.	1 to 5	1.53
Programming		

DIMENSIONS	MEASURES	WEIGHTS
The number of simulations written with proprietary simulation packages should be minimized.	1 to 5	1.06
The number of software wrappers required around individual simulations, if any, should be minimized.	1 to 5	0.76
If a distributed simulation infrastructure is used, it should be chosen based on ease of use for the programmers.	1 to 5	1.13
Programmers should be satisfied with the programming environment.	1 to 5	1.00
The time to train programmers to be able to work with the system should be minimized.	not currently measured	1.17
Choices should be made that facilitate the ease of connecting the individual simulations to the infrastructure.	1 to 5	1.74
The data formats between simulations should be compatible.	1 to 5	2.06
Good configuration control should be maintained between simulations.	1 to 5	1.79
Installation		
A detailed log should be kept of all installation details, including troubleshooting actions and results.	Y/N (5/0)	0.87
Personnel of average ability, but taught the job, should be able to install the system.	Y/N (5/0)	0.81
The different skills needed to install the system should be specified.	Y/N (5/0)	0.53
The number of people required to install the system should be minimized.	not currently measured	0.74
Effective troubleshooting capability should be part of the system.	1 to 5	1.47
Installers should be satisfied with the installation scenario.	1 to 5	1.00
Training		
The training should be effective, preparing the trainee to perform the tasks that need to be performed with the system.	1 to 5	1.58
The trainees should be satisfied with the training.	1 to 5	1.00
The trainers should be satisfied with the training scenario.	1 to 5	1.00
Written materials should be available to support the training.	Y/N (5/0)	1.61
On-line materials should be available to support the training.	Y/N (5/0)	1.71
The training should be geared to the knowledge/skill level of the audience.	Y/N (5/0)	1.67
The overall quality of installation training should be good.	1 to 5	0.93
The overall quality of end user interface training should be good.	1 to 5	1.55
The overall quality of programmer training should be good.	1 to 5	0.93
Documentation		
The programming code level documentation should be sufficient.	1 to 5	1.55
The software design should be clearly specified and diagrammed to aid programmers in their work.	1 to 5	1.58
The end user needs and goals should be clearly documented.	1 to 5	1.50
The quality of training documentation should be sufficient.	1 to 5	1.72
The quality of installation documentation should be sufficient.	1 to 5	1.65
The quality of written end user interface documentation should be sufficient.	1 to 5	1.78
The quality of on-line help and support for end users should be sufficient.	1 to 5	1.87

Using the attributes above, the framework can be applied for either formative or evaluative usability of a DSS. After discussing measurements, application of the framework will be described.

Measurements

Measurements of the attributes are either on a scale of 1 to 5 or binary as 5 or 0 depending on whether or not they exist in the system. Attributes measured from 1 to 5 are on a scale where 1 is worst and 5 is best. For the attributes concerning minimizing the number of proprietary simulation packages and software wrappers, the measurement is calculated by taking the percentage of simulations written in open code or the percentage of simulations not requiring software wrappers and multiplying it times 5. Dimensional metrics are calculated by dividing the sum of the measures by the total possible score, resulting in a number ranging from 0 to 1.00. The instructions for both formative and evaluative usability applications follow.

Formative Usability

The design team will work to establish targets for each of the framework dimensions. Although ideally, high scores in all dimensions would be the goal, resource constraints may result in tradeoffs. In addition, the framework is a guide to help designers and managers to consider each of the framework attributes when developing a system. The target scores for each attribute and dimension can be specified, then the resulting system's holistic usability measured to ensure those targets are met. This process is shown in the diagram below.

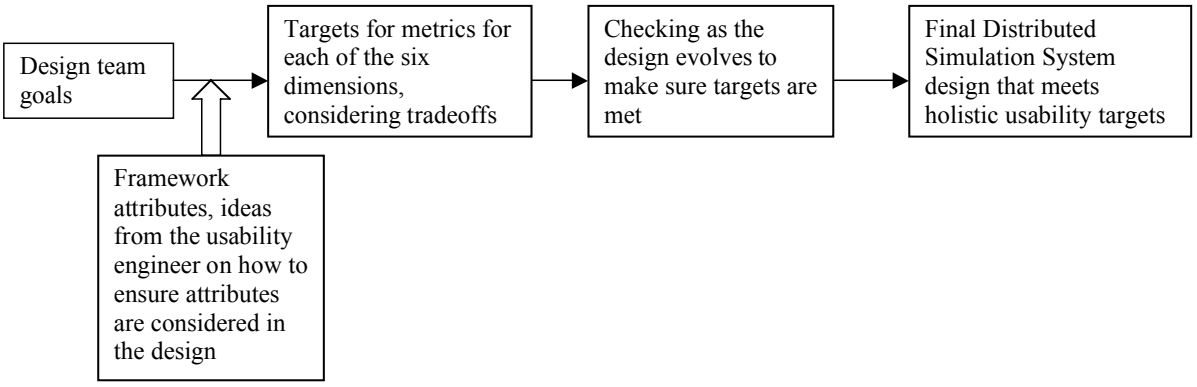

Figure 13. Formative Usability

Evaluative Usability

The procedure for assessing the holistic usability of a DSS is as follows. The evaluator will become familiar with the salient aspects of the system. (A small team, rather than a single evaluator, can also perform the assessment.) A sample of key user types will be given a survey concerning user satisfaction. User satisfaction requires user feedback to measure. System documentation, interface(s), design, and programming/infrastructure aspects will be evaluated from study, observation, and input from and discussion with personnel. If the situation warrants, personnel can be recruited to help with the assessment. A concise report will be generated using inputs from the above process that

- summarizes the metrics of dimensional attributes
- lists the strengths and weaknesses

- makes recommendations for improvements

Each of the system assessments will be reported in four sections: (1) system description, (2) assessment details and observations, (3) summary of results (metrics), and (4) strengths, weaknesses, and recommendations. A flowchart of the holistic usability assessment process is shown below.

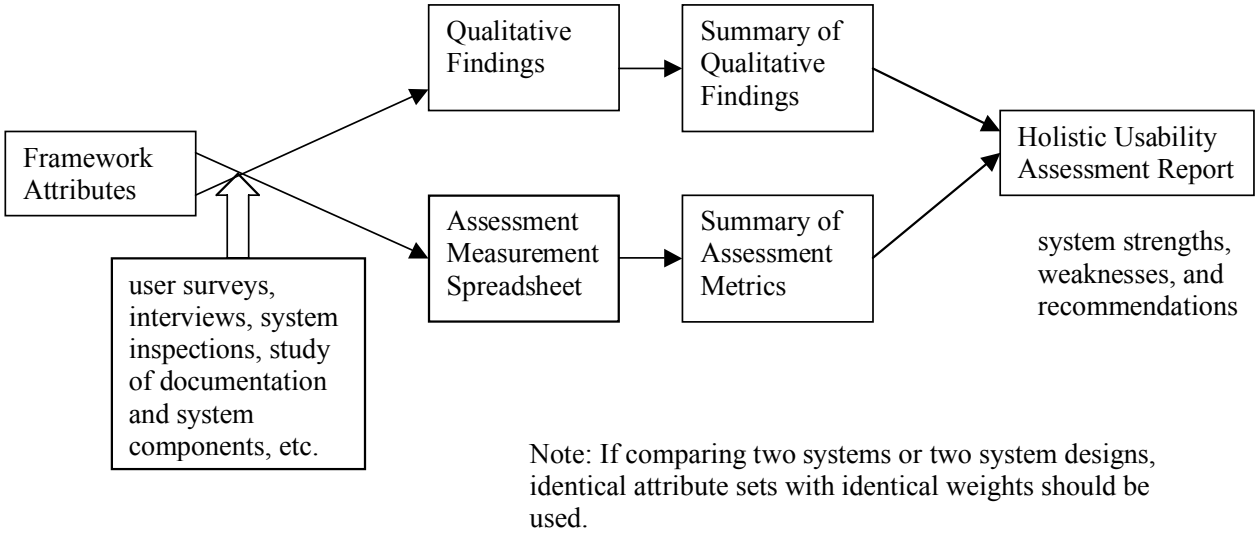

Figure 14. Evaluative Usability

Contributions to the Body of Knowledge and Value Added to the DSS Industry

This dissertation developed a new concept of measuring usability not by measuring efficiency and effectiveness directly from the system as is traditionally done, but by measuring a set of attributes that affect those measures. The measures obtained in the system assessments performed, however, were relative to the attributes, with the assumption that efficiency and

effectiveness would be affected by the quality of and presence or lack of presence of certain attributes. The links between usability measures and attributes were determined by observation of people working with DSSs and study of literature. The attributes were validated by surveying experts in distributed simulation. User satisfaction was measured directly in the standard way, asking users, because that variable cannot be determined without direct user feedback. This concept is shown in the figure below. Weights were also obtained for these attributes using expert survey data. The concept of weighted system attribute sets that affect usability measures was introduced. (It is important not to confuse the system attributes with usability attributes, which are in this discussion termed "measures.")

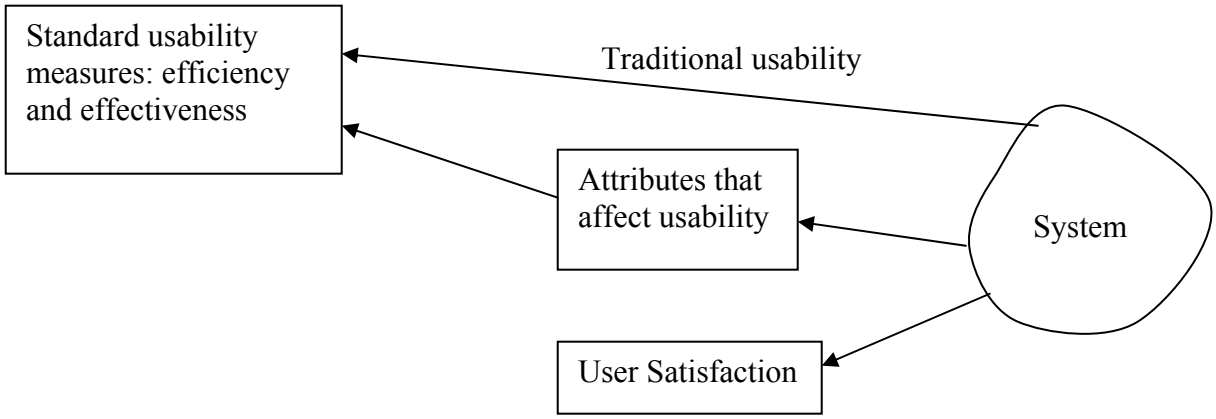

Figure 15. Usability Measures and Attributes Linked to a System

The concept of holistic usability was developed, looking at all people who work with the system as users, not just the end users. This multidimensional view of considering usability sheds light on the need for ease in design, development, and installation, as well as end use. In addition, the needs for good documentation and training were brought into focus. A systematic

way of measuring holistic usability was developed, as was a format for reporting both qualitative and quantitative system assessment results.

The study of what makes working with the system easy for all types of DSS users was the first study of usability for distributed simulation. The attribute set developed for the framework dimensions provides a guideline of items for designers to consider when developing a system and a baseline from which to measure an existing system's holistic usability. The adequate consideration of framework attributes will

- help to increase the productivity of developers and end users
- improve the chances of system success
- increase the utility of a DSS
- help ensure end user needs are met
- lower life cycle costs
- and improve satisfaction levels for all people who work with the system.

Future Research

Research is needed to further clarify the relationships between the attributes in each dimension and efficiency and effectiveness for each user. Efforts are needed to determine how best to adjust weights for different types of systems. The metrics used to measure holistic usability need study and refinement.

For the distributed simulation framework, long-term research to determine attribute sets customized to different types of simulations would be helpful. For instance, one might classify analytical research simulations into four types: engineering, business, medical, and pharmaceutical. A subset of the engineering analytical category could include design and operational modeling, e.g., a tank simulation or aircraft design and operation. Entertainment simulations could be categorized as games or experiential. Military simulations might be categorized as training or battlefield. Training simulations might be categorized as desktop, augmented reality, or virtual reality—all these being single user or multi-user interactive. Customized holistic usability attribute sets could be developed and used in both the formative and evaluative usability stages as tools.

Following is a list of more ideas for research.

- The framework and the concept of holistic usability can be generalized to other types of systems.
- the development of measuring methods to measure efficiency and effectiveness for each type of user relative to attributes
- in DSSs, a study of system mistakes and successes relative to the dimensions and attributes in the holistic usability framework
- refinement of attribute weights, sensitivity studies, a mathematical look at the construct of holistic usability
- specialized attribute sets for different types of systems
- cost studies to measure the benefit of applying holistic usability
- more validation studies on the attributes
- comparison studies between different systems of the same type relative to the attributes

- a study applying the framework to a new design project
- studies of DSS interfaces relative to users' needs and mental models
- use of metaphors and icons in DSS interfaces to reduce cognitive load on the users
- the study of customizable DSS interfaces: beginner, intermediate, and expert users; different types of users; multiple modes of visualization
- studies optimizing documentation aspects for different types of users
- a study relative to the dimensional attributes, of system failures and successes to see what attribute set characteristics led to success and which led to failure
- measurement studies to get a better understanding of the attributes effect on usability measures
- expansion of the framework into subattributes and subdimensions
- instilling good holistic usability for DSSs for engineering design
- facilitating ease of use for real-time, distributed real-estate market simulation

A number of possible subattributes and attributes need to be studied in the Programming dimension. Programmers mentioned that these items could make their work easier in DSSs:

- libraries of code for specialized processes
- programming tools specifically for distributed simulation
- clear conceptual and contextual modeling
- being able to hide detailed subunits of parts of the simulation from an overview so they don’t overwhelm the programmer (or end user)

- configuration control (already an attribute): compatibility between the simulation modules for exchanging data and control, good integration, uniformity in shared variable names between simulations
- clarity, simplicity, and good documentation for the APIs (mentioned several times by the programmers); example code and test cases for the APIs
- A clear separation of the model, its supporting simulation software, and the integration of that simulation software with the distributed simulation infrastructure needs to be strictly maintained. When these features are closely intertwined, long-term maintenance becomes very difficult. (The need to maintain separation between these three items was mentioned by two survey participants independently.)
- debug/remote debug capability

Survey participants mentioned the following items as things that could improve usability for interfaces in DSSs. They also suggest future research areas:

- different modes of visualization
- ease of adjusting parameters or even the basic model as conditions change
- adaptable user interfaces
- ensuring that the simplifications essential to implementing a simulation are congruent with the mental models of the users
- clear definition and easy collection of metrics the end users are interested in
- architecture with an easily-understandable model or metaphor for individual simulation modules

Conclusion

At the outset of this research, it seemed to me that distributed simulation was rare. When I started searching for survey participants and the results began to come in, it became apparent that it is actually quite common. It appears in many forms, from financial market real-time analysis to military war games to multi-user games played over the Internet in real time. An underlying theme for all the different user types in the framework is simplicity: the more clearly defined and organized their task is, the easier it will be for them. One programmer, when asked what would make working with a DSS easier for programmers, said: "Understanding the end goal, the objective. As silly as it sounds, it's not always clear." It is hoped that this holistic usability framework will help designers to focus on the essentials that make working with distributed simulation easy for users.

APPENDIX A: USER SURVEY FOR VALIDATION

Note: this survey reflects the text that was presented online for the validation survey.

1. Do you want to take this survey?

accept decline

A framework is being developed for the usability of distributed simulation systems (DSSs). In this framework, the goal is to make the job of each person who works with the system—whether an end user, designer, programmer, or installer—easier. The framework has six dimensions: end user needs and goals, end user interface(s), programming, training, installation, and documentation. Answers to the following questions will give important information that will be used in the validation and refinement of this holistic usability framework.

2. What category below describes your work area or interest in Distributed Simulation? Select as many categories as apply.

manager researcher end user programmer designer trainer

installer other

3. Which type(s) of DSS(s) do you have experience working with?

military entertainment aerospace business engineering medical

pharmaceutical other

4. How many years of experience do you have working with distributed simulation? ____

5. Which organization are you associated with (e.g., NASA) ?_____

Please indicate how strongly you agree or disagree with the following statements.

6. A central control and monitoring interface improves the efficiency of use of a distributed simulation system.

strongly disagree neutral agree strongly agree
disagree

7. The ability to change simulation parameters from a central control interface makes distributed simulation easier to use.

strongly disagree neutral agree strongly agree
disagree

8. The ability to start and stop individual simulations from a central control interface makes use distributed simulation easier.

strongly disagree	disagree	neutral	agree	strongly agree

9. The ability to communicate with other users who are logged into the system facilitates work coordination in distributed simulation systems.

strongly disagree	disagree	neutral	agree	strongly agree

10. Good exception handling makes working with distributed simulation systems less time consuming (when problems occur).

strongly disagree	disagree	neutral	agree	strongly agree

11. Showing relevant variables in all simulations running simultaneously helps the user learn about relationships between the simulations.

strongly disagree	disagree	neutral	agree	strongly agree

12. The ability to review the data from several simulation scenarios simultaneously results in a more efficient analysis process.

strongly disagree	disagree	neutral	agree	strongly agree

13. The ability to save and analyze statistics in the system makes a system easier to work with.

strongly disagree	disagree	neutral	agree	strongly agree

14. Good data visualization capability helps users understand simulation results faster.

strongly disagree	disagree	neutral	agree	strongly agree

15. The ability to combine information from different simulations in a way that helps the user understand their interrelationships results in a more satisfactory interface.

strongly disagree	disagree	neutral	agree	strongly agree

16. As the complexity of the programming in a DSS increases, the efficiency of the time spent programming decreases.

strongly disagree	disagree	neutral	agree	strongly agree

17. As the number of simulation written with proprietary simulation packages in a distributed simulation system increases, the difficulty of programming the system increases.

strongly disagree	disagree	neutral	agree	strongly agree

18. As the number of software "wrappers" to access individual simulations increases, the difficulty of programming the system increases.

strongly disagree	disagree	neutral	agree	strongly agree

19. The ease of programming the distributed simulation infrastructure affects the productivity of the programmers.

strongly disagree	disagree	neutral	agree	strongly agree

20. The faster programmers can be trained to understand the system, the more quickly their work can be accomplished.

strongly disagree	disagree	neutral	agree	strongly agree

21. The ease of programming the distributed simulation affects the job satisfaction of the programmers.

strongly disagree	disagree	neutral	agree	strongly agree

22. The less coding required to create a distributed simulation system, the easier the programming job.

strongly disagree	disagree	neutral	agree	strongly agree

23. The lower the level of expertise required for programming the distributed simulation system, the faster the programming task will proceed.

strongly disagree	disagree	neutral	agree	strongly agree

24. The design of the software infrastructure for a distributed simulation system affects how easy it is to connect individual simulations to it.

strongly disagree	disagree	neutral	agree	strongly agree

25. Compatible data formats between individual simulations make exchanging data between them easier.

strongly disagree	disagree	neutral	agree	strongly agree

26. Keeping a detailed log of the installation details, including problems encountered and solutions, saves time when questions arise or future problems occur.

strongly disagree	disagree	neutral	agree	strongly agree

27. The installation process will be faster if people of average ability can install the system.

strongly disagree	disagree	neutral	agree	strongly agree

28. Specifying the different skills needed to install the system helps to efficiently manage the process.

strongly disagree	disagree	neutral	agree	strongly agree

29. The fewer the number of people required for installation, the more efficient the installation process.

strongly disagree	disagree	neutral	agree	strongly agree

30. Good troubleshooting capability helps to ensure successful installation.

strongly disagree	disagree	neutral	agree	strongly agree

31. The time required to install a system is a good measure of how efficient the installation process is.

strongly disagree	disagree	neutral	agree	strongly agree

32. The easier the installation process is, the faster it will proceed.

strongly disagree	disagree	neutral	agree	strongly agree

33. The lower the skill level required, the faster the installation process will proceed.

strongly disagree	disagree	neutral	agree	strongly agree

34. Effective training is important to ensure that users can efficiently use the system.

strongly disagree	disagree	neutral	agree	strongly agree

35. A quick training process will help users become productive faster.

strongly disagree	disagree	neutral	agree	strongly agree

36. Written materials increase the efficiency of the training.

strongly disagree	disagree	neutral	agree	strongly agree

37. Having on-line training materials available increases the satisfaction level of trainees with the training.

strongly disagree	disagree	neutral	agree	strongly agree

38. Gearing the training presentation level to the knowledge level of the audience facilitates the learning process.

strongly disagree	disagree	neutral	agree	strongly agree

39. Training installers improves their ability to install a distributed simulation system successfully.

strongly disagree	disagree	neutral	agree	strongly agree

40. The quality of end user interface training affects the speed with which people learn to use the system.

strongly disagree	disagree	neutral	agree	strongly agree

41. Programmer training--familiarization with the system and its software design characteristics --helps programmers become productive quickly.

strongly disagree	disagree	neutral	agree	strongly agree

42. Good code-level programming documentation helps programmers work faster when developing a distributed simulation system.

strongly disagree	disagree	neutral	agree	strongly agree

43. A clearly specified and diagrammed software design makes programming distributed simulation easier.

strongly disagree	disagree	neutral	agree	strongly agree

44. Documenting end user needs and goals helps to ensure they are met.

strongly disagree	disagree	neutral	agree	strongly agree

45. Good training documentation helps trainees successfully learn the material.

strongly disagree	disagree	neutral	agree	strongly agree

46. Good installation documentation facilitates fast system installation.

strongly disagree	disagree	neutral	agree	strongly agree

47. Good user documentation helps users learn an interface faster.

strongly disagree	disagree	neutral	agree	strongly agree

48. The quality of user documentation affects the user's level of satisfaction with the system.

strongly disagree	disagree	neutral	agree	strongly agree

49. The quality of online help affects the level of satisfaction a user has with an interface.

strongly disagree	disagree	neutral	agree	strongly agree

50. Can you think of any other important factors that would make a distributed simulation system easy to use or work with?

---

51. What are the most important factors affecting the ease of programming a distributed simulation system? In other words, what factors most affect usability for programmers?

---

52. Do you have any comments to add that would be helpful for this study?

---

53. Good configuration control between distributed simulations is essential for efficiently programming a distributed simulation system.

strongly disagree	disagree	neutral	agree	strongly agree

APPENDIX B: VALIDATION SURVEY STATISTICAL ANALYSIS SPREADSHEET

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	n	q	p	z	1 - prob(Z<=z)	n*q	n*p
6. A central control and monitoring interface improves the ease of using a distributed simulation system.	0	2	3	38	20	63	0.0794	0.9206	12.3514	0.0000	5	58
7. The ability to change simulation parameters from a central control interface makes distributed simulation easier to use.	0	1	8	33	20	62	0.1452	0.8548	7.9316	0.0000	9	53
8. The ability to start and stop individual simulations from a central control interface makes use of distributed simulation easier.	1	2	8	35	17	63	0.1746	0.8254	6.8034	0.0000	11	52
9. The ability to communicate with other users who are logged into the system facilitates work coordination in distributed simulation systems.	1	2	14	24	22	63	0.2698	0.7302	4.1156	0.0000	17	46
10. Good exception handling makes working with a distributed simulation system less time consuming (when problems occur).	0	2	5	35	21	63	0.1111	0.8889	9.8219	0.0000	7	56
11. Showing relevant variables in all simulations running simultaneously helps the user learn about relationships between the simulations.	2	6	19	25	9	61	0.4426	0.5574	0.9022	0.1835	27	34

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	n	q	p	z	1 - prob(Z<=z)	n*q	n*p
12. The ability to review the data from several simulation scenarios simultaneously results in a more efficient analysis process.	0	3	11	35	14	63	0.2222	0.7778	5.3033	0.0000	14	49
13. The ability to save and analyze statistics in the system makes a system easier to work with.	1	0	10	39	12	62	0.1774	0.8226	6.6488	0.0000	11	51
14. Good data visualization capability helps users understand simulation results faster.	0	2	3	21	37	63	0.0794	0.9206	12.3514	0.0000	5	58
15. The ability to combine information from different simulations in a way that helps the user understand their interrelationships results in a more satisfactory interface.	1	3	10	31	18	63	0.2222	0.7778	5.3033	0.0000	14	49
16. As the complexity of the programming in a distributed simulation system increases the efficiency of the time spent programming decreases.	1	7	29	18	7	62	0.5968	0.4032	-1.5534	0.9398	37	25
17. As the number of simulations written with proprietary simulation packages in a distributed simulation system increases the difficulty of programming the system increases.	1	5	13	23	19	61	0.3115	0.6885	3.1795	0.0007	19	42

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	n	q	p	z	1 - prob(Z<=z)	n*q	n*p
18. As the number of software "wrappers" to access individual simulations increases the difficulty of programming the system increases.	0	3	19	33	7	62	0.3548	0.6452	2.3889	0.0084	22	40
19. The ease of programming the distributed simulation infrastructure affects the productivity of the programmers.	1	2	6	38	15	62	0.1452	0.8548	7.9316	0.0000	9	53
22. The less coding required to create a distributed simulation system the easier the programming job.	0	5	23	23	9	60	0.4667	0.5333	0.5175	0.3024	28	32
23. The lower the level of expertise required for programming the distributed simulation system the faster the programming task will proceed.	1	15	20	21	6	63	0.5714	0.4286	-1.1456	0.8740	36	27
20. The faster programmers can be trained to understand the system the more quickly their work can be accomplished.	0	3	11	37	12	63	0.2222	0.7778	5.3033	0.0000	14	49
21. The ease of programming the distributed simulation system affects the job satisfaction of the programmers.	0	5	28	23	6	62	0.5323	0.4677	-0.5091	0.6946	33	29
24. The design of the software infrastructure for a distributed simulation system affects how easy it is to connect individual simulations to it.	0	0	4	46	13	63	0.0635	0.9365	14.2085	0.0000	4	59

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	n	q	p	z	1 - prob(Z<=z)	n*q	n*p
25. Compatible data formats between individual simulations make exchanging data between them easier.	1	1	3	30	28	63	0.0794	0.9206	12.3514	0.0000	5	58
26. Keeping a detailed log of the installation details including problems encountered and solutions saves times when questions arise or future problems occur.	1	0	5	41	16	63	0.0952	0.9048	10.9445	0.0000	6	57
27. The installation process will be faster if people of average ability can install the system.	0	6	18	29	9	62	0.3871	0.6129	1.8251	0.0340	24	38
28. Specifying the different skills needed to install the system helps to efficiently manage the process.	1	1	17	35	9	63	0.3016	0.6984	3.4314	0.0003	19	44
29. The fewer the number of people required for installation the more efficient the installation process.	0	5	13	35	9	62	0.2903	0.7098	3.6373	0.0001	18	44
30. Good troubleshooting capability helps to ensure successful installation.	0	2	6	39	16	63	0.1270	0.8730	8.8923	0.0000	8	55
31. The time required to install a system is a good measure of how efficient the installation process is.	6	16	19	18	3	62	0.6613	0.3387	-2.6835	0.9964	41	21
32. The easier the installation process is the faster it will proceed.	3	10	23	19	7	62	0.5806	0.4194	-1.2868	0.9009	36	26

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	n	q	p	z	1 - prob(Z<=z)	n*q	n*p
33. The lower the skill level required the faster the installation process will proceed.	1	17	27	15	1	61	0.7377	0.2623	-4.2205	1.0000	45	16
34. Effective training is important to ensure that users can efficiently use the system.	0	4	3	35	19	61	0.1148	0.8852	9.4403	0.0000	7	54
35. A quick training process will help users become productive faster.	1	9	17	32	3	62	0.4355	0.5645	1.0246	0.1528	27	35
36. Written materials increase the efficiency of the training.	0	4	18	33	7	62	0.3548	0.6452	2.3889	0.0084	22	40
37. Having on-line training materials available increases the satisfaction level of trainees with the training.	0	3	21	32	6	62	0.3871	0.6129	1.8251	0.0340	24	38
38. Gearing the training presentation level to the knowledge level of the audience faciliates the learning process.	0	1	4	37	19	61	0.0820	0.9180	11.9022	0.0000	5	56
39. Training installers improves their ability to install a distributed simulation system successfully.	1	0	8	38	14	61	0.1475	0.8525	7.7621	0.0000	9	52
40. The quality of end user interface training affects the speed with which people learn to use the system.	0	2	4	37	17	60	0.1000	0.9000	10.3280	0.0000	6	54

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	n	q	p	z	1 - prob(Z<=z)	n*q	n*p
41. Programmer training--familiarization with the system and its software design characteristics--helps programmers become productive quickly.	0	0	6	45	9	60	0.1000	0.9000	10.3280	0.0000	6	54
42. Good code-level programming documentation helps programmers work faster when developing a distributed simulation system.	0	0	8	31	19	58	0.1379	0.8621	7.9966	0.0000	8	50
43. A clearly specified and diagrammed software design makes programming distributed simulation easier.	0	0	5	41	14	60	0.0833	0.9167	11.6775	0.0000	5	55
44. Documenting end user needs and goals helps to ensure they are met.	0	2	3	30	24	59	0.0847	0.9153	11.4528	0.0000	5	54
45. Good training documentation helps trainees successfully learn the material.	0	0	6	43	11	60	0.1000	0.9000	10.3280	0.0000	6	54
46. Good installation documentation facilitates fast system installation.	0	1	10	39	9	59	0.1864	0.8136	6.1842	0.0000	11	48
47. Good user documentation helps users learn an interface faster.	0	2	9	35	13	59	0.1864	0.8136	6.1842	0.0000	11	48
48. The quality of user documentation affects the user's level of satisfaction with the system.	1	2	7	40	9	59	0.1695	0.8305	6.7665	0.0000	10	49

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	n	q	p	z	1 - prob(Z<=z)	n*q	n*p
49. The quality of on-line help affects the level of satisfaction a user has with an interface.	1	3	13	33	10	60	0.2833	0.7167	3.7244	0.0001	17	43
53. Good configuration control between distributed simulations is essential for efficiently programming a distributed simulation system.	0	1	4	19	14	38	0.1316	0.8684	6.7186	0.0000	5	33

APPENDIX C: INSTITUTIONAL REVIEW BOARD APPROVAL

Office of Research & Commercialization

February 20, 2006

Jeffrey Dawson
2422 Abalone Blvd.
Orlando, FL 32833

Dear Mr. Dawson:

With reference to your protocol #06-3206 entitled, **"Development of a Holistic Usability Framework for Distributed Simulation Systems,"** I am enclosing for your records the approved, expedited document of the UCFIRB Form you had submitted to our office. **This study was approved on 2/16/06. The expiration date will be 2/15/07.** Should there be a need to extend this study, a Continuing Review form must be submitted to the IRB Office for review by the Chairman or full IRB at least one month prior to the expiration date. This is the responsibility of the investigator. **Please notify the IRB office when you have completed this research study.**

Please be advised that this approval is given for one year. Should there be any addendums or administrative changes to the already approved protocol, they must also be submitted to the Board through use of the Addendum/Modification Request form. Changes should not be initiated until written IRB approval is received. Adverse events should be reported to the IRB as they occur.

Should you have any questions, please do not hesitate to call me at 407-823-2901.

Please accept our best wishes for the success of your endeavors.

Cordially,

Barbara Ward

Barbara Ward, CIM
UCF IRB Coordinator
(FWA00000351 Exp. 5/13/07, IRB00001138)

Copies: IRB File
Luis Rabelo, Ph.D.

BW:jm

12443 Research Parkway • Suite 302 • Orlando, FL 32826-3252 • 407-823-3778 • Fax 407-823-3299

LIST OF REFERENCES

Ackermann, D., & Tauber, M. J. (Eds.). (1990). *Mental models and human-computer interaction 1*. Amsterdam: North-Holland Elsevier Science Publishers.

Agarwal, R., & Venkatesh, V. (2002). Assessing a firm's web presence: a heuristic evaluation procedure for the measurement of usability. *Information Systems Research, 12*(2), 168-186.

American National Standards Institute. (2001). *ANSI NICITS 354-2001, for information technology-common industry format for usability test reports*: American National Standards Institute.

Badre, A. N. (2002). *Shaping web usability, interaction design in context* (first ed.). Boston, MA: Addison-Wesley.

Banks, J., John S. Carson, I., & Nelson, B. L. (1996). *Discrete-event system simulation*. Upper Saddle River, New Jersey: Prentice Hall.

Belleman, R. G., & Shulakov, R. (2002). *High performance distributed simulation for interactive simulated vascular reconstruction.* Paper presented at the ICCS 2002.

Bevan, N., & Macleod, M. (1994). Usability measurement in context. *Behavior and Information Technology, 13*, 132-145.

Brinkman, W.-P., Haakma, R., & Bouwhuis, D. G. (2001). *Usability evaluation of component-based user interfaces.* Paper presented at the Interact 2001, Tokyo, Japan.

Brooke, J. (1996). Chapter 21. Sus: A 'quick and dirty' usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester & I. L. McClelland (Eds.), *Usability evaluation in industry* (pp. 189-194). London: Taylor & Francis.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction* (first ed.). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Ceranowicz, A., Dehncke, R., & Cerri, T. (2003). *Moving toward a distributed continuous experimentation environment.* Paper presented at the Interservice/Industry Training, Simulation, and Education Conference.

Chapanis, A. (1959). *Research techniques in human engineering*. Baltimore: The John Hopkins Press.

Chorafas, D. N., & Steinmann, H. (1995). *Virtual reality, practical applications in business and industry* (First ed.). Upper Saddle River, New Jersey: Prentice Hall PTR.

Defense Modeling and Simulation Office. (2004). High level architecture. from https://www.dmso.mil/public/transition/hla/

Eberts, R. E. (1994). *User interface design* (first ed.). Englewood Cliffs, New Jersey: Prentice-Hall.

Fishwick, P. A. (2004). Toward an integrative multimodeling interface: A human-computer interface approach to interrelating model structures. *Simulation, 80*(9), 421-432.

Garrido, J. M. (2001). *Object-oriented discrete-event simulation with java*. New York: Kluwer Academic/Plenum Publishers.

Gentner, D., & Stevens, A. L. (Eds.). (1983). *Mental models*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Green, T. R. G. (1990). Limited theories as a framework for human-computer interaction. In D. Ackermann & M. J. Tauber (Eds.), *Mental models and human-computer interaction 1* (pp. 3-39). Amsterdam: Elsevier Science.

Gulliksen, J., Boivie, I., Persson, J., Hektor, A., & Herulf, L. (2004, October 23-27, 2004). *Making a difference--a survey of the usability profession in sweden.* Paper presented at the NordiCHI '04, Tampere, Finland.

Harmon, P. (2005). The OMG's model driven architecture. Retrieved 18 October 2005, from http://www.cutter.com/research/2002/edge020611.html

Henry, P. (1998). *User-centered information design for improved software usability*. Norwood, Massachusetts: Artech House.

Hornbaek, K., & Frokjaer, E. (2004). Usability inspection by metaphors of human thinking compared to heuristic evaluation. *International Journal of Human-Computer Interaction, 17*(3), 357-374.

Hornbaek, K. (2006). Current practice in measuring usability: Challenges to usability studies and research. *International Journal of Human-Computer Studies, 64*, 79-102.

International Organization for Standardization. (1996). *ISO 9241, ergonomic requirements for office work with visual display terminals -- part 10: Dialogue principles.*

International Organization for Standardization. (1997). *Iso 9241-14, ergonomic requirements for office work with visual display terminals -- part 14: Menu dialogues.*

International Organization for Standardization. (1998). *Iso 9241-11, ergonomic requirements for office work with visual display terminals-- part 11: Guidance on usability.*

James, W. (1890). *The principles of psychology*. New York: Henry Holt and Company.

Johnson-Laird, P. N. (1983). *Mental models, towards a cognitive science of language, inference, and consciousness*. Cambridge, MA: Harvard University Press.

Jordan, P. W., Thomas, B., Weerdmeester, B. A., & McClelland, I. L. (1996). *Usability evaluation in industry* (first ed.). Bristol, PA: Taylor & Francis.

Leeds University. (2005). What do we mean by usability? Retrieved 17 October 2005, from http://www.leedsmet.ac.uk/inn/usabilityservices/whatusability.htm#userexp

Mayhew, D. J. (1999). *The usability engineering lifecycle, a practitioner's guide for user interface design* (first ed.). San Diego, CA: Academic Press.

McCormick, E. J. (1976). *Human factors in engineering and design* (fourth ed.). New York: McGraw-Hill.

McGee, M. (2004, April 24-29, 2004). *Master usability scaling: Magnitude estimation and master scaling applied to usability measurement.* Paper presented at the CHI 2004, Vienna, Austria.

National Institute of Standard and Technology. (1999). *Common industry format for usability test reports, version 1.1*: National Institute of Standard and Technology.

Nielsen, J. (1993). *Usability engineering* (first ed.). San Diego, CA: Academic Press.

Nielsen, J. (1994). Discount usability engineering. In R. G. Bias & D. J. Mayhew (Eds.), *Cost-justifying usability* (pp. 245-272). San Francisco: Morgan Kaufmannn.

Norman, D. A. (1969). *Memory and attention, an introduction to human information processing*. New York: John Wiley & Sons.

Norman, D. A. (1986). Cognitive engineering. In D. A. Norman & S. W. Draper (Eds.), *User-centered system design* (pp. 31-61). Hillsdale, NJ: Erlbaum.

Oakhill, J., & Garnham, A. (Eds.). (1996). *Mental models in cognitive science, essays in honour of phil johnson-laird*. East Sussex, UK: Psychology Press.

Pegden, C. D., Shannon, R. E., & Sadowski, R. P. (1995). *Introduction to simulation using siman* (2nd ed.). New York: McGraw-Hill.

Perumalla, K. S. (2002). *Position paper.* Paper presented at the XMSF Workshop, Monterey, CA.

Preece, J., Rogers, Y., & Sharp, H. (2002). *Interaction design* (first ed.). New York, NY: John Wiley & Sons.

Rubin, J. (1994). *Handbook of usability testing*. New York: John Wiley & Sons.

Shneiderman, B. (1992). *Designing the user interface: Strategies for effective human-computer interaction* (second ed.). Reading, MA: Addison-Wesley.

Snyder, C. (2003). *Paper prototyping*. San Francisco, California: Morgan Kaufmann Publishers.

Sogandares, F. M. (2002). Stone axes and warhammers: A decade of distributed simulation in aviation research. 125-132.

SPSS, I. (2003). *How to get more value from your survey data*: SPSS, Inc.

SPSS, I. (2005). *The how's and why's of survey research*: SPSS, Inc.

Sulistio, A., Yeo, C. S., & Buyya, R. (2004). A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. *Software: Practice and Experience, 34*(7), 653-673.

Trenner, L., & Bawa, J. (1998). *The politics of usability, a practical guide to designing usable systems in industry* (first ed.). Berlin: Springer.

Walpole, R. E., & Myers, R. H. (1978). *Probability and statistics for engineers and scientists*. New York: Macmillan Publishing Company.

Welford, A. T., & Birren, J. E. (1965). *Behavior, aging and the nervous system*. Springfield, Illinois: Charles C. Thomas.

Wickens, C. D. (1992). *Engineering psychology and human performance* (second ed.). New York, NY: HarperCollins.

Young, R. M. (1983). Chapter 3. Surrogates and mappings: Two kinds of conceptual models for interactive devices. In D. Genter & A. L. Stevens (Eds.), *Mental models* (pp. 35-52). Hillsdale, New Jersey: Lawrence Erlbaum Associates.