Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

USABILITY STUDY OF THE VIRTUAL TEST BED AND DISTRIBUTED SIMULATION

Jeffrey W. Dawson

Department of Industrial Engineering
and Management Systems
University of Central Florida
4000 Central Florida Boulevard
Orlando, Florida 32816 U.S.A.

Ping Chen

Institute for Simulation
and Training
University of Central Florida
3100 Technology Pkwy
Orlando, Florida 32826 U.S.A.

Yanshen Zhu

Department of Electrical and Computer Engineering
University of Central Florida
4000 Central Florida Boulevard
Orlando, Florida 32816 U.S.A.

ABSTRACT

Improving the usability of a Distributed Simulation System (DSS) test bed is the focus of this paper. An introduction to the field of usability is given, followed by a discussion of the characteristics of DSSs. Then the usability of DSSs is considered. The Virtual Test Bed (VTB), a sample DSS we have improved the usability of, is described. The methodology used to improve the VTB's usability is given. With the goal of improving usability for end users, prototyping of a graphical user interface is discussed; both software and paper prototypes are considered. Lessons learned provide insights into problems we encountered. Research on important aspects of DSSs that affect usability is reflected in a table that summarizes key issues. Our anticipated future research in this area is also discussed.

1 INTRODUCTION

This paper discusses the usability of Distributed Simulation Systems (DSSs). An introduction to the field of usability is followed by a description of a DSS. Our work towards improving the usability of a sample project—the Virtual Test Bed (VTB)—is discussed, while elaborating on aspects of the system that affect usability. The need for user surveys and task analyses in usability analysis is considered. Following this is a discussion of our experience with graphical user interface (GUI) prototyping for the VTB. Lessons learned are reflected in a framework for the usability of DSSs.

Issues that affect the usability of DSSs for people involved in their design, development, installation, maintenance, operation, and usage involve many challenging areas of research. Typically, usability focuses on the end user of a system. Thus, a central focus of our VTB usability improvement effort is to create a GUI for the end user. There are also usability issues that affect everyone else who works with the system. For instance, software usability is important in the design of the DSS infrastructure. Most end users do not want to write code, though, so the usability of the infrastructure itself is for a different type of user—the programmers.

2 USABILITY

Usability is a measure of how easy a system is to use. Standardized industry measurements are often used when measuring usability: effectiveness—whether or not tasks can be successfully performed by users, efficiency—how fast those tasks are performed, and user satisfaction—how much users like the system. These three measures are important, but there are many other considerations in a system's usability. Usability is a multidisciplinary field that requires knowledge of cognitive psychology, human engineering, anthropology, technical writing, human factors, computer science, and other disciplines.

Usability can either be instilled into a design as part of the design process or evaluated after the design is finished. When incorporating usability into the design of a software product, expertise is needed in usability, user interface design, writing for users, and user interface development. Focusing on system users early in the design process can help ensure that a simulation project is successful. When usability is evaluated for an existing system, recommendations may lead to improvements, but often these improvements are slated for the next generation of the system.

At the requirements phase of a project, one develops user profiles, performs task analyses, and considers technology constraints and design principles, which are used to develop usability goals (Mayhew 1994). This process is followed by iterative design development. The development of profiles of users and the analysis of tasks they will perform are key to incorporating usability in design. While creating prototypes, heuristics—lists of good usability design principles—are used as a guide. Experimental observation using typical users is essential in evaluating prototypes. Each user is different, reflecting the infinite variations in human skills, behaviors, attitudes, motivations, and abilities. Giving users predefined tasks, then watching them perform them, is the key to improving an interface.

When the usability of an existing design is measured, experts can evaluate the design using a set of usability heuristics, and users can be observed performing a set of tasks in order to measure the *system's* performance. A survey is often used to measure user satisfaction. A specification matrix can be used to quantitatively list usability goals and measurements. The establishment of quantitative usability goals that are measured as a design evolves can help ensure that the goals are achieved.

3 DISTRIBUTED SIMULATION SYSTEMS

DSSs involve simulations that are distributed over more than one computer, which are often 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. Different organizations and companies sometimes need to interface their simulation systems in order to solve problems; DSSs offer an attractive way to do this. Currently, the biggest users of DSSs are military organizations, who use them for a variety of purposes, such as war games and training.

One of the 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 Object Management Group adopted the HLA as the facility for distributed simulation 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.

4 USABILITY IN DISTRIBUTED SIMULATION

In looking at DSSs 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 the 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.

Connecting individual simulation models in a distributed simulation environment is a nontrivial task. Boer and Verbraeck (2003) noted that "the interoperability in distributed simulation involves at least the *data transfer* and the *time synchronization* between the simulation models" (p. 829). They further noted that the data transfer could be an event or an entity transfer. Also provided in their paper is a formal theoretical framework for interfacing commercial off-the-shelf simulation models with an architecture, such as the HLA-RTI or the FAMAS Simulation Backbone Architecture. For their example they chose FAMAS over the HLA-RTI, noting that because their system was simple, the power and complexity of the HLA-RTI was not needed. Also discussed is the need for a "wrapper" in order to access internal data in programs such as Arena. Arena is a commercial off-the-shelf product from Rockwell Software. All types of DSS users must address the issue of the interconnectivity of individual models, and to a novice user of a DSS system, it is a key usability issue.

Fowler and Rose (2004) wrote that an "emerging grand challenge" is "true plug-and-play interoperability of simulations and supporting software within a specific application domain" (p. 474). They mentioned HLA as a partial solution, but noted a number of weaknesses it has, which leaves its future as a long-term solution an open issue.

Distributed simulation systems are usually a large team effort. Each member of the team can be considered a user in some way. Indeed, from a managerial perspective, the usability depends 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 is how easy it is to obtain the desired data and how good the data are. From a maintainer's perspective, usability is how easy the system is to maintain. The usability of a DSS is not simply the usability of its user interfaces.

A DSS has multiple users at several levels of system interaction. Activities include researching, analyzing, and studying; starting and coordinating models; inputting data and updates; constructing models; performing training exercises; and selecting simulation modules to incorporate into the distributed system. Users include researchers, system operators, domain experts, programmers, trainees, trainers, and experimental subjects. The overall usability of a DSS reflects the needs of all these types of users. This concept goes beyond an analysis of traditional GUI usability, taking a holistic view of usability. However, if one considers the usability measures of efficiency, effectiveness and user satisfaction, interactions of the system with each type of user affects the usability of the DSS.

5 VIRTUAL TEST BED USABILITY STUDY

We undertook a project to improve the usability of the Virtual Test Bed (VTB), a DSS developed to simulate NASA spaceport and related systems. This system is constructed using individual simulation models, some of which use advanced modeling techniques. The VTB is a prototypical system used to test and develop concepts for distributed simulation for NASA.

5.1 The Virtual Test Bed

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 its flow 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 causing 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.

At present, modifications are being made to the VTB that will enhance its current capabilities and provide new functionality. The VTB is a good research vehicle for usability, particularly because the system is prototypical, so that the design is not frozen and can be modified as required.

5.2 Usability Improvement

Our team's goal was to improve the usability of the VTB. Without a control GUI, however, there is no end-user usability, so we began an interaction design project to develop a control GUI for the system. Several other important tasks emerged as a result: the need to obtain more detailed system configuration information, the need to improve system documentation, the need to change the design concept to allow evolution of the system (which required reprogramming some aspects of the models), and the need to stabilize a system that had shown some signs of instability.

The first tasks we accomplished were a literature survey of distributed simulation systems and usability. This was followed by an assessment of the current system, which consisted of initial documentation of the system and development of Unified Modeling Language (UML) diagrams of some key software classes (particularly those that show interactions). This was followed by an assessment of possible current and future uses of the system and the tasks that would be performed by end users, and interviewing potential end users.

5.3 User Survey and Task Analysis

Several potential users of distributed simulation were surveyed at Kennedy Space Center (KSC). Both expert and novice users expressed their desire to continue using or adopt the use of simulation in their work areas. These data were used as a partial basis for task analysis and future planning.

It was determined that all of the participants were beyond the beginner level of computer expertise, which indicates that some level of familiarity with standard computer interfaces can be assumed for the typical user. In addition, none was an expert in simulation, which indicates little or no familiarity with simulation should be assumed in the end-user interface. The survey provided not only demographic information but also many comments about what the potential users would like to see in and possible uses for a DSS. This type of information is useful in determining what features are important to users of DSSs.

5.4 Graphical User Interface Design Approach

The main objective of this project was to improve the usability of the VTB. After looking at user requirements and characteristics, the technology available, what other researchers have done, and the existing system, we embarked on an interaction design project to create a control GUI for the VTB.

A review of the literature shows that a number of researchers have constructed control GUIs for the HLA-RTI. For example, Adelanto and Deman (2002) created a control GUI for an airport simulation (also see Adelanto 2004). This GUI allows "the user to capture the current situation (weather conditions, sensors located on the airport layout, expected aircraft schedule, etc.). Representation of the current situation is saved into a set of files that will be loaded by the corresponding federates participating in the HLA airport federation" (p. 107). The approach taken by Adelanto and Deman involves six federates running on separate computers, five for simulation and one for the RTI, plus a Java GUI. Each federate model is programmed in C++, which allows more flexibility than using a proprietary simulation package that does not allow access to programming code. Once all parameters are selected, the simulation is started by pressing a button in the GUI. A notable design aspect of this approach is that the GUI itself is not part of the RTI, but remains outside of it. Another interesting aspect is that a separate animator federate exists to show simulation activity.

In order to begin making a GUI for the VTB, the team was divided up into subteams to approach the problem. The subteams were: (1) user experience/software architecture design, (2) software design, (3) help system design.

In order to design a DSS GUI, first there is the conceptualization of the user experience, which depends on the type of user logged into the system and the tasks to be performed. (We determined that when a user first logs into the system, it will detect the type of user and adjust the options accordingly.) Then there are issues of how the display is to be designed, taking into account good usability practices. Although a large body of decades of human-computer interaction research can be drawn upon to assist in development, a DSS presents unique interface design challenges, particularly with respect to the presentation of a conceptual model of the DSS to the user. As noted by Law and Kelton (2000), there are many pitfalls that can befall a simulation study, one of which is to treat it as "primarily an exercise in computer programming" (p. 92). The primary goal is to provide a tool for end users to make decisions and solve problems. The rationale for having usability be a key component of development is to ensure user goals are always in mind. The technology to accomplish those goals is secondary.

Although creating an interface that serves several types of users (end users, system administrators, programmers) is envisioned, the focus of the initial design was on the end user performing basic tasks. The basic user experience design concept is that of a user sitting down to the interface, logging in, selecting models, and running a simulation.

Several preliminary tasks analyses were performed by team members. After enough task scenarios were created, the next step in the user experience design process was to create low-fidelity prototypes for testing, using the task scenarios as guidance. Advantages of low-fidelity prototyping include low cost, speed of development, and the ability to make changes quickly and inexpensively. Paper prototyping was selected as an avenue of design exploration for this project. Paper prototyping is a tool often used by software vendors and Web site designers (Snyder 2003). Because the software development of an interface for the VTB (or any DSS) is technically complex and time consuming, paper prototypes allow the ability to explore design concepts which not could not be explored with real prototypes. While paper prototyping may sound simple, one is actually designing the software architecture and the user experience; many questions arise at each step of the way. In addition, in a complete prototype there may be hundreds of different screen components, which must be organized effectively.

5.5 Graphical User Interface Software Design

Ultimately, a user interface is needed offering control and monitoring capability for an HLA-RTI federation, visualization of data in real-time and also after the simulation stops, the ability to change parameters in remote federate simulations, and other functionalities. Recent software developments involving distributed computing capabilities using technology such as the XML data format and Java applications suggest that an evolution in distributed simulation interfaces will occur, which could offer the end-user and system operators large increases in usability and productivity. Developing distributed simulation capabilities in a manner that addresses user needs and allows the user to easily achieve those needs without technical support while using the system is important.

The VTB was designed so that each model, running locally, gives local prompts that the user must enter to allow the system to continue. These local prompts were removed in order to allow complete control from a control GUI. Some software development aspects of the control GUI follow.

The Arena models communicate with the HLA-RTI though the NIST-developed manufacturing adapter, written in C++. The adapter provides a limited set of classes that can be accessed via the RTI. In order to start and stop Arena remotely, however, WebLogic server was chosen. WebLogic server runs alongside the RTI and can communicate with Arena through Arena's built-in Visual Basic for Applications (VBA) interface. Java code in WebLogic server was written to start and stop Arena models via the GUI.

Federates written in Java do not require an adapter and can be started in a similar fashion via WebLogic server. Monitoring of the federation during simulation takes place via the RTI, taking advantage of messages that are available in the adapter for proprietary simulations and also through Java for other simulations and real-time participants. (Real-time participants include anything from a spreadsheet used in calculations, to special-purpose software, to data pulled from Web sites.)

The two above-mentioned approaches, a Web server control layer and monitoring information via the RTI, were combined into a single GUI. In addition to control and monitoring, other functionality is needed, such as the ability to send parameters to simulation models and data visualization capability.

Java was chosen as the programming language for the GUI because it offers operating system platform independence. In addition to reusable and easily-modifiable object-oriented code, Java offers flexibility in using and constructing software modules that may be used for multiple projects. The control federate is able to send and receive messages to any federate, both before the simulation starts and while the simulation is running via the RTI.

An important part of any computer system is the component that provides help to the user. The help framework was implemented using the Sun Microsystems JavaHelp framework, which uses Java Swing classes. This is a flexible, modular approach that can be used to provide help in a variety of formats, such as on-line search, context-sensitive menus, and either client-side or Web browser-enabled windows.

The language used to communicate with users via help systems is an important usability component. The language needs to be written in a way that is easy to understand. Professional writers often prepare the messages. Given a software framework with which to provide help, development of the content is the most time consuming part. As in other aspects of usability, ideally this help content would be tested with users to iteratively develop and improve it.

Figure 1 below shows the GUI design approach. 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. A help module provides help and explanatory information to the user.

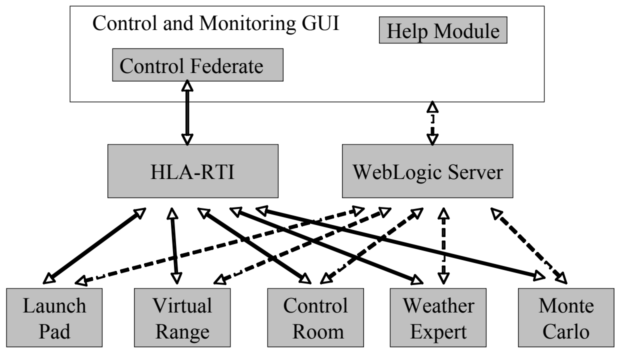

Figure 1: Virtual Test Bed GUI Design Approach

5.6 Lessons Learned

Often the design and implementation of DSSs are difficult, yet a system is eventually delivered that succeeds in fulfilling the customer's requirements. The day-to-day operations of large scale DSSs may be left to contractors or company technicians, so that the end user only uses the system when support personnel are working to keep the system up and running. Although it would be interesting to read about usability problems for all types of users in DSSs, large and small, reporting usability problems is a delicate matter. Even when usability problems are uncovered in a study, organizations are loath to have them publicly (and sometimes even privately) reported. When performing a usability study, a rule is to always report good news to the client first, then to soften the negatives. But the usability problems encountered and recommendations for eliminating them are the key to improvements. Recommendations to improve the usability of a system lower the cost of system use and improve user satisfaction, but are often not implemented until the next iteration or generation of the system.

In setting out to evaluate and improve the VTB's usability, the team learned some lessons. Space does not permit a full discussion of the details of the lessons learned, but three problems we encountered are discussed below.

Documentation of the technical aspects of system constructions was lacking. Many people worked on various models, and they left little documentation either as comments in code or in text. Most programmers do not make much effort in documenting their code; it must be emphasized and tracked by management. Thus, a major unexpected task for the usability team was to create documentation for the system. The first step of this was to create UML Diagrams. Class and sequence diagrams are particularly helpful. Creating documentation required time-intensive exploration of the systems code on its several computers. *Lesson:* Documentation is crucial to follow-on development and team efficiency.

The VTB was designed as a research project and the professional programmers and students who wrote the code generally did not plan for others to take their places and continue the project. This is a major stumbling block to new programmers. *Lesson:* Plan for continuation and for new programmers to join the project by emphasizing the need for usability for programmers in the design.

The HLA-RTI, interfaces between the RTI and legacy systems, and other technical aspects of DSSs are especially challenging, due to a lack of standardized interfaces. *Lesson:* The simulation community needs to work towards designing standardized programming interfaces for simulation programs to interact in a distributed environment.

Our team tackled solving the first two problems. The third problem needs to be addressed by the simulation community on a global basis.

6 FRAMEWORK FOR USABILITY OF DISTRIBUTED SIMULATION SYSTEMS

The focus of this project has been mainly on the typical end user. Assessing the overall usability of a DSS, we can expand our focus to include not just the end users but other people who interact with the system. A large number of issues are involved with the usability issues associated with this expanded list of users. Software usability is a factor, which includes the usability of individual programming languages, the infrastructure on which the distributed simulation runs, the programming and technical aspects of how legacy simulation software—both proprietary and open source—interact, and the effectiveness and ease of team communication (for both local and distributed teams). In addition to the technical software aspects, there are managerial issues about how to facilitate the process. The most important managerial issue, from the standpoint of efficiency and productivity, is documentation—of all aspects of the DSS project. Documentation needs include secure archives of all software code written (with backups in different locations), as well as documentation of all code changes, all computer setup and installation information, the network layer, the goals and objectives of the project, and the clients' needs and desires. This last point—the clients' needs and desires—should be reviewed periodically to make sure the project is on the right track.

Many issues have been identified in this study. One approach taken was that a team exercise was conducted wherein each team member independently suggested a list of items he or she thought necessary to include in a graphical user interface (GUI) for DSSs. In addition to the teamwork exercise, studying other DSSs, surveying the literature, and talking to designers of systems has yielded numerous ways to improve DSS usability. Table 1 below lists twenty-nine issues that have been identified as being important to the usability of DSSs. A number of these issues concern the need for a control federate. Some of the entries are specific to DSSs, but some, such as the need for data visualization, are also true for simulation systems in general. The distributed nature of the system increases the possibilities, however. For example, data visualization on a locally-contained simulation would be restricted to what that computer was accessing. In a distributed environment, with globally-distributed simulation engines and live participants, data visualization is potentially more powerful. This framework is also a checklist for DSS designers.

Table 1: Usability Issues in Distributed Simulation Systems

Category	#	Issue
User needs	1	Are the goals and tasks of the end users fully explored, and is the system designed to meet those needs?
User interface design	2	A central control and monitoring federate is needed.
	3	There is a need to allow users to play with the system; the system needs to be fun to use.
	4	The system needs to be designed so that the interface and its accompanying documentation and help files help the users develop a viable mental model of the system.
	5	Timing over the communication network. End users do not want to wait for responses. This will vary, however, for simulations that are known to take an extended period of time to execute.
	6	The ability to change parameters of individual federates from a control federate is needed.
	7	The ability to start, stop, and pause federates from a control federate is needed.
	8	The user should be able to determine who is logged into the system and to communicate with them.
	9	Multiuser capability
Data visualization and analysis	10	The control federate needs to have a display that shows the relevant factors in other simulations running simultaneously that are affecting the model that is currently running.
	11	Ability to view several scenario's data simultaneously
	12	Ability to save and analyze statistics
	13	Ability to save scenarios, recall them, and temporally examine events
	14	Data visualization capability (real-time during simulation and after the simulation stops)
Programming, configuration and installation	15	Ease of configuration/installation
	16	Exception handling is a major problem. In DSSs, when a federate gets stuck, there is often no way for the users to know.

Table 1 (Continued): Usability Issues in Distributed Simulation Systems

Category	#	Issue
	17	Ease of interconnectivity of individual models
	18	Good documentation of programming code and system configuration is especially important.
	19	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.
	20	Ease of interconnectivity of the network infrastructure(s) required to run the simulation
	21	Ease of integrating legacy systems
	22	Plan ahead for local models to be used in a distributed simulation (e.g., plan for needed program modifications to do such things as allow local GUI interaction prompts and data entry to be remotely executed).
	23	Troubleshooting support for when things go wrong
Training	24	Ease of training for those who use and support the system
	25	Skill levels of personnel needed to operate and maintain the system. Changing the skill level of a person involved in a task requires a re-evaluation of the usability for that task.
Infrastructure	26	Reliability/self healing
	27	Availability (percentage uptime)
	28	Longevity and continuity: As technology in software and hardware evolve, can we transition and maintain the capability?
	29	Ease of upgrading the hardware

A wide variety of usability issues is present in the above list. For example, issue 15 concerns ease of configuration and installation for installers. This relates to how easy it is to install a system and get it operational and would depend on the overall design of the system, the skill level needed to perform the task, and the available documentation. Issue 14, data visualization capability, is a matter of data presentation and manipulation, human cognition and perception, and how best to design the capability to assist in problem solving. These examples hint at the broad areas covered in this holistic usability framework for DSSs.

7 CONCLUSION

The success of distributed simulation systems depends on their usability. The above framework for the usability of DSSs can be used to evaluate existing DSSs or as an aid in developing new ones. Future work with the VTB will involve developing data visualization and more extensive control capabilities in its control GUI. User input will also be used to refine the VTB and work towards its implementation in a workplace environment.

ACKNOWLEDGMENTS

This research was conducted by the University of Central Florida under NASA contract NAS 1003006. Team members Alvaro Cantillo, Ping Chen, Serge Sala-Diakanda, Fred Gruber, Ethling Hernandez, Mario Marin, and Yanshen Zhu contributed to this research.

REFERENCES

Adelantado, Martin. 2004. Rapid prototyping of airport advanced operational systems and procedures through distributed simulation. *Simulation* 80 (1):5-20.

Adelantado, Martin, and Thibault Deman. 2002. A-CMSI: an airport-common modeling and simulation infrastructure using High-Level Architecture. *Simulation* 73 (2).

Boer, Csaba Attila, and Alexander Verbraeck. 2003. Distributed simulation with COTS simulation packages. In *Proceedings of the 2003 Winter Simulation Conference*, eds. S. Chick, P. J. Sanchez, D. Ferrin, D. J. Morrice, 829-837.

Defense Modeling and Simulation Office. *High Level Architecture* 2004. Available via <https://www.dmso.mil/public/transition/hla/>

Ceranowicz, A., R. Dehncke, and T. Cerri. 2003. Moving toward a distributed continuous experimentation environment. In *Proceedings of the 2003 Interservice/Industry Training, Simulation, and Education Conference*, 1-11.

Law, Averill M., and W. David Kelton. 2000. *Simulation modeling and analysis*. Boston: McGraw Hill.

Fowler, John W., and Oliver Rose. 2004. Grand challenges in modeling and simulation of complex manufacturing systems. *Simulation* 80 (9):469-476.

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

McLean, C. and Riddick, F. 2000. The IMS mission architecture for distributed manufacturing simulation. *In*

Proceedings of the Winter Simulation Conference, eds. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1539-1548.

Snyder, Carolyn. 2003. *Paper prototyping*. San Francisco, California: Morgan Kaufmann Publishers.

AUTHOR BIOGRAPHIES

JEFFREY W. DAWSON is currently a doctoral candidate in the University of Central Florida department of Industrial Engineering and Management Systems. He has a bachelor of science degree in industrial engineering from the University of Tennessee, and master of business administration and master of science in industrial engineering degrees from the University of Central Florida. He has worked in the nuclear and aerospace industries performing reliability, system safety, design, and cost analyses. His recent research areas include augmented reality, usability, ergonomics, and simulation. His email address is <jeffrey@geomix.com>.

PING CHEN is currently a Ph.D. student in the University of Central Florida, Institute for Simulation and Training. He has a bachelor of science degree in applied mechanics from the Changsha Institute of Technology, China, and a master of science degree in manufacturing technology from the Minnesota State University, Mankato. He has worked in the areas of computer software development, software performance testing and modeling, manufacturing automation, computer aided testing, and system engineering. His email address is <chenping@bellsouth.net>.

YANSHEN ZHU is currently a PhD student in the University of Central Florida department of Electrical & Computer Engineering. He has a bachelor of science degree in biochemistry from the Wuhan University, China and master of science in computer engineering from the University of Central Florida. His recent research areas include usability, simulation, and optimization. His email address is <yanshen_zhu@hotmail.com>.