

Improving Constructive Simulation Interfaces

Charles Cohen, Ph.D.

Steve Rowe

Joshua Band

Cybernet Systems Corporation

727 Airport Blvd

Ann Arbor, MI 48104

734=668-2567

srowe@cybernet.com, jband@cybernet.com

Erik C. Hofer

Hofer Consulting Services

734-644-7805

ehofer@acm.org

Michael W. Haas, PhD, PE

AFRL/HECP

2255 H Street, Building 33, Area B

Wright-Patterson AFB, OH 45433-7083

937-255-8768

Michael.Haas@WPAFB.AF.MIL

Keywords:

Usability, OneSAF, JSAF, HCI, Human Factors, Interface Design, Performance, Constructive Simulation

ABSTRACT: *Military simulation applications are used in command and staff training, wargaming, analysis and experimentation. Constructive simulations, such as One Semi-Automated Forces (OneSAF), provide a flexible environment for building and simulating exercises, however, when building these new software applications, human factors engineering is often overlooked in the design process.*

At Spring 2006 SIW, Cybernet Systems Corporation presented preliminary results from a human factors analysis of OneSAF Testbed Baseline (OTB) to identify key weaknesses in its interface, and identified several systematic problems. Using GOMS-KLM to model human behavior while interacting with OneSAF, we found several interaction areas ripe for improvement including: excessive mouse movements, precise mouse movements, excessive visual search, finding objects in multiple data views at the same time, and repetitive, successive menu selections. Other problems exist, but these general design flaws are important in that they highlight areas of major performance loss and cognitive loading.

With the recent distribution of OOS, we have undertaken an updated analysis of the new suite of constructive simulation applications. In order to evaluate the user performance within OOS and JSAF, we discovered the need for tools to quickly create and instrument sample interfaces for analysis. This paper discusses the tools and methods we developed in order to extend our previous work[1] to apply to general-purpose web applications. In discussing our test-harness framework, we present recommendations for user interaction and event data that must be captured, quantitative analysis tools that correlate interface arrangements with performance measures, and validation techniques of existing and evolving models of human performance. We also discuss our mixed-method analysis approach that uses theory-derived cognitive modeling techniques to conduct a formative analysis of interface alternatives and the validation of that formative analysis with empirical data.

1. Introduction

In our previous study, we analyzed the HCI of the OneSAF Objective Test Bed, and determined its interface was overly cluttered by its many operating modalities [1]. The graphical user interface for OneSAF is shown in Figure 1 [2]. The overall workflow of the application is modal; that is, each of the buttons on the button bar selects an operating modality (such as creating new units, selecting units, modifying the map, etc.). Modal interfaces tend to decrease the operational complexity of a piece of software by limiting the number of tasks specifically in a particular mode. This benefit is offset by an increase in the mental operation duration because one must remember what mode the application is in, and the need to change the mode of the application in order to perform a goal from a different mode. Since our original research was performed, the production version of the OneSAF software, now called OneSAF Objective System (OOS) has been released [2]. By comparing the HCI for OOS (Figure 2) and for JSAF (Figure 3), one can see that the issues identified by our previous work have not been addressed in any substantive way. Therefore, our work continues to be relevant.

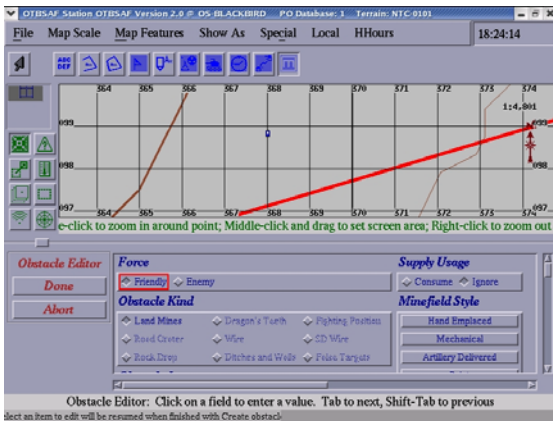


Figure 1: The OTB Graphical User Interface [2]

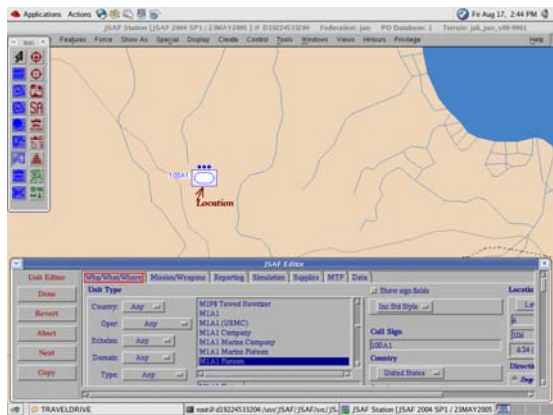


Figure 2 The JSAF Graphical User Interface [3]

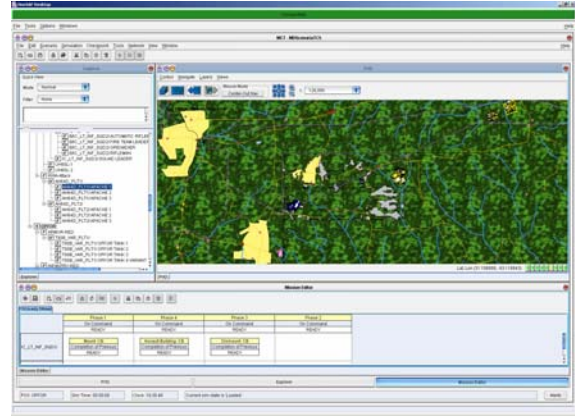


Figure 3: The OOS Graphical User Interface [2]

1.1 Analytical approaches

Traditional approaches to usability testing involve conducting a variety of tests on interface prototypes or final interface designs using either functional software or 'Wizard of Oz' human-guided techniques to simulate interface functionality. 'Wizard of Oz' experiments are those in "which subjects interact with a computer system that subjects believe to be autonomous, but which is actually being operated or partially operated by an unseen human being." [23] These tests include information architecture analysis techniques, such as the development of a global transition network (GTN): a node/edge diagram depicting each UI element and the elements it connects to, heuristic techniques, such as checklist methods that rate portions of the UI for potential problems, user modeling techniques, such as GOMS [4] or cognitive modeling, and user testing, which involves having human operators attempt to use the system to derive empirical results. These usability techniques are critical for gaining an understanding of how users will interact with the system, but traditional approaches to usability engineering includes some limitations that make it challenging to incorporate usability data into the development process.

Direct human observation of the subject by a usability engineer who records and interprets the subject's behavior is helpful to investigate depth of interactions (high to low level) in a single user session, however this method is time-consuming, expensive to run for large subject trials, and leaves little room for automation.

Automated monitoring of high-level task or application-specific performance metrics (scores, timing information, etc) allows for a larger number of subjects but generally doesn't address the depth needed to make a study truly insightful.

Automated monitoring of low-level user behavior (e.g. event captures such as mouse clicks) allows for a more in-depth study of the HCI, however, is difficult to connect this low level data with higher-level tasks and session semantics.

While these approaches present many different views of the same usability problems, it is difficult to merge this data into a coherent picture of a system's overall usability. For instance, the GTN approach provides insights into overall site complexity, but user-modeling analyses at rigorously defined tasks. Combining the results of all of these methods into a meaningful assessment requires significant effort on the part of a trained usability expert. Without this synthesis step, many usability results provide seeming disparate insights into the quality of a system. The need for a usability professional to synthesize these test results means that the usability engineer is effectively a gatekeeper - other development team members are not able to easily see how small design alterations might affect the ways users interact with the system.

Another challenge of the usability testing process is that many of the analyses are run as summative tests. Only a final set of system prototypes or candidate interfaces are typically evaluated, limiting the usability inputs to the initial design and the test results. Often, usability feedback comes too late in the development process to have a real impact as design decisions. Some firms have undergone business process design that places usability engineering in the center of the design process (e.g. Brink, Gergle and Wood, 2001 [5]), but usability professionals still function much like outside consultants on many projects.

We address these usability-engineering challenges by defining a hybrid analytical approach. Our approach builds a network-based model of usability by incorporating usability that is automatically generated through a standards-based data-logging framework. This model, in turn, is used as a common analytical framework for combining test results.

1.2 Hybridized analytical approach

As an alternative to the expert-maintained usability assessment described above, we propose a hybrid analytical assessment that builds and maintains a 'usability network' which visualizes the results of different usability tests. Our goal in this approach is to provide better understanding of how design changes impact the overall system usability. This data is then made accessible to developers and automated usability tests, such as the data-logging framework described below.

Our hybrid approach builds on network-centered analysis methods that are widely used to understand information flow in organizations, social networks, and the spread of communicative diseases. In usability networks, each UI element represents a node, while operator and interface actions represent edges, the human computer interaction is then modeled of the object/action network.. On the surface, this approach greatly resembles a global transition network (GTN), which maps a system's interface in a similar way. In addition to the structure of a system, we also incorporate heuristic and user modeling data in laying out the network. Each edge of the network (or the action connecting two UI elements) is weighted based on the results of usability testing, making the distances between nodes vary based on the ease of use. For instance, two nodes appear to be physically close together if a GOMS analysis suggests that it will take users a short time to navigate from one to the other. The edge weighting can be applied based on a variety of usability test results, depending on what analyses have been conducted. In our experience, this weighted network approach provides a very clear picture of system complexity. The effect of changing a series of UI elements can quickly be determined updating the information for those nodes and edges.

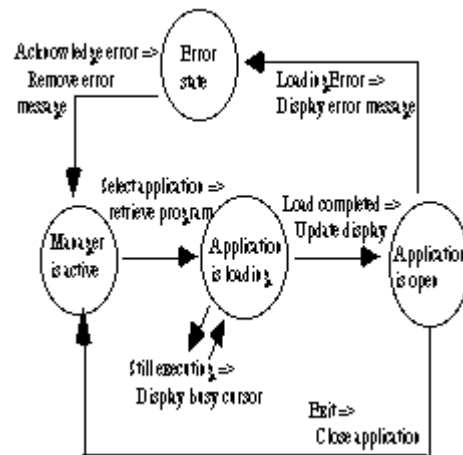


Figure 4 Sample GTN For Opening An Application In The Window Manager [22]

2. Data Logging Framework

Automated capture of user workflows through instrumentation of the UI framework is desirable because it enables:

- Examination of individual sessions for performance case studies
- Correlation of UI events with the emergence of new information

- Gathering of large numbers of user workflows to build statistical models of situation awareness through techniques such as neural networks
- Conduction of controlled studies to understand how UI decisions impact the ability to respond to new information or situational changes
- Quick prototyping and evaluation of new user interfaces

2.1 Usability Data Standards

Instrumenting the UI framework provides a compelling new capability, allowing the capture of rich data while possessing the same ease of administration as server side logging.

We reviewed several options for logging usability data. One option, the W3C Common Log Format (W3LOG) for server logs describes how to record requests for web pages from external clients [6]. The depth of this log is primarily useful for tracking the path of page jumps, not interaction on a single, dynamic page. The server logs for these models track server activity, not subject.

Other event models include browser-based Document Object Models (DOM) stored within Netscape [7], and Microsoft Internet Explorer [8]. These models represent low level, application-independent user activity, however, they don't represent user behavior at a higher level (task performance).

More generic usability standards, such as ISO/IEC 25062:2006 [9], the Common Industry Format, are available but focus on providing a framework for reporting usability information. Another standard, ISO 13407, Human Centered Design Process for Interactive Systems [10] focuses on the guidelines for setting up a study, but not recording data.

We determined that it was desirable to capture this data in a manner consistent with the NIST Framework for Logging Usability Data (FLUD) standard [11].

While most of our interest focuses on the "log" portion of the data model, other components of the model that describe tasks and users are also desirable. Selecting FLUD provides us with immediate tools for visualizing log data that can augment capabilities we develop internally.

The FLUD design paper, "Design of a File Format for Logging Website Interaction," by Cugini, and Laskowski, describes FLUD as a, "format for the representation of user interaction with a website. A widely accepted format enables the development of a set of software tools to process the data, the sharing of data sets for longer term

analysis and research and provides a common language for expressing user interaction with a website." We believe that the FLUD format is generic enough to handle usability testing and logging for standard GUI applications. XML structuring of this format is trivial and provides the researcher output that seamlessly fits into a variety of visualization and statistical software packages.

2.2 Framework for Logging Usability Data (FLUD)

NIST's Visualization and Usability Group has been developing measurement and testing standards related to usability engineering since 1997. FLUD was developed in January 2001 to address the need for a standard format to represent user interactions with Web-based applications. The following is an overview of FLUD file concepts and terminology [11].

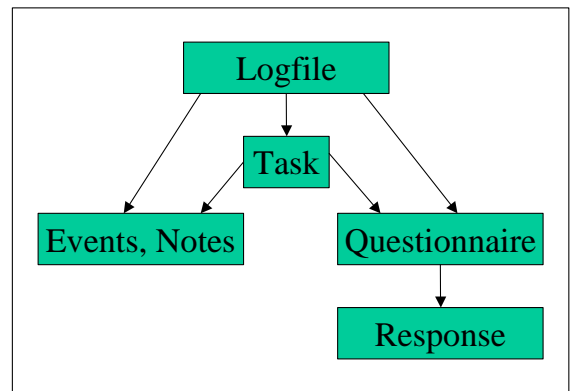


Figure 5 Basic, nested FLUD file structure

Session

A FLUD file records a single user session that is further defined as the "interaction of a single subject with a single fully configured hardware system during a continuous time interval." A subject may visit multiple interfaces and pages while performing several tasks within a session.

Logfile

A FLUD file contains a header which defines the session start time, software, hardware, user and session identification information, and a list of events that will be detected (e.g. key press, slider, enter widget, etc.)

Task

The subjects directed and performed action. Undirected action performance can also be logged with a single "dummy" task.

Events

Events are low-level user actions with interface widgets and elements. Event logging includes timing information,

user actions, widget effects, interface element ID, cursor and event location, window and browser state.

User action

A user action is an action performed directly by the user and associated with a particular input device, typically a mouse or a keyboard.

Widget

A widget is an interface element to which the user action was targeted. Screen objects, such as buttons, textboxes, menus, checkboxes, and sliders, are typical widgets.

System effect

A system effect is used to describe how the state of the system as seen by the user changes (either as a result of the user action or autonomously).

Window state

The window state includes typical window operations, such as open, close, move, re-size, and iconify.

Questionnaire

A questionnaire is the session section wherein the subject responds to direct questions posed by the tester. Tasks are used to simulate intended usage whereas questionnaires are for feedback and reporting purposes. In our current version of our logging implementation, we ignore the questionnaire section as we are more interested in automated information gathering, however, we would easily be able to incorporate it in future software revisions.

Notes

Notes from the subject or tester or others can be reported wherever events are legal. We have not implemented a notation software capability but intend on doing so in future software revisions.

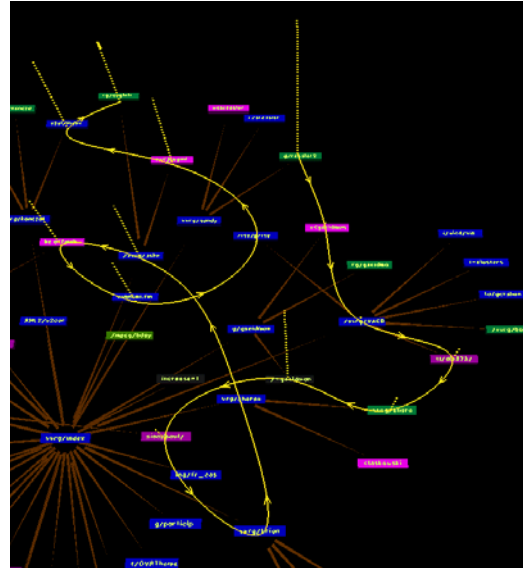


Figure 6 Example output of the NIST Visualization Tool, FLUDViz [20]

2.2 Instrumented GUI Building

One of our research goals was to develop a web-enabled GUI building tool. We evaluated several client-side web development languages and settled on JavaScript. We began our instrumentation effort using the JavaScript toolkit, DOJO [12]. After a good deal of work with the this toolkit, we determined that there were several , open-source toolkits that could more readily accomplish our GUI building task. In deciding to use an open source toolkit, we avoided the time and cost associated with building our own from scratch.

We evaluated several higher level web-based GUI building tools such as Backbone[13], Bindings[14], JackBe[15], Tibco General Interface[16], and OpenLaszlo[17]. We found Tibco to be the best solution as it is available under a BSD open-source license with a large user community. Tibco GI is a comprehensive, stable, feature rich, and extensible platform making it easy to focus on our workflow and user interaction capture feature.

“Tibco offers a full-featured IDE for building browser-based applications that are nearly indistinguishable from apps produced by native code. The client communicates with the server via Web services. This toolkit will be most useful for larger projects aimed at building desktop-like applications that interact with a server farm knitted together with Web services”[18].

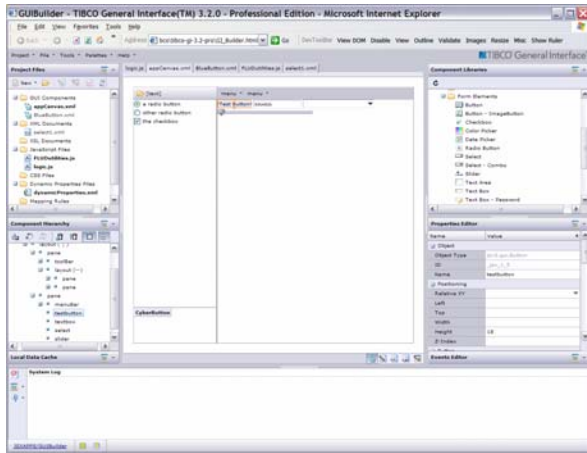


Figure 7 Tibco General Interface

We extended the Tibco GI widgets (buttons, check boxes, images, etc.) and GUI building toolkit to capture all user interactions with the web-based interface. Our extensions were focused on the Situational Awareness domain. The data capture is XML-based and adheres to the FLUD specification. Logging occurs automatically every time an event occurs.

	Key press	Key release	Pointer press	Pointer release	Pointer move	Enter widget	Leave widget	Ad hoc
Example Button			'yes'	'yes?'		'no'	'no'	click 'yes'
Button			Yes	Yes		Yes	Yes	
Radio			Yes	Yes		Yes	Yes	
Checkbox			Yes	Yes		Yes	Yes	
Select			Yes	Yes		Yes	Yes	
Textbox	Yes	Yes	Yes	Yes		Yes	Yes	
Slider			Yes	Yes	Yes	Yes	Yes	
Image			Yes	Yes	Yes	Yes	Yes	

Figure 8 Widget table for GUI Builder implementation

Our goal was to provide the GUI developer a suite of tools to ease and expedite the construction of new web-based situational awareness applications. These web apps deploy within our web portal framework, capture user performance data, and analyze usability and cognitive load.

We encapsulated the GUI building tools within what we call our “Enterprise Portal User Interface Construction Tool” featuring:

- Model-View-Controller Architecture Implementation
- Multi-tiered Functional Layers
 - Client logic layer
 - Presentation layer
 - Data layer
 - Communication layer
- Delivery of Rich Internet Applications (RIAs) with the look, feel, and performance of desktop installed software

- Runs in a standard web browser
- Simplifies deployment
- Support for industry standards and best practices
- Scalable architecture
- Suite of development tools and API's

The Enterprise Portal User Interface Construction Tool provides an interface-based event-recording component that, when enabled, will collect session-based user interaction, timing, and event data while our web application software is in use.

Automated capture of user workflows through instrumentation of the UI framework is desirable because it enables:

- Examining individual sessions for performance case studies
- Correlating UI events with the emergence of new information
- Gathering large numbers of user workflows to build statistical models of situational awareness through techniques such as neural networks
- Conducting controlled studies to understand how UI decisions impact the ability to respond to new information or situational changes
- Quickly prototyping and evaluating new interfaces

While other toolkits exist for instrumenting web applications, they either require extensive software modifications on the client computer or capture lower quality data.

Instrumenting the UI framework provides a compelling new capability, allowing the capture of rich data while providing the same ease of administration as server side logging.

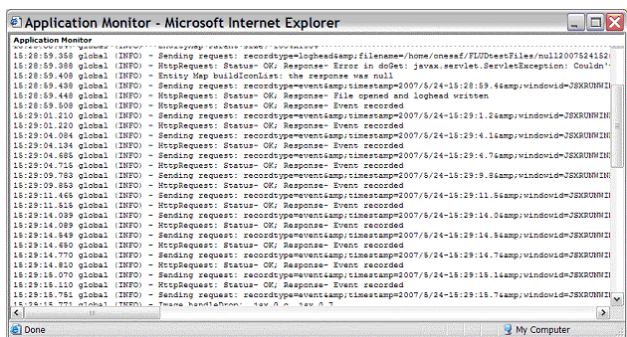


Figure 9 Usability data is logged in NIST FLUD format, a human readable text file.

Our data capture and logging conforms with the NIST Framework for Logging Usability Data (FLUD) standard.

2.3 FLUD Parsing and Visualization

While FLUD has commonly been used to understand user paths through websites, we have adapted NIST's visualization tools to provide a much more detailed view of interfaces, documenting the relationships between all the UI elements in an application. We have used VisVIP[20] and FLUDviz[21] to visualize the user's movements in and out of various interface components (pop ups, menus, floating tool bars, separate pages, etc.). This technique allows us to create a network-based visualization of an interface and overlay user data on top of the network to understand how particular components of an interface are used, both in general terms and in accomplishing specific usability tasks. Additionally, we can draw on summary statistics from usability tests to understand how particular task-sequences impact overall task performance.

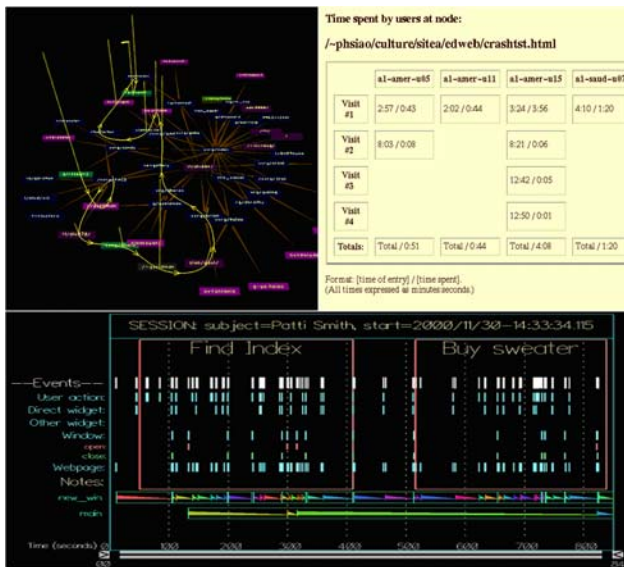


Figure 10 After action, recorded usability data can be processed for visualization and comprehensive analysis.[19][20]

Cybernet Systems Corporation has packaged a group of tools to help analyze the performance data that has been logged in FLUD format. These tools allow visualization and analysis of individual usability sessions, navigational paths of websites, event triggering and timing graphing, and comparison of user interface workflows.

3. Methods and Assumptions

We apply the techniques and technologies described above to develop a workflow for designing constructive simulation interfaces. This workflow unobtrusively collects usability data throughout the design and deployment process.

The workflow begins with identifying a simulation task and conducting a heuristic analysis of existing interfaces that are used to accomplish this task. Based on the results of this initial study, an interface is created in our web applications framework. Once the interface is implemented, a network-based visualization of the interface is generated with equal weighting along all edges of the graph, showing the relationships between all elements.

A group of users then uses the interface to conduct a number of simulation tasks. In addition to gathering information about overall timing and performance, we are also able to construct a use-network that illustrates how users moved between various interface widgets. We then use this usage data to weight the user interface network visualization, highlighting which interface elements are used most often or are used in sequence. This new weighted graph is used to modify the interface to make important interface components more salient and to reorganize elements that are commonly used in sequence.

We can continue to iterate through this process until a satisfactory and high-performing interface is produced.

4. Conclusion and further work

We have presented a development tool for web-based applications that provides built-in usability data logging in a standard format. This tool will enable us to rapidly construct and measure the performance of alternative interfaces for applications. Our approach for creating usability networks by overlaying interface network diagrams with user performance data is particularly useful in identifying opportunities for usability improvement by illustrating dramatic differences in how interfaces are designed and in how those interfaces are actually used.

A future goal for this project is to integrate existing cognitive models into the interface builder to serve as an automated usability test harness. For instance, by constantly tracking and analyzing interface complexity and user focus, we can design widgets that redirect the operator's attention when critical events happen or based on what they are doing. For instance, we can automatically enlarge crowded tool palettes in the operator's focus area or modulate the intensity of alerts based on interface complexity.

In the next phase of our project, we will use the software infrastructure presented here to perform usability experiments targeting JSAF interfaces relevant to tactical command and control operations. We will use the results of these experiments to recommend changes to the tactical

interfaces that are being developed within the Air Force, and also to inform our corporate HCI designs.

5. References

- [1] Rowe, Steve, Joshua Band, Charles J. Cohen, "Improving Human Interfaces in Military Simulation Applications." Simulation Interoperability Standards Organization's Simulation Interoperability Workshop, Spring 2006, Huntsville, AL, September 10-15, 2006.
- [2] OneSAF Public Site. PEOSTRI US Army. <<http://www.onesaf.net/>>
- [3] Joint Semi-Automated Forces (JSAF). US Joint Forces Command. <http://www.jfcom.mil/about/fact_jsaf.html>
- [4] John, B. E., & Kieras, D. E. (1996). "Using GOMS for user interface design and evaluation: Which technique?," ACM Transactions on Computer-Human Interaction, 3, 287-319.
- [5] Tom Brink, Darren Gergle, Scott Wood. Usability for the Web: Designing Web Sites That Work. Paperback - 320 pages, published 15 October, 2001.
- [6] Logging Control in W3C httpd. World Wide Web Consortium. <http://www.w3.org/Daemon/User/Config/Logging.html#common_logfile_format>
- [7] The Document Object Model in Mozilla. Mozilla.org <<http://www.mozilla.org/docs/dom/>>
- [8] About the DHTML Object Model. Microsoft Developer Network. <<http://msdn2.microsoft.com/en-us/library/ms533022.aspx>>
- [9] Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Common Industry Format (CIF) for usability test reports. ISO/IEC 25062:2006. <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43046>
- [10] Human-centered design processes for interactive systems. ISO 13407:1999. <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=21197>
- [11] Cugini, John. The FLUD format: Logging Usability Data from Web-based Applications. Information Technology Laboratory, National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899. 2002. <<http://www.itl.nist.gov/iaui/vvrg/cugini/webmet/flud/specification.html>>
- [12] dojo. The javascript toolkit. <<http://dojotoolkit.org/>>
- [13] backbase. <<http://www.backbase.com/>>
- [14] Bindows. <<http://www.bindows.net/>>
- [15] JackBe. <<http://www.jackbe.com/>>
- [16] Tibco. <<http://www.tibco.com/>>
- [17] OpenLaszlo <<http://www.openlaszlo.org/>>
- [18] Wayner, Peter. "Top AJAX tools deliver rich GUI goodness." Infoworld. November 27, 2006.
- [19] VISVIP. <<http://zing.ncsl.nist.gov/~cugini/webmet/visvip/vv-home.html>>
- [20] FLUDViz. <<http://zing.ncsl.nist.gov/WebTools/FLUDViz/overview.html>>
- [21] The R Project for Statistical Computing. <<http://www.r-project.org/>>
- [22] Johnson, C. "Proceedings of the Design, Specification and Verification of Interactive Systems '96. Berlin, Germany. 1996.
- [23] Kelley, J.F., "CAL - A Natural Language program developed with the OZ Paradigm: Implications for Supercomputing Systems". First International Conference on Supercomputing Systems (St. Petersburg, Florida, 16-20 December 1985), New York: ACM, pp. 238-248

Author Biographies

CHARLES COHEN has been working in the fields of image processing, robotics, human-computer interaction, and artificial intelligence for over a decade. He is currently the Vice President of Research and Development for Cybernet Systems Corporation. He has been the project manager for many projects for the United States Armed Forces (Air Force, Navy, and Army), National Aeronautics and Space Administration, and other government agencies. Dr. Cohen's current research interests are in gesture recognition, image processing, estimation theory, system integration, human computer interaction, visual communications, machine vision, and perceptually coupled systems.

STEVE ROWE is a senior software engineer at Cybernet Systems Corporation. Besides being well versed in simulation environment, intelligent agents, and scenario development work, he has taught in the classroom and also worked in the development of automated teaching materials at KnowledgeNet Corporation.

JOSHUA BAND is a research engineer at Cybernet Systems Corporation. He has significant experience developing massive multi-user networking and simulation environment technologies. He acted as lead interface engineer and project manager on the prototype testing application mentioned in this paper.

ERIK HOFER is an independent consultant. He works on a variety of projects in human-computer interaction, visualization and data analysis.

MICHAEL W. HAAS, PHD, PE is a Principal Electronics Engineer at the Collaborative Interfaces program within the Warfighter Interface Division of the Human Effectiveness Directorate at the Air Force Research Laboratory. He is pivotal in the development of

the Synthesized and Human Aerospace Forces in an immersive Research Environment (SAFIRE) which is a simulation capability linking many of the Warfighter Interface Division's human-in-the-loop simulations together as well as with external AFRL and AFMC assets