1993 N94-24430

NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

MARSHALL SPACE FLIGHT CENTER
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

INTEGRATION AND EVALUATION OF A SIMULATOR DESIGNED TO BE USED WITHIN A DYNAMIC PROTOTYPING ENVIRONMENT

Prepared by:	Loretta A. Moore
Academic Rank:	Assistant Professor
Institution and Department:	Auburn University Department of Computer Science and Engineering
MSFC Colleague:	Joseph P. Hale
NASA/MSFC:	
Laboratory:	Mission Operations
Division:	Operations Engineering
Branch:	Crew Systems Engineering

Introduction

The Human Computer Interface (HCI) prototyping environment is designed to allow developers to rapidly prototype systems so that the interface and functionality of a system can be evaluated and iteratively refined early in the development process. This keeps development costs down by modifying the interface during the requirements definition phase, thus minimizing changes that need to be made during and after flight code development. Problems occur within a system when the user interface is not adequately developed and when designers and developers have an incomplete understanding of the system requirements.

A process has been developed for prototyping on-board payload displays for *Space Station Freedom* (Moore, 1992). This prototyping process consists of five phases: identification of known requirements, analysis of the requirements, development of a formal design representation and specification, development of the prototype, and evaluation of the prototype. The actual development of the prototype involves prototyping the displays, developing a low fidelity simulator, building of an interface (or communication) between the displays and the simulator, integration of these components, and testing to ensure that the interface does what the developer expects.

This research integrates and evaluates a software tool which has been developed to serve as a simulator within the prototyping environment. The tool is being evaluated to determine whether or not it meets the basic requirements which are needed for a low fidelity simulator within this environment. In order to evaluate the architecture and its components, a human computer interface for and a simulator of an automobile have been developed as a prototype. The individual components (i.e., the interface and simulator) have been developed (Moore, 1993), and the current research was designed to integrate and test the complete working system within the prototyping environment. The following sections will describe the architecture and components of the rapid prototyping environment, the development of a system to assess the environment, and the integration and evaluation of PERCNET.

Architecture of the Environment

The architecture for building prototypes of systems consist of four major components: a interface development tool, a test and evaluation simulator development tool, a dynamic, interactive interface which links the interface and the simulator, and an embedded evaluation capability. The interface development tool allows the designer to dynamically develop graphical displays. The test and evaluation simulator development tool will allow the functionality of the system to be implemented and will act as driver for the displays. The dynamic, interactive interface will handle communication between the HCI prototyping tool and the simulation environment. This component consists of a server which sends and receives messages between the other components. The embedded evaluation capability will collect data while the user is interacting with the system and will evaluate the adequacy of an HCI based on a user's performance.

Human Computer Interface Development Tool. Sammi by Kinesix has been chosen as the Human Computer Interface (HCI) development tool. Sammi is a graphical user environment which allows user interfaces to be built which can manage networked information graphically. Sammi combines the functions of a graphical user interface with full network communication support. Within Sammi the user interface and the networked data access can be defined independently of the actual data source or application. This will allow an interface developed under Sammi and communicating with the low fidelity simulator, to later be connected to a high fidelity simulator such as those in the Payload Crew Training Complex (PCTC), and later to the actual on-board flight software. Sammi has a distributed architecture which means that the user interface and the application code are separate, that is, the user interface is no longer embedded within the application code. With this separation users can easily create and modify the human computer interface without affecting the datasource, and vice versa. This will allow concurrent development of the application and the interface. Sammi developed applications can use remote procedure calls to access information from a variety of nodes and servers on an Ethernet network.

Simulator Development Tool. A simulator is a computer program that models a system or process in order to enable people to study it. The simulator development tool should provide the capability to develop a low fidelity simulation of a system or process. The development of a simulator has two important functions. First, the simulator helps the developer to identify and define basic system requirements. Second, potential users can evaluate both the look (in terms of the screen layout, color, objects, etc.) and feel (in terms of operations and actions which need to be performed) of a system. During the requirements definition phase, a high-fidelity simulation of the system will not yet have been developed, so it is important to build a low fidelity simulator, so that the iterative cycle of refining the human computer interface based upon a user's interactions can proceed.

For a piece of software to function as a simulator within this environment there are several requirements which must be met in addition to it just being a simulation tool. These requirements include the ability of the process to communicate with UNIX processes using the TCP/IP protocol; real-time simulation execution, the execution engine must be tied to a real time clock to assure that simulation timing and data collection are accurate; an option for a variable communications mode during execution (i.e., with and without external communication); real time communication with Sammi on a separate platform, via Ethernet; the ability to receive data from Sammi to dynamically control scenario events, modify blackboard variables, trigger scenario events, and track operator actions for post-hoc analysis; the ability to specify and send commands and data to Sammi; and the ability to receive data and commands from multiple Sammi applications/stations. The multiple Sammi stations may include one or more display prototype stations and a monitoring station. A Simulator Director should be able to send commands to this software from a monitoring station (e.g., start simulation, trigger scenario event). Sammi subroutines must be provided that have been developed for the Simulator-Sammi communication and the software must be tested and validated with documentation provided. PERCNET is designed to be used as a knowledge-based graphical simulation environment for modeling and analyzing human-machine tasks. Within PERCNET task models are developed using modified petri nets, a combination of petri nets, frames, and rules. This research evaluated PERCNET to determine whether or not it met the basic requirements which were listed above.

Dynamic, Interactive Interface. This interface will handle communication between the HCI prototyping tool and the simulation environment during execution. This interface is a server which has been developed using the Sammi Application Programmer's Interface (API). It will be a peer-to-peer or asynchronous server which means that messages and commands can be sent and received both ways between Sammi and the application. Once the embedded evaluation tool has been developed, the server can also service requests from this process providing information as to which functions the user has used, errors which have been made, and so forth.

Embedded Evaluation Capability. The embedded evaluation capability will include a capture/playback component and an analysis component. The Capture feature will capture a user's session and save this information to a log. This log can later be "played" back or analyzed. The analysis component will analyze the user's session and provide guidelines for the redesign of the system. Some of the measures will include: frequency of use of specified features, task completion time, error counts, requests for help, amount of work/errors per unit time, and response time to different activities and events.

Development of a prototype within the Architecture

In order to assess the individual components of the architecture a system was chosen and developed (Moore, 1993). The system chosen for pathfinding and initial empirical evaluation of the project was an automobile. An automobile has sufficient complexity and subsystems' interdependencies to provide a moderate level of operational workload. Further, potential subjects in the empirical studies would have a working understanding of an automobile's functionality, thus minimizing pre-experiment training requirements. There were four basic tasks which were completed: (1) requirements were developed for the automobile simulator, (2) the automobile simulator was developed using PERCNET, (3) a human computer interface for operating the automobile simulator was developed using Sammi, and (4) evaluation criteria for the operation of the automobile simulator were developed (Moore, 1993).

Integration and Evaluation of PERCNET

The initial design provided by Perceptronics presented a potential problem. The dynamic, interactive interface component was designed to be embedded within the PERCNET process. This would allow Sammi and PERCNET to communicate, however, there would be no way for other processes to communicate with Sammi and PERCNET. This was a real problem within our environment because the embedded evaluation capability would be a separate process that needed to send messages and receive information from this process during the execution. Once this problem was identified and the importance of this function was understood, the developers from Perceptronics changed the architecture.

PERCNET provides the basic functionality of a tool which can act as a simulator with the changes made in its architecture. However, there are some remaining issues which need to be addressed and major problems with the current system which need to be fixed. One problem concerns the system running out of swap space and exiting because it can no longer allocate memory. A minimal configuration of this tool needs to be presented and the the system should be able to run with this configuration without the system exiting. A second problem involves the tendency of the system to core dump, sometimes in response to specific features (such as trying to use an option from the menu which has not been implemented or is not currently working) and sometimes randomly. A third problem, is that the screen and the keyboard lock up and the system has to be rebooted. It is not clear whether the problem can be attributed to PERCNET or the second screen (a Plasma display) which is attached to the SunSPARC station on which we are running PERCNET. This item needs further investigation. There have been other problems with several features of the system and most of these have been fixed by the developers at Perceptronics. However, there are several functions of the system which have not yet been evaluated yet, such as, communication across the network, having multiple Sammi displays communicate with a single PERCNET model, and being able to start and stop the simulation from the second Sammi window.

Conclusions and Future Work

PERCNET has been integrated within the human computer interface prototyping environment; however, it is recommended that further testing and evaluation be conducted using the automobile interface and simulator to resolve the issues previously discussed. Most requirements have been met but there needs to be a more thorough evaluation of the simulator tool and the architecture of the environment.

Following the automobile prototype development, a second system, based on a Spacelab/Space Station payload should be developed for further evaluation of the environment. This should involve development of the payload simulator requirements from existing experiment simulator requirement documents, development of the payload simulator using PERCNET, development of an interface for the payload using Sammi, and integration and testing of the payload simulator and interface.

References

Moore, L. A. (1993). Assessment of a Human Computer Interface Prototyping Environment (Contract No. NAS8-39131). MSFC, AL: NASA, George C. Marshall Space Flight Center.

Moore, L. A. (1992). A Process for Prototyping Onboard Payload Displays for Space Station Freedom. In M. Freeman, R. Chappell, F. Six, & G. Karr (Eds.), Research Reports - 1992 NASA/ASEE Summer Faculty Fellowship Program (Report No. NASA-CR-184505, pp. XXXVI.1 - XXXVI.4). MSFC, AL: NASA, George C. Marshall Space Flight Center.

Perceptronics User's Manual. (1992). Woodland Hills, California: Perceptronics, Inc.

Sammi User's Guide. (1992). Houston, Texas: Kinesix Corporation.