ABSTRACT
The automatic evaluation of graphical user interfaces can help reduce development costs in the creation of new designs or modification of existing designs. Several standards for the X Window System have been proposed or implemented that could greatly reduce the time spent evaluating GUIs.

We implemented a User Interface Testbed (UseIT) based on the proposed Remote Access Protocol (RAP) [1] standard. UseIT was created to automatically record an end user's interaction with a Motif GUI application without modification or re-linking of existing code. The recorded interaction could then be replayed or displayed visually for interpretation by a human factors specialist. The end goal was to recreate the GUI and automatically recommend design changes based upon the interactions.

Keywords
Interface design, evaluation, agents, user interactions

INTRODUCTION
The motivation for this work comes from the need to automatically evaluate and record a user's interactions with a graphical user interface (GUI). Many publications and reports stress the need for human factors evaluation in the design process. While this project does not seek to replace a human factors specialist, we wish to lower the cost, in terms of dollars and time, of the evaluation process. Some initial evaluation work can be recorded without the presence of a specialist. This work can be completed by the software or interface designer and then reviewed by the specialist.

The usability evaluation process often results in the collection of large quantities of data. This data can typically be recorded, statistically analyzed, and the results presented visually. Interface design recommendations can then be based on the qualitative results of the evaluation. If focus is on quantitatively assessing an interface design, then it is logical to consider implementing a scheme by which the usability of an interface design could be assessed automatically with recommendation for candidate modification presented to the designer. A GUI should be easy to use, meet international and corporate standards, meet windowing system style guides, and be visually appealing. How can a computer system automatically evaluate a GUI to all of the above standards and guidelines? We propose a simple, but hopefully cost saving, evaluation process.

RELATED RESEARCH
The goal of this project was to expedite getting UseIT into the hands of interface developers. We did not seek to copy any research, but to incorporate public domain software, where possible, and design the missing pieces when necessary.

Numerous applications have been developed to record and playback a user's interaction with a GUI (e.g., for X Windows, XRECORD [2] and XTEST [3]). These applications are more appropriate for regression testing than for performing a GUI usability test.

Layout Appropriateness (LA) [4] is a metric developed to
DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
place a cost on a GUI design based on detailed task analysis. The appropriateness of a given layout is computed by weighting the cost of each sequence of actions by how frequently the sequence is performed. An oversimplified explanation of this metric would be to compare it to an architect's design of a kitchen. An architect seeks to place the refrigerator, stove/oven, and sink in the smallest triangle as possible. With this as an example, one could think of LA as a metric to decrease the amount of time traveled between the most frequently accessed GUI widgets, while still allowing less frequently accessed widgets to be part of the design.

A metric-based tool called AIDE (semi-Automated Interface Design and Evaluator) [5] incorporates many metrics into a tool that can help the interface design process by automatically creating initial GUI designs. This tool uses efficiency, alignment, and balance metrics to create initial designs. With this tool, a designer can base their GUI design on the end user's tasks while allowing for overall appearance of the GUI.

The Remote Access Protocol (RAP) provides a protocol which communicates information about GUIs between X applications. RAP can be used to get information about a GUI, monitor its status, and even cause changes in a GUI's status.

The RAP authors use the terminology of a client and an agent in their research and we will use the same terminology here. The term client is a X Windows application that is compiled and linked with X11R6 include files and libraries. This application can use X widgets, Athena widgets, or be based on the popular Motif windowing system. The agent is an external application that communicates with a client using the aforementioned protocols and RAP. A simple agent would request all resources and resource values from the client.

The basic technique RAP uses to capture interface communication is to monitor the exchange of X protocol packets between the X server and clients running on a user's machine. By tapping into the communication between a client and the X server, one can monitor all window creations, map and unmaps, text and graphics rendering and size changes. One major prerequisite of the RAP project was one of transparency. By being transparent, no modification of an application or the X server is necessary. One could immediately begin monitoring a GUI without having to recompile the software.

RAP uses several standards in the X Windows System release 6.1 (X11R6.1) and proposes that RAP be a standard in release 7. X11R6.1 provides the Xt library [6], the Inter-Client Exchange Protocol (ICE) [7], and ICE Rendezvous[7] used by RAP.

The Xt library in X11R6 introduced an implementation-dependent widget called the HooksObject. The Hooks object is associated with the application's display connection and maintains a set of callback lists. These lists hold routines which are called whenever certain changes occur to an application. Additional routines were also added to Xt to allow retrieval of these changes.

ICE provides common functionality for many protocols (opening connections, validating requests, etc.). ICE handles generic protocol communication while RAP provides specific protocol messages and message handlers. ICE Rendezvous was designed to establish an ICE connection between an external agent and a client in a manner which will not require awareness of a specific protocol (i.e., RAP).

RAP uses the protocols and standards mentioned above to create their own protocol to communicate with a client. RAP's protocol can make specific requests about the GUI. For example, RAP could ask for the values of all the resources of the widget tree or ask to be notified when a button press happens.

**USER INTERFACE TESTBED**

The goal of the User Interface Testbed (UseIT) is to create an application that could help a user interface designer or human factors specialist handle the data collection for a usability evaluation. We wanted to use software that could access a GUI without modifying source code. There are many applications on the market today that can record and playback the interactions of a GUI. Many of these record and playback applications work by modifying source code. Most of these applications were implemented to do regression testing and do not help the interface designer. Screen size and location can cause problems with interaction playback. These problems are associated with the individual recording of a button press. Many applications record the x and y location of the button press. This approach is not appropriate for an interface that can change location on the screen. Many recording applications require modification to the source code and relinking to a specific library in order to function.

The UseIT team implemented a recording and playback mechanism to visually evaluate basic task analysis techniques using existing standards based on the X Window System release 6.1 (X11R6.1). With UseIT we are able to communicate with clients linked with X11R6.1 libraries via an external agent without modification to the applications code.

**Recording**

To communicate with a client, the UseIT agent must first establish communication using the ICE protocol. Once this connection is established, the agent can gain access to the actual window IDs of the GUI widgets. The window IDs will be used for playback instead of a x and y location. This will solve the problem of widget location. All events caused by a user's interaction can now be recorded until either the client or agent ends communication.
Playback
Playback is implemented through a procedure similar to recording. The client and agent applications are started and communication is established via RAP. The logfile from the recording session is read into the agent. The actual events are then passed to the X server. An interface designer can watch the interaction from an end user as if it was videotaped.

Link Analysis
Link analysis is an important part of interface design. With link analysis, an interface designer can learn what areas of a GUI are being used the most, if a particular feature is not used, and the traversal paths taken. Many human factors specialists videotape an end user using a GUI during testing. Extensive notes are taken to supplement the videotape. Reviewing this recorded data from the users tested can be difficult and time consuming. A way to visualize this data is needed.

The Layout Appropriateness and AIDE research of Andrew Sears was the basis of our application. Although Sears incorporated many algorithms into his research, we focused on keeping the design simple. With our process the interface designer or human factors specialist can make the final decision on interface design. While automatic design of an interface may save time, there might be room for debate on whether or not we should implement such a capability. If someone could automate a professional psychologist, would you want to see a computer or a real person?

One of the many goals of this project was to save some time and effort for the human factors specialist. Our approach was to present visually data the specialist typically uses when evaluating an interface. We were able to combine the interaction data from the testers and display the result on a duplicate of the GUI.

UseIt duplicated the GUI by taking a bitmap snapshot and used RAP to record the user’s interactions. We then redisplayed the GUI (using the bitmap image) and drew lines between the different interactions with the GUI widgets. Sears displayed his efficiency algorithm [5], in this same method. We wanted to duplicate the GUI using Xtent [8], but we came across some problems with RAP that made the translation nearly impossible. Xtent is an executable specification system that creates GUIs from files similar to X resource files [9].

If duplication of the GUI were possible, we could have a GUI builder running and the interface designer could make changes. This could greatly help in the interface design process.

RAP PROBLEMS
We encountered several problems with obtaining the resource values of a GUI while using RAP. Enough resources were available to duplicate the interface, but the look and feel of the application was not the same. The return values for font and color names in Motif were not valid to duplicate the GUI. RAP returns a pointer to a font structure inside the client application when queried about a font resource for a widget. The actual X Font name is needed. RAP returns an integer value when accessing the color value. These integers are different across platform. The actual color name is needed and even this could cause some problems across the different platforms. Without correct font names and color names the GUI does not look correct.

CONCLUSIONS
We were able to create an application that could record the interactions of a GUI and visually display the interactions. If RAP becomes a standard part of the X Windowing System, no code modifications should be necessary. We have also created a similar application running on Windows 95.

We did not implement any algorithms to automatically recommend interface changes. This might easily be done in the future. There certainly is room for debate on how automatic user interface evaluation should be. Standards seem to change daily and there are too many conflicting corporate standards to efficiently create an automatic evaluation tool. The research work by Sears [4,5] and his predecessors is a great step forward and could easily be included in our work or other research projects.

More research needs to done to determine the effectiveness of automatic evaluation, but we do agree that some form of visualization of the collected data is needed. Afterall isn’t a picture worth a thousand words?

REFERENCES
