An Environment to Support User Interface Evaluation Using Synchronized Video and Event Trace Recording

Albert N. Badre, Scott E. Hudson, and Paulo J. Santos

Graphics, Visualization, and Usability Center
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

ABSTRACT

This paper presents a simple but very powerful technique to support user interface evaluation along with a prototype open environment — *I-Observe*, the Interface OBServation, Evaluation, Recording, and Visualization Environment — which supports a preliminary implementation of this technique. This technique operates by recording user interface sessions in multiple modalities, both as a trace of interesting events and through video images. It then provides tools to allow the user interface evaluator to combine these modalities, analyzing the event stream to search for patterns of interesting or important user actions, then using the recorded timestamps associated with these actions to present only the sections of the video recording of interest. This allows, for example, all places where the user invokes a help system or a particular command to be observed without requiring the evaluator to manually search the recording or sit through long sessions of unrelated interactions. By combining the precise recording of automatic event trace capture with the rich contextual information that can be captured in a video and audio recording, this technique allows analyses to be performed that would not be practical with either media alone.

Keywords: User Interface Evaluation, Evaluation Support Tools, Event Trace Logging, Video-Based User Interface Evaluation Techniques.

INTRODUCTION

In the last decade a great deal of effort has been devoted to developing tools such as user interface design support systems [Harts90], user interface management systems (UIMS) [Fole88, Myer89, Olse89], and user interface toolkits [McCo88, Myer90, Lint87], to reduce the cost of producing high quality user interfaces. A central goal of many of these systems has been to make user interface construction economical enough so that true iterative design could be applied in practice. As a result, most of this effort has concentrated on the design and implementation phases of development. However, comparatively little work has been done on tools to support the user interface evaluation task which is also a key component of an iterative design approach (notable exceptions being [Buxt83, Olse88]). This paper considers a simple but very powerful technique to support evaluation by combining video-based evaluation techniques with analysis of automatically recorded event traces.

Video recording is a common and important technique in usability studies. In controlled experiment settings, subjects are filmed during the experiment sessions. The recorded video is then reviewed, annotated, codified, and analyzed. Similarly in field studies, video is collected from actual usage and analyzed later to isolate user interface problems or to study particular phenomena.

Video offers a number of advantages including recording of the entire context of an interaction (e.g., use of documentation, environmental distractions, etc.), the ability to hear verbal or thinking-aloud style comments [Lewi82] made by the user, and simply the ability to see exactly what the user is doing at a given point.

Unfortunately, the task of reviewing video recordings and annotating them is an expensive and time-consuming endeavor — particularly when field studies involving many hours of usage need to be considered. In addition, the precision with which a human observer can detect and

This work was supported in part by the National Science Foundation under grant IRI-9015407, the Center for Information Management Research under the IUCRC program, and a grant from Sun MicroSystems.

record events from a video sequence is limited (e.g. it would not normally be possible to determine to the millisecond how much delay occurred between keystrokes in a textual interface). As a result, these factors limit the kinds of analysis that can be carried out based on video recorded data.

On the other hand, event trace recordings allow very precise information to be gathered. Event traces — consisting of both base input events and synthetic events which indicate interface or application code action — are relatively easy to capture and can normally be recorded with fairly precise timing information (e.g., with *timestamps* indicating precisely when the event occurred). In addition, since event traces can be recorded automatically and unobtrusively, the recording is guaranteed to be accurate and comprehensive. As a result, evaluation techniques based on analytical models (see for example [John90]) can be employed.

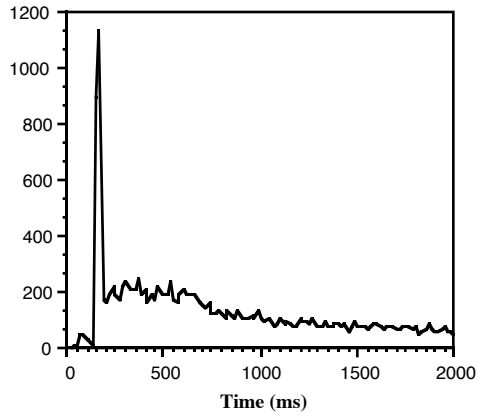

Figure 1. Distribution of "Think-Time" Delays

As an example of this sort of technique, Figure 1 shows the results of an analysis we performed "by hand" (i.e., with a long series of AWK [Aho79] scripts) before beginning development of the I-Observe environment. This graph shows data collected from the Apple Info-Booth Kiosk presented at CHI '89 and distributed on CD-ROM [Solo90]. In this example we were interested in the distribution of times between the point at which a new screenful of information was presented and the point at which the user acted on that information by pressing a key or mouse button.

The distribution that was obtained was not what was expected. In particular, the distribution contained a fairly high percentage of very small delays that were counter to our initial intuition. After a long series of additional analyses we hypothesized and partially verified that this anomalous spike was due to a particular common user action where a series of cards showing the pictures of conference attendees were "flipped through" very quickly by rapidly clicking the mouse button.

With the aid of better visualization, and particularly with the use of synchronized video data, this kind of analysis would have been dramatically easier and could have been performed in minutes rather than hours. In particular, we might have been able to select the region of the distribution that was unexpected and turn directly to corresponding segments of the video to determine the nature of the interactions involved

Tools supported by the I-Observe evaluation environment are designed to do precisely this sort of work. They are intended to combine the advantages of both the analytically strong event recording and the semantically rich video data. In particular, precise information from the event trace can be analyzed and searched for patterns of activity. Based on selections made in the event trace, the richer contextual information of the video media can be used. This richer source of information can allow the evaluator to gain a more precise understanding of what the user was doing (and perhaps why). For example, the video data can record the nature of *external* behavior (actions that do not involve direct interaction with the computer) such as reading the screen, consulting the manual, thinking, or answering the phone, which by definition cannot be captured in the event trace. This knowledge can in turn be used to refine the selection process or to insert additional information into the event stream for further analysis.

RELATED WORK

The work described here builds on a number of techniques appearing elsewhere, attempting to combine them in a way that makes new analysis and evaluation approaches possible. For example, keystroke and other low level logging or event trace tools have been popular with usability researchers for a long time (see for example, [Good85]). With the advent of graphical user interfaces, logging has become more elaborate, for it now has to record not just keystrokes but also

spatial information such as mouse clicks, and mouse trajectories. Several tools have appeared recently which record events in graphical user interfaces such as [Badr91, Korn90].

Some user interface management systems have also provided logging capabilities. For example, one early UIMS [Buxt83] provided event logging and analysis as a central feature of the system. Since UIMSs (or UI toolkits) are in control of the interaction and have semantic knowledge of the state of the application and interface, they can produce logs at a higher level, and include with each event important context information. An example of this approach can be found in the MIKE system [Olse88]. A stand-alone tool which provides a range of techniques for analysis of event traces is also described in [Sioc91].

A number of tools to support video based annotation and evaluation have been produced as well. These systems assist evaluators in replaying and annotating video recordings of sessions. Most integrate the functions of VCR control panel with an annotation mechanism which allows humans to recognize and mark the time of significant events. Once the annotations have been entered, the evaluator may use them as index marks to the video: the VTR controller may roll the tape to a previously annotated position, and replay the related segment. VideoNoter [Trig89, Rosc90], U-Test [Kenn89], and the Virtual VCR [Buxt90] are examples of tools that allow some form of video indexing. The EVA system [Mack89a, Mack89b] goes further by providing synchronized display of logged events. However, events, or patterns of events, cannot be used as a basis for search and selection of video segments.

Finally in the most closely related previous work, a system which supports selection and display of video segments using patterns of events was demonstrated at CHI '92 [Hamm92]. However few technical details are available and the stand-alone system did not support an evaluator's interface, extensible environment, or any form of analysis tools.

ARCHITECTURE AND IMPLEMENTATION

The I-Observe prototype environment takes an open approach — it supports a loosely connected set of (semi-)independent programs which can be combined to carry out evaluation tasks. These independent programs communicate using a simple common data representation. Four major classes of system components are supported by the environment: data collection systems, selection tools, analysis tools, and visualization tools. These components fit into an overall architecture as illustrated in Figure 2. Because of the open approach taken, each component of the system can be easily replaced or extended to meet specific evaluation needs, and a variety of different tools can be employed to meet particular needs. A typical use of the environment is shown in Figure 3. Here a pattern matching process is used to identify interesting segments in the video recording which are then viewed by the evaluator (without sitting through the more voluminous uninteresting portions).

Data collection system components are responsible for capturing event stream and other data which forms the subject of evaluation. This data is stored in a file using the common (text based) data format for easy manipulation by other tools. The current prototype uses the CHIME data collection system [Badr91]. This system works with any X window system [Sche86] application by interposing between the window server and the client application. Each event that passes from the server to the client is recorded with a timestamp. In addition to the CHIME system, a new data collection system embedded in the Artkit user interface toolkit [Henr90] is also under development. This collection system will allow higher level information to be collected and related back to the software driving the interface.

In addition to event stream recording, the environment also supports simultaneous recording on a read-write optical video disk as well as the alternative of a video tape recorder. Use of a video disk offers the advantage of rapid random access for playback during evaluation. However, this equipment is still relatively expensive and has somewhat limited recording capacity. Video tape offers a less expensive option. While the sequential nature of the recording may produce significant delays in playback when selected sequences are far apart, our experience indicates that this does not occur too frequently and that video tape is still a viable option.

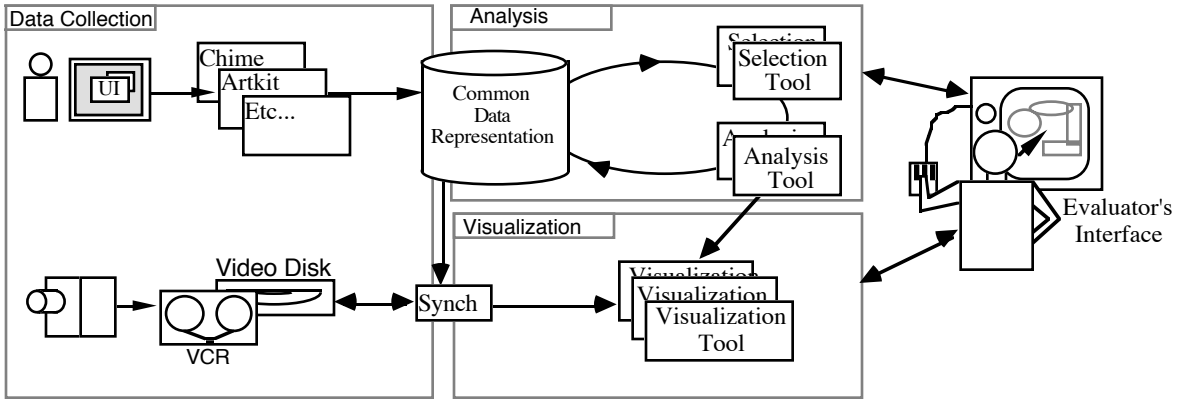

Figure 2. I-Observe System Architecture

For video tape recording, our system employs a high-end VTR which is capable of single-frame accurate positioning using time code information recorded directly on the video tape. However, our initial experience indicates that, unless multiple video streams must be displayed in a synchronized fashion, single-frame accuracy is not essential and in many cases adequate results could probably be obtained using any VTR capable of program controlled positioning to within 1/4 to 1/2 second.

Analysis components of the environment support both selection and analysis tools. Selection tools allow the interface evaluator to identify sets of interesting user actions, while analysis tools are designed to support calculations on the data and synthesis of new data elements. The prototype environment currently provides a single powerful selection tool based on a pattern recognition system which searches the event stream for patterns of actions in the form of extended regular expressions.

As illustrated in Figure 4, this tool provides an easy to use visual language for expressing patterns. This language allows patterns to be constructed without explicit programming using a notation very similar to syntax-diagrams (sometimes called "railroad-track" diagrams). For example, the pattern shown in Figure 4 selects all sequences where either the evaluator had placed a particular annotation in the event stream, or the user had invoked the "Help" command

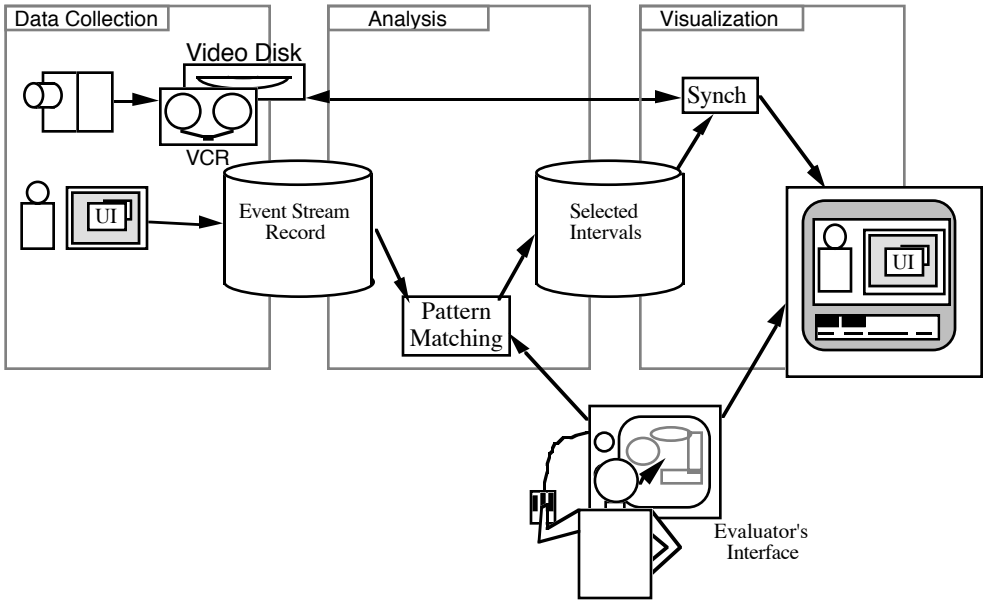

Figure 3. Typical Usage of the Environment

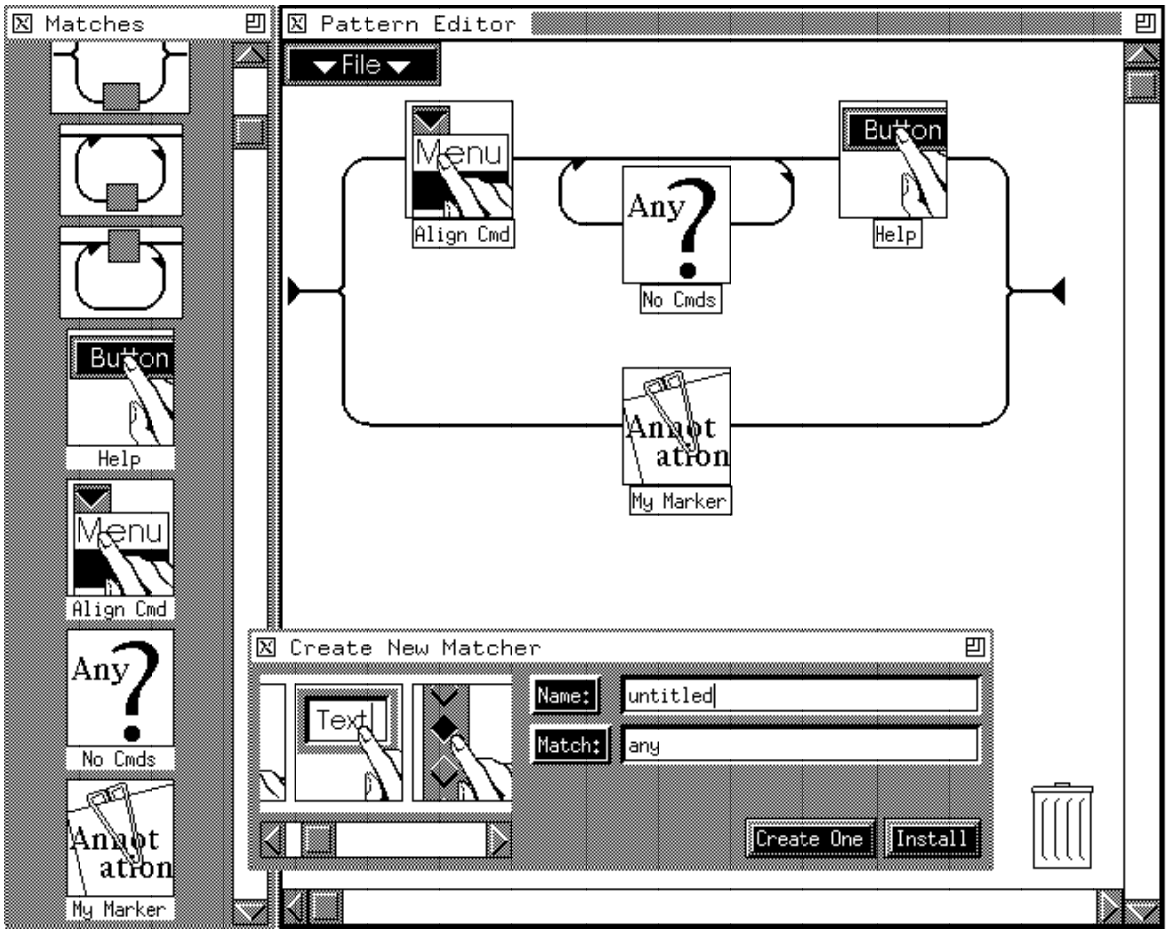

Figure 4. The Pattern Specification Interface

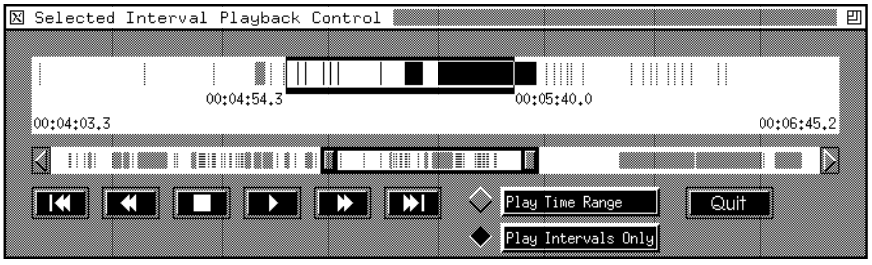

Figure 5. The Time Line Interface and Playback Controller

immediately after the "Align" command (found via its position in a particular menu)

In general, the system allows any predicate (expressible in C) over the fields contained in a single event record to be matched as atomic items. Each match predicate is represented by a named icon in the visual language. The window in the lower portion of Figure 4 provides a mechanism for constructing new custom match predicates and installing them in the scrolling palette at the left. These items can then be edited into a complete syntax diagram by dragging and snapping them together with looping and alternation constructs.

When searches are requested, the syntax diagram patterns are automatically compiled into equivalent regular expressions encoded in a textual notation and passed to the pattern matching subsystem. The use of this textual notation as an intermediate form allows the system to be easily extended to

support new tools which automatically generate patterns.

The textual notation for regular expressions is compiled into a non-deterministic finite state machine [Aho86] for pattern matching. A non-deterministic machine must be used since the predicates which take on the role of the alphabet in this situation are not necessarily disjoint in the events that they match. After removing empty transitions for this non-deterministic machine, the system uses it to search the event stream, returning the set of intervals (start and end events) matched by the pattern. These intervals can then be passed to a visualization tool within the environment.

In the current prototype, the environment supports a visualization tool for selection and playback of video sequences based on selected intervals (e.g. the intervals identified by the pattern matching tool). The system takes each interval sequence and extracts the timestamps from the selected events. Overlapping intervals are then merged, a minimum interval time is imposed to ensure that sequences are long enough to view, and a short user-defined *pre-roll* and *post-roll* time is added to each interval so that the action being viewed is shown with at least some context. Finally, the set of selected intervals are displayed in a timeline interface as shown in Figure 5.

This interface is used as the basis for interactive control of synchronized video playback. The evaluator can zoom in and out of the timeline using the special two-part slider shown in the center of Figure 5. Within the zoomed view a region for display can be selected and VCR-like controls at the bottom can be used to interactively control video playback.

To display a particular video sequence, the visualization tool simply calculates the time code corresponding to the timestamp of the start event, sends commands to move the disk or tape to that position, and directs the device to play until the time code corresponding to the end event of the interval is reached.

NEW ANALYSIS TECHNIQUES

In this section we consider in detail an example analysis that makes use of the combined advantages of precise event trace recording and the rich information that can be recorded on video in a way that would not be possible in either media alone.

This example is based on analysis of chunking behavior. There is ample experimental evidence that people represent and manipulate information in coherent units we call chunks. Further, studies such as [Chas73, Reit76, Badr82a, Badr82b] strongly suggest that the size and content of a mental chunk is revealed through characteristic patterns of behavior. When performing a task, a person typically engages in a series of actions followed by a pause, indicating the boundary between two chunks.

Chunking behavior can be interesting for a number of reasons. For example, it can be predictive of a user's ability on a scale of expertise. Novices will form smaller chunks, with fewer elements per chunk. As a user progresses and becomes more expert, the duration of the inter-chunk pauses becomes shorter, less frequent, and less variable [Badr82a]. Consequently, determination of the user's chunks, their components, sizes, and complexity can be useful in answering a range of experimental and practical questions.

In one experiment, we are analyzing pauses and flurries of activity to identify and characterize user chunks. However, analysis of the sort we need for this experiment cannot be performed with either pure video or pure event analysis techniques alone. Since the timing of actions and pauses is critical in this analysis, the analytical accuracy that can be attained from the event recording is a must. However, for this experiment, we also needed to classify what the user was doing, both during the chunk, and during the pause. Since this includes actions outside direct interaction with input devices, as well as a classification of fairly abstract human behavior, at least some of this information could not possibly be attained from the event stream alone.

The facilities provided by the I-Observe system are well suited to this problem. The pattern matching and analysis capabilities can be used to identify and characterize chunking behavior analytically based on frequency and duration of activity and pauses. This information can then be used to control a very targeted presentation of the video recording — showing only the actions of interest with a small amount of context — in order to classify the nature of the activity during selected chunks (e.g., as waiting for the system, observing

system operation, problem solving, movement from device to device, distraction, etc.).

By combining the analytical analysis possible with the timestamped event recording with the more informal, but still critical, analysis of the video information, the I-Observe environment offers a unique opportunity for perfecting and validating these chunk based analysis techniques and allows analyses which could not have been done using one recording media alone.

FUTURE WORK

The environment described in this paper represents a basic framework into which a series of tools will eventually be placed. For example, we are currently exploring several event recording systems that operate at different levels of "intrusion". These range from toolkit independent systems, such as CHIME, which can record from any X window system application, but provides only low level events, up to systems embedded in a particular toolkit, that can record much higher level actions. In addition we are exploring new visualization techniques for event streams, and analysis results, as well as techniques for combining these visualizations with synchronized video playback.

CONCLUSION

This paper has presented a new open user interface evaluation environment supporting the simple but very powerful technique of capturing user interface sessions both as a stream of events and as a synchronized video recording. Using pattern matching and analysis tools, this event stream can be used to select interesting or relevant user actions. These selections can then be used to drive the display of corresponding sections of the video tape. This technique allows evaluators to use the rich information provided by video recording in a highly targeted fashion. For example, all invocations of a particularly troubling command could be viewed without requiring the evaluator to sit through hours of irrelevant interaction. With this technique, it should be possible to use existing evaluation strategies more easily and to develop new evaluation methods that were not previously practical using either pure video or event trace recording alone.

REFERENCES

[Aho 79] Aho, A., Kernighan, B. and Weinberger, P., "AWK — A Pattern Matching and Scanning Language", Software — Practice and Experience, 9(4), pp. 267-280, 1979.

[Aho86] Aho, A., V., Sethi, R., and Ullman, J. D., Compilers: Principles, Techniques, and Tools, Addison-Wesley Publishing Company, Reading Mass, 1986.

[Badr82a] Badre, A., "Selecting and representing information structures for visual presentation", *IEEE Transactions on Man, System, and Cybernetics*, pp. 495–504, 1982.

[Badr82b] Badre, A., "Designing chunks for sequentially displayed information", in Badre and Shneiderman, eds., *Directions in Human/Computer Interaction*, Ablex, 1982.

[Badr91] Badre, A.N., and Santos, P.J., "A knowledge-based system for capturing human–computer interaction events: CHIME — Observations and Issues", *Georgia Institute of Technology technical report GIT-GVU-91-21*, 1991.

[Buxt83] Buxton, W., Lamb, M., Sherman, D., and Smith, K., "Towards a Comprehensive User Interface Management System", *Proceedings of SIGGRAPH '83*, pp. 35-42, 1983.

[Buxt90] Buxton, W. and Moran, T., "EuroPARC's Integrated Interactive Intermedia Facility (IIIF): Early Experiences", in Gibbs, S. and Verrijn-Stuart, A. A. (eds), *Multi-User Interfaces and Applications*, North-Holland, pp. 11-34, 1990.

[Chas73] Chase, W., and Simon, H., "Perception in Chess", *Cognitive Psychology*, vol. 4, pp. 55–81, 1973.

[Fole88] Foley, J., Gibbs, C., Kim, W., and Kovacevic, S., “A knowledge-based user interface management system”, *Proceedings of CHI'88*, pp. 67-72, 1988.

[Good85] Good, M., “The Use of Logging Data in the Design of a New Text Editor”, *Proceedings of the CHI'85 Conference on Human Factors in Computing Systems* (1985), pp. 93–97.

[Hamm92] Hammontree, M. L., Hendrickson, J. and Hensley, B. W., "Integrated Data Capture and Analysis Tools for Research and Testing on Graphical User Interfaces" (Demonstration), *Proceedings the CHI '92*, pp. 431-432, 1992.

[Harts90] Hartson, H. R. , Siochi, A., and Hix, D., “The UAN: A User-Oriented Representation for Direct Manipulation Interface Designs”, ACM Transactions on Information Systems, 8(3), pp. 181-203, July 1990.

[Henr90] Henry, T., Hudson, S and Newell, G., “Integrating Gesture and Snapping into a User Interface Toolkit”, *Proceedings of the ACM Symposium on User Interface Software and Technology*, pp. 112-122, 1990.

[John90] John, B., “Extensions of GOMS analyses to expert performance requiring perception of dynamic and auditory information”, *Proceedings of CHI'90*, pp. 107-115, 1990.

[Kenn89] Kennedy, S. "Using Video in the BNR Usability Lab", *SIGCHI Bulletin*, **21**(2), October 1989, pp. 92-95.

[Korn90] Kornbrot, D, Macleod, M., Diaper, G., Gilmore, D., Cockton, G., and Shackel, B., "Monitoring and Analysis of Hypermedia Navigation", *Proceedings of INTERACT '90*, 1990, pp. 401-406.

[Lewi82] Lewis, C. and Mack R., "Learning to Use a Text Processing System: Evidence from "Thinking Aloud" Protocols", *Proceedings of Human Factors in Computer Systems*, 1982, pp. 387-392.

[Lint87] Linton, M.A., and Calder, P.R., “The design and implementation of InterViews”, *Proceedings of USENIX Association C++ Workshop*, pp. 256-267, 1987.

[Mack89a] Mackay, W. E. and Davenport, G., "Virtual Video Editing in Interactive Multimedia Applications", Communications of the ACM, 32(7), July 1989, pp. 802-810.

[Mack89b] Mackay, W. E., "EVA: An Experimental Video Annotator for Symbolic Analysis", *SIGCHI Bulletin*, **21**(2), October 1989, pp. 68-71.

[McCo88] McCormack, J. and Asente, J., “An Overview of the X Toolkit”, *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software*, pp. 46-55, 1988.

[Myer89] Myers, B.A., “User Interface Tools: Introduction and Survey”, *IEEE Software*, 6(1), pp. 15-23, November 1990.

[Myer90] Myers, B.A., Giuse, D.A., Dannenberg, R.B., Vander Zanden, B., Kosbie, D.S., Pervin, E., Mickish, A., and Marchal, P., “Comprehensive support for graphical, highly-interactive user interfaces: the Garnet user interface development environment”, *IEEE Computer*, **23**(11), pp. 71-85, November 1990.

[Olse88] Olsen, D. R., and Halversen, B. W., “Interface Usage Measurements in a User Interface Management System”, In *Proceedings of ACM SIGGRAPH Symposium on User Interface Software* (Banff, Alberta, Canada, Oct. 17–19). ACM Press, 1988, pp. 102–108.

[Olse89] Olsen, D., “A Programming Language Basis for User Interface Management”, *Proceedings of CHI'89,* pp. 171-176, 1989.

[Reit76] Reitman, J., “Skilled perception in Go: Deducing memory structures from inter–response times”, *Cognitive Psychology*, vol. 8, pp. 336–377, 1976.

[Rosc90] Roschelle, J., Pea, R., & Trigg, R., “VideoNoter: a Tool for Exploratory Video Analysis”, IRL Technical Report IRL90-0021, March 1990.

[Sche86] Scheifler, R. W. and Gettys J., "The X Window System", *A C M Transactions on Graphics*, v5, April 1986, pp. 79-109.

[Sioc91] Siochi, A. C., and Ehrich, R. W., "Computer Analysis of User Interfaces Based on Repetition in Transcripts of User Sessions", *ACM Transactions on Information Systems*, v9 n 4, October 1991, pp. 309-35.

[Solo90] Solomon, G., et al., "CHI'89 Info Booth", CD-ROM containing user interface usage trace data, Apple Computer Inc., 1990.

[Trig89] Trigg, R., “Computer Support for Transcribing Recorded Activity”, in *SIGCHI Bulletin* **21**(2), October 1989.