TOWARDS AUTOMATIC EVALUATION OF MULTIMODAL USER INTERFACES

Sandrine Balbo, Joëlle Coutaz, Daniel Salber

Laboratoire de Génie Informatique, IMAG
B.P. 53X, 38041 Grenoble Cedex, France
Tel. +33 76 51 48 54, e-mail: balbo@imag.fr, joelle@imag.fr, salber@imag.fr

CONTACT ADDRESS OF MAIN AUTHOR
Joëlle Coutaz
Laboratoire de Génie Informatique, IMAG
B.P. 53X, 38041 Grenoble Cedex
Tel. +33 76 51 48 54
Fax +33 76 44 66 75
e-mail: joelle@imag.fr

ABSTRACT

The evaluation of the usability and the learnability of a computer system may be performed with predictive models during the design phase. It may be done on the executable code as well by observing the end user in action. In this case, data collected in vivo must be processed. Our goal is to provide a software support for performing this difficult task.

This article presents an early analysis and experience towards the automatic evaluation of multimodal user interfaces. With this end in view, a generic Wizard of Oz platform has been designed to allow the observation and the automatic recording of subjects'behavior while interacting with a multimodal interface. We then show how recorded data can be analyzed to detect behavioral patterns, and how deviations of such patterns from a data flow-oriented task model can be exploited by a software usability critic.

KEYWORDS

Capture of behavioral data, multimodal user interface, Wizard of Oz, user interface evaluation techniques.

TOWARDS AUTOMATIC EVALUATION OF MULTIMODAL USER INTERFACES

Sandrine Balbo, Joëlle Coutaz, Daniel Salber

Laboratoire de Génie Informatique, IMAG
B.P. 53X, 38041 Grenoble Cedex
Tel. +33 76 51 48 54, e-mail: balbo@imag.fr, joelle@imag.fr, salber@imag.fr

ABSTRACT

The evaluation of the usability and the learnability of a computer system may be performed with predictive models during the design phase. It may be done on the executable code as well by observing the user in action. In this case, data collected in vivo must be processed. Our goal is to provide a software support for performing this difficult task.

This article presents an early analysis and experience towards the automatic evaluation of multimodal user interfaces. With this end in view, a generic Wizard of Oz platform has been designed to allow the observation and the automatic recording of subjects'behavior while interacting with a multimodal interface. We then show how recorded data can be analyzed to detect behavioral patterns, and how deviations of such patterns from a data flow-oriented task model can be exploited by a software usability critic.

KEYWORDS

Capture of behavioral data, multimodal user interface, Wizard of Oz, user interface evaluation techniques.

INTRODUCTION

The development of interactive systems is an iterative process composed of three steps: design, construction and evaluation. Software tools such as interaction toolkits and UIMS technology, or software architecture models such as PAC [7] and the Abstraction-Link-View paradigm [14], have been developed to facilitate the construction of graphical user interfaces (GUI). Although the construction of user interfaces has been widely addressed by the software engineering community, little attention has been paid to software support for user interface design and evaluation.

Parallel to the development of graphical user interfaces, natural language processing, computer vision, and gesture analysis have made significant progress. Clearly, the combination of medias and modalities open a complete new world of experience but our current understanding on how to design, build and evaluate such interactive systems is very primitive.

This article presents our early analysis and experience with the automatic evaluation of multimodal interfaces. Our goal is to provide designers with a Wizard of Oz multimodal software platform flexible enough to support the evaluation of multiple interactive systems. The next two sections define the problem space: first, the main streams for user interface evaluation are presented and our own approach is localized in this framework. Then, a taxonomy for the study of multimodal interfaces is presented. The last two sections are dedicated to our own solution to the problem: the description of the platform and a first experience with automatic evaluation.

AN OVERVIEW OF EVALUATION TECHNIQUES

As shown in figure 1, evaluation techniques for interactive systems are divided in two broad categories.

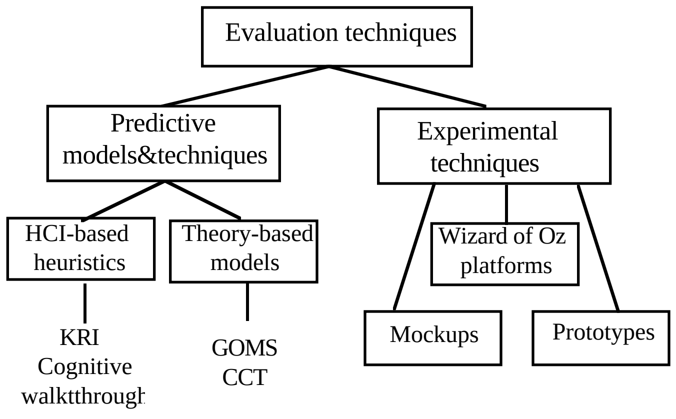

Figure 1: An overview of evaluation techniques for interactive systems.

Predictive methods are applicable during the design phase. They do not require any system implementation, nor do they need effective users. At the opposite, experimental techniques rely on the existence of a physical apparatus ranging from mock-ups of the real system up to the full implementation of a running prototype.

Predictive techniques

In general, predictive techniques are theory-based. For example, GOMS [5] and its related models such as the Cognitive Complexity Theory (CCT) [15], rely on an explicit hierarchical decomposition of the user's tasks. This static representation is supposed to model the user's plan for accomplishing a particular task. The genuine GOMS is a pure analytic model of errorless performance: it is able to predict the time required to accomplish a task without errors. Recently, GOMS has been extended to predict errors due to cognitive overload [16]. CCT, on the other hand, is useful for comparing several designs in terms of learnability and knowledge transfer.

At the opposite of GOMS and CCT, PUM builds a dynamic model of the user's plan. It predicts errors through a programmable cognitive architecture [27]. The designer specifies the knowledge that the user needs to accomplish a particular task. This description, which includes domain knowledge as well as knowledge about the user interface, is compiled in terms of rules. These rules represent the user's ability to accomplish this particular task with that particular user interface. Based on this knowledge, the PUM cognitive architecture tries to elaborate a plan. If no plan can be generated, then the designer is notified of a potential usability or learnability problem.

Predictive evaluation techniques may also be based on HCI heuristics. Typically, the assessor looks for properties in the user interface design that, he knows from experience, lead to usability or learnability problems. Such knowledge, exemplified by the Smith & Mosier's work, may be embedded in an expert system such as KRI [18]. KRI is able to detect anomalies from a formal description of the user interface. However, only the lexical and syntactical levels of the interaction are covered by the critic. Task modelling and any high level cognitive activity are discarded.

Assessing a design through HCI heuristics is a difficult task. The task-based "cognitive walkthrough" method proposed by Lewis et al. provides a useful framework for extracting evaluation guidance from a formal theory of human-computer interaction [17]. It consists of "a list of theoretically motivated questions about the system" such as "how will user access description of action?" or "how will user associate description to action?".

In summary, the main benefit from predictive models and techniques is that they allow the evaluation of user interfaces at the design stage. A design can be improved before a costly implementation takes place. On the other hand, specifying data to a predictive model may be as time consuming as the implementation per se. In addition, predictions made by theoretical models are based on hypotheses, not on real data. As demonstrated by Pollier [22] as well as by Nielsen et al. [21], heuristic evaluation is hard. At least three assessors are necessary to discover a reasonable number of usability problems (i.e., half of the problems at best!)

Experimental techniques

Experimental techniques and methods deal with real data observed from real users accomplishing real tasks with a physical artefact. This artefact may include paper scenarios, mock-ups, computer system prototypes, or Wizard of Oz (WOz) platforms.

With a WOz setting, "designers can illustrate how users will interact with yet-to-build software" [19]. In general, MOz experiments have been applied to natural language interfaces only. Corpus collected in vivo would be used to tune the linguistic parameters of the system, and thus would improve the robustness of the interaction [9].

In general, behavioral data from MOz experiments, are tape-recorded. As a result, they must be retrieved and interpreted by hand. This is a time consuming task which requires expertise and patience. However, recent MOz platforms are able to capture and mix digitized and analogical behavioral data [12]. By doing so, automatic tools can be developed to support the evaluation process. However, to our knowledge, none of the MOz platforms has tackled the problem of multimodal interfaces.

In summary, analysis from experimental methods are performed on real data, not on uncertain hypothetical values. This benefit is counterbalanced by the volume of behavioral data to process and by the difficulty to identify the appropriate parameters for a particular experiment. We believe that a MOz computer platform which automatically captures selected behavioral data, provides a good basis for the development of evaluation and design tools. In addition, a MOz computer platform to study multimodal interaction is certainly a promising enterprise. Our approach to this problem is presented in the rest of the article.

A TAXONOMY FOR MULTIMODAL USER INTERFACES

In psychology, a modality refers to a human sensory channel such as vision, audition and touch. In the theoretical framework of the Model Human Processor [5] as well as in ICS [3], these channels are modelled as specialized processors. Whereas a modality denotes a type of human communication channel, a media such as a computer system, is an artefact that conveys information by triggering one or several human communication channels. According to these definitions, how can a user interface be qualified as being multimodal? The ICS model can provide us with a useful starting point.

Multimodal User Interface: a Definition

In ICS, the human information processing system is subdivided into a set of specialized subsystems. The sensory subsystems transform sense data into specific mental codes that represent the structure and content of incoming data. These representations are then handled by subsystems specialized in the processing of higher-level representations: the morphonolexical subsystem for processing the surface structure of language, the object subsystem for processing visuospatial structures, and the propositionnal and implicational subsystems for more abstract and conceptual representations. The output of these higher-level subsystems are directed to the effector subsystems (articulatory and limb).

Using a similar process, a multimodal system is able to represent and manipulate information at multiple levels of abstraction. This transformation process makes possible the automatic extraction of meaning from raw input data, and conversely the production of perceivable information from symbolic abstract representations[1]. If “striving for meaning” denotes multimodal user interfaces, fusion and temporal constraints define a classification for multimodal interaction per se.

Two dimensions for classifying multimodal interfaces

Fusion covers the combination of different types of modalities. The absence of fusion is called exclusiveness, while the existence of fusion forms a synergy. Thus, a multimodal user interface is

• exclusive if input (or output) expressions are built up from one modality only,

• synergic if input (or output) expressions are built up from multiple modalities.

Figure 2 illustrates the discussion.

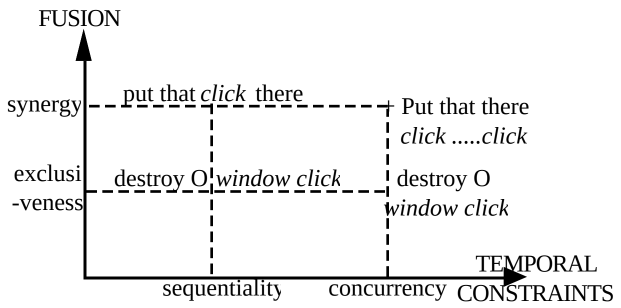

Figure 2. Examples of input expressions in multimodal user interfaces. Text appearing on one line denotes sequentiality, while text appearing on 2 lines denotes concurrency. Italic expresses mouse gesture. Normal text indicates spoken words.

[1]At the opposite of multimodal systems, multimedia systems 1) do not elaborate abstract concepts automatically (these are encoded manually as meta-information), and 2) they do not transform the information per se but apply encapsulation on it.

As an example of exclusive multimodal user interface, we can imagine the situation where, to open a window, the user can choose among double-clicking an icon or say "open window". One can observe the redundancy of the ways for specifying input expressions but, at a given time, an input expression uses one modality only. As an example of synergic multimodal system, the user of a graphics editor can say *put that there* while pointing at the object to be moved and showing the location of the destination with the mouse or a data glove.

Time constraints express the possibility for the user and the system to build exclusive or synergic expressions sequentially or concurrently. Sequentiality or concurrency may be studied at the input/output expression level as well as at finer grain, such as the physical actions involved in the specification of expressions. Sequentiality in synergic multimodality implies that modalities be interleaved during the construction of an expression. If we consider the *put that there* example, sequentiality would require the user to say *put that* followed by a mouse click to denote *that*. He would then say *there* and click a second time to indicate the destination. Although not desirable, sequentiality may be imposed for technical reasons. In a concurrent synergic multimodal system, the user would utter the sentence and perform the mouse clicks in a “natural” way, i.e., without necessarily interleaving modalities.

In summary, the two dimensions, fusion and temporal constraints, define a problem space which, as shown in the next paragraph, provides a useful framework for reasoning about multimodality.

Benefits from the classification

As discussed in [8, 10], one can study the implications of fusion and temporal constraints on software architectures. Bourguet and Caelen in [4] exploit the framework for the interpretation of multimodal expressions in a dialogue model.

For example, in a graphic editor supporting concurrency, consider the vocal command *rotate the triangles* combined with the selection of a set of triangles with the mouse. Depending on the presence of fusion or not, the interpretation of the vocal command may have different effects:

• In an exclusive multimodal user interface, the vocal and gesture commands are independent. Then, *rotate the triangles* can be interpreted in two ways: a) rotation of the triangles selected in a previous command (i.e., the pronoun *the* acts as an anaphoric reference); b) rotation of all of the triangles in the picture if no triangle has been previously selected.

• In a synergic multimodal user interface, the vocal and gesture commands can be coupled. Thus, in addition to interpretations a) and b), *rotate the triangles* allows a third interpretation c) where *the* acts as a deictic related to the concurrent gesture. The choice between the 3 interpretations relies heavily on the dialogue model used to drive the interface. In particular, if parallelism at the interface prevails, then solution c is selected. If sequentiality dominates, then solutions b and c are good candidates. In addition, if vocal modality is privileged, then c prevails over b.

Psychological theories and human factor principles claim that a dialogue model should consider the task domain, the task and the user. Although some of their recommendations are useful for GUI technology, they are not directly extensible to multimodal user interfaces. In the absence of hard-core theory, we opt for the Wizard of Oz experimental approach.

NEIMO, A MULTIMODAL WIZARD OF OZ PLATFORM

Objectives

The goal of Neimo is to provide designers with a "Wizard of Oz" environment to observe and evaluate how users interact with multimodal interfaces. Wizards are used to supplement missing functions such as recognizers or generators for a particular modality.

Modalities studied with the platform include graphics, oral communication, and gesture. Gesture covers facial expression and 2-D pen scribbling. We consider that the redundancy provided by facial expression can be used to disambiguate the interpretation of end-user's behavior.

An essential requirement for such an environment is flexibility. Neimo must support any number of wizards, any number of modalities, and any type of application whose source code is available[2]. In addition, for a given application and a given set of modalities, it should allow multiple experiments with fusion and time constraints.

Hardware configuration

Figure 3 illustrates a typical hardware configuration for the Neimo platform. It includes one workstation for the user (i.e., the subject) and several workstations for wizards. All of the workstations are Apple Quadras.

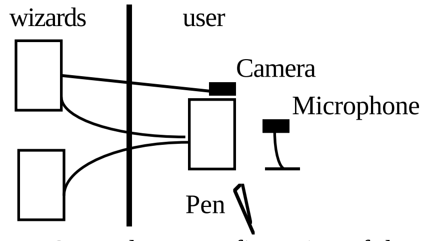

Figure 3: Hardware configuration of the Neimo platform.

The user's workstation is equipped with a pen computer, a microphone and a speaker to study pen gesture coupled to speech and facial expression. Voice Navigator, a word-based Markov network-based pattern matching system, is used as the speech recognizer. Because of its poor performance with regard to "naturalness", we intend to replace Voice Navigator with a wizard and evaluate the behavioral differences. The pen gesture recognizer is based on Rubine's work [23].

A CCD camera is focussed on the user's face and connected to a video acquisition board installed in one of the wizards' workstation. Thus, it is possible for that wizard to observe an image of the user's face digitized in real time (2 to 3 images per second).

Wizards are classified in two categories: the functional wizards and the modal wizards. A functional wizard accomplishes the task domain dependent services that are not implemented in the application. A modal wizard replaces or complements software components specialized in the processing of a particular modality or in the fusion of multiple modalities. For example, when Voice Navigator is turned off, a speech modal wizard comes into play. Similarly, a modal vision wizard is used to interpret facial expressions[3] [26].

Software organisation and services

In order to satisfy the flexibility requirement, the Neimo platform is organized around a minimal communication kernel, NeimoCom, interfaced by libraries.

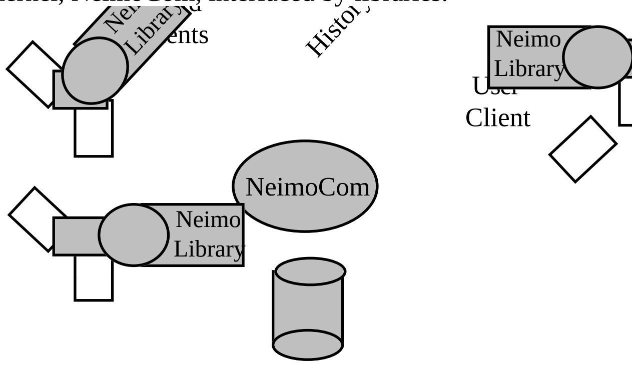

Figure 4: The software organisation of the Neimo platform. Dimmed areas denote common services and white areas, specific components.

As shown in Figure 4, NeimoCom acts as a communication server for transferring messages between workstations, and serves as a message recorder in history files. Each workstation runs a set of client functions linked to a NeimoCom library. These functions are not provided by the kernel but are developed for specific purpose. For example, the client functions of the speech wizard workstation support the wizard's task. In particular, a list of prerecorded answers is proposed to the wizard in order to alleviate his cognitive load as well as to guarantee a consistent behavior with regard to the user.

[2]An application is a functional core that implements task domain concepts.

[3]Turk's face recognition system developed at MIT is being adapted for the Quadra environment.

A Neimo library provides client functions with the following main services:

• open and close a connection. A wizard workstation can dynamically open or close a connection during a session.

• declare new types of message or redefine previously defined types with additional fields and/or suppression of obsolete fields. A message type is declared as a data structure with named typed fields. Since data fields are named, their order is not significant. By so doing, old clients can be modified or new clients added to the environment without jeopardizing previous settings. For example, a color field may be added to the original type*face* which was first designed for black and white pictures of faces.

• dynamically subscribe to a set of message types. This service allows clients to express their interest for a category of messages. For example, the vision wizard client subscribes to the *face* type to receive images of the user's face in order to be able to process and display them onto the wizard's screen. In addition, the dynamicity of the subscription allows wizards to change roles on the fly.

• send and receive messages synchronously. Messages are time-stamped by NeimoCom which maintains the universal time.

• open, close history files and record messages in an opened history file[4]. In addition to navigation functions such as *go to the last record of history file h*, or *select record i from history file h*, clients can define views. A view acts as a filter on a history file. It includes a start and a stop date to identify the temporal window of interest, the origin (i.e., wizard VS user) as a selector of the source of the recorded messages, and a list of the message types of interest (for example face, mouse and speech).

A more detailed description of the run time kernel is available in [1]. To summarize, the originality of the Neimo kernel is three-fold:

1) messages from multiple medias are processed in a uniform and integrated way,

2) message types are not imposed by the system. Instead, their level of abstraction is client defined. It is then possible to record information from low level events such as mouse clicks up to high level commands;

3) messages are recorded on request in dedicated files. Messages can be subsequently retrieved either directly according to their record sequencing number, or indirectly through the notion of views. Views allow client programs to extract messages through temporal windows on a set of message types. A user interface critic is one of such potential clients.

[4]History files format takes advantage of QuickTime which supports time-based data from different medias.

In addition to the run time kernel, a minimal user interface common to all of the wizards has been designed. This user interface allows a wizard to:

• set up the configuration for the session: which user's workstation to observe, which message types to receive, etc.;

• control message recording and message load during the session;

• observe the user's behavior: reception of the user's utterances, message sequence received from the user, replication of the user's screen as well as the user's mouse movements[5].

The Neimo platform is under development using MacApp in the MPW environment. Although we have not yet been able to make full-fledged experiments with Neimo, an early experiment of message recording and interpretation under X window [2] will be re-implemented in the Neimo environment. The model developed in this experiment is described in the next section.

OUR APPROACH TO AUTOMATIC EVALUATION

Our automatic evaluation of user interfaces is a four step process: 1) definition of a task model, 2) acquisition of behavioral data, 3) identification of behavioral patterns, and 4) critic per se.

A task model defines the optimal way of performing the task in a particular context for a particular domain. It is a behavioral reference model. Recorded data, as those captured by Neimo, reflect the effective behavior of the user performing a task in a quasi-realistic setting. As in the MRP technique [25]), behavioral patterns are repeated user actions that may reveal usability problems. The critic per se combines general heuristic HCI knowledge with data specific to the case at hand: it detects deviations of the behavioral patterns from the reference task model. These four aspects are developed in the following paragraphs.

The task model

The definition of a task model depends on its end use.

In the design stage, the task model expresses the logical use of the system. It structures the work space in terms of tasks and subtasks showing the relationships between clusters of logically connected tasks. As in GOMS and CLG [20], the representation is a task hierarchy whose leaves denote the tasks that are conceptually indivisible. This is the conceptual task model.

When considering the running prototype, the task model aims at specifying the way the system functions. At the opposite of the conceptual task model, it specifies the way the user should perform the task with the real system. It

[5]In the first version of the system, this last function is implemented via Timbuktu.

does not necessarily describe a manner that is convenient for the user. Ideally, the conceptual task model and the effective task model should be isomorphic. In particular, the elementary tasks of the conceptual model should correspond to commands in the effective task model.

Although mapping elementary tasks to commands is straightforward, the correspondence for compound tasks is difficult to formalize. This situation results mainly from the discrepancy between the points of view adopted at the design and implementation steps. In the design phase, attention is focussed on the domain. As a result, syntactic tasks such as window manipulations which, by definition are domain independent, are not considered. Clearly, our evaluation technique, based on the acquisition of real data, speaks in favor of an effective task model: the sequence tree.

A sequence tree represents the sequence of possible elementary tasks. An elementary task is indivisible and modifies the system state. A node denotes an elementary task. As shown in figure 5, a node may be decorated with preconditions P. For example, to open a file from the Macintosh desktop, the folder which contains the file must be opened.

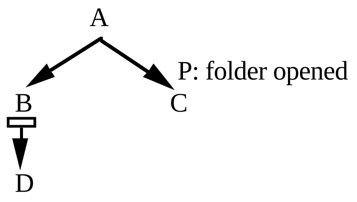

Figure 5: The sequence tree model.

Edges express ordering between elementary tasks. If A and B denote two elementary tasks, then A $\mapsto$ B means that A must be executed before B, and A cannot be executed again. In A $\rightarrow$ B**,** A must be executed before B but A may be executed again in the future. If a node has multiple siblings, then the user may choose any one of them and switch freely between the subtrees. This property allows for the expression of interleaving between tasks.

In summary, a sequence tree shows the possible migration of the user through an organized and constraint space of elementary tasks. In turn, elementary tasks are expressed in terms of the physical actions that the user can perform with input medias. A notation like UAN [13] can be used to specify the correspondence.

Collecting behavioral data

Recorded data correspond to the physical actions of the user.

In an X Window environment, actions such as mouse clicks and key presses, are modelled as events. We have recorded them in a history file with additional information such as the name of the elementary task to which it belongs. In a Neimo environment, actions are conveyed through messages and additional actions such as facial expressions and meta-comments, can be interpreted by a wizard and recorded.

Data has been recorded for a set of simple tasks selected from a complex problem space: the design of user interfaces for everyday cars. This design environment, Compo, is similar to an authoring system like HyperCard, where scripts are expressed with an iconic language [6].

Behavioral patterns

The analysis of the recorded data within Compo led us to identify three classes of interesting behavioral patterns: direction shift, action repetition and action cancellation as part of syntactic tasks.

A direction shift occurs when the user stops following a downward path in the sequence tree. For example, although it is useless to do so, he may systematically select an icon file in the Macintosh desktop before invoking the *page setup* item in the *file* menu.

Action repetition as part of syntactic tasks cover actions that do not modify the functional core but concern the user interface portion only. Typically, we have observed systematic resizing, or iconification, or scrolling tasks on newly opened windows.

Action cancellation as part of syntactic tasks refer to closing newly opened windows or navigating through menus without selecting any item.

The critic

Behavioral pattern types are then related to usability problems based on human factor and psychological principles. These are numerous, inflationary or even contradictory. As a first step experience, we have used the simple taxonomy provided by Scapin [24] with the notions of compatibility, homogeneity, concision, feedback pertinence, explicit control, cognitive load, and error management.

Rules have been encoded in the critic to point out user interface flaws through behavioral patterns. For example, the systematic occurrence of a direction shift at a particular node in the sequence tree expresses a cognitive dead-end. The interface does not lead the user to build the appropriate model. There is an incompatibility between the effective task model and the user's mental model. In the same way, systematic iconification of newly opened window may correspond to a messy screen or expresses the irrelevance of the information contained in the window.

CONCLUSION

In summary, we have developed an early experience with the automatic evaluation of interfaces using a simple apparatus. From low level captured data (i.e., user's mouse clicks and key presses), from a task model, and from general HCI heuristic knowledge, we have been able to detect anomalies for a graphical user interface. This technique needs to be extended to multimodal interfaces.

With this end in view, we have designed Neimo, a generic platform, able to register a wide range of behavioral data in an integrated way. These data may result either from the automatic interpretation of the user's behavior by the system, or from on the fly interpretation by human wizards. With such a tool, experimenters should be able to build rich but focussed history files. As an additional benefit from history files, we expect to devise user models applicable to the design of multimodal interfaces.

ACKNOWLEDGEMENTS

This article was influenced by stimulating discussions with our colleagues of the pole IHM-Multimodal of PRC Communication Homme-Machine including Jean Caelen (ICP, Grenoble) and Claude Valot (CERMA). Gilles Ambone and Bernard Noz deserve special thanks for their active participation in the design and implementation of the Neimo platform.

REFERENCES

1. G. Ambone, B. Noz, D. Salber, "Projet Neimo, Spécifications externes", rapport interne, équipe IHM, LGI-IMAG, 1992.
2. S. Balbo, J. Coutaz, "Un pas vers l'évaluation automatique des interfaces homme-machine", to appear in the ERGO-IA'92 proceedings, Biarritz, 1992.
3. P.J. Barnard, "Cognitive Resources and the Learning of Humnan-Computer Dialogs", in Interfacing Thought, Cognitive Aspects of Human-Computer Interaction, J.M. Carroll Ed., MIT Press Publ., 1987, pp.112-158
4. M.L. Bourguet & J. Caelen, "Interfaces Homme-Machine Multimodales: Gestion des Evénements et Représentation des Informations", to appear in the ERGO-IA'92 proceedings, Biarritz, 1992.
5. S.K. Card, T.P. Moran & A. Newell, "The psychology of Human Computer Interaction", Lawrence Erlbaum Associates, 1983
6. A. Chabert, "La programmation visuelle", Rapport de DEA d'Informatique, Institut National Polytechnique de Grenoble, Juin 1991
7. J. Coutaz, "PAC: an Implementation Model for Dialog Design", Proceedings of the Interact'87 conference, Stuttgart, H-J. Bullinger, B. Shackel ed., North Holland, september 1987, pp. 431-436.
8. J. Coutaz, "Multimedia and Multimodal User Interfaces: A Taxonomy for Software Engineering Research Issues", St Petersburg HCI Workshop, August, 1992.
9. Nils Dahlbäck and Arne Jönsson: "Empirical studies of discourse representations for natural language interfaces", Fourth Conference of the European Chapter of the ACL Proceedings, 1989, pp. 291-298.
10. A. Gourdol, L. Nigay, D. Salber, J. Coutaz, "Two cases studies of software architecture for multimodal interactive systems: VoicePaint and a Voice enabled graphical notebook", IFIP'92 Congress, Oct, 1992, Madrid.
11. Dan Diaper: "The Wizard's Apprentice: A Program to Help Analyse Natural Languages Dialogues", proceedings of the fifth conference of the British Computer Society, Human-Computer Interaction Specialist Group, University of Nottingham, 5-8 September 1989, pp. 231-243.
12. M.L. Hammontree, J.J. Hendrickson, B.W. Hensley, "Integrated Data Capture and Analysis Tools for Research and Testing on Graphical User Interfaces", in the CHI'92 Conference Proceedings, ACM Press Publ., 1992, pp. 431-432
13. R. Hartson, P.D. Gray, "Temporal Aspects of Tasks in the User Action Notation", Human-Computer Interaction, Laurence Erlbaum, vol. 7, No 1, pp. 1-45, 1992
14. R. Hill, "The Abstraction-Link-View paradigm: using constraints to connect user interfaces to applications", in Proceedings of CHI'92, ACM Press, May 1992, pp. 335-342
15. D. Kieras, P.G. Polson "An Approach to the Formal Analysis of User Complexity", International Journal of Man-Machine Studies, 22, 1985, pp. 365-394
16. F.J. Lerch, M.M. Mantei, J.R. Olson, "Skilled financial planning: The Cost of Translating Ideas into Actions", in Proceedings of CHI'89, ACM Press, 1989, pp. 121-126.
17. C. Lewis, P. Polson, C. Wharton, J. Rieman, "Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces, in Proceedings of CHI'90, ACM Press, 1990, pp. 235-241
18. J. Löwgren & T. Nordqvist, "A Knowledge-Based Tool for User Interface Evaluation and its Integration in a UIMS", Human-Computer Interaction-INTERACT'90, pp. 395-400
19. Wendy E. Mackay: "Video: Data for Studying Human-Computer Interaction", CHI '88, 1988, pp. 133-137.

20. T. Moran, ““The Command Langage Grammar : a representation for the user interface of interactive computer systems”, International Journal of Man-Machine Studies, 15, 1981, pp. 3-50

21. J. Nielsen, R. Molich, “Heuristic Evaluation of User Interfaces”, in CHI’90 Proceedings, ACM Press Publ., 1990, pp. 249-256

22. A. Pollier, “Evaluation d’une interface par des ergonomes : diagnostics et stratégies”, Rapport de recherche INRIA no 1391, Février 1991

23. D. Rubine, “The automatic recognition of gesture”, PhD thesis, School of computer Science, Carnegie Mellon University, CMU-CS-91-202, 1991.

24. D. L. Scapin, “Guide ergonomique de conception des interfaces homme-machine”, Rapport Technique INRIA no 77, Octobre 1986

25. A. Sochi & D. Hix, “A study of Computer-Supported User interface Evaluation Using Maximal Repeating Patern Analysis”, in Proceedings of the CHI’91 Conference, ACM Press, pp. 301-305, 1991.

26. M. Turk and A. Pentland, “Eigenfaces for recognition”, Journal of Cognitive Neuroscience, Vol. 3, No. 1, pp. 71-86, 1991.

27. R.M. Young & J. Whittington, “Interim Report on the Instruction Langage”, AMODEUS Project Document: Deliverable D5, ESPRIT Basic Research Action 3066, 11th August 1990