REQUIREMENTS FOR MULTIMODAL WIZARD OF OZ PLATFORMS

Daniel Salber, Joëlle Coutaz

Laboratoire de Génie Informatique, IMAG
B.P. 53 X,
38041 Grenoble Cedex, France
Phone: +33 76 51 44 40, E-mail: salber@imag.fr, joelle@imag.fr

ABSTRACT
The Wizard of Oz (WOz) technique is a mechanism for the experimental evaluation of user interfaces. It allows the observation of a user operating an apparently fully functioning system whose missing services are supplemented by a hidden wizard. From our analysis of existing WOz systems, we observe that this technique has primarily been used to study natural language interfaces. With recent advances in interactive media, multimodal user interfaces are becoming popular but our current understanding on how to design such systems is still primitive. In the absence of generalizable theories and models, the WOz technique is an appropriate approach to the identification of sound design solutions. We show how it can be extended to the analysis of multimodal interfaces and we formulate a set of requirements for multimodal WOz platforms. The Neimo system is presented as an illustration of our early experience in the development of such platforms.

KEYWORDS: Multimodal interaction, Wizard of Oz, Evaluation techniques.

INTRODUCTION

Communication between the user and the computer has been shown to be significantly enhanced when different input media are simultaneously available [17]. There is a high potential for systems allowing the use of combined input media but our knowledge for designing, building, and evaluating such systems is still primitive. In the absence of generalizable theories and models, evaluation techniques and user observation provide ways to improve the design of interactive systems.

In this paper, we focus on the Wizard of Oz (WOz) experimental evaluation technique. We show how this technique can be extended to the analysis of multimodal interfaces and formulate a set of requirements for multimodal WOz platforms. The Neimo system is presented as an illustration of our early experience in the development of such platforms.

EVALUATION TECHNIQUES AND WOZ

Evaluation techniques

Evaluation techniques are twofold. They may be based on predictions from theory or they may rely on experimental data.

Predictive models and techniques do not require any system implementation nor do they need effective users. Examples of such techniques include GOMS [6] and its related models such as CCT [19], theory-based models such as ICS [4], KRI [21], and the "cognitive walkthrough" method [20]. Their main benefit is to allow user interface evaluation at an early stage of the development process. However, such predictions rely on theoretical hypotheses, not on real data. Thus, predictive models and techniques may lack precision or they may be limited in scope when the case study (e.g., multimodal interaction) is not supported by the underlying theory. In addition, the setting and interpretation of such models are sometimes complex and as time consuming as an effective implementation.

At the opposite, experimental techniques deal with real data observed from real users accomplishing real tasks, and operating a physical artefact. Artefacts may be paper scenarios, mock-ups, computer system prototypes, or Wizard of Oz systems. A WOz system allows the observation of a user (i.e., a subject) operating an apparently fully functioning system whose missing services are supplemented by a hidden wizard. The subject is not aware of the presence of the wizard and is led to believe that the computer system is fully operational. The wizard may observe the subject by any means such as a dedicated computer system connected to the observed system over a network, an internal video circuit, or a combination of both. When the subject invokes a function that is not available in the observed system, the wizard simulates the effect of the function. Through the observation of subjects' behavior, designers can identify subjects' needs when accomplishing a particular set of relevant tasks and they can evaluate the particular interface used to accomplish the tasks.

Existing WOz Systems and Lessons Learned

Existing WOz Systems. Most of existing WOz systems have been primarily developed to study the usage of natural languages for retrieval information systems. Telephone information services such as telephone directories, flight or train information and reservation services, have been an interesting field for experiments [12, 23]. The experimental set-up is quite simple: the wizard answers phone calls and pretends callers are talking to an automatic information system. In order to give callers the illusion that they are actually talking to a computer, the wizard's voice is filtered through a distortion system (e.g., a vocoder) that gives the voice a robotic flavour. Questions and answers are tape-recorded for later manual transcription and analysis. Other case

studies involve databases or advisory systems interrogation [15, 18, 27] as well as dialogues with expert systems [9, 22]. All of them aim at collecting vocabulary corpus in order to tune and augment the robustness of spoken or written natural language recognizers. To our knowledge, the platform described in [8] is the only documented attempt that supports the observation of graphical direct manipulation combined to natural language.

Although limited in scope, results from WOz experiments already form an interesting body of knowledge about subjects, wizards, and evaluators.

Lessons Learned about Subjects. When subjects believe they are talking to a computer, the complexity of the language is much lower than in natural human to human communication [9, 10, 23]. The following experiment described in [5] confirms this result: some of the subjects were told that the computer system was simulated (i.e., they knew they were interacting with a human being) and other subjects were not aware of the simulation. The wizard did not know whether the current subject was aware or not of the "man behind the curtain"; so, the wizard behaved in a consistent way for both types of subjects. The results showed remarkable differences in the language used by each category of subjects.

Lessons Learned about Wizards. A second interesting result from WOz experiments is that wizards' tasks, although apparently simple, are cognitively expensive. The realism of the apparatus requires wizard's actions to be consistent in content, style, and pace. In particular,

- in similar contexts, a given command from the subject must trigger the same behavior from the wizard,
- response time must comply to the subject's expectation: if the wizard is too slow at reacting, the subject may avoid using simulated functions or may believe that the system is overloaded.

In summary, wizards cannot afford improvisation. To achieve an acceptable consistent behavior, wizards must be trained at well defined tasks, and must be assisted by powerful tools. To this end, some WOz systems include limited but useful mechanisms such as a set of predefined replies or menus containing prestored parts of answers [7].

In order to alleviate cognitive overload, recent studies suggest a two-wizard configuration where one wizard is specialized in I/O whereas the second one performs task level processing [1]. The I/O wizard acquires subject's requests and transmits simulated answers; the task wizard interprets the requests translated by the I/O wizard and generates the answers to be formulated by the I/O wizard. This collaborative task sharing is more likely to guarantee consistency. It doesn't add noticeably to the response time provided that the wizards are appropriately trained. Another experiment using a two-wizard configuration has proven to be successful [11].

Tools for Analysis. The third lesson learned from WOz experiments is the need for analysis tools that efficiently support evaluators'tasks. Manual analysis of a large body of data requires expertise and is time-consuming. When collected data is recorded electronically, the analysis can be partly automated. For example, in natural language dialogues, the use of pronouns can be identified [7] or, as in the Wizard's Apprentice, the dialogue structure can be analysed [9]. Although the apparatus described in [16] is not a WOz platform, it provides a precise searchable record of subject's behavior: events related to graphical menus and buttons are captured and aggregated into higher abstractions, then linked through timestamps to videotape frames. Although this system tracks direct manipulation actions, the salient features of subject behavior must be extracted by hand.

In summary, most of existing WOz systems have been developed on a case-per-case basis and support the observation of one modality only. Similarly, automated analysis tools are limited in scope and are rarely integrated into the WOz platform from the start. Although [8] is intended for the combined use of graphics and natural language, the system is limited by technical constraints and supports one wizard only. There has been no attempt to produce a generic, multiwizard, reusable WOz platform that would make possible the observation and analysis of multimodal interaction. With multimodal interaction, new problems arise with specific requirements.

REQUIREMENTS FOR MULTIMODAL WOZ PLATFORMS

Requirements for multimodal WOz platforms can be defined from the users' perspective (i.e., the wizards and the evaluators) as well as from the software point of view. Before discussing these issues, we must introduce the notion of multimodal interaction.

Multimodal Interaction

Multimodal interaction is characterized by the possibility for a user to exploit multiple communication channels. It requires the system to represent and manipulate information at multiple levels of abstraction including the dynamic maintenance of meaning in the task domain [3, 14].

As discussed in [3, 14], multimodal interaction may be characterized along two dimensions:

- the temporal use of modalities (i.e., communication channels) which can be sequential or concurrent,
- the level of fusion between modalities: modalities may be used in a combined way (e.g., the "put that there" paradigm) or used independently.

The usage of multiple modalities along the fusion and temporal constraints have direct implications on software design as well as on users. The wizard is one such user.

Requirements from the Wizards' Perspective

From the wizard's perspective, multimodality increases the complexity of the task as well as information bandwidth.

Task Complexity. In a traditional WOz system, the wizard supplements a single missing service such as the interpretation of typed natural language. Within an interaction where the subject can use multiple modalities concurrently, the wizard must simulate more complex functions such as the synergistic combination of modalities. Thus, the wizard must have available very efficient means for observing the subject. As shown by early experiments in face-to-face communication, internal video circuits are not accurate enough to track everyone of the subject's actions [13]. Clearly, the wizard must have a dedicated computer of his own. The computer is less prone to tracking failure than the human wizard and, as discussed above, it can provide ready for use answers or encapsulate elementary subject actions into higher abstractions such as commands.

Information Bandwidth. In a multimodal system, the subject has many ways for providing inputs. The same semantic content may take multiple forms. Thus, the amount of information to be conveyed to the wizard is likely to be much more important than in a traditional system. The required input processing bandwidth may become too high for a single wizard and may result in inconsistent behavior with respect to content, form, and response time. Thus a multiwizard configuration seems a reasonable way to go.

Multiwizard Configuration. With a multiwizard configuration, the difficulty is to organize the collaboration between the wizards in a way that guarantees consistent behavior. Ideally, the workload should be equally distributed among the wizards. However, the workload is difficult to estimate a priori since it heavily depends on the subject's behavior. In these conditions, it should be possible for the wizards to dynamically change their respective roles. In addition, we feel the need for a specialized wizard who would act as a supervisor. This superwizard would not accomplish any regular wizard task but would regulate wizards' behavior, monitor the session, and make proper decisions in case of system malfunction.

Task allocation among wizards has not been investigated or reported in the literature. Although we know that it depends on the peculiarities of the experiment, we are working on a set of general features that could help in supporting task distribution. For example, we have found useful to structure a typical wizards' task in three steps: acquisition, interpretation, and formulation.

- acquisition consists of analyzing the message issued by the subject, and of extracting the relevant information in the problem space,
- interpretation corresponds to the task level problem solving in the domain,
- formulation covers the form and emission of an answer to the subject.

Modelling a wizard's task as a three step process suggests the allocation of a combination of steps to different wizards. The I/O wizard and the task wizard discussed above are one such possibility. However, when multiple modalities have to be combined, the fusion process requires an extra acquisition step that complements the acquisition over each communication channel. In this case, it might be wise to define a specialized wizard for performing the fusion of multiple modalities.

Requirements from the System Perspective

The overall property of a platform for multimodal WOz experiments is performance and flexibility. In turn, flexibility covers configuration issues as well as software communication protocol.

Performance. As discussed above, the wizard is an important element in response time conformance. However, if the system is slow, the wizard cannot compensate the flaw. Thus the system has to provide the effective foundations for efficiency.

Configuration Flexibility. Configuration flexibility is the ability of the system to support a variable number of wizards as well as a variety of I/O devices. Multiple wizards, each equipped with a dedicated computer, are recommended when task overload must be shared. Consequently, a WOz platform should be envisioned as a multiprocessor distributed system. In addition, a multimodal WOz system should be flexible enough to accept new interaction devices. The adjunction of a new device should simply result in the implementation of a module that would allow the subject to use the new modality and let the wizards track, interpret and simulate its operation. By doing so, devices that are not yet supported by the software under test could be conveniently simulated by a wizard.

Communication Protocol Flexibility. In an ordinary WOz system, communication between the subject's computer and the wizard's computer is a major issue. In a multimodal WOz system, this point is even more critical for two reasons. First, such a system may require multiple wizards with multiple workstations possibly exchanging information with each other. Second, a generic multimodal WOz platform requires that the information exchanged over the communication channels must be of any level of abstraction. As an illustration, consider the design of a drawing program where modalities available to the subject are mouse pointing and voice-enabled commands. Designers should be able to perform the following experiments:

- in the very first step, the tested program would implement mouse gestures only, and voice recognition would be performed by wizards. In this experiment, low level events should be transmitted to the wizards' workstation (e.g., mouse movement in terms of "x,y" coordinates along with the replication of the subject's utterances).
- in a further development stage, the program would be enhanced with a speech processing system. Events of a higher level of abstraction such as the recognized sentence and the identification of the selected object, should be sent to the wizards' computer.

This example shows the need for a flexible communication protocol, allowing communication between the subject and the wizard systems, as well as communication between the wizards. This protocol should also be flexible enough to allow easy transmission of any level of information, from low-level user events up to high-level information of any kind.

Requirements for Evaluators: Analysis Tools

A long-term goal for multimodal WOz systems is to include computerized automatic evaluation tools. This objective, however, still requires theoretical research. In the short run, it is realistic to provide an integrated way to retrieve and manipulate data collected during the experiment. With this perspective in view, we propose the following requirements:

- data should be collected on the same physical support; a support that allows recording from different medias and that offers great flexibility is a computer file. Since we intend to have a fully-computerized WOz system, integration of all history data into computer files is a natural choice.
- data should be correctly and automatically synchronized to allow real time replay and analysis.
- automatic processing should be used whenever possible; for example, if speech is recorded in history data, detection of syllables should be performed automatically while analyzing the history data.
- a powerful filtering mechanism should be available to provide an efficient way to reduce the amount of data to be analyzed, or to allow focussing on a particular aspect of the experiment. For example, in the case of the voice-controlled drawing program, one should be able to focus on the choice of modalities used for drawing new objects; the analysis tool would locate the instants of the session where the subject decided to draw a new object and would display the modalities used.
- statistical computations on history data should also be integrated. For example, one would be able to detect and study “maximal repeating patterns” from the subject’s behavior [25].

NEIMO, A MULTIMODAL WOZ PLATFORM

Based on the requirements described above, we have designed and developed Neimo, a WOz platform to observe and analyze multimodal interaction [2]. This section presents the system configuration used in Neimo to conduct our current experiments as well as the software organization and kernel services provided to support WOz requirements.

System Configuration

As shown in Figure 1, the hardware configuration of Neimo consists of a workstation for the subject, and several wizard workstations. Modalities currently supported are: 2D-pen gesture, speech, and facial expressions [26]:

- pen gesture recognition is based on Rubine’s work [24]. A "pen wizard" controls the correctness of the recognition and supplies interpretation for ambiguous gestures.
- speech recognition and interpretation is simulated by a dedicated "speech wizard". Subject's utterances are digitized and saved in history files using a sound digitizing board.
- a CCD camera is focussed on the subject’s face and connected to a video acquisition board. The "face wizard" can observe the subject’s face digitized in real time and interprets facial expressions. The result of the interpretation may be recorded and/or transmitted to other wizards to help them in their task.

Our first test application is the simulation of an "electronic paper calculator": the subject draws an arithmetic operation on the tablet and the result is automatically computed and displayed. The pen wizard checks that the drawn numbers have been accurately recognized by the system. To do this, the pen wizard can observe the drawn figures in a window of his own, click a button in a dialog box to validate the result, or, if recognition has failed, type in the correct value. The face wizard detects specific facial expressions that will be used as a redundant or complementary information for further evaluation. Simple commands such as "new blank page" or “erase all”, can be expressed with voice and are simulated by the speech wizard.

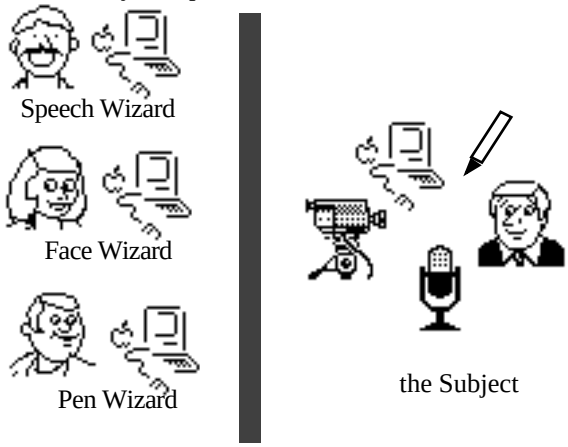

Figure 1. A typical Neimo hardware configuration.

Software Organization

As shown in Figure 2, a Neimo configuration consists of a Neimo kernel and of several client programs. In turn, the Neimo kernel includes a communication kernel, NeimoCom, and specific libraries that allow client programs to exchange information through NeimoCom. Clients are the application under test and the wizard software. We claim that this organization is necessary to support the flexibility and performance requirements presented above.

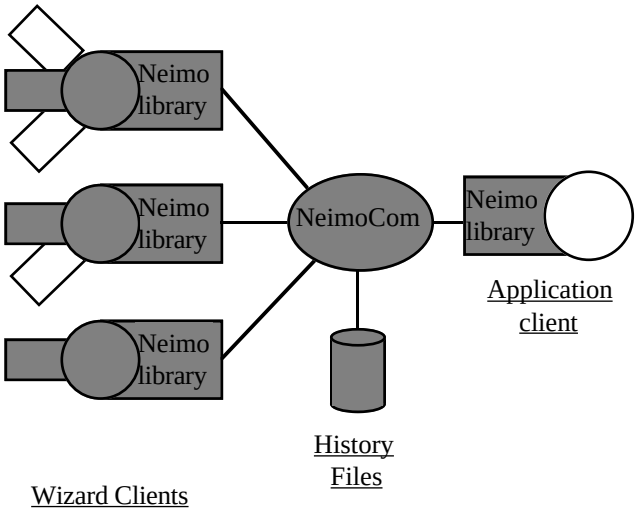

Figure 2. The software components of the Neimo platform. Dimmed areas denote the common services provided by Neimo; white areas represent specific components within client programs.

NeimoCom. NeimoCom is the communication control center of the platform. All of the messages exchanged between client programs transit through this dispatcher. This approach has multiple advantages. First, NeimoCom can be used to determine the time-base for the whole system. Consequently, messages are time-stamped in a consistent way. Second, it can record pertinent information in history files. Third, it can handle network-related errors and failures. Finally, it can arbitrate possible conflicts between clients and manage the dynamicity of the whole system in a consistent way. In particular, it can dynamically support any number of wizard clients. This feature allows human wizards to come into play when needed during an experiment.

Neimo libraries. Neimo libraries provide client programs with access to the communication services. Communication is handled through messages. A message is a time-stamped container. It has a sender client and possibly multiple receivers. Its type, i.e., the format of the content, is client-defined. More precisely, Neimo supports a set of standard message types for utility functions and usual modalities, but application-specific or modality-specific messages can also be defined. This mechanism guarantees the flexibility of the platform and allows for future evolution.

On the application client side, the library allows the application under test to send and receive messages, as well as saving information in history files. Since message types are client-defined, these can be of any level of abstraction. On the wizards side, the library provides the following services:

- reception of messages from the application under test. A dynamic subscription mechanism allows the wizard clients to express their interest in specific categories of messages. For example, the speech wizard client would subscribe to the speech message type in order to receive the subject's utterances. Note that the ability for wizard clients to dynamically subscribe to message types allows human wizards to change roles during the experiment.
- emission of messages to the application under test. These messages may then be interpreted by the application client as operations to be performed. This facility allows human wizards to remotely control the observed application.
- saving interesting information into history files, e.g., the result of the interpretation of facial expressions, or wizards' actions if wizards' behavior needs to be analyzed.
- exchange of messages between wizards. Up to now, only a Unix talk like exchange tool has been designed to allow the communication of short typed messages. We plan to provide wizards with the ability to exchange audio and video messages.

The wizard clients. As shown in Figure 2, each human wizard is attached to a wizard client program that suits his specific needs. Specific modules can be plugged onto a wizard program to provide specific services. Every message type sent by the application program under test has a corresponding module in one of the wizards' programs. For example, the speech wizard program includes a speech module that processes speech messages; it allows the wizard to hear the subject's utterances and perform a set of possible actions in response to a spoken command. A wizard program can accept many plugged-in modules that allow the human wizard to play multiple roles at once or successively during the session.

Analysis of History Data

As emphasized above, the analysis of recorded data is a heavy task. The Neimo platform provides fundamental services from which sophisticated analysis tools may be developed. These services come in the form of a library, the "history library", that allows the manipulation of history files.

History files collect and synchronize messages recorded during the Neimo experiments. Since message types are client-defined, history files must be processed by the clients that are aware of these types. The history library allows client programs to read and perform selective browsing operations within history files. Selective browsing is achieved using the "view" mechanism. A view acts as a filter on history files. It is defined by a temporal window of interest (i.e., a start and stop date), an origin that specifies the source of the recorded messages (i.e., wizard or subject), and a list of the message types of interest (e.g., face and speech). The view mechanism allows for the dynamic specification of the region of interest discarding irrelevant information.

A Few Technicalities

The Neimo system runs on Apple Macintosh Quadras connected through an Ethernet network. It is written in C and C++, and is developed with MPW (Macintosh Programmer's Workshop) and MacApp [28], an object-oriented application framework. Communication relies on Apple Events [29], a powerful high-level inter-application communication protocol introduced by Apple with System 7. History files use a QuickTime-based format; QuickTime [30] is an extension provided by Apple for manipulating and storing time-based data from different medias. The wizard clients take advantage of Dinker [31], a dynamic linking facility for applications written with MacApp. Implementation of the Neimo system on other platforms (Sun, NeXT) is currently under study.

CONCLUSION AND PERSPECTIVES

We have defined a set of requirements for multimodal WOz platforms. With Neimo, we have shown how these requirements can be fulfilled within software design. The genericity and extensibility of the Neimo platform constitutes a unique attempt in the domain of the WOz technique. In particular, it is characterized by the following interesting properties:

- messages from different medias are processed and recorded in a uniform way within a common time-base. This integration provides a sound basis for building history files and defining useful operations such as the view filter mechanism;
- message formats are client-defined. This facility allows Neimo experimenters to define the granularity and the level of abstraction of the information to be recorded ranging from low-level events such as mouse clicks, to high-level commands and facial expressions;

- messages are automatically broadcasted and dispatched according to their type and the subscriptions made by the wizard clients;
- wizard clients are provided with flexible services such as the dynamic subscription to message types, the dynamic connection to the kernel, and the possibility to record human wizards' actions. This flexibility makes it possible to test multiple configurations including a varying number of wizards possibly changing roles dynamically.

Two major areas however need further investigation. First, the organization of the wizards' tasks requires a lot of testing and experiments. The design of the Neimo platform is intended to test multiple configurations from which we plan to identify a set of rules that would help configurate wizards' operations most efficiently. Second, the tools for analyzing history files must be enhanced in order to reach higher levels of abstraction with full automation (e.g., detection of subject's intention, and usability problems). As a first step in this direction, we are currently developing a critique which, from recorded subject's actions, detects behavioral patterns and deviations from a data flow-oriented task model [3].

ACKNOWLEDGEMENTS

This work has been supported by project ESPRIT BR 7040 AMODEUS as well as by the Programme Cogniscience of the CNRS, France. We would like to thank B. Noz and G. Ambone (LGI) for their participation in the design and development of the Neimo platform. J. Caelen (ICP, Grenoble), R. Amalberti and C. Valot (CERMA), J-M Francony and E. Kuijpers (CRISS, Grenoble) were kind enough to share their skills and experiences.

REFERENCES

1. R. Amalberti and C. Valot: "Le Magicien d'Oz", CERMA, 1992.
2. G. Ambone, B. Noz, D. Salber: "Projet Neimo, Spécifications Externes", internal technical report, équipe IHM, LGI-IMAG, 1992.
3. S. Balbo, J. Coutaz and D. Salber: "Towards Automatic Evaluation of Multimodal User Interfaces", Workshop on Intelligent User Interfaces '93.
4. 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
5. S.E. Brennan: "Conversation as Direct Manipulation: An Iconoclastic View", in The Art of Human-Computer Interface Design, B. Laurel ed., 1990.
6. S.K. Card, T.P. Moran and A. Newell: "The Psychology of Human-Computer Interaction", Lawrence Erlbaum Associates, 1983.
7. N. Dahlbäck and A. Jönsson: "Empirical studies of discourse representations for natural language interfaces", Fourth Conference of the European Chapter of the ACL, proceedings 291-8 1989.
8. N. Dahlbäck, A. Jönsson and L. Ahrenberg: "Wizard of Oz studies — why and how", Third Conference on Applied Natural Language Processing, Trento, Italy, 31 march—3 april 1992.
9. D. Diaper: "The Wizard's Apprentice: A Program to Help Analyse Natural Language Dialogues", People and Computers V, proceedings of the 5th Conference of the British Computer Society, 1989.
10. P. Falzon: "Ergonomie Cognitive du Dialogue", Presses Universitaires Grenobloises, 1989.
11. J.-M. Francony, E. Kuijpers and Y. Polity: "Towards a methodology for Wizard of Oz experiments", Third Conference on Applied Natural Language Processing, Trento, Italy, 31 march—3 april 1992.
12. N. Fraser, N. Gilbert and C. McDermid: "The Value of Simulation Data", Third Conference on Applied Natural Language Processing, Trento, Italy, 31 march—3 april 1992.
13. R.S. Fish, R.E. Kraut, R.W. Root: "Evaluating Video as a Technology for Informal Communication", in the CHI'92 Conference Proceedings, ACM Press Publ., 1992, pp. 37-48
14. A. Gourdol, L. Nigay and D. Salber: "Multimodal Systems: Aspects of Event Fusion and a Taxonomy", IFIP Congress '92 Proceedings, Madrid.
15. R. Guindon and K. Schuldberg: "Grammatical and Ungrammatical Structures in User-Adviser Dialogues; Evidence For Sufficiency of Restricted Languages in Natural Language Interfaces to Advisory Systems", proceedings of the 25th Annual Meeting of the Association for Computational Linguistics.
16. 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
17. A.G. Hauptmann: "Speech and Gesture for Graphic Image Manipulation", CHI '89 Proceedings, pp. 241-245.
18. A. Jönsson and N. Dahlbäck: "Talking to A Computer is Not Like Talking To Your Best Friend", Proceedings of the Scandinavian Conference on Artificial Intelligence '88, pp. 53-68.
19. 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
20. C. Lewis, P. Polson, C. Wharton, J. Rieman: "Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces", CHI '90 Proceedings, pp. 235-241.
21. J. Löwgren and T.Nordqvist: "A Knowledge-Based Tool for User Interface Evaluation and its Integration in a UIMS", Human-Computer Interaction—INTERACT '90, pp. 395-400.
22. Y. Polity, J.-M. Francony, R. Palermiti, P. Falzon, S. Kazma: "Recueil de dialogues homme-machine en langue naturelle écrite", Les Cahiers du Criss, n° 17, 1990.
23. M. Richards & K. Underwood: "How Should People and Computers Speak to Each Other", Proceedings of Interact '84 — First IFIP Conference on Human-Computer Interaction, pp. 268-273.
24. D. Rubine: "The Automatic Recognition of Gesture", Ph.D Thesis, School of Computer

Science, Carnegie-Mellon University, CMU-CS-91-202, 1991.
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. S. Whittaker and P. Stenton: "User Studies and the Design of Natural Language Systems", Fourth Conference of the European Chapter of the ACL, proceedings 291-8 1989.
28. MacApp Programmer's Manual, Apple Computer Inc.
29. The Event Manager, in New and Improved Inside Macintosh, Apple Computer Inc.
30. QuickTime Reference Guide, QuickTime CD, Apple Computer Inc.
31. Dinker User's Manual, E.T.O. #7 CD, Apple Computer Inc.