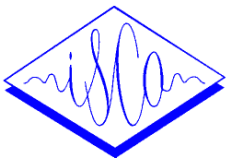


2nd International Conference on
Spoken Language Processing (ICSLP 92)
Banff, Alberta, Canada
October 12-16, 1992

SPEECH INTERFACE FOR A MAN-MACHINE DIALOG WITH THE UNIX OPERATING SYSTEM

P.Lefebvre[+], *F. Poirier*[*], *G. Duncan*[+]

[+]Siemens-Nixdorf Information Systems
[*]14, Av. des Béguines, 95802 Cergy, France
[*]Télécom Paris - Dep SIG - CNRS URA 820
46, rue Barrault, 75634 Paris, France

ABSTRACT

Natural language and graphic interface techniques go some way towards facilitating competent, reliable use of the Unix operating system. Such methods, however, remain insufficient particularly where newcomers to the use of Unix are concerned. This paper presents a novel, multimodal approach to interaction with the Unix operating system. The main aim of this type of interface is to provide for an optimum combination of various communication modalities (voice, keyboard, graphics) within an integrated architecture, so as to promote a more natural man-machine dialog. Here, the role of each mode is defined together with their complementarity within a multimodal architecture. The multimodal event analyser is then presented, together with preferred options in respect of the communication modalities exercised for system response.

1. INTRODUCTION

This paper presents a multimodal dialog architecture whose main aim is to facilitate use of Unix for newcomers to this operating system. With system dialog in its native, command-line format, acquiral of an adequate level of proficiency is difficult to achieve for occasional system users, and highly problematic for completely novice computer system users. Nevertheless, Unix has become a standard operating system, and current market trends strongly predict that its use will become ever more widespread in the future. In this light, methods to enhance the man-Unix dialog are therefore of significant interest.

Previous research [1] [2] [3] has underlined the weaknesses of current man-Unix dialog interfaces, and have led to two approaches to improve the quality of dialog. The first, termed a "direct manipulation interface" [4] essentially exploits the concept of metaphors. The second approach involves use of natural language for dialoguing with the system [5] so as to approximate as closely as possible a man-man dialog.

This paper presents an integrated approach, permitting interaction with the system via direct manipulation, but also through the use of natural language by voice or keyboard input. This strategy of integrating communication via both graphics and natural language modalities offers significant advantages in the design of the Unix operating system interface.

The first section of this paper discusses the role attributed to each communication modality in the multimodal interface: that is, graphics, voice, and keyboard modes. Particular attention is given to the new perspectives offered by the synergistic use of several concomitant modalities in a man-machine dialog. This is followed by an description of the software architecture, including the interpretation mechanism for event handling in this multimodal man-Unix interface. Finally, rule based criteria are explored, which govern the selection of appropriate multimodal response employed by the system in various circumstances.

2. THE AIM OF MODALITIES

2.1 Graphics to paliate insufficiencies of voice

To represent most concepts of the Unix system (e.g. files, processes, users), an icon-based approach has been employed within a multi-window environment. With this kind of environment, interaction between user and system is made possible through the use of graphical objects such as menus, dialog boxes, and *drag and drop* operations.

An advantage of visual representation in this way is to facilitate understanding of the Unix system. Moreover, it is in fact quite necessary for speech interaction. This is since it is impossible to represent verbally all the elements of the Unix operating system, e.g. files and processes, because the vocabulary size of the speech system is limited. In addition, the name of such elements is highly variable, depending on either the Unix system or on specific choices made by users. Therefore, the variability of these elements precludes their prescence in any standard lexicon relating to system interaction.

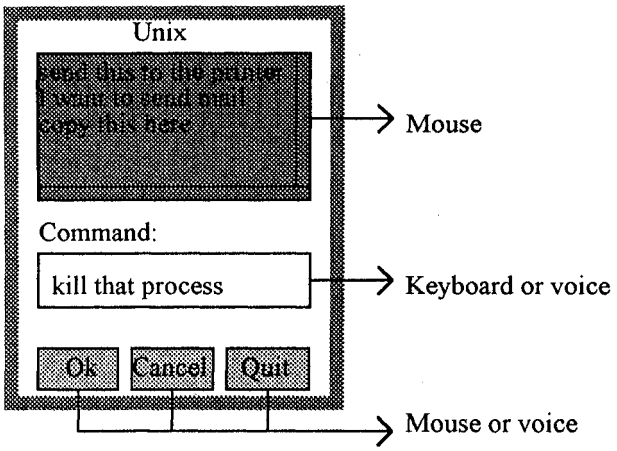
One consideration in the design of such an interface is the modalities available to the user during an interaction. Here, colour has been used to represent and uniquely identify modalities. Figure 1 shows the *multimodal communication* box which allows the user to input requests and commands to the system by voice or keyboard. This box allows the user to view at any instant a complete history of the interaction. Such a box is composed of various subsections which can be

10.21437/ICSLP.1992-381

selected by one or several modalities. For example, the pushbuttons *ok, cancel, quit* can be selected either by mouse click, or by speaking the name inscribed on the button. Listing of previous commands used during the interaction can however only be achieved through using mouse and scrollbar in the appropriate dialog box.

Dialog boxes configured in this way offer enhanced flexibility for a novice user, since he/she can then choose in certain cases according to his/her personal or situational preference for one modality over another.

Figure 1 - The multimodal communication box



2.2 Voice in a multimodal environment

The speech recognition board used in the system is the Vecsys Datavox. This product is oriented towards connected word recognition, in single-speaker mode. Training and recognition algorithms use Dynamic Time Warping, and the system works under the MS-DOS operating system. Events from the speech input system are therefore signaled to the target Unix-based user platform via the serial port. The request appears finally in the multimodal communication box (displayed using the X-Window graphic interface).

In such a multimodal architecture, each interaction is based on the following model:

Action - Objects - Parameters

Note that the order of appearance of each of these elements neither imposes nor implies the same order of production in the request issued to the system by the user.

The *action* is one of the Unix system commands. The available set of voice commands (of around 50 words) has here been defined only for the *user* level of the Unix operating system. The *administration* level has not been considered here as it generally is only of concern to users who are already expert with the Unix operating system.

Objects , represented by graphical icons, are the base entities of the Unix system (e.g processes, users) on which actions are to be carried out. Graphics-based metaphors here offer appropriate metaphors for illustrating some important concepts of an operating system.

The number of equivalent Unix parameters input by voice is limited to two per command. This limitation is due difficulties which have been observed in pronouncing long sentences (more than ten words) without interruption or hesitation. Of course, certain system requests require specification of more than two parameters. A natural solution to this problem is that not all Unix commands in fact require use of a multimodal dialog box, and due account is taken of this fact in the box generation process. Such boxes are implemented here by using the UIL (User Interface Language) description language under the Motif graphics environment. They allow completion of a request issued by voice with dialog boxes including parameters which characterize a Unix command. The user can then select via mouse or voice input those parameters which complete his basic command. Figure 2 demonstrates a multimodal dialog box for completing the voice request:

"I want to send mail".

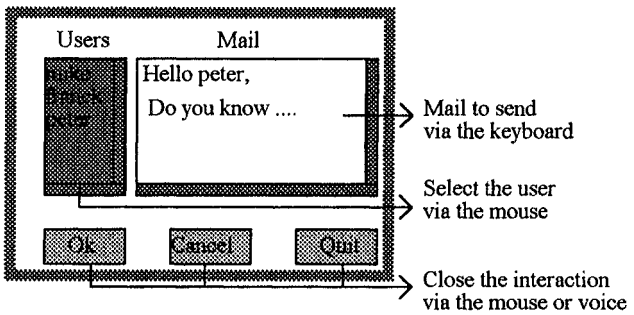


Figure 2 - Send mail to a user

Voice input is therefore used principally to act on graphical objects displayed by the Unix interface system. Handling these objects, however, demands the ability to process deixis in expressions such as:

"send *this* to the printer"
"kill *these*"
"copy *this* into the directory"

where the deictic designators *this* and *these* - which must be recognized within the voiced expression - refer to objects selected with the mouse.

2.3 The role of keyboard input

Use of this modality is mandatory for certain Unix

system tasks. For instance, in response to the request:

"I want to create a directory"

a dialog box will appear and will ask to the user to type in an appropriate name for the directory.

In this system, it is also possible at any time to substitute keyboard for voice input in interacting with the Unix system. The advantage in this case is that limitations on vocabulary size and syntactic structure are relieved to some extent. .

Nevertheless, for short requests, voice is a more rapid and natural input modality compared to keyboard entry [6].

3. INTERPRETATION OF MULTIMODAL EVENTS

3.1 Multimodal architecture

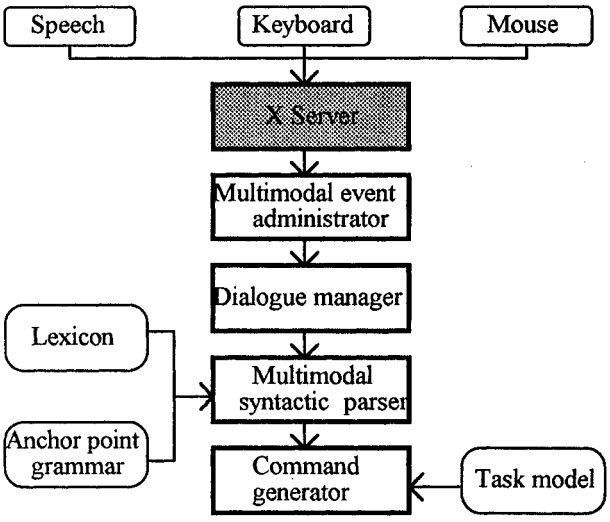


Figure 3 - Multimodal architecture for the Unix system

Figure 3 presents the general architecture of the system, which is characterised by the following modules:

- **X server:** inserts into an event queue events from keyboard, mouse and speech input system via the serial port.
- **Multimodal event administrator:** labels the various objects, using the event identifiers transmitted by the server, according to their source (keyboard, speech, mouse) and their time of arrival.
- **Dialog manager:** constructs a unified representation of the user request, independant of the communication modality by which each object was signalled.
- **Multimodal syntactic-semantic parser:** finds and interprets anchor points within the complete request structure, using the application-specific lexicon in conjunction with a context-free grammar.
- **Command generator:** generates the Unix command within an X-Window, using the anchor points in conjunction with the task model.

3.2 Multimodal dialog characteristics

Studies have shown that, in a specific task context, users tend to employ a highly specialized vocabulary, with a random free-form syntax characterised by short phrases [7].

There are four important characteristics of multimodal dialog with Unix to be considered:

- **optionality:** in a request, certain words may be optional.
- **specialization:** the user employs a specialized vocabulary directly related to the Unix task (e.g. shell, kill, copy).
- **asynchronism:** the various communication modalities can work in parallel, and the dialog administrator is tasked with unifying the various elements of the user request. There can be great flexibility in their order of production.
- **deixis:** certain words (e.g this, that, these, it) refer to objects selected by the mouse.

3.3 Anchor point grammar

The syntactic parser employed in the system presented here implements an anchor point grammar. Although for the present application, the underlying anchor point grammar model has been simplified. The grammar used is a context-free grammar, and associated rules have the following form:

$I_1 \rightarrow X_1\ [X_2] \ldots X_n$ where I_1 is an anchor point
X_i is an element in the lexicon
or an anchor point
$[X_i]$ is an optional element

As an illustration of the formalism, consider the following rules:

Kill_process --> Kill [Process]
[With_extreme_prejudice]
Kill --> stop OR kill OR can you kill
Process --> [the] process OR Deictic_object
Deictic_object --> that OR this OR these
With_extreme_prejudice -> with no pity

From these rules, the following requests are considered as legal:

"kill the process"
"kill that"
"with no pity kill these"

3.4 The task model

The aim of this module is to generate the Unix command with its options and arguments. The response will be either textual or graphical.

The command and its options are deduced from anchor points found in the request, and associated arguments correspond to variable elements in the request transmitted by the graphical or keyboard modality. For instance:

"kill that with no pity"

anchor points: Kill, With_No_Pity, Deictic_objec
request: kill -9 *141* (141 is the value returned by the graphical modality)

Some requests are ambiguous both at the semantic level and with regard to the appropriate action which should consequently result. For example, in the request:

"destroy *that*"

"that" can represent either an ordinary file or a directory. Therefore, script shells have here been used to automatically test arguments and possible commands before generation of the final command, and hence remove ambiguity. Failure to arrive at a conclusion will initiate a dialog with the user so as to disambiguate the request.

3.5 Multimodal system response

The system uses two possibilities to generate the Unix command: textual or graphical. Criteria used are the following:

- Metaphoric approach to the Unix command
- Context of the interaction
- User preferences
- Complementarity of modalities

The first of these uses a possible transposition of computer objects into real-world objects. For instance, the following voice requests:

"What's the time"

"call the calculator"

generate respectively the graphical representations of a clock showing the current time, and a calculator.

The interaction context determines the choice of modality. If the user is in the file system context, and he asks:

"return to the parent directory"

the system must destack and close the window representing the file structure in the current subdirectory.

The third criterion allows acccount to be taken of user choices. Preference can be explicitly indicated in the user request for either textual or graphical visualization, e.g:

"I want to see graphically the free space on disk"

Lastly, complementarity of modalities allows for ease of handling of complex requests such as:

"I want to find a file"

A dialog box will appear allowing specification of search criteria so as to complete the request.

4. CONCLUSIONS

The paper has presented a novel multimodal approach to the design of a user-Unix interface. The architecture detailed here offers significant enhancements, over the use of either graphics-based or natural language modalities used in isolation, by exploiting the complementarity of each for various task situations.

Further work is currently under way to incorporate a user model, so as to automatically adapt system response to correspond to the level of expertise of individual users. Initial trials indicate that this feature further enhances the suitability and value of a multimodal approach to the user-Unix interface.

Acknowledgements

The authors and their respective establishments wish to extend their thanks to the Association Nationale pour la Recherche Technique, France, for their support of this project under CIFRE contract no. 146/89.

References

[1] J. Bradford et al, "What kind of errors do Unix users make?", Human-Computer Interaction - Interact '90.

[2] L. Coventry, "Some effects of cognitive style on learning Unix", Int. Journal of Man-Machine Studies, pp349-365, 1989.

[3] S. W. Draper, "The nature of expertise in Unix", Interact, pp 182-186, 1984.

[4] M. Beaudouin-Lafon et al, "Iconic shells for multitasking workstations", Proc. ACM Symposium of small systems, May 1988.

[5] R. Wilensky et al, "Talking to Unix in English: an overview of UC", Com. of the ACM, June 1984, v.27(6), 1984.

[6] D. Jones et al, "Design guidelines for speech recognition interfaces", Applied Ergonomics, 1989.

[7] P. Falzon, "Ergonomie cognitive du dialogue", Presse Universitaire de Grenoble, 1989.