Multimodal Systems: Aspects of Events Fusion and a Taxonomy

Arno P.J. Gourdol, Laurence M.B. Nigay, Daniel Salber and Joëlle Coutaz

Université Joseph Fourier
Laboratoire de Génie Informatique
BP 53 X — F-38041 Grenoble CEDEX — France
gourdol@imag.fr

Abstract
This paper discusses multimodal systems. We have defined a taxonomy of multimodal interactions and built two example multimodal systems. We consider several levels where the fusion can take place in the software architecture of those systems. We emphasize the need for a common abstract representation to represent input data and commands, and to combine them successfully. This is still an area of exploration.

Keyword Codes: H.1.2; D.2.2; H.5.1
Keywords: Information Systems,User/Machine Systems; Software, Software Engineering, Tools and Techniques; Information Systems, Information Interfaces and Presentation, Multimedia Information Systems

1. INTRODUCTION

Communication between the computer and the user has been shown to be significantly enhanced when different input modes are simultaneously employed [Hauptmann 89]. There is a high potential for systems combining input modes, but our knowledge for designing and building them is still primitive.

This paper will present a definition and classification of such multimodal systems. We will then discuss some possible software architectures for such systems, allowing data fusion.

We have built two examples multimodal systems, VoicePaint and Notebook.

- *VoicePaint* is a colour, pixel-oriented Apple® Macintosh™ drawing programme (*à la* MacPaint®). It handles speech, keyboard and mouse inputs. VoicePaint was developed using a custom application framework, Human Interface Kernel (HIK) [Gourdol 89] and is more completely described in [Gourdol 91].
- The *Notebook* application is a voice enabled, electronic notebook that handles both speech, mouse and keyboard inputs. The application is developed on the NeXT® workstation with NeXT Interface Builder®. The voice interface services are provided by The Carnegie Mellon Spoken Language Shell [Lunatti 91]. More complete references are available in [Nigay 91a].

2. DEFINITIONS AND TAXONOMY

A *modality* is associated with a physical hardware device. Keyboards, mice, microphones, graphics displays, allow the user to communicate with a system using the corresponding modalities (typing, graphic input, speech, graphic output).

A multimodal system is a system that supports many modalities for input and output. For example, a system using a keyboard (typing modality) and a mouse (graphic input modality) is multimodal.

We propose a classification for multimodal systems, based on a continuous classification space defined by two axes:

- *Supported use of the modalities:* values along this axis indicates to what extent different modalities can be used at the same time. We distinguish two main levels: sequential (the modalities must be used one after another) and parallel (modalities may be used simultaneously).
- *Level of interpretation of input data or generation of output data:* values along this axis indicates how a single command or result is issued by combining modalities. Independent commands are created with only one modality. Combined commands are created with several modalities.

To classify a system according to these criteria, we identify a set of its features f_i (e.g. supported commands). Each feature f_i is weighted according to its estimated importance (w_i) and given an estimation of its place along the two previously defined axes, giving a point p_i.

$$f_i = (p_i, w_i)$$

The position γ of the considered system in the classification space is defined by equation (1).

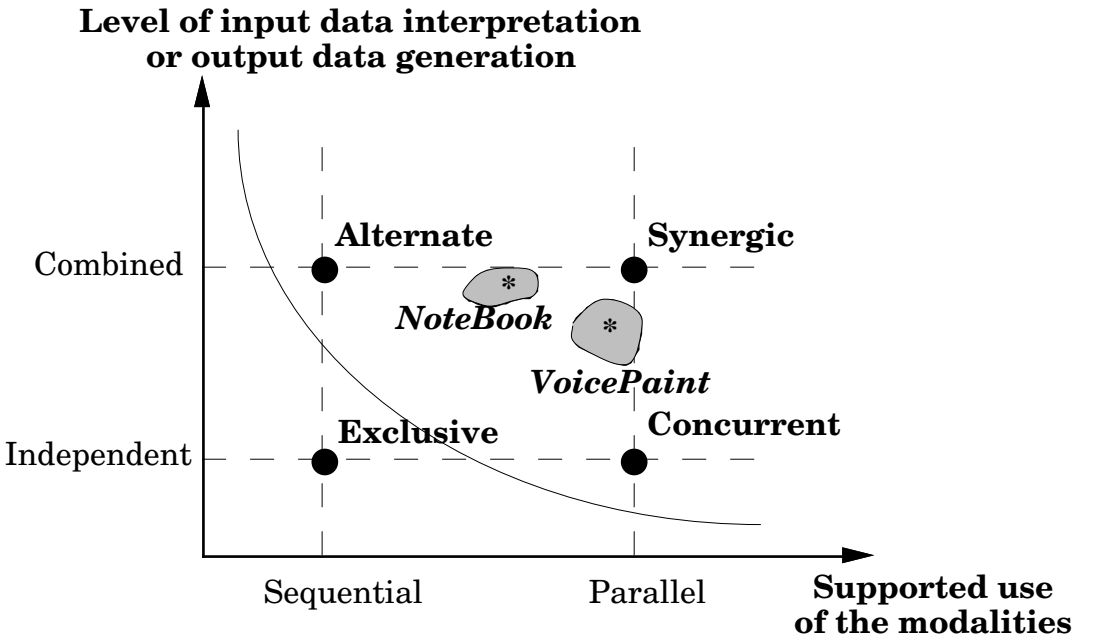
$$\gamma = \frac{1}{\Sigma_w} \times \sum_i p_i \times w_i \qquad \Sigma_w = \sum_i w_i \tag{1}$$

Figure 1. A Taxonomy of Multimodal Interactions and Classification of NoteBook and VoicePaint

Figure 1 shows the diagram and the approximate classification of VoicePaint and Notebook. Systems above the curved line are considered multimodal.

We consider four particular points in the classification space, corresponding to particular values of the axes: exclusive, alternate, concurrent, synergic.

- *Exclusive:* data expressed through different modalities are interpreted independently. For example, a graphical and vocal interface where the user has to switch explicitly between modes offers two different modalities but only one can be used to express a single command.
- *Alternate:* data expressed through different modalities can eventually be combined. For example, the Apple Macintosh Finder™ allows the user to select a document with the mouse and then open it with the keyboard (by typing the



keyboard short-cut Command–O). The two modalities (pointing and typing) are used sequentially and combined into a single action.

- *Concurrent:* if input data are interpreted independently. For example, the VoiceFinder is a system that adds voice input to the Macintosh Finder [Articulate 90]; with this system, the user can issue the voice command "Empty the Trash" while simultaneously opening a document by double-clicking it. The two modalities (speech and mouse) are used in parallel from the user perspective, but they express two independent commands.
- *Synergic:* if input data are combined. For example, in the VoicePaint or NoteBook applications, the user enters a single command using two modalities (speech and mouse) in parallel; the input data are combined to be interpreted by the system.

Systems allowing modalities to be combined seem to be most promising. We classify VoicePaint and Notebook as synergic multimodal applications, because of their closeness to the synergic point.

3. ARCHITECTURE

Both VoicePaint and NoteBook use a similar software architecture model. This model is based on the Seeheim model which makes a clear distinction between the core of the application and its user interface. Some of its components are common with those of the Arch model [UIMS 91] and the PAC-Amodeus model [Nigay 91b]. The major components are briefly described below and illustrated in the Figure 2.

- The *Functional Core* (FC) implements domain specific concepts in a presentation independent way.
- The *Dialogue Controller* (DC) is the keystone of our model. It has the responsibility for task-level sequencing. Each task of the end-user corresponds to a thread of dialogue. This observation suggests a multiagent decomposition: an agent or collection of agents can be associated to each thread of dialogue.
- The *Interface with the Functional Core* (IFC) maps conceptual objects from the Dialogue Controller to domain objects from the Functional Core and back.

- The *Presentation Techniques Component* (PTC) defines two multivalued mapping functions between Presentation Objects and Interaction Objects.

  The PTC describes the presentation (i.e. image of the system as defined by D. Norman [Norman 86]). It implements the perceivable behaviour of the application for outputs as well as input commands. It is at this level of abstraction only that the interaction media is taken into account.
- The *Low Level Interaction Component* (LLIC) denotes the underlying platform, both software and hardware. It supports the physical interaction with the user. It manages (time-stamps and queues) end-user's events (or stimuli) from different media and has the responsibility for their lexical analysis.

  Some of the low-level events are not transmitted to the Presentation Technique Component. Indeed, lexical tasks such as resizing a window, are locally performed by the Low Level Interaction Component. In addition, in the case of spoken-utterances, this component could include mechanisms for confirmation allowing the user to intercept a recognition.

The multiagent architecture of the Dialogue Controller offers an interesting conceptual framework. It provides a pathway to structure an interactive software. Its non sequential, hierarchical and distributed features make it particularly well suited to multimodal interfaces. Each communication mode can be represented with as many event messages sent to the agents as necessary. Their processing can be done in parallel or in synergy.

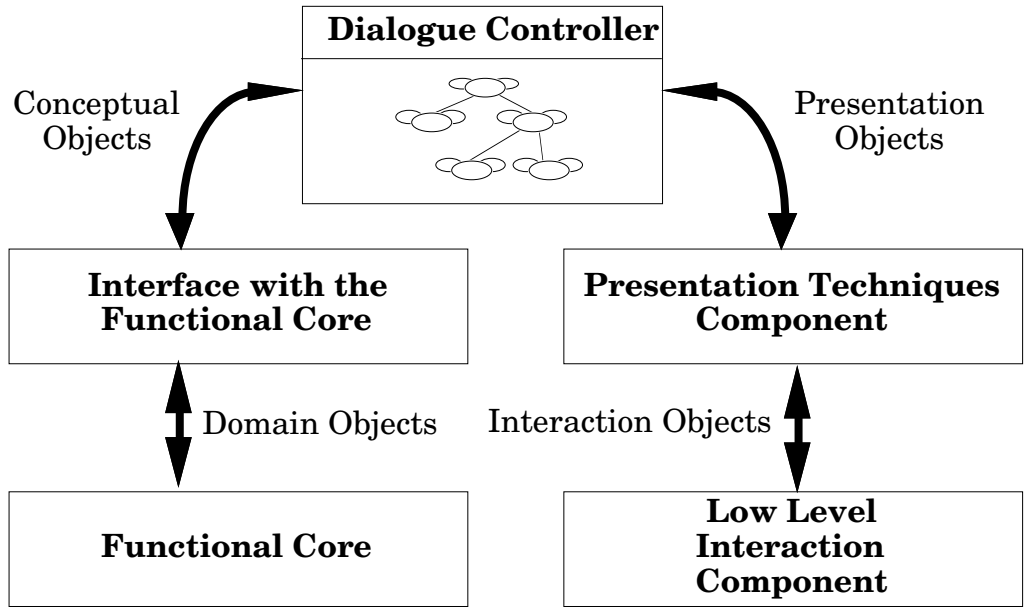


Figure 2. Software architecture

To sum up, let's see how the system reacts to a user action. Input events from the user are going into the Low-Level Interaction Component. They are then transformed in terms of interaction objects, and handled to the Presentation Techniques Component, which may transform them in presentation objects. Those

objects are then processed by the Dialogue Controller. The Controller eventually creates Conceptual Objects and passes them to the Interface with the Functional Core. This "domain adaptor" then transforms them to Domain Objects which are then processed by the Functional Core. Other Domain Objects are created, and follow the reverse path, up-to, if needed the Low-Level Interaction Component. At each stage, objects are processed locally and discarded if possible. That is, the Functional Core will probably never deal with each and every key presses, but rather with string of characters, or still more abstract objects.

Inside this framework, fusion of "event" objects can take place at several levels.

4. LOCALISATION OF FUSION

As non-multimodal systems transform each object (from interaction objects, presentation objects, conceptual objects, domain objects and back), multimodal systems too, perform this transformation. But their task may be more complex — especially for synergic systems. They may have to fuse ("combine") several different objects, eventually of different kind, to go to the next step. We have identified three levels of fusion:

- *Semantic fusion:* The Dialogue Controller is a good place for combining the results of two commands. To fuse these results we use a dedicated agent dispatching the results to other agents. This is the strategy used in Voice Paint.
- *Syntactic fusion:* The hierarchy of Dialogue Controller agents is also a good place to implement multimodal interactions where multimodal events can be combined into a higher level to complete a command. It is often the case that the specification of a command involves user actions distributed over multiple agents. These user's actions could be performed using different modalities. To combine the distributed multimodal actions into a higher abstraction (command level), we use a dedicated agent, father of the agents which receives the elementary actions. This strategy has been used in Notebook.
- *Lexical fusion:* The lexical combination is performed by the Low-Level Interaction Component. For example, depressing the shift key while clicking allows multiple selections.

We have identified three levels here. However, we feel that some fusions may be done at the Presentation Techniques Component level, if realistic feedback or efficiency is a concern. Because we performed the fusion at the Dialogue Controller level in the two systems we built, we are now going to detail how the fusion is done at the Dialogue Controller level.

5. FUSION AT THE DIALOGUE CONTROLLER LEVEL

In our systems, multimodal fusion is implicit. We do not do complex syntax analysis, but let our agent architecture do the work for us.

To do this, we use:

- in VoicePaint, a lightweight multitasking kernel built into HIK, our application framework, to allow the input of several media at the same time,
- in Notebook, two processes are created, one for speech input and one managing graphic behaviour.

As a result, for example, while a mouse command is processed, voice messages can be acquired, recognized, and sent to the appropriate agents. We chose to have the fusion of events done by a supervisor agent.

Synchronous data input allows synergic input, but is not required to the fusion, the hierarchy of agents is enough. No explicit combination has to be specified either in the mouse or voice handler.

Although our approach does not allow complex, media dependent analysis (of the kind of Put That There [Bolt 80]), it is an easy way to allow multimodal input. It is expandable, and significantly enhance otherwise monomodal application. One could not do what is possible with VoicePaint without multimodal input.

To be able to fuse modalities, an important question is how to represent the elements of fusion.

6. ABSTRACT COMMON REPRESENTATION

Fusion is facilitated if it takes place among uniform objects. A common representation of data might be used to represent data to be fused as well as the result of the fusion. The following examples illustrate the use of common representation.

6.1. Common representation for fusion

VoicePaint and Notebook use a common representation for multimodal fusion. Because of the level of fusion (inside the Dialogue Controller), we use abstract command representation. Indeed, the Dialogue Controller needs only to know the command and its parameters. Incomplete commands and synergic multimodal commands can be specified.

6.2. Common representation as a result of fusion

Other programmes, such as ICPDraw [Wretö 89], use frame as a common media-independent abstract representation of commands. Each attribute of the frame may be specified using different media.

The MMI2 Esprit project defines a structure called Common Meaning Representation to communicate between its sub-systems. This Prolog-based representation is oriented towards natural language representation. It is also used as a common representation for fusion.

6.3. Media-dependent data

Whenever possible, media-independent commands should be used. They give a greater flexibility to the user by allowing him to choose a specific modality without increasing the complexity of the Dialogue Controller. Media fusion is also easier, the interface component being in charge of filling an appropriate abstract command structure.

Media dependent data however, should not always be discarded. There are some times when a direct access to the data is needed, for example because the processing is data dependent. At least a mention of which media were used to specify the abstract command should be available, so that the results of this command be issued using the most appropriate output media.

None of the reviewed mechanisms allow for the smooth handling of media needing continuous data input, such as stylus, voice or mouse, except for the command mechanism of MacApp which defines a class of “tracking commands” to specifically track the mouse.

7. CONCLUSION

We successfully used our software architecture model to build two multimodal systems (VoicePaint and Notebook). However, we feel the urge to study a more sophisticated abstract common representation mechanism providing:

- Media-independent command representation
- Indication of which media was used, and the initial media-dependent data
- Incomplete commands management
- Continuous data capture (speech, pen strokes)

In our next step, we will study the consequences the abstract common representation will have on software architectures.

8. ACKNOWLEDGEMENT

We wish to thank members of the multimodal working group of the IHM '91 conference whose thoughts served as a basis for this paper.

9. REFERENCES

[Bolt 80] R. A. Bolt, "«Put that there»: Voice and Gesture at the Graphics Interface", ACM *SIGGRAPH 1980*, 1980, 226-270.

[Gourdol 89] A. P.J. Gourdol, *Human Interface Kernel: Developer's Guide*, 1989

[Gourdol 91] A. P.J. Gourdol, *Architecture des Interfaces Homme-Machine Multimodales*, Master's Thesis, Université Joseph Fourier, 1991.

[Hauptmann 89] A. G. Hauptmann, "Speech and Gestures for Graphic Image Manipulation", CHI *'89 Proceedings*, 1989, 241-245.

[Lunatti 91] J.-M. Lunatti, A. I. Rudnicky, *Spoken Language interfaces: The OM system*. In proceedings of CHI'91 Conference, New orleans, April 27-May 2, 1991, pp. 453,454.

[Nigay 91a] L. M.B. Nigay, *A case Study of Software Architecture for Multimodal Interactive System: a voice-enabled graphic notebook*, Technical Report, October 1991.

[Nigay 91b] L. M.B. Nigay, J. Coutaz, *Building User Interfaces: A Cookbook for Organizing Software Agents*. ESPRIT Basic Research Action 3066 AMODEUS (Assimilating MOdels of DEsigners, Users and Systems), 1991.

[Norman 86] D. A. Norman, S.W. draper, *User Centred System Design*. Lawrence Erlbaum Associates Publ., 1986.

[UIMS 91] The UIMS Workshop Tool Developers A Metamodel for the Runtime Architecture of an Interactive System, 1991.

[Wretö 89] J. Wretö, J. Caelen, *ICPDraw, Rapport final du projet ESPRIT MULTIWORKS n° 2105,* 1989.