Software Implications for Multimodal User Interfaces

Joëlle Coutaz, Arnaud Gourdol

Laboratoire de Génie Informatique (IMAG)
BP 53 X, 38041 Grenoble Cedex, France
email: joelle@imag.imag.fr, gourdol@imag.imag.fr
tel. (33) 76 51 48 54, fax (33) 76 44 66 75

Abstract

This paper discusses software considerations for multimodal user interfaces, that is, systems able to support human-to-human modalities of communication (such as gesture and natural language). A definition and a classification of multimodal systems is proposed and the distinction between multimodal and multimedia systems is clarified. Then multiagent models and techniques used in graphical user interfaces are compared with concepts and approaches developed in computer vision and natural language processing. This analysis leads the way to the definition of a general framework aimed at the integration of multiple modalities.

Keywords: multiagent model, multimedia user interface, multimodal user interface.

1. Introduction

Graphical user interfaces (GUI) are now common practice. Although not fully satisfactory, concepts in GUI are well understood and software tools are widely available. Parallel to the development of graphical user interfaces, natural language processing, computer vision, and gesture [Krueger 85] analysis have made significant progresses.

This paper presents our early analysis and experience with the integration of multiple modalities of communication between a user and an interactive system. Next section defines the notion of multimodality and clarifies the distinction between multimodal and multimedia systems. In section 3, we first summerize our own observations about GUI's software architecture, then relate them to current practices in the domain of "human modalities" processing (e.g. natural language processing, computer vision). This analysis opens the way to the definition of a general framework aimed at the integration of multiple modalities. This framework is presented in the last section.

2. Multimedia and Multimodal User Interfaces

In this section, we define the notion of multimedia user interface and suggest a classification illustrated with current interactive systems. Multimodal user interfaces are then analyzed along the same lines.

2.1. Multimedia User Interfaces

2.1.1. Definition

A media is a technical means which allows written, visual, or sonic information to be communicated among humans.

By extension, a multimedia system is a computer system able to acquire, deliver, memorize, and organize written, visual, and sonic information. In the domain of computer science:

- written material is not restricted to physical hard copies. It is extended to textual and static graphical information which are visually perceivable on a screen;
- visual material is usually associated with full motion video, more rarely with realistic graphical animations such as those produced in image synthesis;
- sonic information includes vocal or musical pre-recorded messages as well as messages produced with a sound or voice synthetizer.

2.1.2. Classification

Multimedia systems may be classified in two categories: first generation multimedia systems and full-fledged multimedia systems.

First generation multimedia systems are characterized by "internally produced" multimedia information. All of the information is made available from the standard hardware such as bitmap screen, sound synthetizer, keyboard and mouse. Such basic hardware has led to the development of a large number of tools such as user interface toolkits and user interface generators. With some rare exceptions such as Muse [Hodges 89] and the Olivetti attempt [Binding 89], all of the development tools have put the emphasis on the graphical media. Apart the SonicFinder, a Macintosh finder which uses auditory icons [Gaver 86], computer games have been the only applications to take advantage of non speech audio information.

Full-fledged multimedia systems are able to acquire non digitized information. The basic apparatus of first generation systems is now extended with microphones and CD technology. Fast compression/decompression algorithms such as JPEG [Wallace 91] make it possible to memorize multimedia information. While multimedia technology is making significant progresses, user interface toolkits and user interface generators keep struggling in the first generation area. Since the basic user interface software is unable to support the new technology, multimedia applications are developped on a case per case basis. Multimedia electronic mail is made available from Xerox PARC, NeXT and MicroSoft: a message may include text, graphics as well as voice annotations. FreeStyle from Wang, allows the user to insert gestural annotations which can be replayed at will. Authoring systems such as Guide, HyperCard and Authorware allow for the rapid prototyping of multimedia applications. Hypermedia systems are becoming common practice although navigation is still an unsolved problem.

To summarize, a multimedia computer system includes multimedia hardware to acquire, memorize and organize multimedia information. From the point of view of the user, a multimedia computer system is a sophisticated repository for multimedia information. At the opposite of multimodal computer systems, it ignores the semantics of the information it handles.

2.2. Multimodal User Interfaces

2.2.1. Definition

A modality may be the particular form used for rendering a thought, the way an action is performed. In linguistics, one makes a distinction between the content and the attitude of the locutor with regard to the content. For example, the content "workshop, interesting" may be expressed using different modalities such as: "I wish the workshop were interesting"; "The workshop must be interesting"; "The workshop will be interesting". In addition to these linguistic modalities one must consider the important role played by intonation and gesture. Thus human to human communication is naturally multimodal.

By extension, a computer system is multimodal if it is able to support human modalities such as gesture, written or spoken natural language. As a result,

- a multimodal system must be equipped with hardware to acquire and render multimodal expressions in "real time", that is, with a response time compatible with the user's expectations,
- it must be able to choose the appropriate modality for outputs,
- it must be able to understand multimodal input expressions. The implication of this fundamental feature, understanding, will be analyzed further in section 3.2.

2.2.2. Classification

Current practices in multimodal user interfaces lead to the following taxonomy: exclusive, and synergic multimodal user interfaces. In addition to the modality per se, we need to consider the effect of concurrency.

A user interface is *exclusive mudimodal* if:

- multiple modalities are available to the user, and
- an input expression is built up from one modality only.

An input expression is a "sentence"

- produced by the user through physical input devices, and
- meaningful for the system. In particular, a command is a sentence.

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, using a keyboard shortcut, or say "open window". One can observe the redundancy of the means for specifying input expressions but, at a given time, an input expression uses one modality only.

Xspeak [Schmandt 90] extends the usual mouse-keyboard facilities with voice recognition. Vocal input expressions are automatically translated into the formalism used by X window [Scheifler 86]. Xspeak is an exclusive multimodal system: the user can choose one and only one modality among the mouse, keyboard and speech to formulate a command. In Grenoble, we have been using Voice Navigator to extend the Macintosh Finder to an exclusive multimodal Finder. Similarly, Glove-Talk [Fels 90] is able to translate gesture acquired with a data glove into speech (synthesis). Eye trackers are also used to acquire eye movements and interpret them as commands. Although spectacular, these systems are by no means exclusive multimodal only.

A user interface is *synergic multimodal* if:

- multiple modalities are available to the user, and
- an input expression is built up from multiple modalities.

For example, the user of a graphics editor such as ICP-Draw [Wretö 89] and Talk and Draw [Salisbury 90], can say *put that there* while pointing at the object to be moved then showing the location of the destination with the mouse or a data glove. In this formulation, the input expression involves the synergy of two modalities: speech events, such as *that* and *there*, call for complementary input events such as mouse clicks and/or data glove events interpretable as pointing commands.

Clearly, multimodal events must be linked through temporal relationships. For example, in Talk and Draw, the speech recognizer sends an ASCII text string to Gerbal, the graphical-verbal manager. The graphics handler time-stamps high level graphics events (e.g. the identification of selected objects along with domain dependent attributes), and registers them into a blackboard. On receipt of a message from the speech recognizer, Gerbal waits for a small period of time (roughly one-half second), then asks the blackboard for the graphical events that occurred after the speech utterance has completed. Graphical events that do not pertain to a window of time are discarded. It results from this observation that windowing systems which do not time-stamp events are the wrong candidates for implementing synergic multimodal platforms.

One important feature in user interface design is concurrency. Concurrency makes it possible for the user to perform multiple physical actions simultaneously, to carry multiple tasks in parallel (multithread dialogues), to allow the functional core and the user interface to perform computations asynchronously. In our case of interest:

- concurrency in exclusive muldimodal user interfaces allows the user to produce multiple input expressions simultaneously, each expression being built from one modality only. For example, it would be possible for the user to say "open window" while closing another one with the mouse;

- concurrency is necessary to synergic multimodal user interfaces since, by definition, the user may use multiple channels of communication simultaneously. The absence of concurency would result in a strict ordering with conscious pauses when switching between modalities. For example, the specification of the expression *put that there* would require the user to say *put that*, then *click*, then utter *there*, then *click*.

Figure 1 summarizes our taxonomy illustrated with examples.

	Non Concurrency	Concurrency
Synergic Multimodal	Not desirable	ICP-Draw Talk-And-Draw
Exclusive Multimodal		XSpeak
Mono-Modal	non agent-based UIMS's	multiagent-based UIMS's

Figure 1: Dimension space for multimodal user interfaces.

2.3. Multimedia and Multimodal: Summary

Multimedia and multimodal user interfaces use similar physical input and ouput devices. Both acquire, maintain and deliver visual and sonic information. Although similar at the surface level, they serve distinct purposes:

- a multimedia system is a repository of information produced by multiple communication techniques (the medias). It is an information manager which provides the user with an environment for organizing, creating and manipulating multimedia information. As such, it has no semantic knowledge of the information it handles. Instead, data are encapsulated into typed chunks which constitute the units of manipulation (e.g. creation, deletion, and, in the particular case of hypermedia systems [Conklin 87], linkage between chunks). Chunk contents are ignored by the system;

- a multimodal system is supposed to have the competence of a human interlocutor. At the opposite of multimedia systems, a multimodal system analyzes the content of the chunks produced by the environment in order to discover a meaning. Conversely, it is able to produce multimodal output expressions that are meaningful to the user.

In the current state of the art, one can identify another distinctive feature between multimedia and multimodal systems: multimedia information is the subject of the task (it is manipulated by the user) whereas multimodal information is used to control the task. With the progress of concepts and techniques, this distinctive usage will grow blurred over time.

So far, we have tried to clarify the distinction between multimedia and multimodal systems, and we have proposed a classification for multimodal user interfaces. We need now to analyze the implication of multimodality on software architectures.

3. Software Architecture: Observations

In this section, we first summerize our own observations about GUI's software architecture, then relate them to current practices in the processing of "Human modalities".

3.1. Graphical User Interfaces

3.1.1. The Seeheim model

The Seeheim model has set the foundations for the separation between the functional core (the application) and the presentation (the user interface) [Pfaff 85]. Unfortunately, the ambiguity of the term "separation" has led to a number of wrong interpretations. One of them is the linguistic mapping of the three components identified in the Seeheim model with semantics, syntax and lexical analysis. This sequential pipe-line model of the interaction, based on compiler technology, is convenient for automatic generation of simple user interfaces. It does not support multithread dialogues nor a fortiori full-fledged concurrency as required by multimodal user interfaces [Coutaz 90]. These observations have encouraged computer scientists to investigate different architectural models. One of them is a highly parallel modular organization around the notion of agent.

3.1.2. Multiagent models

Multiagent models structure an interactive system into a collection of specialized agents that produce and react to events. An agent is a complete information processing system: it includes event receivers and event transmitters, a memory to maintain a state, and a processor which cyclically processes input events, updates its own state, and may produce events or change its interest in input event classes. Agents communicate with other agents including the user.

Multiagent models such as MVC [Goldberg 84] and PAC [Coutaz 87] or multiagent-based tools such as Interviews [Linton 89], Serpent [Bass 88], or Sassafras [Hill 87] do not deny the principle of separation of the functional core from the user interface. In fact, they go further: They generalize the distinction between concepts, dialogue control and presentation techniques by applying the separation of concerns at multiple levels of abstraction among cooperating agents. They differ however in the way they express distribution and cooperation. As an illustration, let us consider PAC and MVC.

In the MVC model, a Model corresponds to a PAC Abstraction: both of them represent a competence which is media independent. A MVC View is responsible for outputs whereas a MVC Control handles inputs only. Thus, the combination of a MVC View and a MVC Control corresponds to a PAC Presentation. The notion of a PAC Control, that mediates between the media independent perspective of an agent (the Abstraction) and its the media dependent perspective (the Presentation), has no direct correspondence in MVC. It can however be considered as a particular class of Model. At the opposite of PAC, MVC does not convey any conceptual levels of abstractions. Levels of abstractions are defined by the implementation tool (Smalltalk hierarchy of classes) not by the model per se. In PAC, an interactive system is explicitly organized into multiple levels of abstraction. Heuristics rules have been devised to help the software designer to identify the intermediary agents needed in the hierarchy [AMODEUS 90].

Although multiagent models differ in the way they organize the distribution and the cooperation between agents, they all have to face the same problem space: that of including the appropriate mechanisms for expressing the various classes of agents involved in the implementation of

GUI's. From our own experience, we have identified two classes of agents in the user interface portion of an interactive system: interactors, and hidden agents (see Figure 2).

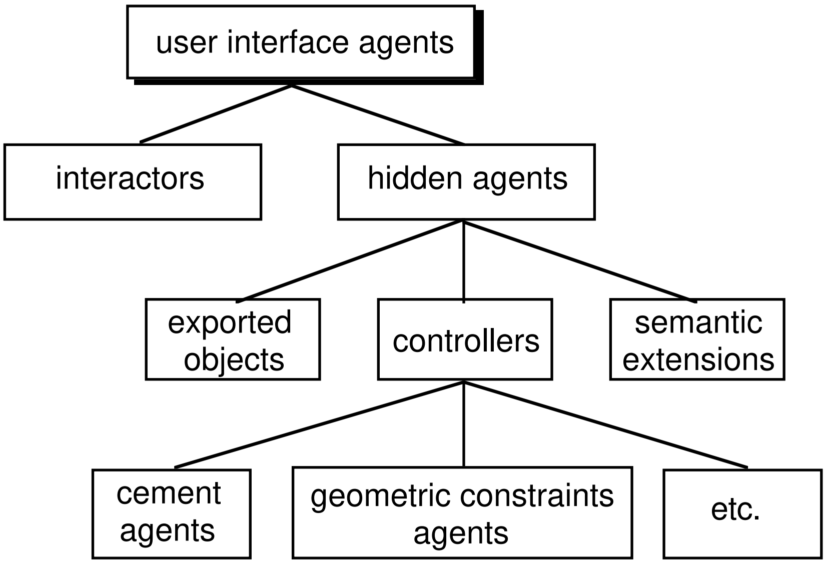

Figure 2: Agent types and relationships.

Interactors communicate with the user directly. From the user's point of view, they act as the partners of the user all involved in a common task. From the software point of view, they serve as the "proxies" of a set of conceptual agents. They are the intermediaries between conceptual agents and the user. Consequently, they are related to such agents according to some mapping mechanism [Coutaz 91]. Depending on the software platform, they are named presentation objects, interactive objects, widgets, etc.

Hidden agents have no direct interaction with the user. A hidden agent may be an exported entity, a conceptual extension, or a controller. It is always linked to another agent and possibly to entities of the functional core.

- An *exported object* is an entity that both the functional core and the user interface understand. It is supposed to be presentation independent and to match the mental representation that the user has for a particular concept in the task domain. It is located at the boundary between the functional core and the user interface. As a result, it is linked to the functional core and to other agents in the user interface (interactors or conceptual extension).

- A *conceptual extension* serves as a semantic enhancement of the functional core. Experience shows that exported objects may not match the user's needs because of a wrong design decision. It may also be the case that the tool used for building the user interface does not support the definition of the desired types of exported objects. For example, in a number of UIMS's, exported objects cannot be compound entities. It results from this limitation an undesirable dissemination of logically connected information which then must be glued together. Conceptual extensions provide a convenient way for defining new perspectives on data from the functional core or for combining data and behaviour into a new concept in the user interface. Sometimes, these concepts, which help in the accomplishment of a task, are referred to as "syntactic objects" as opposed to "semantic objects" which implement domain dependent concepts of the functional core [Bass-Coutaz 91].

An example of semantic enhancement is the concept of "zone" for the user interface we have designed for a mobile robot [Bass-Coutaz 91]. The task analysis contained no concept of zone but we observed that the specification of identical values might be repeated. For example, the operator might tell the robot that at places p1, p2 and p3, it should follow any humans, warn the operator in case of fire and move as fast as possible. According to the Keystroke model [Card 1983], it should be more convenient for the operator to describe an expected robot behavior in terms of the zone concept and allow the specification of which places belong to which zone. Since the zone concept had no meaning for the functional core (i.e. the intelligent supervisor of the robot), it has been implemented as a conceptual extension in the user interface.

- A *controller* is a computational unit which governs the behaviour of a set of related agents. These agents may be conceptual extensions or interactors. The notion of controller has a large number of illustrations. In particular, constraint-based geometry managers such as that of Garnet [Myers 90], organize spatial relationships between subagents. Serpent's view controllers dynamically determine the creation and deletion of interactors [Bass-Coutaz 91]. As shown in figure 3, cement agents in PAC implement distributed syntax analyzers.

  Syntax analysis is distributed when the specification of a command involves user's actions distributed over multiple agents. This is particularly true for CAD interactive systems where a palette, which is an agent, displays the possible options and where a window, which is yet another agent, is used as a scratch area to create and edit an artifact (e.g. a graphical document). The creation of a circle implies selecting the circle icon in the palette then drawing the circle in the scratch area. The combination of the two actions has a meaning - that of creating a circle - but the palette and the scratch area are not aware of each other. They would otherwise be dependent on each other and thus the first one would not be reusable without the existence of the second one. In addition, multithread dialogue authorizes logically connected actions to be interleaved with "foreign" actions. As a consequence, there is a conflicting need for maintaining "mutual ignorance" between two agents, for remembering logically connected actions and for combining them into a higher abstraction (e.g. a command). The multiagent framework provides a natural way for satisfying such requirements: As shown in Figure 3, an intermediary agent is introduced to cement logically related, but role independent, agents.

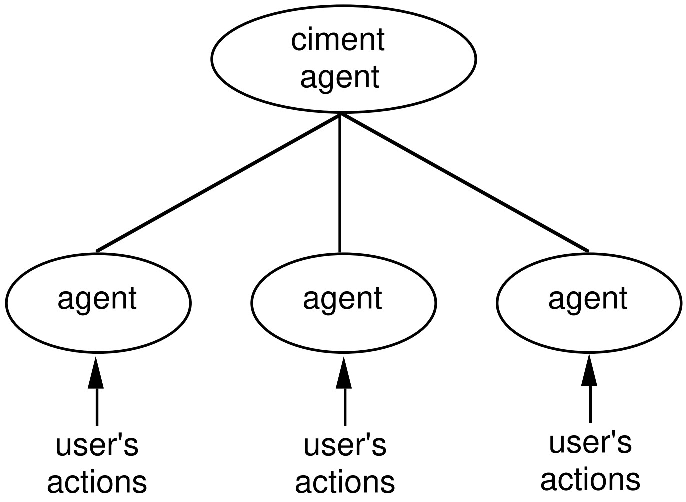

Figure 3: An example of a controller agent to enhance code reusability: a distributed syntax analyzer agent which links logically related but role independent agents.

Current multiagent models have been designed based on experience with direct manipulation graphical user interfaces [Nanard 90, Hutchins 85]. In order to build multimodal user interfaces, we first need to identify the similitudes and differences between GUI techniques and that of other human modalities.

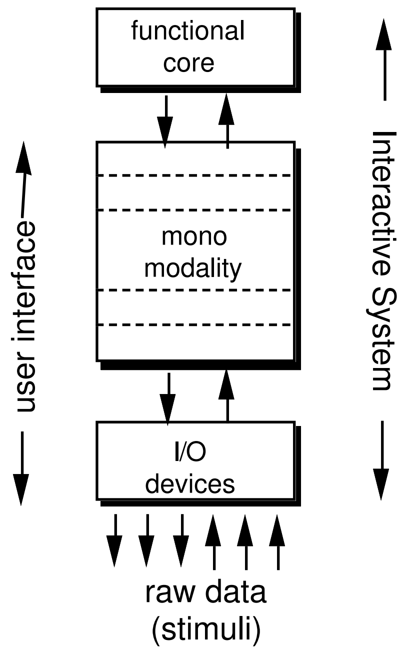

Figure 5: The general software framework for monomodal interaction.

3.2. GUI's and other modalities

Computer vision systems as well as natural language processing systems are all based on a framework similar to that of multiagent architectures used in GUI's. As shown in Figure 4, stimuli (or events) are received through physical input devices. The functional core sits at the other end of the spectrum. Between these two extremes, a number of layers of abstractions transform input stimuli into semantic abstractions understood by the functional core.

As an illustration, consider the layers of abstraction involved in a computer vision system [Crowley 89] (Figure 6). At the lowest, most basic level we find the image or signal, as generated by the acquisition process. This information is then described at a second level as geometric information expressed in a coordinate space which is inherent to the signal. The third level in this hierarchy is typically concerned with scene geometry. Although geometric in nature, the coordinate system for this level switches from 2-D image coordinates to a 3-D scene based coordinates. A scene based coordinate space makes it possible to integrate information from different viewing positions. Viewpoint invariance then leads to such fundamental concepts as "object constancy". It becomes possible to "see beyond" the visual stimulus to an imagined (internal) world of individual "objects". However, to associate the perception with known objects, the geometric information in both the image description and in the scene description must be grouped to form cues. The idea of individual "objects" leads to the final, most abstract level. The interpretation level expresses perceptions as symbolic information. Symbols permit perceptions to be associated to previous experiences. Association with known "objects" leads to expectations concerning nearby objects as well as expectations of behaviors. Both the geometric scene description and the description of sensor signals provide information for inference of a symbolic scene description.

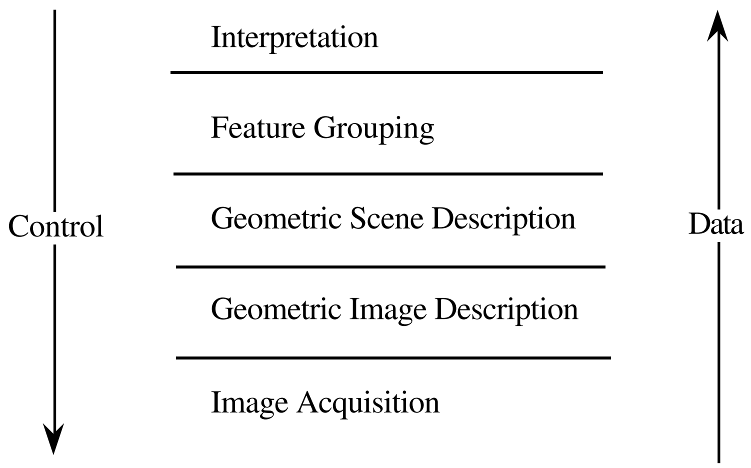

Figure 6: Levels of abstractions in a computer vision system.

Although one can observe similitudes in the way problems are tackled in the various disciplines, layers in NL and computer vision involve sophisticated modelling techniques which, in GUI's, are either unnecessary or avoided. Indeed, understanding NL expressions requires the computer system to perform two types of complementary activities:

- translate input expressions into an internal abstract model which represents the semantics of the expressions,
- maintain a dynamic model of the interaction including domain modelling, task modelling, and user modelling. This model sets the foundations for reasoning about the most recent input expressions. One class of such reasoning deals with uncertainty and ambiguity such as those inherent to anaphoras and ellipses.

An anaphoric reference is a partial description (denotation) of a concept which has been previously mentioned in the discourse. For example, in the following conversation: "Do you have a message from FooBar? Delete <u>it</u>", *it*, which refers to the message from FooBar, is a metaphoric phenomenon. An ellipsis is an omission of a syntactic construct without necessarily loosing the meaning of the sentence. For example, in "Do you have a message from Foo? <u>And from Bar</u>?" *and from Bar*, from which the syntactic construct *do you have a message from* has been eliminated, is an ellipsis.

Clearly, anaphoras and ellipses, which both refer to past expressions, cannot be solved without a dynamic model of the interaction. For example, in Partner, a speech understanding system, a history mechanism is used to process anaphoras, and the user's goals are traced to formulate hypotheses for solving ellipses [Pierrel 90]. At the opposite of natural language user interfaces, traditional graphical user interfaces do not maintain any dynamic model of the interaction, or if they do, the model is poor and implicit. For example, the notions of "current selected object" and "current cut buffer", which avoid the user to specify an operand to a command, support limited cases of ellipses. Similarly, the notion of "currently available commands", which structures the task space, supports context modelling in a very crude way.

In addition to linguistic problems, "human modalities" techniques are unreliable: raw data are noisy and recognition is flawful. In order to enhance recognition, models are maintained at multiple levels of abstractions, each model being used to update confidence factors in adjacent layers. These techniques can be observed in speech recognition as well as in computer vision. In GUI's, on the other hand, raw data, such as mouse clicks and keyboard events, are deterministic. When ambiguity arises as for overlapping objects in graphical editors, an ordering policy is imposed on the user.

4. Software Framework for Multimodality

Although similar principles of software decomposition are exploited in all of the HCI disciplines, each modality has its own system of representation. The implementation of synergic multimodal user interfaces requires the integration of the various modelling techniques into a uniform framework. This combination may be performed along three axes:

- keep the current modelling techniques as they are and define appropriate gateways between them;
- define a unique abstract representation onto which different modelling techniques can be hooked;
- use a medium range approach which keeps some elements of the current modelling techniques and introduce layers for integration.

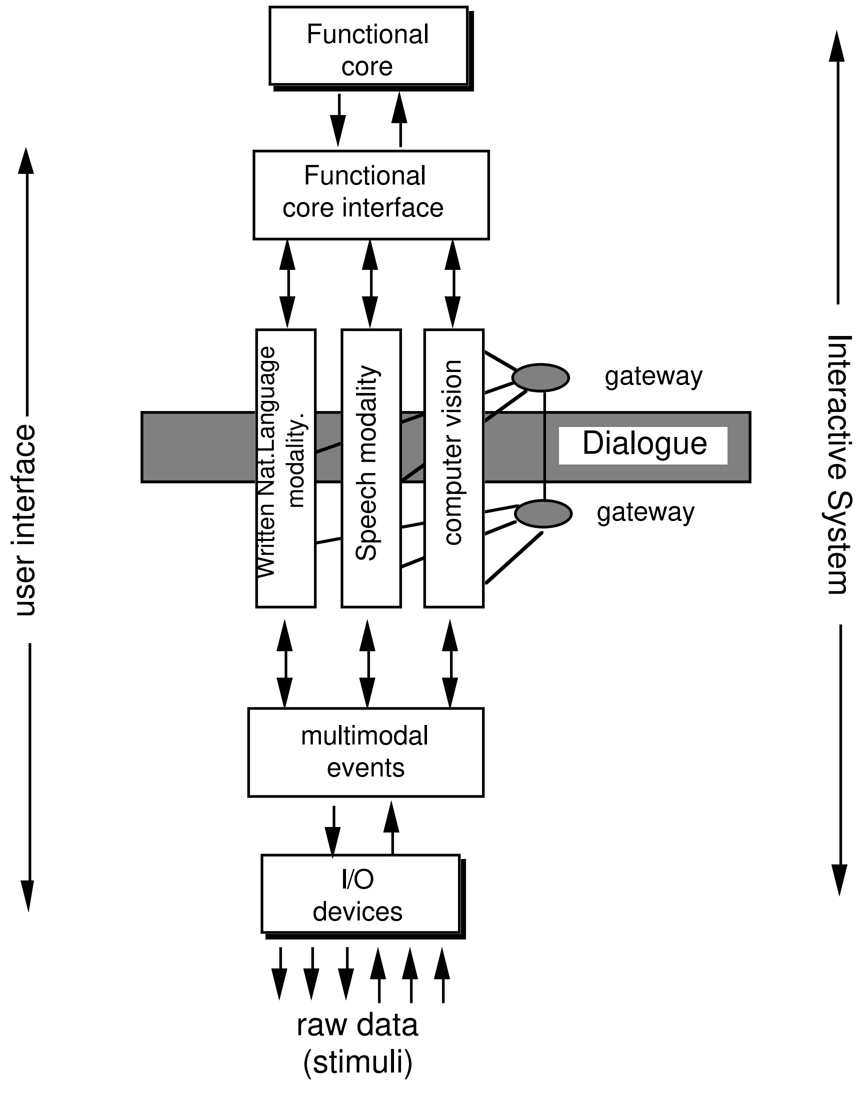

Figure 7: A framework for integrating multiple modalities of interaction.

In the short term, we think that the third approach is a reasonable way to go. As illustrated in Figure 7, the two extreme layers, the interface with the functional core and the event layers, are good candidates for integration. The intermediate layers, which implement highly specialized concepts and know-how, should be kept unchanged. However, a common model of the dialogue should be shared by all of the modalities through specific gateways.

Modelling the dialogue should be performed at a high level of abstractions, i.e. at the same level as the interface with the functional core. As discussed above, dialogue modelling in GUI is very poor or inexistent whereas dialogue modelling in human modalities is unavoidable. However, the representation techniques are too much biased by the modality they serve. We need to jump one step further and model the dialogue at a more psychological level.

We have developed a first experience in integrating voice and graphics modalities at the event level. Conceptually, it is a very simple extension of events as managed by windowing systems. Agents, which used to express their interest in graphics events only, can now express their interest in voice events. As graphics events are typed, so are voice events. Events are dispatched to agents according to their interest. We have applied this very simple model to the implementation of a Voice-Paint editor on the Macintosh using Voice Navigator, a word-based speech recognizer board: as the user draws a picture with the mouse, the system can be told to change the attributes of the graphics context (e.g. change the foreground or background colors, change the thickness of the pen or the filling pattern, etc.). Our toy example is similar in spirit to the graphics editor used by Ralph Hill to demonstrate how Sassafras is able to support concurrency for direct manipulation user interfaces.

Voice-Paint illustrates a rather limited case of multimodal user interface: concurrency at the input level. This is facilitated by Voice Navigator whose unit of communication is a "word". From the user's point of view, a word may be any sentence. For Voice Navigator, pre-recorded sentences are gathered into a data base of patterns. At run time, these patterns are matched which the user's utterances. The combination of Voice Navigator and graphics events into high level abstractions (such as a command) does not require a complex model of the dialogue. Thus, Voice-Paint does not demonstrate the integration of multiple modalities at the higher level of abstractions. This work is the topic of a new project launched in the Grenoble scientific community.

5. Conclusion

This article describes a first step experience aimed at the implementation of synergic multimodal user interfaces. It does not claim any ready-for-use solutions. Instead, it presents a possible framework for bundling multiple modalities into a consistent organization. Our first experimental results encourage us to extend our expertise in multiagent architectures for GUI's to multimodal user interfaces.

6. References

[AMODEUS 90]
G. ABOWD, J. COUTAZ, M. HARRISON, L. NIGAY: System Modelling, ESPRIT BRA AMODEUS, Deliverable D2, 1990.

[Bass 88]
L. BASS, E. HARDY, K. HOYT, R. LITTLE, R. SEACORD:*The Serpent run time architecture and dialogue model*; Carnegie Mellon Technical Report, CMU/SEI-88-TR-6, January, 1988.

[Bass-Coutaz 91]
L. BASS, J. COUTAZ: Developing Software for the User Interface; Addison Wesley, 1991.

[Binding 89]
C. BINDING, S. SCHMANDT, K. LANTZ, M. ARONS: Workstation audio and window based graphics: similarities and differences; Proceedings of the 2nd Working Conference IFIP WG2.7, Napa Valley, 1989, pp. 120-132.

[Card 83]
S. CARD, T. MORAN, A. NEWELL: The Psychology of Human-Computer Interaction, ISBN 0-89859-243-7, Lawrence Erlbaum Associates, Publish., 1983.

[Conklin 87]
J. CONKLIN: Hypertext, an Introduction and Survey; IEEE Computer, 20(9), September, 1987, 17-41.

[Coutaz 87]
J. COUTAZ: PAC, an Implemention Model for Dialog Design; Interact'87, Stuttgart, September, 1987, pp. 431-436.

[Coutaz 90]
J. COUTAZ: Interface Homme-Ordinateur: Conception et Réalisation ; Dunod, 1990.

[Coutaz 91]
J. COUTAZ, S. BALBO: Applications: a Dimension Space for UIMS's; to appear in CHI'91 proceedings.

[Crowley 89]
J.L. CROWLEY: Knowledge, Symbolic Reasoning and Perception; Intelligent Autonomous Systems II, Amsterdam, December, 1989.

[Gaver 86]
W. W. GAVER: Auditory Icons: Using Sound in Computer Interfaces; Human Computer Interaction, Lauwrence Erlbaum Ass. Publ. , Vol. 2, 1986, 167-177.

[Fels 90]
S.S. FELS: Building Adaptative Interfaces with Neural Networks: the Glove-Talk Pilot Study; University of Toronto, Technical Report, CRG-TR-90-1, February, 1990.

[Goldberg 84]
A. GOLDBERG: Smalltalk-80: The Interactive Programming Environment; Addison-Wesley Publ., 1984.

[Hill 87]
R.D. HILL: Supporting Concurrency, Communication and Synchronization dans Human-Computer Interaction-The Sassafras UIMS; ACM Transactions on Graphics 5(2), April, 1986, pp. 179-210.

[Hodges 89]
M. E. HODGES, R.M. SASNETT, M.S. ACKERMAN: A Construction Set for Multimedia Applications; IEEE Software, January, 1989, pp. 37-43.

[Hutchins 85]

E. L. HUTCHINS, J. D. HOLLAN, D. A. NORMAN: Direct Manipulation Interfaces; Human Computer Interaction, Lawrence Erlbaum Ass. Publ., 1(4), 1985, 311-339.

[Krueger 85]
M. W. KRUEGER, T. GIONFRIDDO, K. HINRICHSEN: Videoplace, An Artificial Reality; CHI'85 Proceedings, ACM publ., April, 1985, 35-40.

[Linton 89]
M.A. LINTON, J.M. VLISSIDES, P.R. CALDER: Composing User Interfaces with Interviews; IEEE Computer, February, 1989, pp. 8-22.

[Myers 90]
B.A. MYERS, D.A. GIUSE, R.B. DANNEBERG, B.V. ZANDEN, D.S. KOSBIE, E. PERVIN, A. MICKISH, P. MARCHAL: Garnet: Comprehensive Support for Graphical, Highly Interactive User Interfaces; IEEE Computer, Vol. 23, No. 11, 1990, pp. 71-85.

[Nanard 90]
J. NANARD: La Manipulation directe en Interface Homme-Machine; Thèse de Doctorat d'Etat, Université de Montpellier II, décembre, 1990.

[Pfaff 85]
User Interface Management Systems; G. E. Pfaff ed., Eurographics Seminars, Springer-Verlag, 1985.

[Pierrel 90]
J.-M. PIERREL, J. CAELEN: Compréhension de la parole: Méthodes et Applications; 12èmes journées francophones sur l'informatique, La Machine Perceptive, EC2 Ed., Grenoble, Janvier, 1990, pp. 53-103..

[Salisbury 90]
M. W. SALISBURY, J. H. HENDRICKSON, T. L. LAMMERS, C. FU, S. A. MOODY: Talk and Draw: Bundling Speech and Graphics; IEEE Computer, 23(8), August, 1990, 59-65.

[Scheifler 86]
R.W. SCHEIFLER, J. GETTYS: The X Window System; ACM Transaction on Graphics, 5(2), April, 1986, 79-109.

[Schmandt 90]
C. SCHMANDT, M. S. ACKERMAN, D. HINDUS: Augmenting a Window System with Speech Input; IEEE Computer, 23(8), August, 1990, 50-58.

[Wallace 91]
G.K. WALLACE: The JPEG Still Picture Compression Standard for Multimedia Applications; CACM, Vol. 34, No.4, April, 1991, pp. 30-44.

[Wretö 89]
J. WRETO, J. CAELEN: ICP-DRAW, rapport final du projet ESPRIT MULIWORKS no 2105.