A Design Space For Multimodal Systems: Concurrent Processing and Data Fusion

Laurence Nigay, Joëlle Coutaz

Laboratoire de Génie Informatique (IMAG)
BP 53 X, 38041 Grenoble Cedex, France
Tel: +33 76 51 44 40
E-mail: nigay@imag.imag.fr, joelle@imag.imag.fr

ABSTRACT

Multimodal interaction enables the user to employ different modalities such as voice, gesture and typing for communicating with a computer. This paper presents an analysis of the integration of multiple communication modalities within an interactive system. To do so, a software engineering perspective is adopted. First, the notion of "multimodal system" is clarified. We aim at proving that two main features of a multimodal system are the concurrency of processing and the fusion of input/output data. On the basis of these two features, we then propose a design space and a method for classifying multimodal systems. In the last section, we present a software architecture model of multimodal systems which supports these two salient properties: concurrency of processing and data fusion. Two multimodal systems developed in our team, VoicePaint and NoteBook, are used to illustrate the discussion.

KEYWORDS: Modality, multimodal interaction, taxonomy, design space, software architecture, data fusion, concurrency.

INTRODUCTION

In parallel with the development of graphical user interfaces (GUI), significant progress has been made in natural language processing, computer vision and gesture analysis. Systems integrating these techniques as multiple modalities open a complete new world of experience [1]. But as pointed out in [2], differences of opinion still exist as to the meaning of the term "multimodal".

This paper presents our analysis of the integration of multiple communication modalities between a user and an interactive system. To do so, a software engineering perspective is adopted. First, the notion of "multimodal system" is clarified. Based on a precise definition of multimodality, we then propose a design space and a method for classifying multimodal systems. In the last section, we present a software architecture model that supports the most salient properties of such systems: concurrent processing and data fusion.

MULTIMODAL SYSTEM: A DEFINITION

In the general sense, a multimodal system supports communication with the user through different modalities such as voice, gesture, and typing [3]. Literally, "multi" refers to "more than one" and the term "modal" may cover the notion of "modality" as well as that of "mode".

- Modality refers to the type of communication channel used to convey or acquire information. It also covers the way an idea is expressed or perceived, or the manner an action is performed [4].
- Mode refers to a state that determines the way information is interpreted to extract or convey meaning.

In a communication act, whether it be between humans or between a computer system and a user, both the modality and the mode come into play. The modality defines the type of data exchanged whereas the mode determines the context in which the data is interpreted. Thus, if we take a system-centered view, multimodality is the capacity of the system to communicate with a user along different types of communication channels and to extract and convey meaning automatically. We observe that both multimedia and multimodal systems use multiple communication channels. But in addition, a multimodal system is able to automatically model the content of the information at a high level of abstraction. A multimodal system strives for meaning.

Our definition of multimodality is system-oriented. A user-centered perspective may lead to a different definition. For instance, according to our system-centered view, the NeXT voice electronic mail [5] is not multimodal. It is multimedia only. Indeed, it allows the user to send mail that may contain graphics, text and voice messages. It does not however extract meaning from the information it carries. In particular, voice messages are recorded but not interpreted. On the other hand, from the user's point of view, this system is perceived as being multimodal: the user employs different modalities (referring to the human senses) to interpret mail messages.

Our system-centered definition of multimodality conveys two salient features that are relevant to the software design of multimodal systems:

- the fusion of different types of data from/to different I/O devices, and
- the temporal constraints imposed on information processing from/to I/O devices.

Data fusion and temporal constraints provide the basis for the design space presented in the next section.

MULTIFEATURE SYSTEMS: A DESIGN SPACE

Previous attempts to systematize the description of interfaces have already been made. However, these approaches focused primarily on input devices. In this paper, we are concerned with both the input and output attributes of an interface. At the lower level of abstraction, classifications of input devices such as those proposed by Buxton [7] and by Card et al. [8], are based on physical properties (such as motion and pressure), the data that a device returns (discrete or continuous) and the dimensions of input a device provides.

At a higher level of abstraction, Foley et al. focus on graphics sub-tasks and propose a taxonomy according to the sub-tasks a device is capable of performing [9]. Our design space is located at this higher level of abstraction; it deals with tasks at the granularity of commands. We address the issues of how a command is specified using the different available modalities and how a command is built from raw data.

Recently, D. Frohlich proposed a framework for describing the design space of interfaces. This framework includes both input and output design spaces [6]. It embeds the different types of modalities and takes into account the human senses. Our design space also includes both input and output attributes of an interface, but our goal is different. Our design space is intended for classifying systems within a framework, and for helping software designers to identify the software implications and constraints for the development of a system.

Our design space is defined along three dimensions: Levels of Abstraction, Use of modalities and Fusion. Figure 1 illustrates some possible values along each dimension and the corresponding classes of systems.

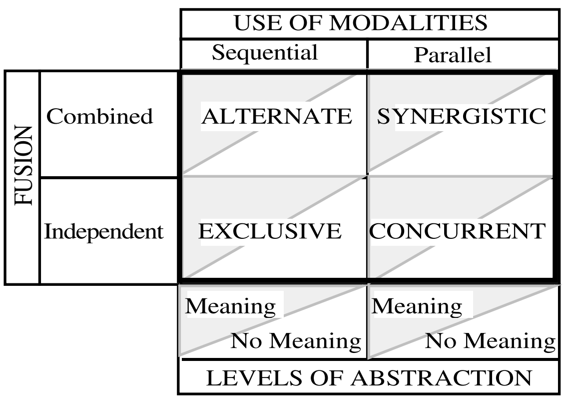

Figure 1: The multi-feature system design space.

Levels of Abstraction

As far as inputs are concerned, data received from a particular device may be processed at multiple levels of abstraction. For example, speech input may be recorded as a signal, or described as a sequence of phonemes, or interpreted as a meaningful parsed sentence. Each represention corresponds to a particular level of abstraction. For outputs, the process is similar: data may be produced from symbolic abstract data or from a lower level of abstraction without any computational detection of meaning. For example, a vocal message may be synthesized from an abstract representation of meaning, from pre-stored text or may simply be replayed from a previous recording.

The important point is that data is represented and processed at multiple levels of abstraction. This transformation process makes possible the extraction of meaning from raw data and conversely the production of data from symbolic abstract representations. To simplify the presentation, we consider only two values along the axis “Levels of Abstraction”: “Meaning” and “No Meaning”. As discussed in the previous section, a multimodal system falls in the “Meaning” category of Figure 1.

Use of Modalities

“Use of modalities” expresses the temporal availability of multiple modalities. This dimension primarily covers the absence or presence of parallelism at the user interface. The granularity for concurrency ranges from the physical actions at the I/O device level to the task-command level. Absence of parallelism is referred to as "Sequential use" whereas presence is called "Parallel use".

A system that supports “Parallel use” allows the user to employ multiple modalities simultaneously. Conversely, a system characterized by the sequential use of modalities, forces the user to use the modalities one after another.

Fusion

Fusion covers the possible combination of different types of data. As discussed above, a data type is associated with a particular modality. The absence of fusion is called "Independent" whereas the presence is referred to as "Combined".

According to the design space, fusion may be performed with or without knowledge about the meaning of the data exchanged. For example, synchronization of audio and video data as supported in the ACME platform [10], is a temporal fusion which does not involve any knowledge of meaning. The ACME platform (Abstractions for Continuous MEdia) is based on the concepts of strands which correspond to streams of audio or video data, of ropes which are combinations of strands, and a logical time system that allows several strands and ropes to be played synchronously. This example of fusion is distinct from the fusion that involves meaning as in the “put that there” paradigm. Fusion based on meaning mixes modalities to build an input or output expression which results in an interpretation at a high level of abstraction in the task domain.

The design space as a whole

The three orthogonal dimensions of our design space (Levels of abstraction, Use of modalities, and Fusion) define eight distinct classes for multi-feature systems. According to our definition, a multimodal system takes the value "Meaning" along the "Level of abstraction" axis.

Having selected the value "Meaning" for "Levels of abstraction", let us considerer the four classes of systems resulting from the combination of the axis "Fusion" and "Use of modalities". We get the following categories: "Exclusive", "Alternate", "Concurrent", and "Synergistic". These classes are discussed in a following section and illustrated with our own multimodal systems: VoicePaint and NoteBook [11].

VOICEPAINT AND NOTEBOOK

VoicePaint is a graphics editor implemented on the Macintosh using Voice Navigator, a word-based speech recognizer board. As a picture is drawn with the mouse, the user can talk and ask the system to change the attributes of the graphics context (e.g., the foreground or background colors, the thickness of the pen, the brightness, the filling pattern, etc.). This system 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 [19].

NoteBook is a personnal electronic book implemented on the NeXT machine using Sphinx, a continuous multi-locutor speech recognition system [21]. It allows a user to create, edit, browse, and delete textual notes. In particular, to insert a note between two notes, the user can say "Insert a note" while simultaneously selecting the location of insertion with the mouse. To edit the content of a note, one modality only is available: typing. Browsing through the set of notes is performed by clicking dedicated buttons such as "Next" and "Previous" or by using spoken commands such as "Next note". To empty the note book, a "Clear notebook" command may be specified using voice or clicking the mouse on the "Clear" button.

Next section shows how the design space can be used to classify a particular multimodal system.

CLASSIFYING MULTIMODAL SYSTEMS

Any classification is based on a set of relevant features f_i. In our case, an interesting set of features is the commands that the system supports. Each feature f_i is weighted according to an estimated importance and has a position p_i within the design space shown in figure 2. For example, the weight w_i can be defined as the frequency of use. As shown in Figure 3, four weighting values have been defined but other rules may be applied. Position p_i can take one of the four discrete values: Exclusive, Alternate, Concurrent and Synergistic. Thus a feature, f_i is formally defined as the couple:

$$f_i = (p_i , w_i)$$

The position, **C**, of a system corresponds to the center of gravity of its features as expressed by the following equation:

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

To illustrate the method, we consider the NoteBook commands presented above.

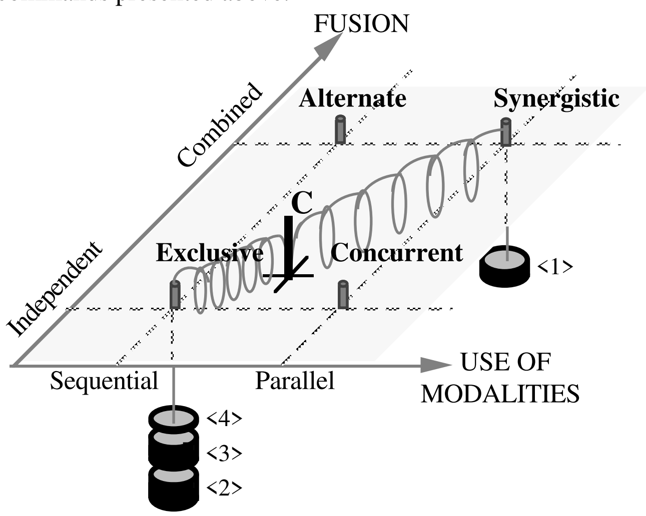

Figure 2: A method to classify multimodal systems: the NoteBook example.

"Insert a note", denoted as Feature <1> in Figure 2, is a synergistic command: it is specified using speech and mouse clicks simultaneously and it requires the fusion of data from multiple input devices. In addition, it is used frequently. Thus <1> is defined as the couple (synergistic, frequent use). The second command "edit the content of a note", denoted as Feature <2> in Figure 2, is characterized by the couple (exclusive, very frequent use): while editing the content of a note only one modality is available (typing) and no other command can be invoked in parallel. For example, it is not possible to turn the pages of the note book while writing the content of a note. Although exclusive, this task is performed very often. Similarly, commands about browsing, denoted as Feature <4>, are used very frequently but are exclusive: the user has the choice between multiple modalities to express the command (speech or mouse clicks) but only one modality is used to specify the command. In addition, no other command can be issued in parallel. The last command to be considered, "clear the note book", is exclusive for the same reason as <3> but is rarely used. Location C of NoteBook in the design space is obtained by applying the formula (1) (see Figure 2). We observe that NoteBook is close to the exclusive class of multimodal systems.

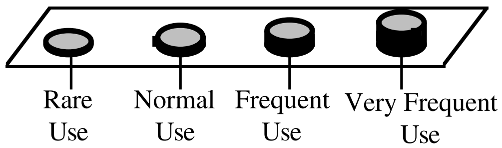

Figure 3: For example four weights which can be associated with a command according to its frequency of use.

A similar process has been applied to VoicePaint and showed that this system is mostly synergistic. Indeed, all of the very frequent commands, which are concerned with drawing, allow the combined and parallel use of speech and mouse gesture.

NoteBook and VoicePaint provide examples for exclusive and synergistic commands only. As an example of concurrency at the command level, one may consider VoiceFinder, a system that adds voice input to the Macintosh Finder. Within VoiceFinder, the user can issue a voice command like "empty the trash" while simultaneously invoking another command such as opening a document with the mouse. "Alternate" requires the fusion of data from/to multiple devices to build up a command but these devices must be used in a sequential manner. For example, the MMI2 [20] system is primarily alternate. In this system, the interaction is driven by natural written language. Deictic references that may occur in a sentence such as "this", are solved by looking for mouse selections in the next following act of interaction: modalities are combined but acquired in a sequential manner.

In summary, the contribution of our design space and classification scheme is three-fold:

- the design space makes it explicit the way different modalities are supported by a particular system,
- the classification scheme makes it precise the location of a system within the design space,
- the design space can be used in conjunction with the classification scheme to study the effect of shifting commands within the design space with regard to the user's expertise or to the task to be performed [12]. Using this methodology, the usability of an interface can be measured.

Usability is the extent to which a user can exploit the potential utility of a system [13]. Usability can be evaluated from the command language itself as illustrated in [13]. Given that the command language barrier is surpassed, usability can be further tested by establishing whether particular modalities are adequate for expressing a given command. For instance, if a command has a small weight in the synergistic position and a large weight in the exclusive position, the choice of modalities for the synergistic command may be inadequate and/or the chosen modalities may be incompatible.

Having identified two salient features for multimodal systems, concurrency and data fusion, we need now to address their implications on software design. Next section describes a model that describes the software organization of synergistic multimodal systems.

SOFTWARE ARCHITECTURE FOR SYNERGISTIC SYSTEMS

Technically, synergistic systems subsume the other three classes of multimodal systems. Although synergistic systems provide a powerful style of interaction [14], they are functionaly more demanding and therefore more complex to build than, for example, exclusive systems. The following architectural model, based on PAC-Amodeus [16], is concerned with the most sophisticated case.

PAC-Amodeus

As shown in Figure 4, the PAC-Amodeus model reuses the components of Arch [15] but refines the dialogue controller in terms of PAC agents [22]. This refinement has multiple advantages including an explicit support for concurrency.

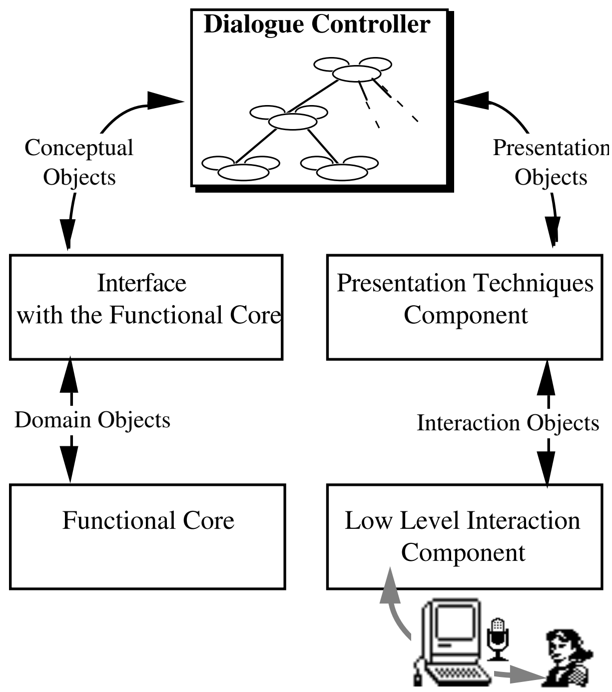

Figure 4: The software components of the PAC-Amodeus model and the interfaces between them.

The Functional Core (FC) implements domain specific concepts in a presentation independent way. The Interface with the Functional Core (IFC) maps Domain objects from the Functional Core onto Conceptual objects from the Dialogue Controller and vice versa.

The Dialogue Controller (DC) is the keystone of the model. It has the responsibility for task-level sequencing. Each task of the user corresponds to a dialogue thread. This observation suggests a multi-agent decomposition where an agent, or a collection of agents, can be associated with each thread. A set of rules that identify the agents necessary for a particular system is presented in [16].

The Presentation Techniques Component (PTC) defines two multi-valued mapping functions that link Presentation and Interaction objects. The PTC describes the presentation (i.e., input and output interfaces). It implements the perceivable behavior of the application for output and input commands. It is only at this level of abstraction that the modality of interaction is taken into account.

The Low Level Interaction Component (LLIC) denotes the underlying software and hardware platform. It supports the physical interaction with the user. It manages user's events from different media (time-stamps and queues) 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 window resize, are locally performed by the Low Level Interaction Component. In addition, in the case of spoken-utterances, this component can include mechanisms for confirmation allowing the user to intercept a wrong recognition.
The roles of the PAC-Amodeus components can be compared to a similar architecture devised for virtual worlds. In [17], dialogue is structured by a set of three level rules. The Specific Level rule set is linked to specific hardware. It corresponds to the LLIC component of the PAC-Amodeus model. The Generic Level rule set transforms events into more general interaction. It can be mapped onto the PTC. Finally, the Executive Level rule set manages tasks and thus corresponds to the DC.

We need now to show how concurrent processing and data fusion are performed within our architectural framework.

Concurrent processing of data

Concurrent processing of data is achieved at different levels of abstraction. Raw data is captured in the LLIC component by event handlers. There is one event handler per input device. Event handlers correspond to the strands in the ACME system [10]. They process in parallel.

Concurrency is also supported in the PTC which receives low level events (Interaction objects) from the LLIC and transforms them into more abstract interaction techniques. For example, a mouse click is transformed into the Select interaction technique. There is one abstracting process per supported modality. A modality, for instance a gestural language, may be supported by different physical devices (and so different events handlers), such as a mouse and a data glove combined with computer vision.

Finally, the multi-agent architecture of the Dialogue Controller offers an interesting conceptual framework to support concurrency. Agents can process data (i.e., Presentation objects) received from the PTC in parallel.

Data fusion

Non-multimodal systems transform data from Interaction objects to Presentation objects, then from Presentation objects to Conceptual objects, up to Domain objects and vice versa. These transformations must also be performed in a synergistic system, but in this case, the task may be more complex. The synergistic use of modalities implies fusion of data from different modeling techniques. Each technique is associated with a modality.

We have identified three levels of fusion: lexical, syntactic and semantic that can be mapped to the three conceptual levels defined by Foley et al. [18]. Lexical fusion corresponds to the Binding level which establishes the interface with the hardware primitives. Therefore lexical fusion is performed in the LLIC component. The syntactic and semantic fusions correspond respectively to the Sequencing and Functional levels. These fusions are thus handled by the component responsible for task-level sequencing: the dialogue controller.

Lexical fusion. Lexical fusion is performed in the LLIC. A typical example of lexical fusion may be found in the Macintosh where the shift key combined with a mouse click allows multiple selections. Lexical fusion involves only temporal issues such as data synchronization.

Syntactic and semantic fusion. The Dialogue Controller is responsible for syntactic and semantic fusions. Syntactic fusion involves the combination of data to obtain a complete command such as the “Insert a note” in the NoteBook system. Semantic fusion combines results of commands to derive new results. For instance, in VoicePaint, the combination of the command “Draw line” with the command “Modify color” results in a two color line. (These two commands can be specified simultaneously.)

Syntactic and semantic fusion requires a uniform representation: the melting pot object. As shown in Figure 5, a melting pot object is a 2-D structure. The structural parts correspond to the structure of the commands that the Dialogue Controller is able to interpret. Events generated by user's actions are abstracted within the PTC and mapped onto the structural parts inside the Dialogue Controller. These events may have different time-stamps. A command is complete when all of its structural parts are filled up by at least one piece of data. Multiple data for the same structural part may denote redundancy or reveal inconsistencies.

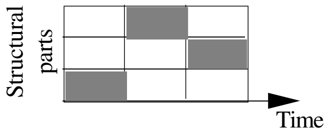

Figure 5: The melting pot object as a common representation for data fusion within the DC.

The non-sequential, hierarchical and distributed features of the multi-agent architecture adopted for the Dialogue Controller make it particularly well suited to perform fusion. Data is combined in parallel and incrementally along the levels of the hierarchy. The fusion mechanism is composed of a set of micro-fusions performed within each agent. The fusion process is based on two criteria: the time (e.g., data belonging to the same temporal window) and the structure of the objects to be combined. Furthermore, an agent may add new data from its own state, to the fusion process.

An example of fusion. Figure 6 illustrates a two-level fusion process for a graphics editor that supports speech and mouse gesture. In this example, the user says "put that there" and at the same time, uses the mouse to select the object to be moved and to indicate the destination in a distinct workspace. A workspace is a drawing area. As in most graphics editors, each workspace has a companion window, a palette that displays the graphics tools. By applying the heuristics rules described in [16], one obtains the architecture shown in Figure 6.

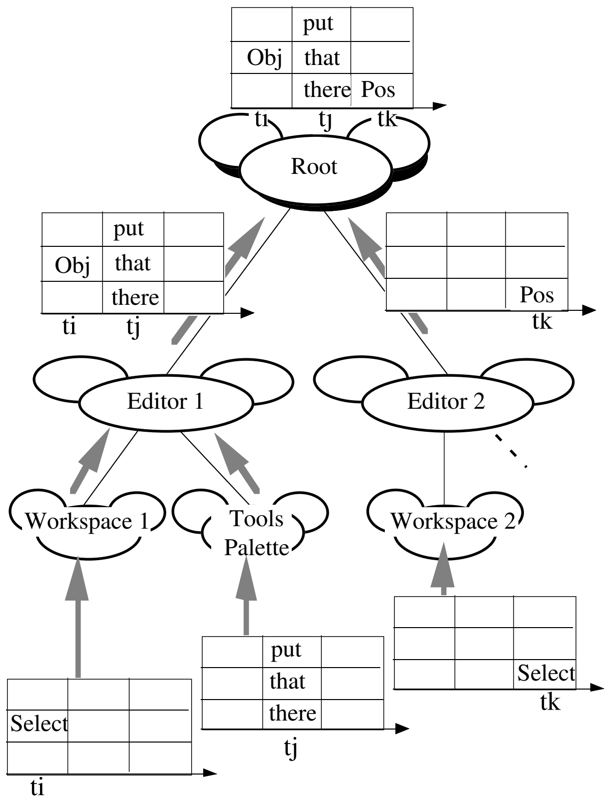

Figure 6: An example of different levels of fusion inside the hierarchy of PAC agents.

At the bottom of the hierarchy, agents Workspace1 and Workspace2 interpret the events that occurred in the drawing areas. Similarly, ToolPalette agent is in charge of the events issued in the palette. Editor agents, such as Editor1 and Editor2, combine information from lower levels into higher abstractions. For the particular example, the three agents Workspace1, ToolPalette, and Workspace2 each receive a melting pot object from the PTC. Each melting pot object corresponds to a user's actions. The agent Workspace1 translates the Select action into the selected graphical object Obj while in parallel, the agent Workspace2 translates the Select action into a position Pos. The cement agent, Editor1, then performs a first level of fusion by combining the "put that there" with the selected object. A second level of fusion is then performed by the Root agent to obtain the complete command to be sent to the functional core.

Abiding with the requirements for building synergistic systems, our model supports concurrent processing of data and offers a framework to perform data fusion. Furthermore our model supports immediate feedback and satisfies the flexibility criterion. Immediate feedback can be performed by each agent, even before full fusion is accomplished. Specific feedback can be generated at each level of fusion. A high degree of flexibility is achieved by this model for the following reasons:

- a physical hardware device may be changed by modifying the Low Level Interaction Component,
- a modality may be changed by modifying the Presentation Techniques Component,
- the Dialogue Controller is modality-independent. It depends only on the structural composition of the commands that the system supports.

SUMMARY AND CONCLUSIONS

We have presented a classification space that describes the properties of both input and output interfaces of multimodal systems. From the software perspective, this classification space highlights two main characteristics of such systems:

- concurrency of data processing, and
- data fusion.

The contribution of our classification space is two-fold:

- four salient classes of systems can be used as the extrema of a reference space,
- the reference space provides a way to characterize and reason about the I/O properties of interactive systems. In particular, it may be useful to compare the location of a command or the whole system devised at the design stage with the effective location measured through usability testing.

As a complement to the classification space, a software architecture model that supports concurency and data fusion is proposed. Three levels of fusion have been identified with the appropriate method to implement them. In addition, the model satisfies three crucial quality criteria: code re-usability, support for immediate feedback and flexibility. In the near future, we will continue to test and verify our results through the design of systems supporting multiple output modalities. We will also need to study how to enhance the robustness of the interaction with pragmatics and with an embedded user model.

ACKNOWLEDGMENTS
This work has been supported by project ESPRIT BR 7040 AMODEUS as well as by PRC Communication Homme-Machine, France. This paper was influenced by stimulating discussions with J. Caelen (ICP, Grenoble) and A. Gourdol and D. Salber (LGI). We wish to thank Alex Rudnicky, of CMU, for providing the facilities (and the enthusiasm) to make possible the implementation of the Notebook example. Many thanks to G. Serghiou for reviewing the paper, and to James L. Crowley for help with style and English Grammar.

REFERENCES

1. Kjelldahl L., Introduction. In Proc. 1st Eurographics Workshop, Stockholm, Sweden (April 18/19,1991), Springer Verlag, pp. 3-5.
2. Blattner M.M. and R.G. Dannenberg R.G. CHI'90 Workshop on multimedia and multimodal interface design. SIGCHI Bulletin 22, 2 (Oct. 1990), pp. 54-58.
3. Byte, Special Issue on Computing without Keyboard, (July 1990), pp. 202-251.
4. Coutaz J. Multimedia and Multimodal User Interfaces: A Taxonomy for Software Engineering Research Issues. In Proc. Second East-Weat HCI conference (St Petersburg, Aug. 1992), pp.229-240.
5. Webster B.F. *The NeXT Book*. Addison Wesley, New York, 1989.
6. Frohlich D.M. The Design Space of Interfaces. Multimedia Systems, Interaction and Applications. In Proc. 1st Eurographics Workshop, Stockholm, Sweden (April 18/19,1991), Springer Verlag, pp. 53-69.
7. Buxton W.A.S. Lexical and pragmatic considerations of input structures. ACM SIGGRAPH Computer Graphics 17 (Jan. 1983), pp. 31-37.
8. Card S.K., Mackinlay J.D. and Robertson G.G. A Morphological Analysis of the Design Space of Input Devices. ACM Transactions on Information Systems 9, 2 (April 1991), pp. 99-122.
9. Foley J.D., Wallace V.L. and Chan P. The Human factors of computer graphics interaction techniques. IEEE Computer Graphics and Applications 4,11 (Nov. 1984), pp. 13-48.
10. Anderson D.P., Govindan R. and Homsy G. Abstractions for Continuous MEdia in a network window system. Technical report UCB/CSD 90/596, Computer Science Division (EECS), University of California, Berkeley (Sept. 1990).
11. Gourdol A., Nigay L., Salber D. and Coutaz J. Two Case Studies of Software Architecture for Multimodal Interactive Systems: VoicePaint and Voice-enabled Graphical NoteBook. In Proc. IFIP Working WG2.7 Working Conference, Engineering for Human-Computer Interaction (Ellivuori, Aug. 1992).
12. Chalfonte B.L., Fish R.S. and Kraut R.E. Expressive richness: a comparison of speech and text as media for revision. In Proc. CHI'91 (April 27-May 2 1991), ACM Press, pp. 21-26.
13. Vainio-Larsson A. Evaluating the usability of user interfaces: research in practice. In Proc. INTERACT'90 (Amsterdam, 27-30 August 1990), Elsevier Science, pp. 323-328.
14. Hauptmann A.G. Speech and Gestures for Graphic Image Manipulation. In Proc CHI'89 Human Factors on Computing Systems (April 1989), ACM Press, pp. 241-245.
15. The UIMS Workshop Tool Developers A Metamodel for the Runtime Architecture of an Interactive System. SIGCHI Bulletin, 24, 1 (Jan. 1992), pp. 32-37.
16. Nigay L. and Coutaz J. Building User Interfaces: Organizing Software Agents. In Proc. ESPRIT'91 Conference (Bruxelles, Nov. 1991), pp. 707-719.
17. Lewis J. B., Koved L. and Ling D. T. Dialogue structures for virtual worlds. In Proc CHI'91 (April 27-May 2 1991), ACM Press, pp. 131-136.
18. Foley J.D., van Dam A., Feiner S.K. and Hughes J.F. *Computer Graphics, Principles and Practice*. Addison-Wesley, 1990.
19. Hill R.D. Supporting Concurrency, Communication and Synchronization dans Human-Computer Interaction-The Sassafras UIMS. ACM Transactions on Graphics 5, 2 (April 1986), pp. 179-210.
20. Wilson M. The first MMI2 Demonstrator, A Multimodal Interface for Man Machine Interaction with Knowledge Based Systems. Deliverable D7, ESPRIT project 2474 MMI2, Tech. report Rutherford Appleton Laboratory, Chilton Didcot Oxon OX11 0QX, RAL-91-093, 1991.
21. Lunati J.M. and Rudnicky A.I. Spoken Language Interfaces: The OM system. In Proc. CHI'91 Human Factors on Computing Systems (News Orleans, April 27-May 2 1991), ACM Press, pp. 453-454.
22. Coutaz J. PAC: an Implementation Model for Dialog Design. In Proc. Interact'87, (Stuttgart, Sept. 1987), H-J. Bullinger, B. Shackel ed., North Holland, pp. 431-436.