Multimodality from the User and System Perspectives

Joëlle Coutaz, Laurence Nigay, Daniel Salber
CLIPS-IMAG, BP 53, 38041 Grenoble cedex
email: {Joelle.Coutaz, Laurence.Nigay, Daniel.Salber}@imag.fr

Abstract. This article is concerned with the usability and implementation of multimodal user interfaces. We show how the usability of such systems can be characterized in terms of the relations they are able to maintain between the modalities they support. Equivalence, assignment, redundancy, and complementarity of modalities form an interesting set of relations relevant to usability assessment and software design. We use the notion of compatibility between user preferences and system properties to show how the CARE properties interact with user modelling to predict usability during the design of a system. In addition we demonstrate how experimental evaluations can be based on the CARE properties. We then depart from the HCI perspective to consider the implications of such properties on software design and techniques: we present PAC-Amodeus, a software architecture model, in conjunction with a generic fusion mechanism.

Keywords: Multimodal user interfaces, properties, usability, software architecture, software design, CARE properties, PAC-Amodeus.

1. Introduction

The combined use of multiple interaction techniques such as speech and gesture opens a new world of experience. Although the potential for innovation is high, the current understanding about how to design, build, and evaluate multimodal user interfaces is still primitive.

Within the ESPRIT Basic Research Amodeus project, we have developed scientific tools that help designers and implementers to reason about multimodality from both the system and the user's perspectives. For example, the MSM framework makes explicit the notion of multiple grains of parallelism as well as the fusion and fission of information at various levels of abstraction [Coutaz 93]. Parallelism and fusion are not novel aspects of interaction but the complexity engendered by multimodality makes these phenomena first class issues. In particular, they convey new properties that can be used to characterize multimodal user interfaces.

In this article, we propose the CARE properties as a simple framework for reasoning about multimodal interaction from both the user and the system perspectives: the Complementarity, Assignment, Redundancy, and Equivalence that may occur between the interaction techniques available in a multimodal user interface. The TYCOON framework offers another approach to the analysis of multimodal systems [Martin 95]. In TYCOON, a modality is modelled as a computational process similar to the interactor-based modelling technique developed in Amodeus [Paterno 94, Duke 94]. Multimodality is discussed in terms of various types of composition between modality processes. Although TYCOON is a useful computational model for reasoning about software design, it is not primarily driven by end-user concerns.

Figure 1 summarizes our scientific approach. The CARE properties whose formal definition is provided in Section 2, serve as a common framework for reasoning about the usage and usability of multimodal interaction techniques as well as for designing software. The user-centered CARE properties are discussed in Section 4 along two complementary perspectives: the predictive theory-based approach and the experimental observation of the user. The implications of the CARE properties on system design are presented in section 5. The discussion will be illustrated with MATIS [Nigay 94], a multimodal interactive system developed in collaboration with colleagues at Carnegie Mellon University. The main features of MATIS are presented in the following section.

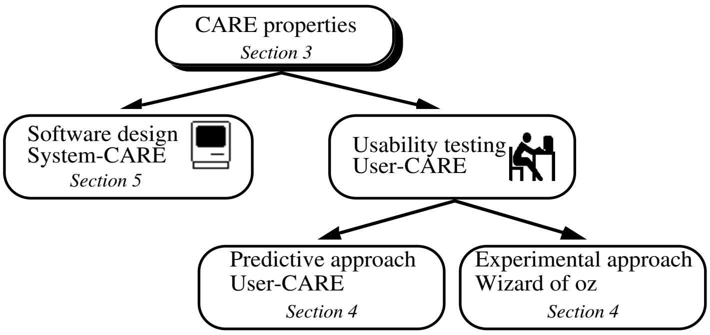

Figure 1: The CARE properties as a framework for reasoning about multimodality in software design and usability testing.

2. The case example: MATIS

MATIS (Multimodal Airline Travel Information System) allows a user to retrieve information about flight schedules using speech, direct manipulation, keyboard and mouse, or a combination of these techniques [Nigay 94]. Speech input is processed by Sphinx, a continuous speaker independent recognition engine developed at Carnegie Mellon University [Lunati 91]. MATIS supports both individual and synergistic use of multiple input modalities [Nigay 94]. For example, using one single modality, the user can say “show me the USAir flights from Boston to Denver” or can fill in a form using the keyboard. When exploiting synergy, the user can also combine speech and gesture as in “show me the USAir flights from Boston to this city” along with the selection of "Denver" with the mouse. MATIS does not impose any dominant modality: all of the modalities have the same power of expression for specifying a request and the user can freely switch between them.

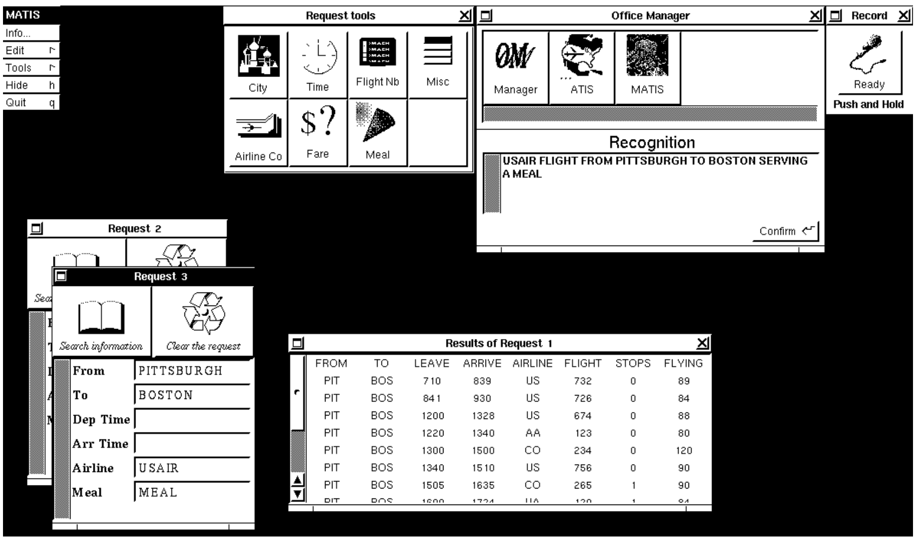

Figure 2: A snapshot from the MATIS application.

In addition, the system is also able to support multithreading: a MATIS user can disengage from a partially formulated request, start a new one, and later in the interaction process, return to the pending request. Figure 2 illustrates this facility: two requests (bottom left of the screen) are currently being formulated. To make an old request active, the user has to select the corresponding window. The request will come to the front and will constitute the new current context of interaction.

Although our earlier studies focused on input user interfaces such as the MATIS system, we wish to demonstrate the relevance of our results to the design of multimodal output interfaces. The CARE properties that we describe in the following section embeds input as well as output interfaces.

3. The CARE properties

3.1. Concepts

The formal expression of the CARE properties relies on the notions of state, goal, modality, and temporal relationships.

- A *state* is a vector of observables, that is, a set of properties that can be measured at a particular time to characterise a situation.
- A *goal* is a state that an agent intends to reach.
- A sequence of successive steps (or states) is called an *interaction trajectory*.
- An *agent*, e.g., a user, or the system, or a component of the system, is an entity capable of initiating the performance of actions.
- A *modality* is an interaction method that an agent can use to reach a goal. To model the expressive power of a modality m, that is, its capacity to allow an agent to reach state s' from state s in one step, we use the function *Reach(s, m, s')*.

This generic definition of a modality can be interpreted at different levels of refinement. For example, a modality could be specified in general terms as 'using speech', or more specifically as 'using a microphone'. Both of these interpretations are valid.

A *temporal relationship* characterises the use over time of a set of modalities. The use of these modalities may occur simultaneously or in sequence within a *temporal window*, that is, a time interval. Alternatively, only one modality from a set may be used. Let *Pick(s, m, s')* be a predicate that expresses the use of *m* among a set of modalities to reach *s'* from *s*.

Modalities of a set M are used simultaneously (or in parallel) if, within a temporal window, they happen to be active at the same time. Let *Active (m, t)* be a predicate to express that modality *m* is being used at some instant *t*.

The simultaneous use of modalities of a set M over a finite temporal window tw can be formally defined as:

$$Parallel\ (M, tw) \Leftrightarrow (Card\ (M) > 1) \wedge (Duration(tw) \neq \infty) \wedge (\exists t \in tw \cdot \forall m \in M \cdot Active\ (m, t))$$

where *Card (M)* is the number of modalities in set M, and *Duration(tw)* is the duration of the time interval *tw*.

Sequential events may have to occur within a temporal window to be interpreted as temporally related. If they occur outside this window, then they may be interpreted differently. Modalities M are used sequentially within a temporal window tw if there is at most one modality active at a time, and if all of the modalities in the set are used within tw:

$$Sequential\ (M, tw) \Leftrightarrow (Card\ (M) > 1) \wedge (Duration\ (tw) \neq \infty) \wedge (\forall t \in tw \cdot (\forall m, m' \in M \cdot Active(m, t) \Rightarrow \neg Active(m', t)) \wedge (\forall m \in M \cdot \exists t \in tw \cdot Active(m, t))$$

Temporal windows for parallelism and sequentiality need not have identical durations. The important point is that they both express a constraint on the pace of the interaction. For example, using multiple modalities simultaneously may be appropriate for the user but may require extra processing resources from the system side or imply a specific software architecture as shown in the following section. The absence of temporal constraints is treated by considering the duration of the temporal window as infinite.

Figure 3 shows the relationships between the concepts used in CARE: an agent having reached state s, may have (or has) the goal g to reach state s' by means of the non-empty set of modalities M={m1, m2 ... mn}. These modalities are linked by temporal relationship TR (TR::=|| | ; | 1, for parallelism, sequentiality, and selection of one interaction technique respectively), and constrained by a temporal window TW (TW::=<interval> | ∞). The symbol T at the upper left denotes the CARE property (T::=C | A | R | E).

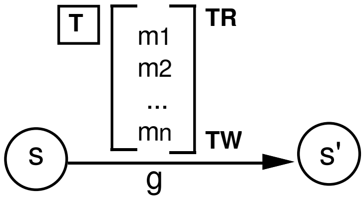

Figure 3: A notation for expressing CARE properties.

The CARE properties, which characterise four types of relationships between states and modalities, can be defined and illustrated with MATIS [Nigay 94]. Our discussion will concentrate on input (i.e., from the user to the system) although the definitions hold for output as well.

3.2. CARE properties: Formal Definition

Equivalence: Modalities of set M are *equivalent* for reaching s' from s, if it is necessary and sufficient to use any *one* of the modalities. M is assumed to contain at least two modalities. More formally:

$$Equivalence\ (s, M, s') \Leftrightarrow (Card(M) > 1) \wedge (\forall m \in M\ Reach\ (s, m, s'))$$

Equivalence expresses the availability of choice between multiple modalities but does not impose any form of temporal constraint on them. Figure 4 shows an example of equivalence between several modalities for specifying "Pittsburgh" as the destination of a trip. Users have a choice of speaking or typing the sentence "Flights to Pittsburgh", or keying "Pittsburgh" in the destination slot of the request form. Alternatively, they may interact with the Tool window and pick up "Pittsburgh" as a destination from the menu of known cities.

Assignment: Modality m is *assigned* in state s to reach s', if no other modality is used to reach s' from s. In contrast to equivalence, assignment expresses the absence of choice: either there is no choice at all to get from one state to another, or there is a choice but the agent always opts for the same modality to get between these two states. Thus we can define two types of assignment:

$$StrictAssignment\ (s, m, s') \Leftrightarrow Reach\ (s, m, s') \wedge (\forall m' \in M.\ Reach(s, m', s') \Rightarrow m'=m)$$

$$AgentAssignment\ (s, m, M, s') \Leftrightarrow$$
$$(Card(M) > 1) \wedge (\forall m' \in M.\ (Reach\ (s, m', s') \wedge (Pick\ (s, m', s')) \Rightarrow m'=m)\ )$$

In the case of an agent assignment, it is interesting to analyse the interaction trajectories to explain and justify its occurrence. In MATIS, window management is performed by direct manipulation only. In particular, speech cannot be used as an alternative. Therefore, the system imposes a strict assignment upon the user for window based tasks. Conversely, a user who always uses speech to specify trip destinations would turn the Equivalence offered by the system into an Agent Assignment. This issue will be developed further in Section 4. Equivalence and assignment both measure the choice available at some point in the interaction trajectory. Redundancy and complementarity go one step further by considering the combined use of multiple modalities under temporal constraints.

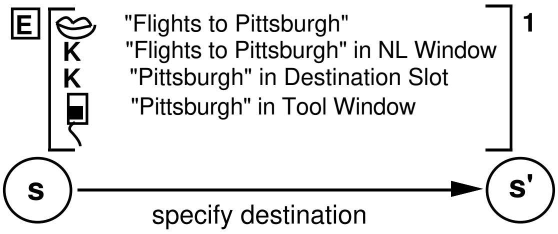

Figure 4: Examples of equivalence in MATIS. The “lip” symbol denotes speech, K represents the use of the keyboard, and the mouse symbol, a mouse selection.

Redundancy: Modalities of a set M are used *redundantly* to reach state s' from state s, if they have the same expressive power (they are equivalent) and if all of them are used within the same temporal window, tw. In other words, the agent shows repetitive behaviour without increasing its expressive power:

Redundancy (s, M, s', tw) $\Leftrightarrow$
Equivalence (s, M, s') $\wedge$ *(Sequential (M, tw)* $\vee$ *Parallel (M, tw))*

Redundancy can comprise two distinct temporal relationships – sequentiality and parallelism – which may have different implications for usability and software implementation. In particular, parallelism puts restrictions on the types of modalities that can be used simultaneously: modalities that compete for the same system or human resources cannot be activated in parallel. The agent can then only act sequentially if it can comply with the temporal constraints (i.e., it must act quickly for the multiple inputs to be treated as if they were parallel).

For example, the MATIS system is able to support parallel-redundancy between speech acts and any one of the other equivalent modalities presented in Figure 4. As shown in Figure 5, redundant typing in the NL window and in the destination slot competes for the same system and human resources. Therefore these two methods must be used sequentially (unless two keyboards and two users were available) and within the temporal window tw, which in turn must be kept within tw'. When the user does not satisfy the temporal constraint, MATIS creates a new request initiated with the extra “redundant” information. This system decision may not be in accordance with the user’s expectation or intention.

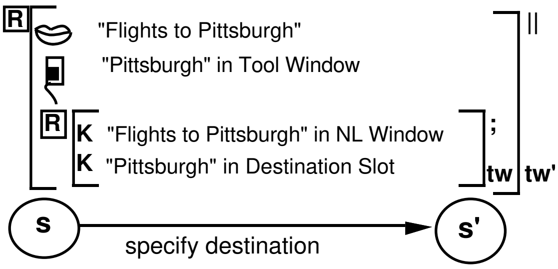

Figure 5: Example of redundancy in MATIS.

According to our formal definition, redundancy requires equivalence but equivalence only stipulates the existence, not the activation, of multiple methods. This asymmetry has implications for the system’s robustness. Suppose for example that a user is sending the same content concurrently via the equivalent modalities m and m'. If the system does not support redundancy for m and m', either it is unable to sense m while sensing m', or it senses both of them and may get confused. Therefore, software designers must have a clear architectural model as well as the appropriate software mechanisms to handle these issues appropriately.

Complementarity: Modalities of a set M must be used in a *complementary* way to reach state s' from state s within a temporal window, if all of them must be used to reach s' from s, i.e., none of them taken individually can cover the target state. To express this adequately, we need to extend the notion of reachability to encompass sets of modalities: REACH(s,M,s') means that state s' can be reached from state s using the modalities in set M.

Complementarity (s, M, s', tw)
$\Leftrightarrow$ $(Card(M) > 1) \wedge (Duration(tw) \neq \infty) \wedge (\forall M' \in \mathbf{P}M\ (M' \neq M \Rightarrow \neg REACH\ (s, M', s'))) \wedge$
$REACH\ (s, M, s') \wedge (Sequential\ (M, tw) \vee Parallel\ (M, tw))$

Deictic expressions, characterised by cross-modality references, are examples of complementarity. As shown in Figure 6, a MATIS user can type or speak the sentence "flights to this city" (or simply "flights to") and select a city name on the screen. Here, the sentence specifies the focus of interest (i.e., the destination of the trip) while the mouse selection denotes a location. These two modalities complement each other and must be combined to reach the intended goal. As with redundancy, complementarity may occur in parallel or sequentially within a temporal window. In contrast to redundancy, which does not favour any modality, complementarity may be driven by a dominant modality, which requires the use of others. Typically, in MATIS, deictic references in speech require the use of the mouse to point to a screen object.

Cross modality references may draw upon both complementarity and redundancy. For example, a MATIS user may say "flights to this city" while typing "Pittsburgh" in the destination slot of the request form. In this case, from the system's perspective, the speech act denotes the topic of interest while the typing action specifies both the topic and its value. The speech act, which is covered by the typing act in the destination slot, should be ignored by the system (i.e., the system should not wait for the resolution of the deictic reference). Again, the CARE properties have implications for system implementation.

Another source of complexity for software designers is that distinct actions produced within the same temporal window through different modalities are not necessarily complementary. In this case, fusion must not be performed. For example, a MATIS user may say "Flights to Pittsburgh" while selecting an irrelevant object on the screen. In section 5, we describe how these issues can be supported consistently through a reusable software mechanism.

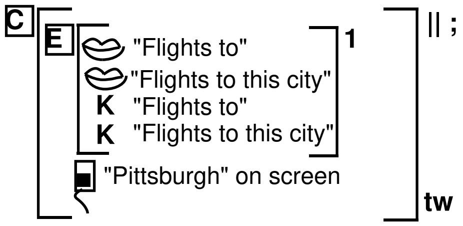

Figure 6: Examples of complementarity in MATIS. Complementarity can be parallel or sequential.

Our formal definitions of the CARE properties provide conceptual foundations for reasoning about multimodal interaction. To put them to work, we need to discuss issues such as coverage and refinement.

Coverage of a modality. Coverage has to do with the set of states for which a particular property holds. For example, in MATIS, the situations described in Figures 4 to 6 hold for any goal related to request specification. Coverage of a property over states can be used as a metric for assessing consistency. The CARE properties, which include the notions of modality and goal, can also be instantiated at multiple levels of refinement.

Modality refinement. Modality refinement in terms of devices and interaction languages is discussed in [Nigay 95]. To summarize,

- a *physical device* is an artefact of the system that acquires (input device) or delivers (output device) information. Examples of devices in MATIS include the keyboard, mouse, microphone and screen.
- An *interaction language* defines a set of well-formed expressions (i.e., a conventional assembly of symbols) that convey meaning. The generation of a symbol, or a set of

symbols, results from actions on physical devices. In MATIS, examples of interaction languages include pseudo-natural language and direct manipulation.

We define an *interaction technique* (or a modality) as the coupling of a physical device d with an interaction language L: <d, L>.

Considering MATIS,

- speech input is described as the couple <microphone, pseudo natural language NL>, where NL is defined by a specific grammar,
- written natural language input is defined as <keyboard, pseudo natural language NL> (a MATIS user can also type in NL sentences in a dedicated windows),
- graphic input is described in terms of <mouse, direct manipulation>, and
- graphic output corresponds to the couple <screen, tables>. (Flight schedules returned by MATIS are always presented in a tabular format.)

From these definitions, it is then possible to reason at a finer grain of concerns and consider the Complementarity, Assignation, Redundancy and Equivalence between physical devices as well as between interaction languages. For example in MATIS, NL sentences can be produced using either the microphone or the keyboard. The microphone and the keyboard are functionally equivalent regarding the goal "producing a NL sentence". Thus, in a noisy environment, the Matis user can easily switch to the keyboard.

Goal refinement. Typically, goals are recursively decomposed into subgoals. In our modelling technique, this decomposition is expressed as a refinement of interaction trajectories. Depending on the level of refinement, interaction trajectories can be viewed either as a one step encapsulation or as a sequence of steps. For example, in Figure 7 the intended goal is to reach a state where both the departure and the destination of the trip are specified. This goal may be seen as a single chunk or as an encapsulation of two subgoals: specify departure and specify destination.

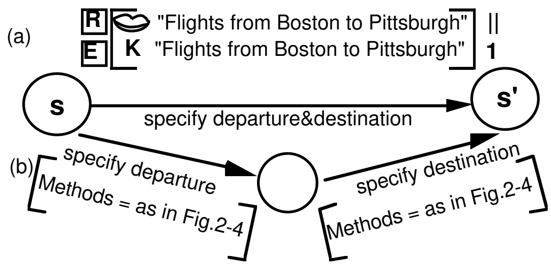

Figure 7: CARE and goal refinement.

The user may use one modality, saying the sentence “Flights from Boston to Pittsburgh”, or typing it into the speech recognition window, or doing both redundantly (case a). Alternatively, they could specify the departure using any method described in Figures 4 to 6 and then specify the destination (case b). In case (a), the user’s actions must be performed within some time interval and the system provides feedback once the temporal window has elapsed. In (b), the system provides feedback for each subgoal. Considering the description in Figure 5 at a high level, methods (a) and (b) are functionally equivalent. At a finer grain of analysis however, (a) and (b) differ in the interaction trajectory. As such, they may not be perceived as equivalent.

In summary, the CARE properties can be used at multiple levels of goal refinement. Designers can exploit the recursive nature of CARE to reason about multimodality at the appropriate level. For example, in the early stage of the life cycle, one may reason at a coarse grain to flesh out the most salient requirements about multimodality. Once the system is designed, one may need to go into more details to make sound predictive assessments as shown in the following section. Once the system is implemented, effective use of the system can be observed and interaction traces may be analyzed at fairly low level of details.

4. CARE and User-CARE properties

The CARE properties of the computer system have a counterpart in corresponding properties of the user: the CARE-like or User-CARE properties.

4.1. User-CARE properties

The user properties are concerned with the choice between different modalities for communicating with the computer. As above, our discussion will be confined to the choice of modality for communication in the input direction, that is, from the user to the system. Because of the user's circumstances – including her task, her background, her training, her knowledge, and the physical and interactive behaviour of the computer interface – the user may well have preferences as to how she communicates with the computer. A familiar example is that if the user is engaged in a task which occupies her hands, she may prefer to use speech. We refer to such preferences by the user, affecting her choice of input modalities, as *U-preferences*.

Certain patterns of U-preferences are worth identifying. (a) If only one modality is acceptable to the user, or if she has a strong preference for one particular modality, then we have a case of U-assignment. (b) If there exists a subset of the possible modalities which she prefers to all others, but between which she is indifferent, then we have a case of U-equivalence. (c) If the user prefers to employ two or more means of communication to convey the same information, then we have a case of U-redundancy. (d) If the user's preference is to use one modality for one aspect of the task and another modality for another aspect, then we have a case of U-complementarity. The crucial requirement on the design of the system is that its properties must be compatible with the user's U-preferences. We regard a system design as being *compatible* with the user's needs provided there exists at least one modality which is acceptable to both system and user.

Compatibility may be assessed using predictive evaluation techniques or using experimental approaches grounded on users observation. Predictive techniques such as discount usability testing [Nielsen 89], draw upon a large body of HCI heuristics. Currently, heuristics frameworks do not cover multimodal interaction properly. Alternatively, predictive evaluation may be based on a theory of cognition such as GOMS. In this section, we outline an early work on predictive assessment using the ICS and PUM theory-based modelling techniques developed within the Amodeus project (section 4.2).

As a complementary means of assessing the usability and usage of multimodality, we have developed NEIMO, a generic and flexible multiworkstation usability lab, to observe and analyze multimodal interaction experimentally [Salber 93] (Section 4.3).

4.2. Predictive assessment

In [Coutaz 95] the issue of compatibility is addressed by considering the requirements for each of the CARE properties to be compatible with each of the U-CARE properties. One approach is to consider what a user needs to know about a system in order to develop any of the U-preferences described above.

A technique such as Programmable User Modelling [Blandford 93] can be used to assess the likelihood that a user will be able to acquire the particular knowledge needed to develop the appropriate U-preference. Another approach is to consider the cognitive resources a user will require in order to use or choose between the modalities included in the various U-preference sets. To do so a general cognitive architecture such as ICS [Barnard 93] can be used. In [Coutaz 95] we further develop the notion of compatibility between user preferences and system properties to show how the CARE properties interact with user modelling to predict usability at the design stage of the software development.

4.3. Experimental assessment: the NEIMO usability lab

4.3.1. Principles
As shown in Figure 8, using the NEIMO platform involves a two-step process: the experimentation session followed by an analysis phase. In phase 1, a subject executes a set of scenarios on a dedicated workstation. In a different room, human factor experts observe the subject, make annotations, or simulate the missing functions of the system (e.g., speech recognition) using their own workstation. Meanwhile, behavioral data about the subject as well as experimenters' annotations are recorded automatically. NEIMO captures information at various levels of abstraction from keystroke level such as mouse events and speech acts, to high level tasks such as sending a fax.

In its current version, the NEIMO platform includes 4 Apple Macintosh Quadras connected by Ethernet. The user interfaces for the subject and the wizards workstations are prototyped with HyperCard. Apple Events are used as the standard communication mechanism but a specific tool has been developed for efficient transmission of video over Ethernet. (Sound is not yet transferred over the network.) Behavioral data are recorded using the QuickTime format.

In phase 2, behavioral data are used by specialists to assess the usability of the system. In the context of our research, the motivation is to elicit the usage of modalities according to the CARE framework.

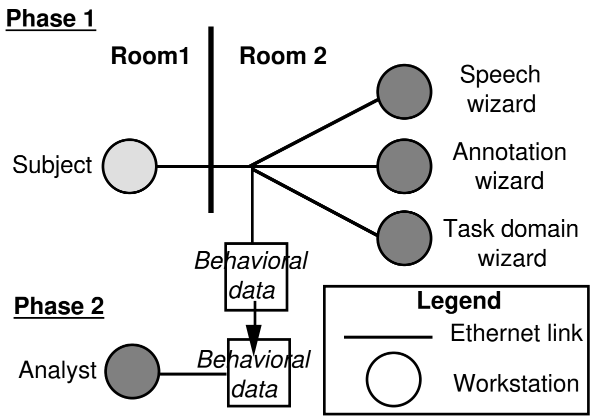

Figure 8: Configuration of the NEIMO platform.

Most usability platforms are not computer-supported. NEIMO, which is able to *digitally record behavior* at various levels of abstraction, opens the way to the development of automated analysis tools that alleviate the time-consuming manual analysis of a large body of data such as repetitive pattern of behavior.

In addition to observation and annotation, NEIMO supports Wizard of Oz experiments. Most existing Wizard of Oz systems support the observation of one modality only or are limited by technical constraints. NEIMO has been designed from scratch to support *multimodality*. A significant amount of effort has been dedicated to implementation issues to satisfy performance requirements.

NEIMO is *multiworkstation*, *generic and flexible*: 1) It supports any number of wizards; 2) It is organized around a reusable and extensible kernel of common services onto which specific user interfaces can be plugged; 3) Workstations are configurable at start up time: wizards roles (e.g., speech recognition, annotations, etc.) can be freely allocated among the workstations. In addition, data capture can be set up at the appropriate level of abstraction.

4.3.2. Illustration
Figures 9 and 10 show the screens of the speech-wizard and of the annotator to observe a subject performing telecommunication tasks (e.g., sending a fax and Vphoning) in a multimodal interaction setting (i.e., speech and direct manipulation). The speech-wizard translates multimodal commands such as "Call this person" into actions understandable by the system. To accomplish this, he can hear the subject talking and a miniature reproduction of the

subject's screen allows the speech-wizard to track the user's mouse and keyboard actions. If the subject makes a linguistic mistake such as uttering a wrong command name, the speech-wizard sends an error message through a dedicated tool. As shown in Figure 9, error messages are predefined and organized into categories (lexical&syntactic and domain-dependent errors), or may be customized on the fly. In the normal case, the speech-wizard simulates the subject's actions using direct manipulation on the miniature screen.

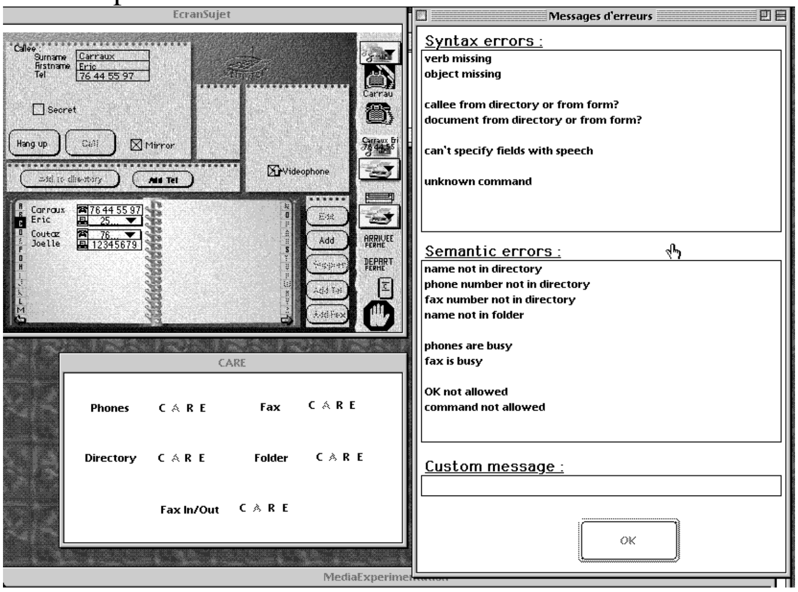

Figure 9: The screen of the Speech-Wizard. On the top left, a miniature of the subject's screen; on the bottom left, the CARE window shows the CARE properties that are currently authorized for the tasks of the experiment; on the right, the error messages tool.

The annotation-wizard who can observe the subject's behaviour through his own workstation (sound+miniature screen), can record comments describing the subject's hesitations (see Figure 10). This information, which complements the subject's wrong mouse clicks, will be pointed out by the analysis tool in the next phase.

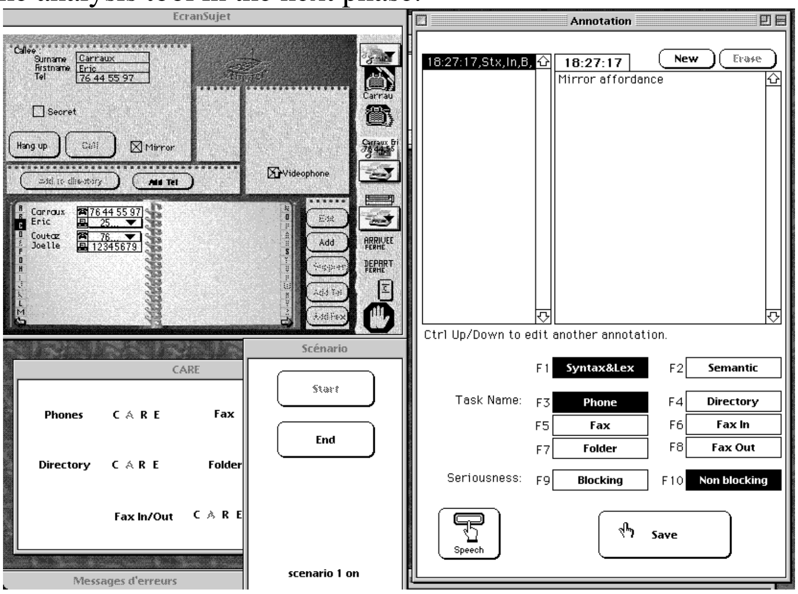

Figure 10: The screen of the annotation-wizard. On the top left, the miniature of the subject's screen; on the right, the form to create annotations. In this example, the problem is concerned with the lexical level of the interaction, it is related to the telephone task and is not blocking. The annotator can complement the "standard" attributes with specific comments: here, a problem of affordance with the mirror facility.

In its current form, the analysis tool provides quantitative data such as the duration of scenarios as well as statistical information through the facilities provided by a spreadsheet program. It does not yet support editing facilities nor the rendering of multimodal usage. It does however replay the set of scenarios (just like a VCR) and provides browsing facilities such as rewinding the "VCR to the previous lexical error" (See Figure 11). In addition, the tool shows a PERT diagram that makes tasks interleaving explicit. The diagram is enhanced with clickable "bubbles" that reveal the annotations recorded on the fly by the annotation-wizard.

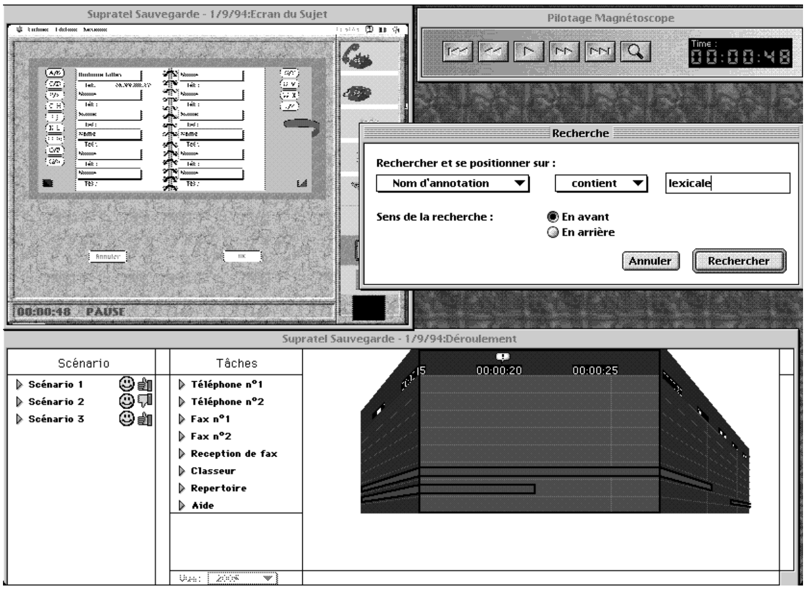

Figure 11: The screen of the analysis tool using the VCR metaphor. In the top left, the replay window that reproduces the subject's screen. On the top right, VCR buttons to control the navigation. At the bottom, the list of scenarios and for each scenario, a PERT diagram displayed on a perspective wall [Mackinlay 91] of the tasks executed during the scenario. A search window is currently opened to position the "VCR" on the next lexical error.

4.3.3. Perspectives

In the near future, we will conduct full-fledged experiments to study the relevance of multimodality for telecommunication tasks (preliminary results for drawing tasks show that in deictic expressions, pointing is often performed first [Catinis 95]). We also need to find the balance between digital and analog recording in order to conciliate precision, volume of recorded data, and potentiality for automatic evaluation. Having primarily developed NEIMO for capturing behavior, we need now to augment our analysis tool with new computation and visualization facilities.

Having analysed the CARE properties in the light of user-centered concerns, we now discuss their implications on software design.

5. CARE and System-CARE properties

Our technical solution for supporting the CARE properties draws upon our software architecture model: PAC-Amodeus [Nigay 93].

5.1. The PAC-Amodeus model

PAC-Amodeus is intended as a guide for developing software architectures at a conceptual level. It is a blend of the components advocated by the Arch model [UIMS 92] and the PAC refining process expressed in terms of agents [Coutaz 87]. As shown in figure 12, the Dialogue Controller of the Arch model is decomposed into a set of cooperative PAC agents. The refinement of the Dialogue Controller in terms of PAC agents has multiple advantages including explicit support for concurrency and data fusion. Data fusion occurs within the Dialogue Controller to build commands at a high level of abstraction. In particular, it is triggered when two modalities are used in a complementary way. A complete description of PAC-Amodeus can be found in [Nigay 93].

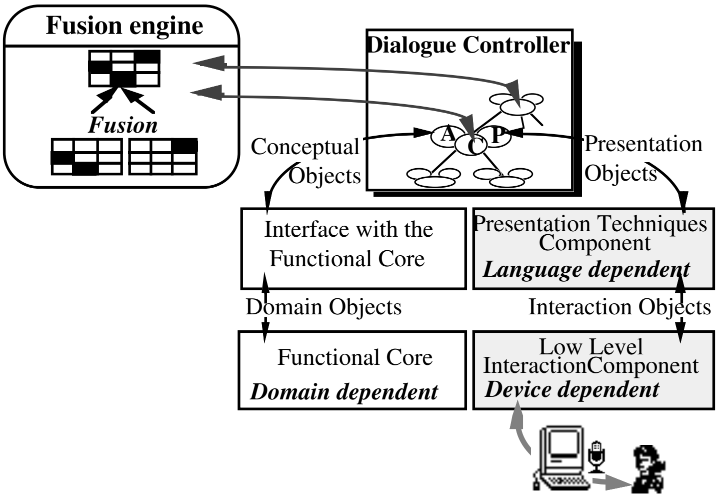

Figure 12: The PAC-Amodeus software components and their relationship with data fusion.

The fusion engine uses a uniform representation model to perform fusion: the melting pot. As shown in Figures 12 and 13, a melting pot is a 2-D structure. On the vertical axis, the "structural parts" model the structure of the task objects that the Dialogue Controller is able to handle. Events generated by user's actions are abstracted and mapped onto these structural parts. The Low Level Interaction and Presentation Techniques components are in charge of this process. Events are time-stamped: a mapped event defines a new column on the horizontal temporal axis. The structural decomposition of a melting pot is described in a declarative way outside the fusion engine. As a result, the fusion mechanism is domain independent: structures that rely on the domain are not “code-wired”. They are used as parameters for the fusion engine.

Using MATIS as an illustration, Figure 13 shows how redundancy and complementarity are handled by the fusion mechanism. In the redundancy example, the user has uttered the sentence “Flights to Boston” while selecting “Boston” with the mouse in the menu of known cities. The speech act is translated into the bottom left melting pot: at time t_i, the slot “to” is filled in with the value “Boston”. The melting pot next on the right results from the mouse selection. The fusion engine combines these two melting pots into a new one (top left). A similar reasoning applies to complementarity.

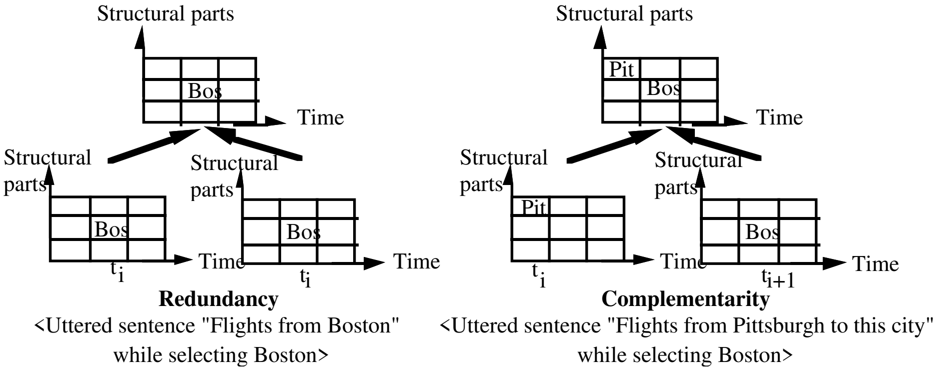

Figure 13: The effect of redundancy and complementarity on the fusion mechanism.

5.2. The fusion mechanism

As demonstrated above, our fusion mechanism is a reusable domain independent algorithm. This algorithm adopts an “eager” strategy: it always makes attempts to combine input data. This approach has the advantage of providing the user with immediate feedback before the functional core is accessed. The drawback is the possible occurrence of incorrect fusions which must be undone. Incorrect fusion occurs due to the different time scales required to process data specified through different modalities. As a result, the sequence of melting pots is not necessarily identical to the user's actions sequence. For example, in MATIS melting pots that correspond to direct manipulation expressions are built faster than those from voiced utterances. We will show how our engine deal with undesirable fusions.

The criteria for triggering fusion are threefold: the logical structure of commands, time, and context. When triggered, the engine applies three types of fusions in the following order: microfusion, macrofusion, and contextual fusion.

- Microfusion is performed when input data is structurally complementary and very close over time: i.e., microfusion combines inputs if they have been produced in parallel or in pseudo-parallelism. (Intersection of time intervals.)

- *Macrofusion* is performed according to the same criteria as microfusion but combines data that belong to a given temporal window. (Temporal proximity.) Macrofusion implies proximity of time intervals as opposed to microfusion which implies the intersection of time intervals.

- *Contextual fusion* is performed according to the structure of the data to be combined and the current context. For example in MATIS, the context corresponds to the current request. Contextual fusion combines new input data with the current request if their respective structures are compatible.

Having presented the driving principles of the fusion mechanism, we now focus on the technical details. Our fusion algorithm has been implemented in C and embedded in a PAC-Amodeus architecture for MATIS. We first introduce the metrics associated with each melting pot. We then explain how microfusion is implemented and show how the fusion mechanism supports redundancy and complementarity. Finally, we present the management of the set of melting pots and the exchanges of melting pots within the hierarchy of PAC agents.

5.2.1. Metrics for a melting pot

Figure 14 portrays the metrics that describe a melting pot m_i:

$m_i = (p_1, p_2, \ldots, p_j, \ldots, p_n)$: m_i is comprised of n structures p1, p2, ...pn.
$info_{ij}$: piece of information stored in the structural part p_j of m_i.
$Tinfo_{ij}$: time-stamp of $info_{ij}$.
$Tmax_i$: time-stamp of the most recent piece of information stored in m_i.
$Tmin_i$: time-stamp of the oldest piece of information stored in m_i.
$Temp\_win_i$: duration of the temporal window for m_i.
Δt: Remaining life span for m_i.

A melting pot encapsulates a set of structural parts $p_1, p_2, \ldots p_n$. The content of a structural part is a piece of information that is time-stamped. Time stamps are defined by the LLIC when processing user's events. The system computes the boundaries (Tmax and Tmin) of a melting pot from the time stamps of its informational units :

So, for $m_i = (p_1, p_2, \ldots, p_j, \ldots, p_n)$, $Tmax_i = Max(Tinfo_{ij})$ and $Tmin_i = Min(Tinfo_{ij})$

The temporal window of a melting pot defines the temporal proximity (+/- Δt) of two adjacent melting pots: When Δt reaches a threshold (i.e., a pre-specified critical value), the engine undertakes a macrofusion. Temporal windows are used to trigger macrotemporal fusion.

So, for $m_i = (p_1, p_2, \ldots, p_j, \ldots, p_n)$, $Temp\_win_i = [Tmin_i - \Delta t, Tmax_i + \Delta t]$

The last metrics used to manage a melting pot is the notion of life span, Exp_i:

$Exp_i = Tmax_i + \Delta t = Max(Tinfo_{ij}) + \Delta t$. This notion is useful to remove a melting pot from the set of candidates for fusion.

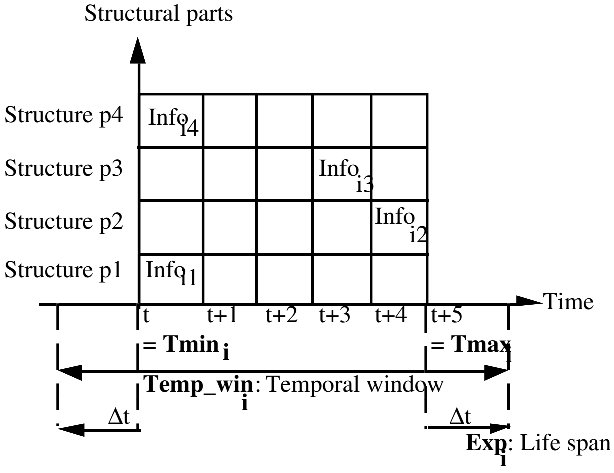

Figure 14: Metrics used to define melting pot m_i.

5.2.2. The mechanism

The fusion mechanism is driven by a set of rules.

Rule 1 deals with microfusion. Because the strategy is “eager”, microfusion is first attempted and triggered on the arrival of a new melting pot. Since it models a user’s action at instant t', this melting pot is composed of one column only. Rule 1 makes it explicit the occurrence of microfusion: if the content of the new melting pot ($col_{i't'}$) is complementary with a column (col_{it}) of an existing melting pot (m_i) and if the time-stamps of this column is close enough to t' (i.e., within Δmicrot), then microfusion is performed.

<u>Rule 1</u> **Microfusion**
Given:

- $col_{it} = (p_1, p_2, \ldots, p_j, \ldots, p_n)$: one column at time t of an existing melting pot m_i.
- $col_{i't'}$ = (p'1, p'2, ..., p'j, ..., p'n): a one column melting pot $m_{i'}$ produced at time t'
- $i \neq i'$

col_{it} and $col_{i't'}$ are combined if:

- they are complementary: Complementary (col_{it} , $col_{i't'}$) is satisfied if:

$\forall k \in [1..n] : \exists\, info_{ik} \ \Lambda \ \neg\, (\exists\, info_{i'k})$

- their time stamps are temporaly close in the following way:

Close (col_{it}, $col_{i't'}$) is satisfied if: $t' \in [t-\Delta microt, t+\Delta microt]$

Microfusion may involve undoing a previous fusion. This exception case is illustrated in Figure 15 that shows the fusions performed in the context of the following example: the user has already specified the destination slot (i.e., Denver) as well as the departure slot (i.e., Boston) of the current request. The result of this specification is modelled in Figure 15 as the melting pot m_1. The user then utters the sentence "Flights from Pittsburgh" while selecting "TWA" using the mouse. Because mouse clicks are processed faster than speech input, the mouse selection is first received by the Dialogue Controller. The mouse click is modelled as the melting pot m_2 which contains [TWA]. The new coming melting pot m2 is combined with m1 by contextual fusion. The result of this contextual fusion is represented in Figure 15 as m_1' which contains [BOS, DEN, TWA]. Meanwhile, melting pot m_3 which corresponds to the sentence "Flights from Pittsburgh", is received by the Dialogue Controller. The current set of candidates for fusion is now {m1, m2, m3}. Because the time intervals of m_2 [TWA] and m_3 [PIT] overlap, they are combined by microtemporal fusion and m2 becomes [PIT, TWA] (rule 1 applies). The previous contextual fusion [BOS, DEN, TWA] is undone. This gives birth to two melting pots and from now on, the user has elaborated two requests.

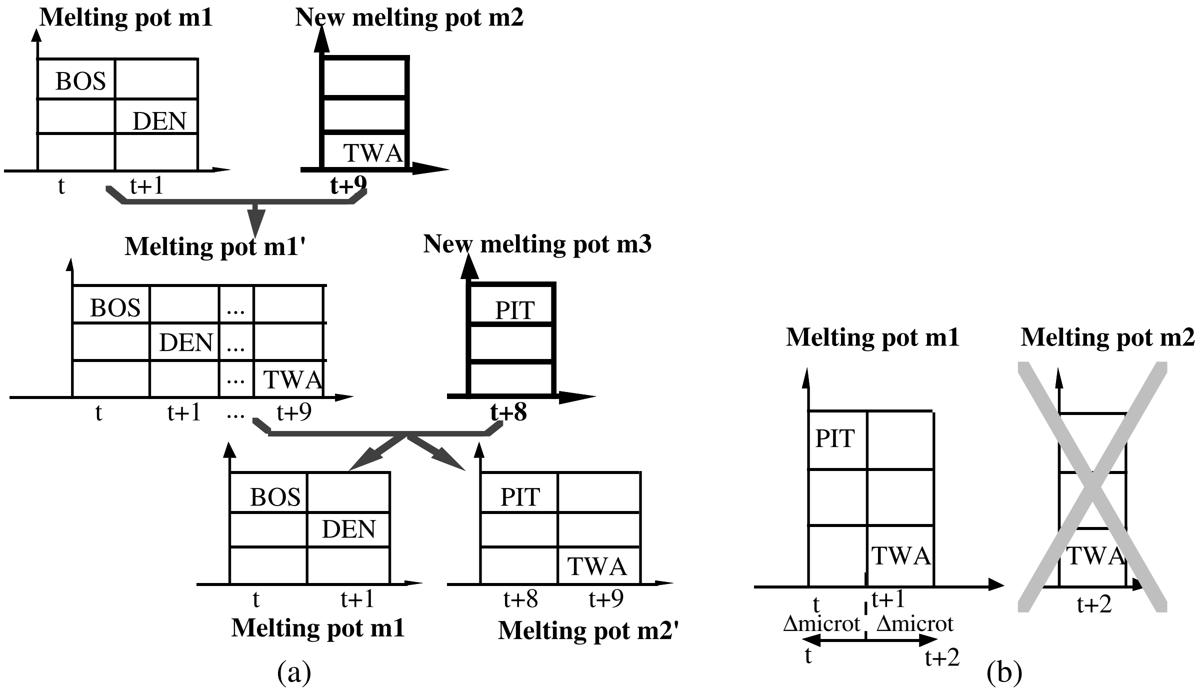

Figure 15: Interacting with MATIS:
(a) undoing fusion due to microtemporal fusion. (b) an example of redundancy.

Before applying the rule on Microfusion, the mechanism checks for redundancy using Rule 2. Redundancy is defined as two close columns that contains the same information. Figure 15 gives an example of redundancy.

<u>Rule 2</u> **Redundancy**
Given:

- $col_{it} = (p_1, p_2, \ldots, p_j, \ldots, p_n)$: one column at time t of an existing melting m_i
- $col_{i't'} = (p'1, p'2, \ldots, p'j, \ldots, p'n)$: one column at time t' of a new melting pot $m_{i'}$
- $i \neq i'$

col_{it} and $col_{i't'}$ are redundant if:

- they contain the same information in the same slots:

Redundant (col_{it}, $col_{i't'}$) is satisfied if: $\forall k \in [1..n]$:

$\exists\ info_{ik}\ \Lambda\ \exists\ info_{i'k}\ \Lambda\ info_{ik} = info_{i'k}$

$\Lambda\ \forall k' \in [1..n] :\ \neg (\exists\ info_{ik'})\ \Lambda\ \neg (\exists\ info_{i'k'})$

- their time stamps are temporaly close: Close (col_{it}, $col_{i't'}$) is satisfied if:

$t' \in [t-\Delta microt, t+\Delta microt]$

Macrofusion is driven by rules similar to those used for microfusion where Δmicrot is replaced by temporal windows. Whereas time has a primary role in micro- and macro- fusions, it is not involved in contextual fusion as illustrated in Figure 15. As described above, contextual fusion is the last step in the fusion process. The driving element for contextual fusion is the notion of context. In MATIS, contexts are in a one-to-one correspondence with requests. There is one context per request under specification and the current request denotes the current context. (The user may elaborate multiple requests in an interleaved way.) When a melting pot is complete (all of its informational units have been received), and its life span expectancy Exp_i expires, it is removed from the set of candidates for fusion. A melting pot removed from the fusion pool is sent to a PAC agent of the Dialogue Controller for further processing. Rule 3 expresses these conditions formally. Exp_i is used for making sure that incorrect fusions have not been performed: when a melting pot is complete, the engine keeps it for a while in the pool of candidates in case the next new melting pots trigger "undo" fusions.

<u>Rule 3</u> **Conditions to remove a melting pot from the list of candidates for fusion:**

Melting pot $m_i = (p_1, p_2, ..., p_j, ..., p_n)$ is removed if:

• m_i is complete: $\forall p_j \in m_i,\ \exists\ info_{ij}$
• and its span life is over: current date = Exp_i

We have shown the generic nature of the fusion mechanism using the criteria of time and structural complementarity. Each melting pot processed by the fusion mechanism may have any number of structures (e.g., lines) that are filled independently. The PAC-Amodeus model along with the fusion mechanism define a reusable platform for implementing input multimodal interfaces. Our natural next step is to study systems that support multiple languages and devices for output. This may lead to the development of a "fission" mechanism as introduced in MSM [Coutaz 93]. Such a fission mechanism can be based on the melting pot representation: By considering the example of Figure 13, and by inverting the direction of the arrows, we can sketch how the fission mechanism can work to support complementarity and redundancy in output interfaces. Several melting pots are deduced from a single one and each derived melting pot is made perceivable by the user through different output interaction techniques.

6. Summary

We have formally defined a set of properties, the CARE properties, useful in three complementary ways. The CARE properties can be exploited:

1) to predictively assess the usability and usage of multimodal user interfaces using theory-based cognitive models such as PUM and ICS,
2) to structure usability testing experiments as in the NEIMO usability lab,
3) to constrain software architecture modelling of interactive systems.

We have developed a generic and extensible platform, NEIMO, that captures behavior at various levels of abstraction, and we have designed a software architectural model, PAC-Amodeus, augmented with a fusion mechanism to support the CARE properties.

Acknowledgements

This work has been supported by project ESPRIT BR 7040 AMODEUS2, by PRC Communication Homme-Machine, and by CNET France Telecom. Special thanks to Eric Carraux for the development of NEIMO and to Lionel Villard for the implementation of the perspective wall within the analysis tool.

References

[Barnard 93]
P.J. Barnard, J. May, Cognitive Modelling for User Requirements, Computers, Communication and Usability, P. F. Byerley, P. J. Barnard, et J. May (éd.), Elsevier, 1993.

[Blandford 93]
A. Blandford, R. M. Young, Developing Runnable User Models: separating the problem solving techniques from the domain knowledge, People and Computers VIII, J. Alty, D. Diaper and S. Guest (éd.), Cambridge University Press, 1993.

[Catinis 95]
Catinis, L & Caelen, J. Analyse du comportement multimodal de l'usager humain dans une tâche de dessin; to appear in Proc. IHM'95, oct. 1995.

[Coutaz 87]
J. Coutaz, PAC, an Object Oriented Model for Dialog Design, Interact'87 Proceedings, North Holland, Stuttgart, Sept., 1987, pp. 431-436.

[Coutaz 93]

J. Coutaz, L. Nigay, D. Salber, The MSM framework: A Design Space for Multi-Sensori-Motor Systems, EWHCI'93 Proceedings, East/West Human Computer Interaction, Moscow, August, 1993 (Lecture notes in Computer Science, Vol. 753, 1993, pp. 231-241).

[Coutaz 95]

J. Coutaz, L. Nigay, D. Salber, A. Blandford, J. May, R. Young, Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The CARE Properties, INTERACT'95 Proceedings, 1995, pp. 115-120.

[Duke 94]

D. Duke, M. Harrison, "Folding Human Factors into Rigourous Development", in the Proc. of *Eurographics Workshop "Design, Specification, Verification of Interactive Systems"*, F. Paterno' Ed., 1994, pp. 335-352.

[Lunati 91]

J-M Lunati, A. Rudnicky, Spoken Language interfaces: The OM system, CHI'91 Proceedings, ACM Press, New Orleans, April 27-May 2, 1991, pp. 453-454.

[Mackinlay 91]

J. Mackinlay, G. Robertson, S. Card. The Perspective Wall: Detail and Context smoothly integrated, In Proc. CHI'91, ACM, 1991, pp. 173-179.

[Martin 95]

J.-C. Martin, Coopératrions entre modalités et liage par synchronie dans les interfaces multimodales, PhD dissertation, TELECOM, Paris.

[Nielsen 89]

J. Nielsen, "Usability Engineering at a Discount", in Designing and Using Human Computer Interfaces and Knowledge-Based Systems, Salvendy & Smith Eds., Elsevier North Holland, 1989, pp. 394-401.

[Nigay 93]

L. Nigay, J. Coutaz, A design space for multimodal interfaces: concurrent processing and data fusion, INTERCHI'93 Proceedings, ACM Press, Amsterdam, May, 1993, pp. 172-178.

[Nigay 94]

L. Nigay, Conception et modélisation logicielles des systèmes interactifs : application aux interfaces multimodales, PhD dissertation, Grenoble University, 1994, 315 pages.

[Nigay 95]

L. Nigay, J. Coutaz, A Generic Platform for Addressing the Multimodal Challenge. CHI'95 Proceedings, ACM Press, Denver, May, 1995, pp. 98-105.

[Paterno 94]

F. Paterno', A. Leonardi, S. Pangoli, "A Tool Supported Approach to the Refinement of Interactive Systems", in the Proc. of *Eurographics Workshop "Design, Specification, Verification of Interactive Systems"*, F. Paterno' Ed., 1994, pp. 85-96.

[Salber 93]

D. Salber, J. Coutaz, Applying the Wizard of Oz Technique to the Study of Multimodal Systems, EWHCI'93 Proceedings, East/West Human Computer Interaction, Moscow, August, 1993 (Lecture notes in Computer Science, Vol. 753, 1993, pp. 219-230).

[UIMS 92]

The UIMS Tool Developers Workshop, A Metamodel for the Runtime Architecture of an Interactive System, SIGCHI Bulletin, 24, 1, Jan., 1992, pp. 32-37.