Multifeature Systems: The CARE Properties and Their Impact on Software Design

Laurence Nigay and Joëlle Coutaz

Multifeature user interfaces support multiple interaction techniques which may be used sequentially or concurrently, and independently or combined synergistically (Nigay, Coutaz 1993a). New interaction aspects must be considered, such as the fusion and fission of information, and the nature of temporal constraints.

The availability of multiple interaction techniques opens a new world of experience, but our understanding of how they relate to each other is still unclear. We propose here a unified framework based on the notions of interaction language and physical device. The framework illuminates the relationship between interaction languages and physical devices. Such relationships are useful for eliciting design criteria, for classifying existing multifeature systems (Nigay 1994) and for evaluating the usability of a system. In this paper, we focus on usability aspects and show how the usability of a system can be correlated with the relationships that the system is able to maintain between the interaction languages and the devices it supports. We then depart from the HCI perspective to consider the implications of such relationships on software techniques and tools.

We have developed NoteBook, a voice-enabled electronic notebook (Nigay 1993) and MATIS, a Multimodal Airline Travel Information Systems (Nigay 1994) (Nigay, Coutaz 1993b). These systems handle both speech, mouse and keyboard inputs. Speech input is processed by Sphinx, a continuous speaker independent recognition engine (Lunati, Rudnicky 1991). MATIS allows a user to retrieve information about flight schedules using speech, direct manipulation, keyboard and mouse, or a combination of these techniques (Nigay 1994). MATIS supports both individual and synergistic use of multiple input modalities. 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. This is in contrast to for example the GEORAL system presented in (Siroux et al. 1996), a touristic information retrieval system, in which speech is the dominant modality. In this chapter, MATIS is used to illustrate the discussion.

Although our earlier studies focused on input user interfaces, we wish to demonstrate the relevance of our results to the design of multimodal output interfaces. The framework that we describe in the following two sections embeds input as well as output interfaces. The next section introduces a software architecture model as a guideline in designing the software of multimodal input and output interfaces. We then present a generic software, called fusion mechanism, which handles user's inputs specified through different interaction techniques. We note though that the fusion mechanism is a tool for developing input user interfaces only.

The Design Space: Physical Devices and Interaction Languages

In his theory of action, Norman structures the execution gulf into a semantic and an articulatory distance that the user needs to cover in order to reach a particular goal (Norman 1986). A similar reasoning holds for information acquisition. This user-centered approach pays little attention to the processing steps that occur within the computer system. Our Pipe-lines model makes these stages explicit (Nigay 1994). By so doing, we extend Norman's theory in a symmetric way within the computer system. As shown in Figure 1, information acquired by input digital channels (physical devices) is transformed through multiple process activities. This sequence of input transformations forms the interpretation function (input interface). In the other direction (output interface), internal information (e.g., system state) is transformed to be made perceivable to the user. This sequence of output transformations defines the rendering function. The interpretation and the rendering functions can be both characterized with four intertwined ingredients: level of abstraction, context, fusion/fission, and parallelism. These dimensions of the MSM framework are fully described in (coutaz, nigay & salber 93). In particular the notion of level of abstraction expresses the degree of transformation that the interpretation and rendering functions perform on information. And this "level of abstraction" dimension of the MSM framework enables us to distinguish multimedia and multimodal systems: a multimodal system is a multimedia system with a high capacity of interpretation and/or rendering. The first definition of multimodality introduced in (Gellersen 1996) is in accordance with our definition : multimodal = multimedia + media interpretation.

The Pipe-Lines model highlights three levels of abstraction in the interpretation and rendering functions:

- physical action in relation with a physical device,

- informational unit in relation with an interaction language,
- system action in relation with a task.

Two relevant concepts therefore emerge from this model: the notion of physical device and that of interaction language. Interestingly, these concepts cover the semantic and articulatory distances of Norman's theory. They are resources and knowledge that the system and the user need to share to accomplish a task successfully.

Definitions

A *physical device* is an artifact or an organ necessary to a system or a human in order to acquire (input device) or deliver (output device) information. Examples include keyboard, loudspeaker, microphone, ears and mouth.

A *physical action* is an action performed either by the system or the user on a physical device. Examples include highlighting information on the screen, synthesizing sound, pushing a mouse button and uttering a sentence.

An *interaction language* is a language used by the user or the system to exchange information. A language defines the set of all possible well-formed expressions, i.e., the conventional assembly of symbols, that convey meaning. The generation of a symbol or a set of symbols, results from a physical action. Examples include pseudo-natural language and direct manipulation language.

An *informational unit* is an atomic unit that conveys meaning. An informational unit is related to an interaction language which defines a set of informational units. Examples include part of command and selection of an icon.

An *interaction technique* is defined by the couple : (physical device, interaction language).

Adopting Hemjslev's terminology (Hemjslev 1947), the physical device determines the substance (i.e., the non analyzed material) of an expression whereas the interaction language denotes its form or structure. For example:

- speech input is characterized by the couple (device = microphone, language = the pseudo natural language NL), where NL is defined by a grammar,
- graphic output is characterized by the couple (device = screen, language = tables).

Modality as defined in Bernsen's modality theory (Bernsen 1996) corresponds to an interaction technique. The theory includes 70 modalities including dynamic analog graphics. But the theory does not consider the device and language concepts in defining a modality. The information mapping (IMAP) methodology (Bernsen 1996) focuses on the choice of a modality starting from the concepts related to the task. The IMAP methodology establishes links between the task level of Pipe-Lines and both the interaction language and physical device levels in the case of the rendering function (output interface). A similar approach can be found in (Karagiannidis, Koumpis, & Stephanidis 1996): a methodology is proposed to select a modality according to the information content to be rendered and the interaction context. Again rules are proposed to establish links between the task level and both the language and device levels of Pipe-Lines. One advance in the latter approach is that it provides temporal combinations of modalities. In the following section we also consider relationships between interaction techniques which are orthogonal to the temporal combination.

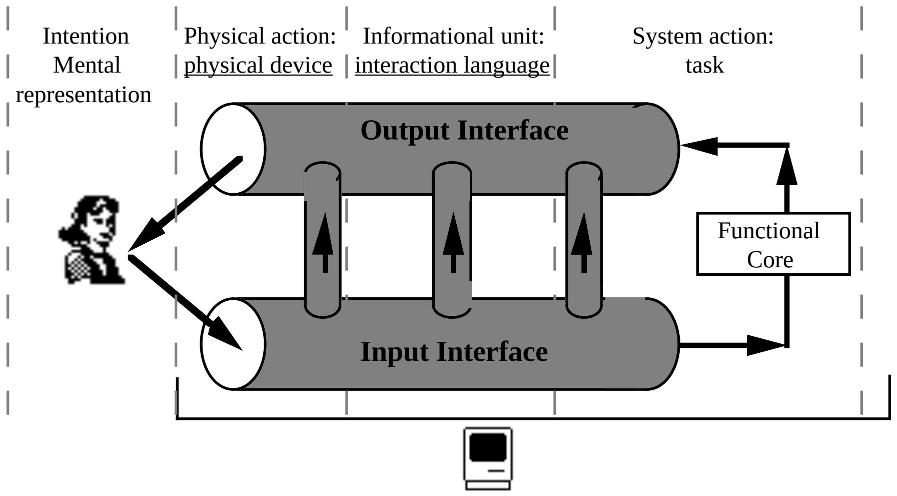

Figure 1: The Pipe-Lines model.

Usability and Relationships between Interaction Languages and Devices

In (Abowd, Coutaz & Nigay 1992), we discuss the usability of a system in terms of two properties: interaction flexibility and interaction robustness. Interaction flexibility covers the multiplicity of ways with which the user and the system exchange information. Interaction robustness denotes the features of the interaction that support the successful achievement and assessment of goals. We here propose the CARE properties as a simple framework for reasoning about multimodal interaction from both the user and the system perspectives: i.e., the Complementarity, Assignment, Redundancy, and Equivalence that may occur between the interaction techniques available in a multimodal user interface. The notions of equivalence, assignment, redundancy, and complementarity have been primarily introduced in the TYCOON design space (Martin, Beroule 1996). We define these four notions as relationships between devices and interaction languages and between interaction languages and tasks (the three levels of the Pipe-Lines model): the CARE properties. These properties affect flexibility and robustness.

The TYCOON framework offers another approach to the analysis of multimodal systems (Martin, Beroule 1996). In TYCOON, a modality is modeled as a computational process similar to the interactor-based modeling technique developed in Amodeus (Paterno & Mezzanotte 1996), (Duke & Harrison 1994). 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.

In the following paragraphs we define and illustrate using examples *the system CARE properties* (the CARE properties supported by the system) applied to tasks, interaction languages and physical devices. We then show that these properties affect both flexibility and robustness.

The System CARE Properties

As shown in figure 2, the CARE properties involve both devices and languages. In addition, they can be either permanent or transient, and either total or partial. In this context, a relation is permanent if it holds for any state of the system ; otherwise, it is said to be transient. A relation is total if it holds for any task supported by the system. It is partial when the relation is true for a subset of the task space.

Having presented the general picture of our framework, we need now to define the relations more formally. To do so, we will consider the notions of system state and set of tasks to express the coverage of the relations over time and over the user's task space.

Language equivalence: L-Equivalence(L, s, T)

Interaction languages of a set L are equivalent over a state s and a non empty set T of tasks, if all of the tasks in T can be expressed using either one of the languages in L. Equivalence is permanent if the relation holds for any state. Equivalence is total if the relation holds for all of the tasks that can be performed with the system. In MATIS, the user can specify a request using direct manipulation or natural language. For example, the departure time of a flight may be specified by clicking "morning" on the screen or by saying "arriving in the morning." Direct manipulation and natural language are permanently and totally equivalent for tasks that are related to the specification of requests.

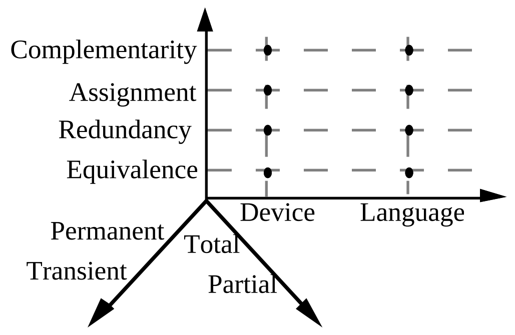

Figure 2: A framework to characterize multifeature user interfaces with relations between interaction languages, physical devices and tasks.

Language Assignment: L-Assignment(l, s, T)

An interaction language l is assigned in state s to a set of tasks T, if there exists no other interaction language equivalent to l over s and T. Assignment is permanent if the relation holds for any state. Assignment is total if the relation holds for all of tasks that can be performed with the system. For example, in MATIS, window manipulation such as cut and paste tasks can be performed using direct manipulation language only. In particular, speech is not available for any task other than request specification tasks.

Language Redundancy: L-Redundancy(L, s, t)

Interaction languages of a set L can be used redundantly in some state s for a task t, if these languages are equivalent over s and t, and if they can be used simultaneously to express t. In MATIS, the user can articulate the sentence “flights from Boston” while selecting “Boston” with the mouse in the menu of known cities. Here, speech and direct manipulation are used redundantly. As another example drawn from the literature (Neal et al. 1988), a wall is represented redundantly by the system via a red line (graphics interaction language) and the vocal message “mind the red wall!” (natural interaction language). Redundant usage has been identified in the WOZ experiment carried out by Siroux et al (Siroux et al. 1996). In (Dowell, Shmueli & Salter 1996) redundant usage of text and speech for outputs are refined into two categories: verbatim reinforcement (text and speech are exactly the same) and priming reinforcement (text is an abbreviated version of the spoken sentence).

Language Complementarity: L-Complementary(L, s, T)

Interaction languages of a set L are complementary over a state s and a non empty set T of tasks, if T can be partitioned such that for each partition Tp of T, there exists a language l of L assigned to Tp over s. Language complementarity is best illustrated by coreferential expressions. For example, in MATIS, natural language and direct manipulation are complementary deictic expressions. In particular, the user can articulate the sentence “Flights from this city to that city” while selecting “Boston” and “Pittsburgh” in the menu of known cities. In this example, speech is assigned to denote the slots concerned in the request (i.e., the source and destination), and direct manipulation is assigned to the specification of the slot values (i.e., Boston and Pittsburgh). Again complementarity use has been pointed out by Sirioux et al (Sirioux et al 1996) as an observed behavior resulting from a WOZ experiment.

Similar relations hold between devices and languages. To illustrate the approach, we will consider the equivalence of a set of devices for a particular language l:

Device Equivalence: D-Equivalence(D, s, E, l)

Devices in a set D are equivalent over a state s and a non empty set E of expressions of an interaction language l, if all of the expressions of E can be defined using either one of the devices in D. Equivalence is permanent if the relation holds for any state. Equivalence is total if the relation holding for E also holds for the set of expressions that define l. For example, in MATIS, the user who chooses to specify a request in natural language, can either type-in or utter the sentence: thus keyboard and microphone are permanently and totally equivalent within the natural language domain.

Similarly, device assignment connects a device to a particular language. In MATIS, the mouse is permanently assigned to the direct manipulation language. An example of device redundancy would be the situation where the user spells a character using the microphone and, in the mean time, types in the same character.

Figure 3 schematically shows the CARE properties in terms of interrelationships between the three levels of abstraction of the Pipe-Lines model: devices, languages and tasks. In addition we can apply the CARE properties at a coarse grain and define the CARE properties in terms of interrelationships between interaction techniques and tasks. At this level of abstraction, the CARE properties could be used to combine unimodal modalities in Bernsen’s modality theory

(Bernsen 1996). If we go one step forward and consider several users (and not only one user and several modalities) as in CSCW systems, the CARE properties can be defined as relationships between users and tasks. For example one user is assigned to a particular task, or two users are "complementary" in performing a given task. Within such systems the CARE properties can therefore help define the role of each participant. In conclusion, 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.

The System CARE Properties and Usability Assessment

From the above definitions, we conclude that equivalence provides a way to enhance flexibility: for input, the user has multiple choices among languages and/or devices and, for output, the system can choose among multiple renderings. Equivalence is also a good ingredient for robustness. For example, in a noisy environment, a MATIS user may switch to direct manipulation or key-in natural language sentences. For critical systems, equivalence of languages and/or devices may be required to overcome device breakdowns or processing limitations. In contrast to this, assignment can be viewed as a restrictive feature. Redundancy provides freedom and may also be used to increase robustness. In complementarity however, usage of multiple devices and languages may generate cognitive overload and extra articulatory synchronization problems. "Permanentcy" and "totalness" of the CARE relations are also important usability factors. A permanent relation, which holds for any state, and a total relation which holds for all of the tasks and/or languages available, express the absence of exceptions. "Partialness" and "transiency," on the other hand, fall into the "inconsistency trap." For input, the user will have to remember the context (e.g., the tasks) for which the relations hold. For output, the system may appear unstable with regard to its rendering strategy. The CARE properties can be analyzed from additional perspectives. In particular, time and processing resources may provide useful insights. For example, in MATIS the keyboard and the microphone are not equivalent for specifying sentences in NL if one considers the time needed to express the sentence. From the system perspective, the acquisition of a NL sentence involves more processing resources when uttered than when it is typed in.

In (Coutaz et al. 1995) we adopt a formal approach and show that the CARE properties of the computer system have counterparts in the corresponding properties of the user: *the User-CARE properties*. As with *the system CARE properties*, the user properties are concerned with the choice between different modalities for communicating with the computer. We refer to user's preferences, affecting her/his choice of input modalities, as *U-preferences*.:

° *U-assignment*: If only one modality is acceptable to the user, or if she/he has a strong preference for one particular modality.

° *U-equivalence*: If there exists a subset of the possible modalities which she/he prefers to all others, but between which she/he is indifferent.

° *U-redundancy*: If the user prefers to employ two or more means of communication to convey the same information.

° *U-complementarity*: If the user's preference is to use one modality for one aspect of the task and another modality for another aspect.

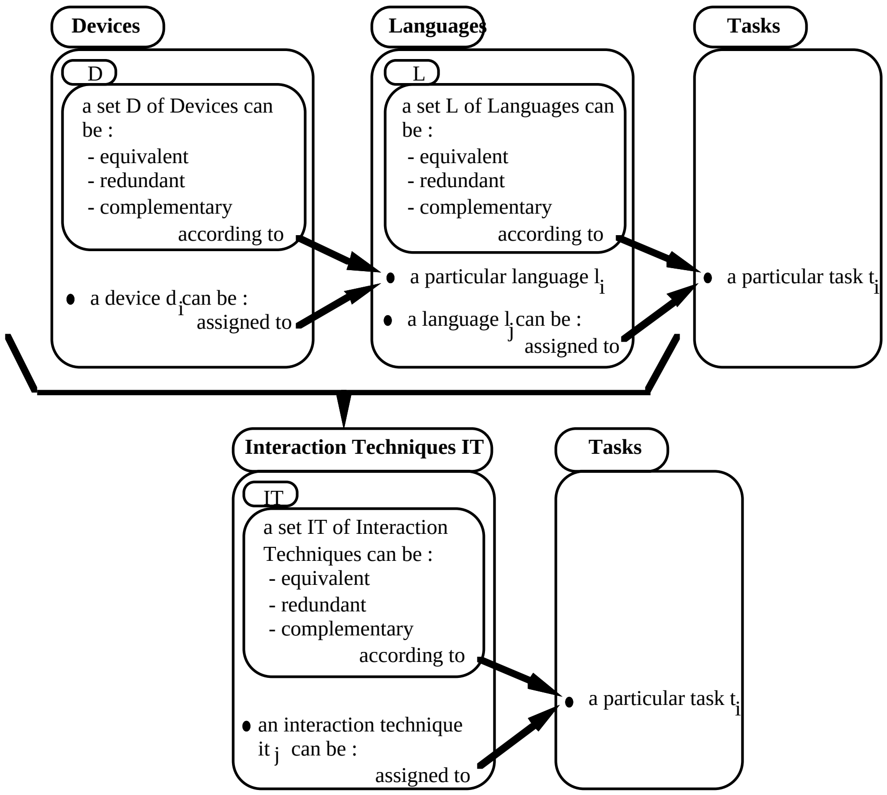

Figure 3: Overall representation of equivalence, assignment, redundancy, and complementarity between devices, languages and tasks.

The requirement on the design of the system is that its properties must be compatible with the user's U-preferences. In (Coutaz et al. 1995) we use the notion of compatibility between user preferences and system properties to show how the CARE properties interact with user modeling and predict usability during system design.

In summary, we have shown that the CARE relations provide insights for usability assessment of multifeature systems. We are aware that our current definitions may be enriched in many different ways. Additional constraints such as time and processing resources used by the user and/or the system are two relevant features. For now, we stick to the definitions provided above and discuss their implications on software design. Our technical solution for implementing the system CARE properties draws upon our software architecture model: PAC-Amodeus.

PAC-Amodeus—A Software Architecture Model

PAC-Amodeus (Nigay 1994) (Nigay, Coutaz 1993a) is a software architecture model. It provides a framework for performing functional partitioning and allocation of function to structural components based on system and user-centered properties.

PAC-Amodeus uses the Arch model (UIMS workshop 1992) as the foundation for the functional partitioning of an interactive system and populates the key element of this organisation, i.e., the Dialogue Controller, with PAC agents (Coutaz 1987). Indeed PAC-Amodeus incorporates the two adaptor components of Arch, the Interface with the Functional Core (IFC) and the Presentation Techniques Component (PTC), to insulate the keystone component (i.e., the Dialogue Controller, DC) from modifications occurring in its unavoidable neighbors, the Functional Core (FC) and the Low Level Interaction Component (LLIC).

As shown in Figure 4, PAC-Amodeus makes explicit the boundaries of the Presentation Techniques and the Low Level Interaction components using the notions of physical device and interaction language. In PAC-Amodeus, the functional partitioning of an interactive system should be defined according to the following rule:

- the Low Level Interaction Component should be device dependent and language dependent;
- the Presentation Techniques Component is device independent but still language dependent;
- the other components of the interactive system, including the Dialogue Controller, should be both device and language independent.

This rule is mainly guided by two software engineering quality criteria, namely re-usability and modifiability. The PAC-Amodeus model however, does not indicate how the CARE properties can be supported within the architecture. Implementation of the CARE properties are detailed in the following section. We here consider the CARE properties defined as relationships between interaction techniques and tasks (bottom layer of figure 3).

Implementation of the System CARE Properties Within PAC-Amodeus

Equivalence and assignment properties do not require a specific code. On the one hand, equivalence implies that the same piece of information resulting from the usage of different interaction techniques is passed from the PTC to the same PAC agent of the DC. Equivalence can therefore be achieved within the PTC. On the other hand, assignment implies that only one interaction technique can be used to convey a particular piece of information to a particular PAC agent of the DC. Again assignment is a property that can be implemented in the PTC.

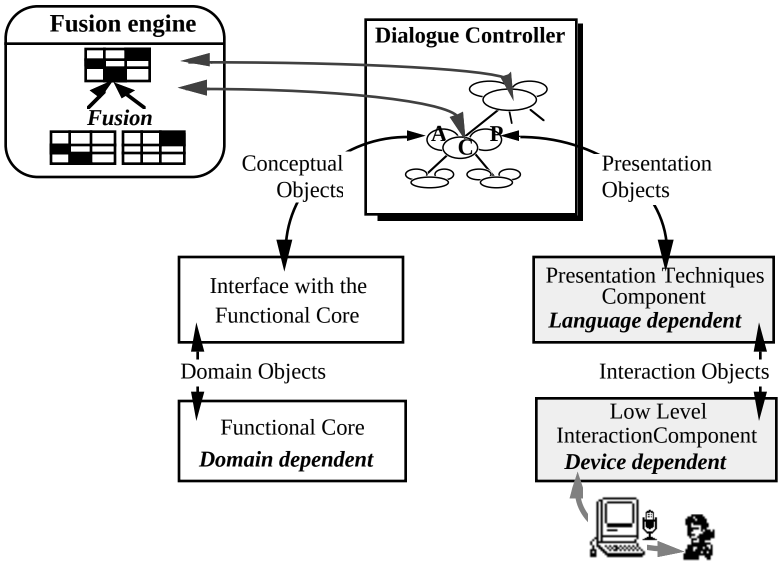

Figure 4: The PAC-Amodeus model and the fusion engine which implements the CARE properties.

To implement the redundancy and the complementarity properties, we have developed a fusion engine (Nigay, Coutaz 1995) at the Dialogue Controller level. A related approach consisting of a merging process can be found in (Sirioux 1996). Because our fusion algorithm is generic and can be reused, it disappears from the architectural design solution. It is not part of a PAC-Amodeus architecture. It operates behind the scene as a reusable mechanism.

Each PAC agent has access to the fusion mechanism through its Control facet. The fusion mechanism is responsible for combining information specified by the user using different interaction techniques. Criteria to combine information include time and structural complementarity.

The fusion mechanism is based on 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 tasks that the Dialogue Controller is able to handle. Events generated by user's actions are abstracted and mapped onto the structural parts. These events have different time-stamps. The structural decomposition of a melting pot is described in a declarative way outside the engine. By doing so, 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.

Before describing the types of fusion which define the overall behavior of the engine we consider an example using MATIS tasks: Figure 5 shows how redundancy and complementarity are handled by the fusion mechanism. In the redundancy example, the user has uttered the sentence "Flights from Boston" while selecting "Boston" with the mouse in the scrollable list of known cities. The speech act generates the bottom left melting pot: at time t_i, the slot "from" is filled in with the value "Boston." The melting pot to its right results from the mouse selection. The fusion engine combines these two melting pots into a new one (top left) which contains the value "Boston." A similar reasoning applies to complementarity.

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 are structurally complementary and very close in 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 involves 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.

In addition to the three types of fusion which implement complementarity, the mechanism checks for redundancy. This is done before applying the Microfusion rule. As opposed to complementarity, only one type of redundancy is implemented, the microredundancy. Microredundancy is defined as two very close columns that contain the same information. As in microfusion, microredundancy compares inputs if they have been produced in parallel or in pseudo-parallelism.

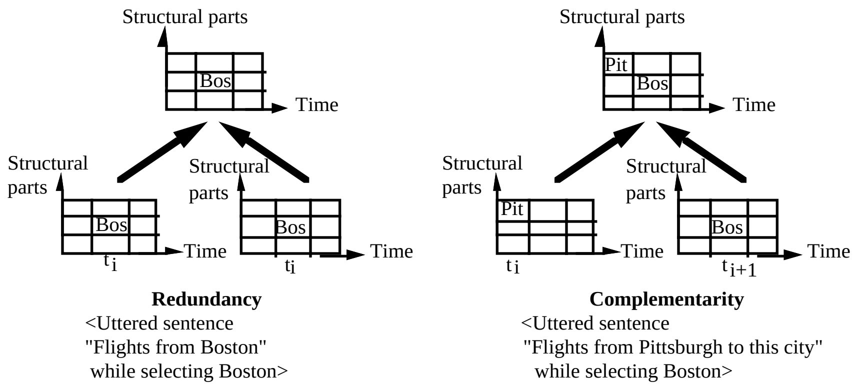

Figure 5: Redundancy and complementarity handled by the fusion mechanism.

The fusion engine and its rules are formally defined in (Nigay, Coutaz 1995). Another formal approach based on LOTOS is presented in (Paterno & Mezzanotte 1996). A time extension of LOTOS is required to describe the MATIS user behavior in terms of an interactor.

We note that the fusion mechanism supports simple interaction of modalities. Moreover there is no dominant modality handled by the fusion mechanism. As future work, the context or history of interaction needs to be taken into account: this can be accomplished by focusing on each PAC agent which locally maintains its state. In addition, we plan to enrich our fusion mechanism with a confidence factor attached to every slot of a melting pot. The notion of confidence factor provides a simple mechanism for modeling uncertainty and can be usefully exploited for solving ambiguities in deictic expressions (Complementarity). Figure

6 shows the relevance of confidence factors using the following example: the user utters the sentence “Flights from Boston to this city” while selecting “Denver” using the mouse.

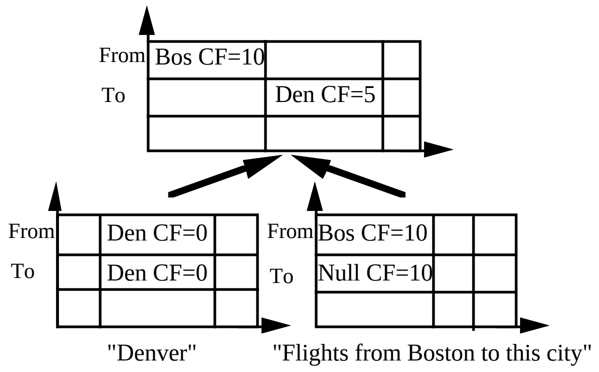

Figure 6: Confidence factor (CF ∈ [0,10]): Example of a deictic expression .

The fusion mechanism is generic and works symbiotically with PAC-Amodeus, our software architecture model. It defines a framework for developing interfaces that support multiple interaction techniques.

Conclusions

We first presented the Pipe-Lines design space which identifies three important levels of abstraction in the interpretation and rendering functions. We then defined the CARE properties as relationships between the three levels of abstraction of Pipe-Lines and demonstrated how they can be exploited to assess the usability of multifeature systems. We have addressed the problem of why and how a system supports different languages and devices. For example:

- Do we use languages and devices synergistically (the complementarity property)?
- Do we use languages and devices for specific tasks (the assignment property)?

  The definitions of the CARE properties provide a formal framework for reasoning about the design of multimodal systems. We have then shown how the CARE properties can be used as constraints for the implementer. We have designed a software architectural model, PAC-Amodeus, augmented with a fusion mechanism to support the CARE properties. The contribution of the software architecture model and fusion engine is two-fold:

- an attempt to bridge the gap between HCI properties and software design,
- a generic platform for implementing multifeature systems.

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 to implement complementarity. Such a fission mechanism can be based on the melting pot representation: by considering the example of Figure 5, 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.

Acknowledgments

This work has been partly supported by project ESPRIT BR 7040 Amodeus II. Many thanks to George Serghiou for help with style and English grammar.

References

Abowd, G.; Coutaz, J.; and Nigay, L. 1992. Structuring the Space of Interactive System Properties, IFIP WG2.7 Working Conference on Engineering for Human Computer Interaction, Elsevier Publ., Finland, 113-128.

Bernsen, N. O. 1996. Rule-based multimodal interface design. IMMI-1, session 3, John Lee ed.

Coutaz, J. 1987. PAC: an Implementation Model for Dialog Design, Proceedings of Interact'87, H-J. Bullinger, and B. Shackel eds, North Holland, Stuttgart, 431-436.

Coutaz, J.; Nigay, L.; and Salber, D. 1993. The MSM framework: A Design Space for Multi-Sensori-Motor Systems, in Proc. EWHCI'93, Moscow, Springer Verlag (Lecture notes in Computer Science, Vol. 753, 1993, 231-241).

Coutaz, J.; Nigay, L.; Salber, D.; Blandford, A.; May J.; and Young, R. 1995. Four Easy Pieces for Assessing the Usability of Multimodal Interaction:The CARE properties, Proceedings of Interact'95, K. Nordby, P.H. Helmersen, D.J. Gilmore and S. Arenesen eds, Chapman&Hall, Norway, 115-120.

Duke, D.; and Harrison, M.. 1994. Folding Human Factors into Rigourous Development. Proceedings of Eurographics Workshop "Design, Specification, Verification of Interactive Systems," F. Paterno ed., 335-352.

Dowell, J.; Shmueli, Y.; and Salter, I. 1996. Applying a cognitive model of the user to the design of a multimodal speech interface. IMMI-1, session 2, John Lee ed.

Gellersen, H-W. 1996. Toward engineering for multimodal interactive systems. IMMI-1, session 4, John Lee ed.

Hemjslev,L. 1947. Structural Analysis of language, Studia Phonetica, vol. 1, 69-78.

Karagiannidis, C.; Koumpis, A.; and Stephanidis, C. 1996. Media/Modalities Allocation in Intelligent Multimedia User Interfaces: Towards a Theory of Media and Modalities. IMMI-1, session 3, John Lee ed.

Lunati J-M.; and Rudnicky, A. 1991. Spoken Lanmguage interfaces: The OM system, CHI'91 Proceedings, New Orleans, 453-454.

Martin, J.-C.; and Beroule, B. 1996. Multimodal interfaces based on types and goals of cooperation between modalities. IMMI-1, session 6, John Lee ed.

Neal, J.G.; Dobes, Z.; Bettinger, K. E.; and Byoun, J. S. 1988. Multi-Modal References in Human-Computer Dialogue, The Seventh National Conference on Artificial Intelligence, Saint Paul, Minesota, Vol. 2.

Nigay, L. 1993. A Case Study of Software Architecture for Multimodal Interactive System: a voice-enabled graphic notebook, Technical Report CLIPS-IMAG, 31 pages.

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

Nigay, L.; and Coutaz, J. 1993b. Problem space, fusion and parallelism in multimodal interfaces, INFORMATIQUE'93 Proceedings, Interface to Real&Virtual Worlds, 2th international conference, Montpellier, 67-76.

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

Nigay, L.; and Coutaz, J. 1995. A Generic Platform for Addressing the Multimodal Challenge. CHI'95 Proceedings, Denver, 98-105.

Norman, D. A. 1986. Cognitive Engineering, User Centered System Design, New Perspectives on Computer Interaction, D. A. Norman, S.W. Draper eds, Hillsdale, New Jersey : Lawrence Erlbaum Associates, 31-61.

Paterno, F.; and Mezzanotte M. 1996. Reasoning on multimodal interactive systems. IMMI-1, session 2, John Lee ed.

Siroux, J.; Guyomard, M.; Multon, F.; and Remondeau C. 1996. Oral and gestural activities of the users in the GEORAL system. IMMI-1, session 2, John Lee ed.

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