JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2004**

Vol. 3, No. 8, September-October 2004

Generic Pipelined Multi-Agents Architecture for Multimedia Multimodal Software Environment

H. Djenidi (1, 2), **A. Ramdane-Cherif** (1), **C. Tadj** (2) and **N. Levy** (1).
(1) PRISM, University of Versailles St.-Quentin, France.
(2) École de Technologie Supérieure, Quebec, Canada.

Abstract

Multimodal human-computer interaction needs intelligent architectures in order to enhance the flexibility and naturelness of the user interface. These architectures have the ability to manage several multithreaded input signals from different input media in order to perform their fusion into intelligent commands. In this paper, a generic comprehensive agent-based architecture for multimodal engine fusion is proposed. The architecture is sketched in term of its relevant components. Each element is modeled using timed colored Petri networks. The generic components of the engine fusion are then included in a pipelined based-agent global architecture for which the architectural quality attributes are outlined.

1 INTRODUCTION

Information and communication technologies should have a main role for helping a broader spectrum of everyday people (especially with physical disabilities) to use computing applications. To respond to this need, multimodal systems that process two or more combined user inputs modes- like speech, pen, touch manual gesture, gaze, and head and body movements- lead, trough their software architectures, to more transparent, efficient, and powerfully expressive means of human-machine communication. The multimodality conveys two striking features that are relevant to the software design of multimodal systems:

the fusion of different types of data from different Input devices, and the temporal constraints imposed on information processing from/to Input/Output devices.

Since the first rudimentary but pertinent system, "Put That There" [Bolt 1980], which processes speech in parallel with manual pointing, different multimodal applications have been developed [Crowley 1997, Bellik 1994, McGee 2000]. Each application is based on a dialog architecture combining modalities to match and elaborate on the relevant multimodal information. Today, there is no agreement on generic architectures that reflects a dialog implementation, independently of the application type. The main objective of this paper is to propose generic comprehensive architectures for multimodal engine fusion. These paradigms use the agent architectural concept to achieve their functionalities and unify them into generic structures. For this purpose, this paper gives a synthesis that sketches the collective and recurrent properties, implicitly used in such dialogs.

Section 2 gives an overview and the requirements necessary in Multimedia Multimodal Dialog Architecture (MMDA) and presents a generic multi-agent architectures in term of components. Section 3 illustrates The engine fusion modeling with a stochastic timed Colored Petri Net (CPN) [Jensen 1997a, 1997b, Jensen et al.1995] and outlines the quality attributes of its architecture. An example of the classical "Copy and Paste" operations is given in more details to demonstrate the proposed generic architecture in Section 4.

2 GENERIC MULTIMEDIA MULTIMODAL DIALOG ARCHITECTURE

In this section a synthesis first gathers an overview and the requirements of MMDA. Then the proposed generic multi-agent architectures are described.

Overview and Requirements

With the increasing complexity of multimedia applications, a single modality becomes insufficient to allow the user to interact effectively across environments. A basic MMDA as shown in Figure 1, gives the user the possibility to decide which modality or combination of modalities are better-suited, depending on the task and environment contexts (see examples in [Oviatt 2000a, 2000b]). The user can combine speech, pen, gaze, manual gestures, and body postures and movements via input devices (key pad, tactile screen, stylus, etc.) to dialog in a coordinate way with multimedia system output. The environmental conditions could lead to more constrained architectures that have to be adaptable during the continuous change of either external perturbations or user's actions. In this context a first framework is introduced in [Hutchins 1986] to classify interactions. It considers two dimensions ('engagement' and 'distance') and decomposes the user/system dialog into four types.

The 'engagement' is a type characterizing the depth implication of the user in the system. The user feels that an intermediary subsystem performs the task, in 'conversation' case, and that he can act directly on the system components in 'model world' case. The 'distance' represents the user cognitive effort taken.

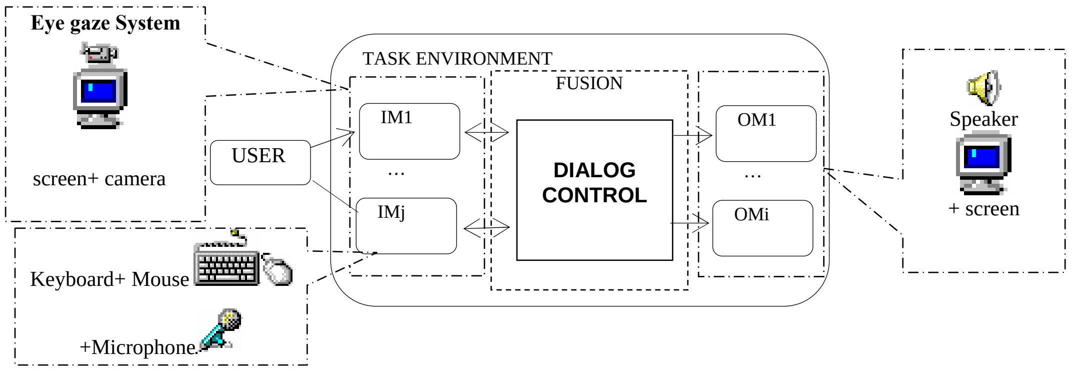

Fig. 1**:** Example of basic multimedia multimodal model
(↔: interaction, →: action, IMj: Input Modality j and Omi: Output Modality i.)

This framework reaches the idea that two kinds of multimodal architectures are possible [Oviatt 2000]. The first one makes fusions based on feature signal recognition. The recognition steps of one modality guide and influence the other modalities in their own recognition steps [Bregler 1993, Project CNRS 1994]. The second architecture uses individual recognition systems for each modality. Such systems are associated with an extra process that performs semantic fusion of the individual recognized signal elements [Bolt 1980, Bellik 1994, Oviatt 1999]. A third hybrid architecture is possible by mixing the two previous types: signal feature level and semantic information level.

At the core of multimodal system design, the information fusion of the input modes is the main challenge. The input modes can be equivalent, complementary, specialized or redundant as described in [Coutaz 1994]. In this context, the multimodal system designed with one of the previous architectures (features or/and semantic levels) needs the integration of the temporal information. The possible types of multimodality depend on the time proximity of the input signals. Time granularity is an important decision criterion when we generate a multimodal semantic sequence. as shown in Figure 2. In this example, it shows that the chosen multimodality type, for mouse clicks and speech, is the synergistic one. This is obvious in the example, because the click occurs only during the time when a sentence is said. The synergistic mouse/speech actions correspond to one statement and the tactile screen actions to another one. Both statements are performed in parallel and could be independent, equivalent, complementary, specialized and/or redundant. In other words, the temporal aspect in MMDA does not handle signals overlapping only.

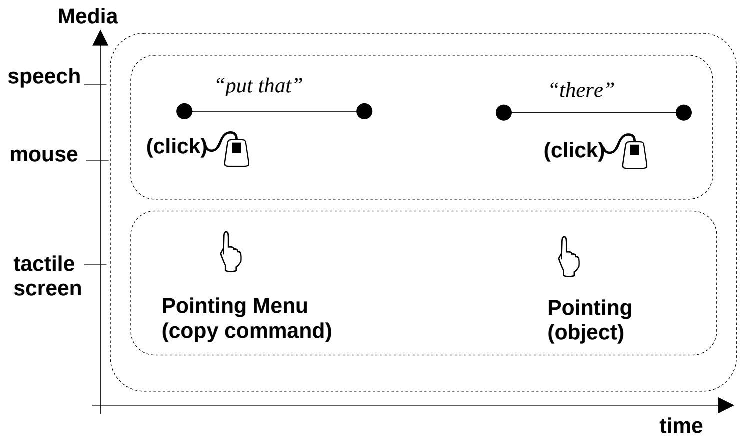

Fig. 2: Example of parallel synergistic multimodality. Because of the time information, tactile screen is in parallel with the synergistic mouse/speech actions.

It helps to decide whether two signal parts should belong to a multimodal fusion set or whether they should be considered as separate modal actions. Therefore, multimodal architectures are better able to avoid and recover errors that mono-modal recognition systems can't recover [Oviatt 1999, Oviatt 2000]. This property results in a more robust natural human-machine language.

Another property is that, the more timed combinations of signal information or semantic multiple inputs grow the more equivalent formulations of the same command are possible. For example, ["Copy that there"], ["copy" (click) "there"] and ["copy that" (click)] are various ways to represent three statements of a same command (copying an object in a place), if speech and mouse clicking are used. This redundancy also increases the robustness in terms of error interpretations.

Figure 3 summarizes the main requirements and characteristics needed in multimodal dialog architectures. As shown in this figure, five characteristics can be used in the two different levels of the fusion operations: the 'early fusion' at the feature fragments level and the 'late fusion' at the semantic one [Oviatt 2000].

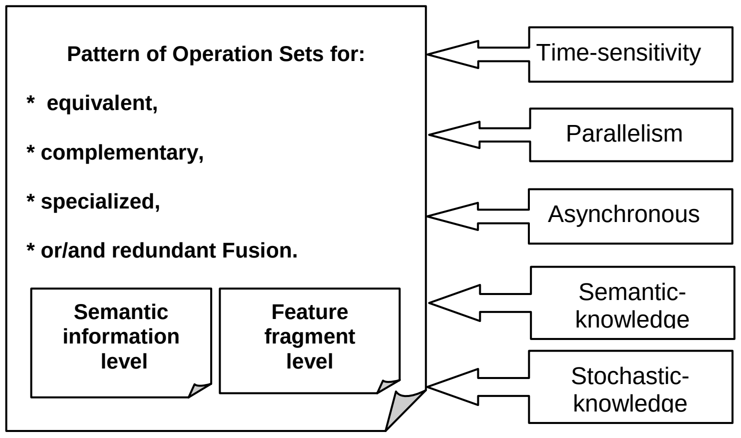

Fig. 3: The main requirements for multimodal dialog architecture (⇐: used by.)

The Asynchronous property gives the architecture the flexibility to handle multiple external events while parallel fusions are still processing. The specialized fusion operation deals with an attribution of a same modality to a same statement type. (For example, in drawing applications, speech is specialized for color statements and pointing for basic shape statements.)

The granularity of the semantic and statistic knowledge depends on the media nature of each input modality. This knowledge leads to important functionalities. It lets the system accept or refuse the multi input information for several possible fusions (selection process); and it helps the architecture choose, between several fusions, the most suitable command to execute or message to send to an output media (decision process).

The property of parallelism is, obviously, inherent to such applications involving multiple inputs. The whole requirements suggest strongly intelligent multi-agent architectures, which are the purpose of the next section.

Generic Multi-Agent Architecture

The Agents are entities that can interact and collaborate with dynamic and synergy for modality combination issues. The interactions should occur between agents and agents should also get information from users. An intelligent agent has three properties. It reacts in its environment at certain times (reactivity), takes initiatives (pro-activity) and interacts with other intelligent agents or users (sociability) to reach goals [Jennings 1998, Weiss 1999, Bird 1993]. Therefore each agent could have several input ports to receive messages and/or several output ports to send ones.

The level of intelligence of each agent varies according to two major options coexisting today in the field of Distributed Artificial Intelligence [Bond 1988, Ishida 1997, Muller 1996]. The first one, corresponding to the cognitive school, attributes the level to the cooperation of very complex agents. This approach deals with agents with strong granularity assimilated to expert systems.

In the second school, the agents are simpler and less intelligent but more active. This reactive school presupposes that it is not necessary to each agent to be individually intelligent to reach an intelligent total behavior [Cohen 1997]. This approach deals with a cooperative team of working agents with low granularity, which can be matched to finite automate.

Both approaches can be matched to the late and early fusions of multimedia multimodal architectures.

Obviously, there are all the possible intermediaries between these options of multi-agent systems (as shown in the proposed approaches developed in the next sections). One can easily imagine systems based on a modular approach, putting sub-modules in competition, each sub-module being itself a universe of overlapping components. This word is usually employed for ‘sub-agents’.

Identifying the generic parts of multimodal multimedia applications and binding them into an intelligent agent architecture requires the determination of common and recurrent communication protocols and their hierarchical and modular properties in such applications.

In most multimodal applications, speech, as input modality, offers speed, a large information spectrum and relative facility of use. It lets both the user’s hands and eyes free to work in other necessary tasks present, for example, in driving or moving cases. Moreover, speech involves a generic communication language pattern between the user and the system. This pattern is described by a grammar with production rules, able to serialize possible sequences of the vocabulary symbols produced by users. The vocabulary could be word set, phoneme set or another signal fragment set depending on the feature level of the recognition system. The goal of the recognition system is to identify signal fragments. Then, an agent organizes the fragments in a serial sequence according to his grammar knowledge and asks other agents for possible fusion at each step of the serial regrouping. The whole interaction can be synthesized in a first generic agent architecture called Language Agent (LA) and depicted by Figure 4.

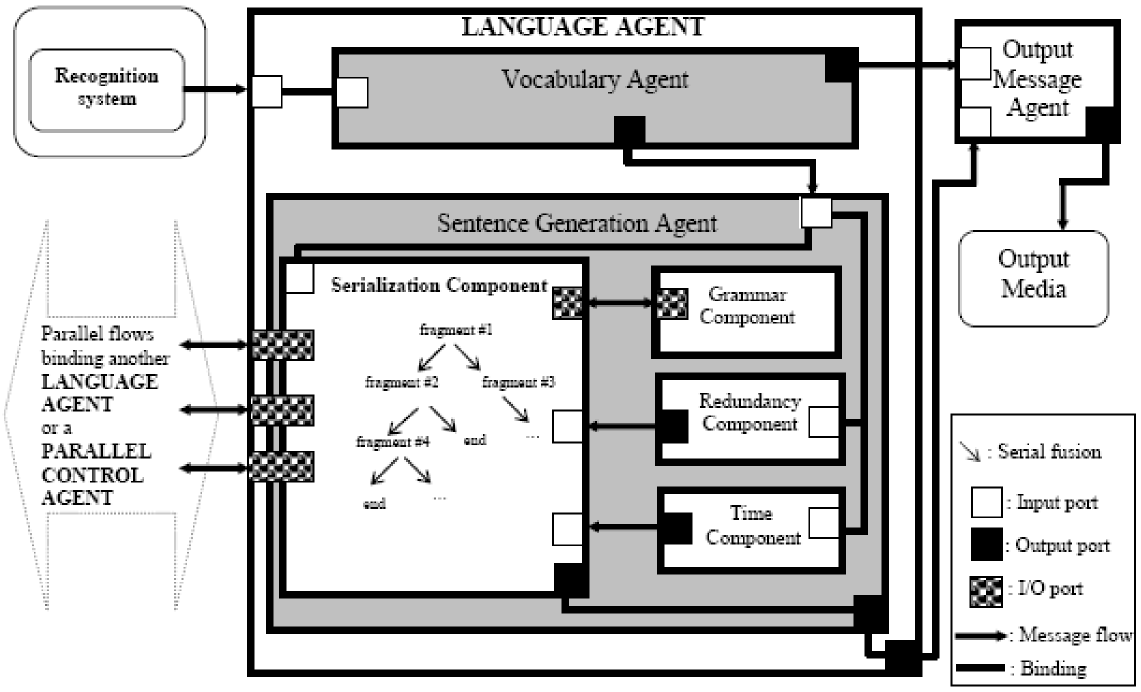

Fig. 4: Generic langage agent corresponding to an input modality.

Each input modality should be associated with an LA. For basic modalities like manual pointing or mouse clicking, the complexity of the LA is strongly reduced. The ‘Vocabulary Agent’ that checks whether or not the fragment is known, is, obviously, no longer necessary. The ‘Sentence Generation Agent’ is also reduced into a simple event thread whereon another external control agent could possibly make parallel fusions. In such a case, the external agent could handle ‘Redundancy’ and ‘Time’ information, with two corresponding components. These two components are agents that, respectively, check redundancies and time neighborhood of the fragments during their sequential regrouping. The ‘Serialization Component’ processes this regrouping. Thus, depending on the input modality type, the LA could be assimilated to an expert system or to a simple thread component.

Two or more LAs can communicate directly for early parallel fusions or, through another central Agent, for late ones (Figure 6). This central agent is called Parallel Control Agent (Figure 5).

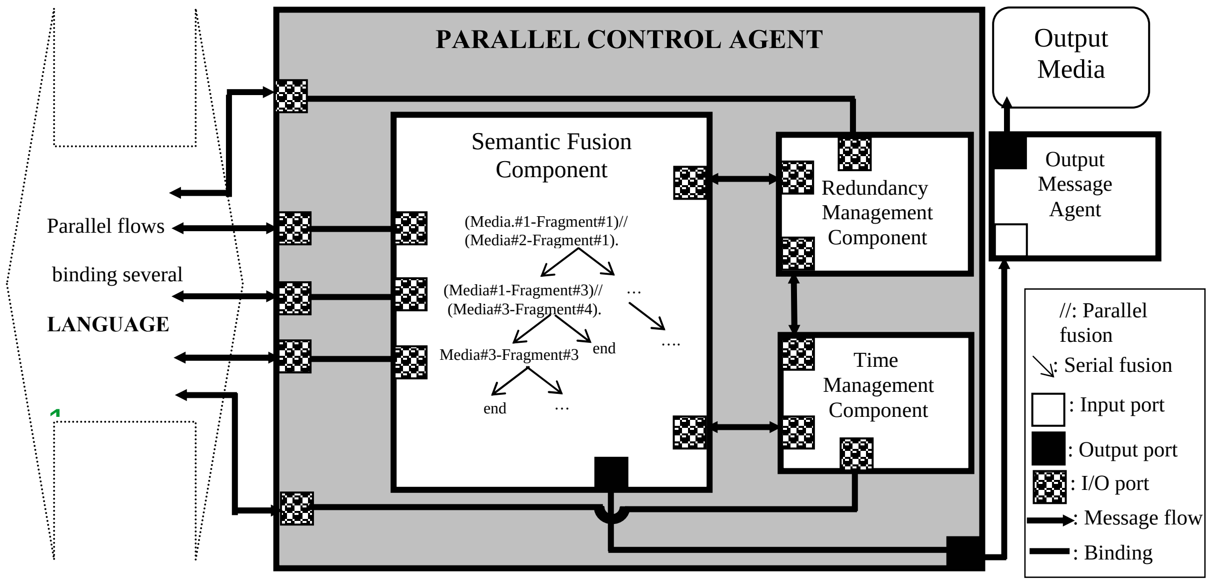

Fig. 5: Generic Parallel control agent for central parallel multimodal fusions.

In the first case, the ‘Grammar Component’ of one of the LAs must carry an extra semantic knowledge for the parallel fusion purpose. This knowledge could also be distributed between the LA’s ‘Grammar Components’, as shown in Figure 6 (left). Several Serializing Components share their common information until one of them gives the sequential parallel fusion output. In the other case (Figure 6 right), a ‘Parallel Control Agent’ (PCA) handles and centralizes the parallel fusions of different LA information. For this purpose, the PCA has two intelligent components for, respectively, Redundancy and Time managements. These components exchange information with other components to elaborate the decision. Then, generated authorizations are sent to the Semantic Fusion Component (SFCo). Based on these agreements, the SFCo carries the steps of the semantic fusion process.

Redundancy and Time Management components receive the redundancy and time information via the Semantic Fusion Component or directly from the LA, depending on the complexity of the architecture and the designer choices.

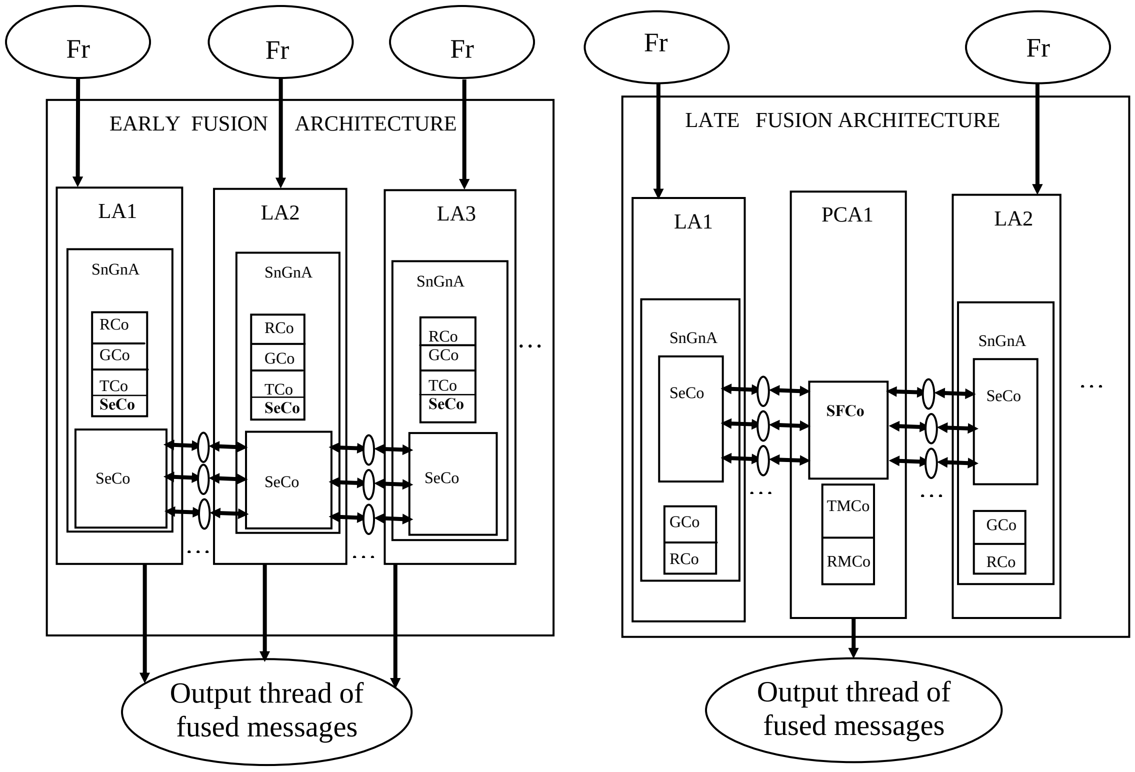
Fr
Fr
Fr
EARLY FUSION ARCHITECTURE
LA1
LA2
LA3
SnGnA
RCo
GCo
TCo
SeCo
Output thread of fused messages
Fr
Fr
LATE FUSION ARCHITECTURE
LA1
PCA1
LA2
SFCo
TMCo
RMCo
Output thread of fused messages

Fig. 6: Principles of early and late fusion architectures (Fr: fragments of signal, L: language, P: parallel, C: control, A: agent, G: grammar, S: semantic, Sn: sentence, Gn: generation, F: fusion, Se: serialization, Co: component, T: time, R: redundancy, and M: management). More connections (arrows that indicate the data flow) could be activated or inhibited by the agents to gather fusion information (an ellipse represents a thread or a locality, a box represents an activity.)

Redundancy and Time Management components receive the redundancy and time information via the Semantic Fusion Component or directly from the LA, depending on the complexity of the architecture and the designer choices.

The paradigms proposed in this section constitute a general but important step in the software development of multimodal user interface: a high level abstraction of informal architectural views. Never less another important phase of the software development, for such applications, concerns the modeling aspect. Different methods like UML, B_method [Abrial 1996], Augmented Transition Networks [Bellik 1995], or Timed CPN [Jensen 1997a, 1997b], can be used to model the multi-agent dialog architectures. Section 4 discusses the choice of Colored Petri Networks to model an example of engine fusion in multimedia multimodal applications.

3 ENGINE FUSION MODELING

This section presents the Petri net modeling of an engine fusion used in multimedia multimodal applications. Small augmented finite-state machines like augmented transitions networks (ATN) have been used in the multimodal presentations system [Chen 1990].

Properties allowed by the specification method / Specification method of the events in theMMDA.		Time granularity in MMDAs			Formal language			Representation of :										Tools free of charges
										Dynamicity with capability of :				Distri-bution:				
										Com-munication		Paral-lelism						
		Assign an arrival time stamp to each event	Sort the incom-ming events	Take into account the time of each processed event (time of reco-gnition, …)	YES	NO	FORMAL LOGIC	Exchan-ged data in the dialog (objects sym-bols, mes-sages, …)	Recipients and data structures (threads, ressources channels, localities, …) & their inter con-nections.	ASYNCHRONOUS	SYNCHRONOUS	INTERLEAVED	CONCURRENCY	FUNCTIONAL	SPATIAL	GRAPHICAL & INTUITIVE	HIERARCHICAL	
Algebra process	CSP				x		x	x			x	x						x
	CCS				x		x	x			x	x						x
	LOTOS				x		x	x			x	x				x	x	x
UML	collaboration diagrams					x		x	x						x	x		
	sequence diagrams	x	x	x		x				x	x	x				x		
	states-transitions diagrams			x		x		x		x	x	x				x	x	
	activity diagrams					x				x	x		x			x		
Z					x		x	x	x								x	x
ATN								x	x	x		x				x		
CPN		x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x

Table 1. Comparison of MMDA specification methods (ATN: augmented transition network, CSP: Communicating Sequential Processes, CCS: Calculus Communicating Systems, LOTOS: Language of Temporal Ordering Specifications, UML: Unified Modeling Language.)

These networks easily conceptualize the communication syntax between input and/or output media streams. However, they have limitations when important constraints such as temporal information and stochastic behaviors need to be modeled in protocols of fusion. Timed Stochastic Colored Petri Networks (CPN) offer a more suitable pattern [Jensen 1997a, 1997b, Jensen 1995] to design such constraints in multimodal dialog. The most important issues of Petri net modeling, in comparison with other formal and informal specification methods used in MMDAs are summarized in Table 1. This table doesn't sketch the timed process algebra because they can not easily and intuitively capture the properties of time granularity presented here.

Multi-Threaded Multimodal Architecture Modeling

For modeling purpose, each input modality is assimilated to a thread where signal fragments flow. Multimodal inputs are parallel threads corresponding to a changing environment that describes different internal states of the system. Multi-agent systems are also multi-threaded: each agent has a control on one or several threads. Intelligent agents observe the states of one or several threads for which they are designed. Then, the agents execute actions that modify the environment. In a more formal way [Weiss 1999],

if $$\mathbf{A} = \{\mathbf{a_1; a_2; \ldots}\} \qquad (1),$$

and $$\mathbf{O} = \{\mathbf{o_1; o_2; \ldots}\} \qquad (2),$$

are the sets of actions and observations of an agent, respectively

and if, $$\mathbf{S} = \{\mathbf{s_1; s_2; \ldots}\} \qquad (3),$$

is the set of states with which the environment is described (including intermediary states), then the Petri network models two kind of activities described by the functions

$$\mathbf{Observation\_function : S \rightarrow O} \qquad (4),$$

$$\mathbf{Environment\_function : S{\times}A \rightarrow 2S} \qquad (5).$$

The first function describes what an agent observes, in a certain state $\mathbf{s_i}$. The second one describes how the environment develop the state $\mathbf{s_i}$ when an action $\mathbf{a_i}$ is executed.

The Petri network models also the actions of the agents described by the function

$$\mathbf{Action\_function : O \rightarrow A} \qquad (6).$$

The characteristic behavior of an agent action in an environment is the set 'History':

$$\textbf{History} = \{ \mathbf{h_1, h_2, \ldots h_i, \ldots} \} \qquad (7)$$

of all sequences of the observations defined by

$$\mathbf{h_i: (s_0) \xrightarrow{a_0} (s_1) \xrightarrow{a_1} \ldots (s_i) \xrightarrow{a_i} \ldots} \qquad (8)$$

with $\mathbf{a_i = Action\_function\ (<s_0, \ldots, s_i>),\ \forall\ i}$ (9)

and $\mathbf{s_i = Environment\_function\ (s_{i-1}, a_{i-1}),\quad \forall\ i, i \neq 0}$ (10)

($\mathbf{s_0}$ is the initial state of the system).

To summarize the precedent transaction, the Petri network has to model the functions (4), (5), (6) and also the input media threads with the design CPN toolkit [Jensen et al.1995].

In the following, it is assumed that this toolkit and semantics are known. However we give a description of the CPN modeling.

Modeling a Multimedia Multimodal Engine Fusion with CPN

The Petri network is a diagram flow of interconnected places (or locations represented by ellipses) and transitions (represented by boxes). A place represents a state and a transition represents an action. Labeled arcs connect places to transitions. The CPN is managed by a set of rules (conditions and coded expressions). The rules determine when an activity can occur and specify how its occurrence changes the state of the places by changing their colored marks (while the marks move from place to place). A dynamic paradigm like CPN includes the representation of actual data, with clearly defined types and values. The presence of dataflow is the fundamental difference between dynamic and static modeling paradigms. In CPN each mark is a symbol that can be of all the data types generally available in a computer language: integer, real, string, Boolean, list, tuples, record and so on. These types are called colorsets. Thus, a CPN is a graphical structure linked to computer language statements. Design CPN toolkit [Jensen 1997b] provide this graphical software environment within a programming language (CPN Meta Language (ML)) to design and run CPN.

In such system each piece of existing information (symbolized by a mark) is assigned to a location. These locations contain information about the system state at a given time and this information can change anytime. This MAS is called distributed in terms of [Tabeling 2002]:

- *Functional distribution*: it means a separation of responsibilities in which different tasks in the system are assigned to certain agents.
- *Spatial distribution*: it means that the system contains multiple locations (that can be real or virtual).

A virtual location is an imaginary location where it already contains observable information or when information can be placed on it, but no assumption of physical information is linked to it. The set of colored marks in all places (locations) before an occurrence of the CPN is equivalent to an observation sequence of a MAS. For the MMDA case, each mark is a symbol that could represent signal fragments (pronounced words, mouse clicks, hand gesture, face attitude, lips move etc.), serialized or associated fragments (comprehensive sentences or commands) or simply a variable.

A transition can model an agent that generates observable values. A location can be observed by multiple agents. The observation function of an agent is simply modeled by input arcs inscriptions and also by the conditions in each transition guard (symbolized by **[conditions]** under a transition box). These functions represent the *facet A* of agents. Input arc inscriptions specify data that must exist for an activity to occur. In Figure 7, the variables, like ‘p11’, ‘p12’, etc (beginning with the character ‘p’), are used to represent the properties of time, grammatical and semantic informations of the signal fragments. When a transition is fired (an activity occurs), a mark is removed from input places and the transition activity can modify the data associated to the marks (or its colors) and thereby changes the state of the system (by adding a mark in at least one output place). If there are colorset modifications to perform, they are executed by a program associated to the transition (and also specified by the output arc label). The program is written in CPN ML inside a dashed box (not connected to an arc and close to the concerned transition-see example in Figure 7.-) Therefore each agent generates data for at least one output location and observes at least one input location. When no code is associated to the transition, output arc inscriptions specify data that will be produced if an activity occurs. The action functions of the agent are modeled by the transition activities and constitute the *facet E* of the agent.

Hierarchy is another important property of the CPN modeling. The symbol $\boxed{\textbf{HS}}$ in a transition means that such transition is a Hierarchical Substitution one (Figure 7). It is replaced by another subordinate CPN. Therefore, Input and output ports of the subordinate CPN correspond as well to the subordinate architecture ones in the hierarchy.

Each transition and each place is identified by its name (written on it). The symbol $\boxed{\textbf{FG}}$ in identical places indicates that the places are ‘Global Fusion’ places [Jensen 1997b]. These identical places are simply a unique resource (or location) shared over the net by a simple graphical artifact: the representation of the place and its elements is replicated with the symbol $\boxed{\textbf{FG}}$ (Figure 7.)

To summarize, modeling MAS can be based on four dimensions which are: Agent (A), Environment (E), Interaction (I), and Organization (O).

- *Facet A* indicates the whole functionalities of internal reasoning of the agent.

- The *facet E* gathers the functionalities related to the capacities of perception and action of the agent in the environment.
- *Facet I* gathers the functionalities of interaction of the agent with the other agents (interpretation of the primitives of the communication language, management of the interaction and the conversation protocols). The structure itself of the CPN, where each transition can model a global agent decomposed in components distributed in a subordinate CPN (within its initial values of variables, and procedures), models this facet.
- The *facet O* can be most difficult to obtain with CPN. It concerns the functions and the representations related to the capacities of structuring and managing the relations between the agents to make dynamic architectural changes.

Sequential operation is not typical of real systems. Systems that perform many operations and/or deal with many entities usually do more than one thing at a time. Activities that happen at the same time are called *concurrent activities*. A system that contains such activities is called a concurrent system. CPN models easily this concept of parallel process.

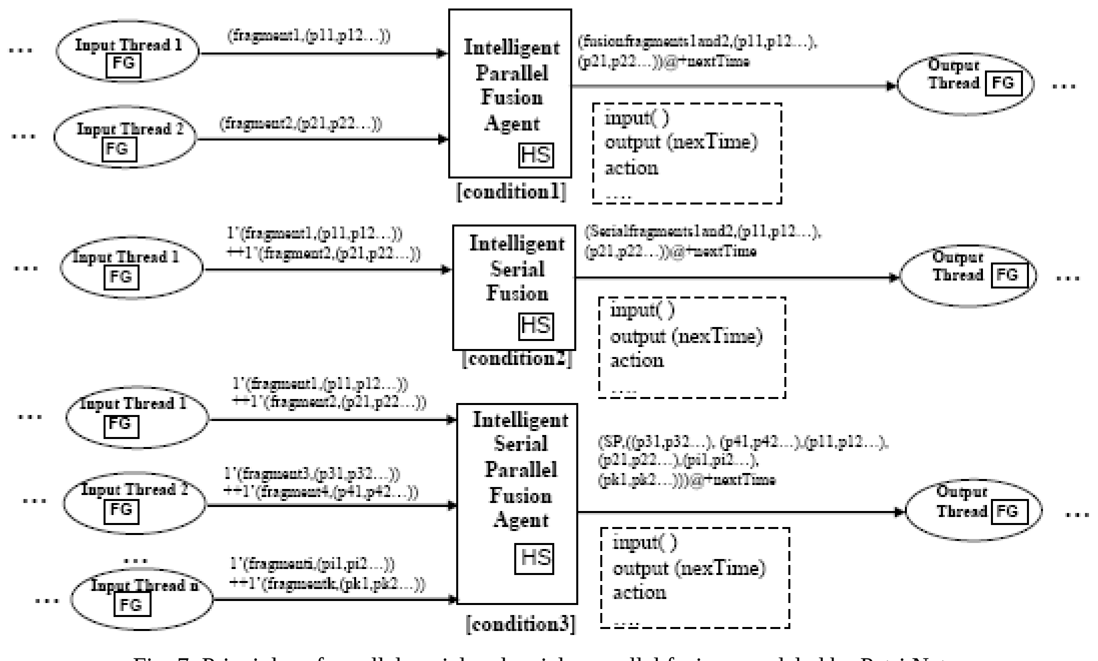
Input Thread 1
FG
Input Thread 2
(fragment1,(p11,p12…))
(fragment2,(p21,p22…))
Intelligent Parallel Fusion Agent
HS
[condition1]
(fusionfragments1and2,(p11,p12…),(p21,p22…))@+nextTime
Output Thread
input( )
output (nexTime)
action
Intelligent Serial Fusion
[condition2]
1'(fragment1,(p11,p12…)) ++1'(fragment2,(p21,p22…))
(Serialfragments1and2,(p11,p12…),(p21,p22…))@+nextTime
Intelligent Serial Parallel Fusion Agent
[condition3]
1'(fragment3,(p31,p32…)) ++1'(fragment4,(p41,p42…))
1'(fragmenti,(pi1,pi2…)) ++1'(fragmentk,(pk1,pk2…))
Input Thread n
(SP,((p31,p32…),(p41,p42…),(p11,p12…),(p21,p22…),(pi1,pi2…),(pk1,pk2…)))@+nextTime

Fig. 7: Principles of parallel, serial and serial→parallel fusions modeled by Petri Nets.

In order to take time into account CPN is timed and provides a way to represent and manipulate time by a simple methodology based on four characteristics :

1. A mark in a place could have an associated number, called a *time stamp*. Such a timed mark had its *timed colorset*.

2. The simulator contains a counter called the *clock*. The clock is just a number (integer or real) whose current value is the current time.
3. A timed mark is not available for any purpose whatever unless the clock time is greater than or equal to the mark's time stamp.
4. When there are no enabled transitions, but there would be if the clock had a greater value, the simulator increments the clock by the minimum amount necessary to enable at least one transition.

These four characteristics give the dimension of simulated time that has exactly the properties needed to model delayed activities. The transition activity can generate an output delayed mark. This mark can reach the output place only after a time equal to a value 'nextTime'(Figure 7.) The value of 'nextTime' is calculated by the code associated to the transition or set by the user.

With all these possibilities CPN provide an extremely effective dynamic modeling paradigm to model MAS like multimedia multimodal fusion engine.

Quality Attributes of the Chosen Architecture

The generic multi-agents architecture chosen for the multimedia multimodal fusion engine within CPN modeling is an intermediary one between the late and early fusion architectures. Section 4 shows an example of this architerture.

The main features appearing in the proposed generic CPN modeled architecture are summarized in four points.

- Distributed architecture: CPN modeling offers the possibility to distribute PCA over the architecture. Each instance of the PCA has its facets of action perception and interaction depending on its contextual position in the network and error-avoidance management. Also, the possibility to decompose each LA into sublevels leads to a model that can assist the code generation in a computer language used in the final implementation of the system (hierarchy, heritage, …). Finally, distribution allows to reduce the perceptions mechanisms of the agents and to spread them out over all the architecture.
- Scalable architecture: The architecture has the ability to sustain a growing load when new modalities are added.
- Parallel architecture: The parallelism gives the possibility to run the application with each LA processed in a separate parallel hardware. It is also possible to easily activate or inhibit a LA (in the case of dynamic architectural reconfiguration) without perturbing the global running application.
- Pipelined architecture: with several input and internal data streams and one output data stream it becomes easy to test and follow the evolution of this multimedia multimodal architecture, under the aspects of error avoidance.

4 EXAMPLE OF AN ENGINE FUSION MODELED BY CPN

Description of the 'Copy and paste' fusion engine

This section presents a typical example of a distributed architecture for fusion, using the paradigm Figure 7. The 'Copy and Paste' fusion engine architecture chosen involves a high level LA, for speech modality, linked, by a distributed PCA, to a rudimentary mouse clicking LA (thread of clicks). The PCA performs the semantic fusion between speech and mouse clicking trough two levels. Tables 2 and 3 give the vocabulary, used by the speech LA, and the basic sentences allowed by the corresponding grammar. Each word has a label used in the CPN design.

Word	Word label	Word	Word label
open	1	paste	5
close	2	cancel	6
delete	3	that	7
copy	4		

Table 2. Vocabulary.

In the following, a few symbolic regular expressions are used to represent semantic elements. These expressions use the arrow operator for sequential concatenation in time domain. For the chosen example, in the semantic expression:

$$(\text{word } 1 \rightarrow \text{word } 2)$$

word 1 is simply followed by (or contiguous to) word 2. The word 'cancel' is a command that automatically cancels the last action among the authorized sentences. Therefore, if the user says one of the words labeled in the set {**1, 2, 3, 4, 5**} just after "cancel", the time proximity between the two words is one of the decision criteria for suppressing the second word or taking it as a next command. For the proposed architecture both scenarios are processed.

The multimodal dialog gives for each sentence a set of possible redundant fusions. The symbol // models these concurrent associations in regular expressions.

For example, depending upon temporal information, the first command given in Table 2 is an element of the following semantic fusion set:

{(click→open→that); (open→click); (click→open);

(click // open); ((click // open)→that); (click // (open→that))}.

Set of Sentences	Command meaning
{ (open→that); (open) }	Open object
{ (close→that); (close) }	Close object
{ (delete→that); (delete).}	Delete object

{ (paste) }	Past last copied object
{ (copy→that); (copy) }	Copy object
{ (cancel)}	Cancel last command

Table 3. Sentences allowed by the grammar.

This semantic set includes the grammatical sentences corresponding to the command 'Open object'. Words, temporally isolated and labeled in the set {1, 2, 3, 4, 7}, are not considered by the PCA. The remaining fusion entities like ((close→open) // click), (click // (delete→open)), etc. or isolated clicks are also ignored by the system. (Thus, some errors made by user are avoided by the model.) The whole sets constitute the semantic knowledge. The associated CPN uses two random generators to design the arrival time of the input media events. The inter-arrival time between two pronounced words as well as the time between two consecutive 'clicks', are exponentially distributed. Events (like words and clicks) are generated or arrived in two different threads (the places named 'ThreadofClick' and 'ThreadofWords'). The time between two click (respectively word) arrivals has a mean = ClickArrival (respectively = WordArrival). The inter-arrival time between 2 click (respectively word) events has an exponential distribution with parameter r =1/ClickArrival (respectively 1/WordArrival). (Mean: 1/r and Variance: $1/(r^2)$). The density function of the inter-arrival time between 2 events is f (***x***) =r ∗ exp (- r ∗ ***x***), if *x* is greater than 0 and f (***x***) = 0 elsewhere. The inter-arrival time follows an exponential law, for the words and also for the clicks. If the time proximity between a word event and a click event is below the variable 'ProxyTime' and if these two events verify the grammatical and semantic conditions (given between brackets under the transition which models the'SFCo'- see Figure 6-) then these two events are fused into one command. Transitions model the PCA components distributed over the network. The mouse click LA is reduced to a simple thread. The transition 'RecognitionSystem' assigns a random label to each word present in the place 'WaitRecognition'. This random assignation does not model a real flowing speech because automatic modeling of user speech is outside the scope of this paper. However, it is sufficient to model times of recognition.

Simulation results:

The Figures 8 (a), (b) and (c) show the simulation results for WorArrival=ClickArrival=5000ms and ProxyTime=10000ms

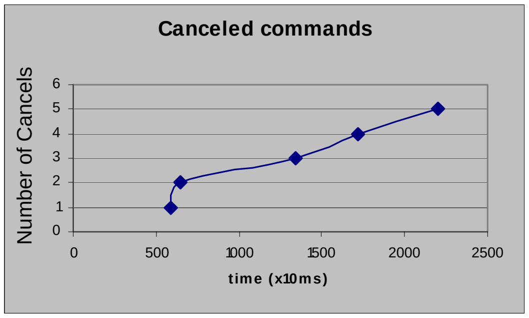

Fig. 8 (a): Canceled command.

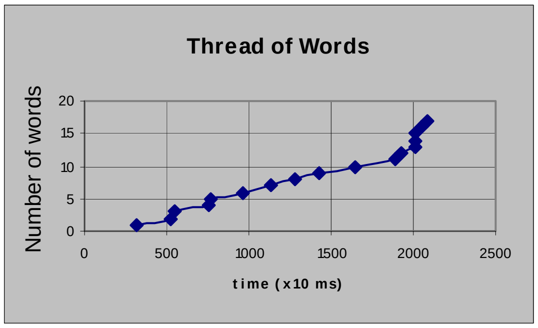

Fig. 8 (b): Thread of words.

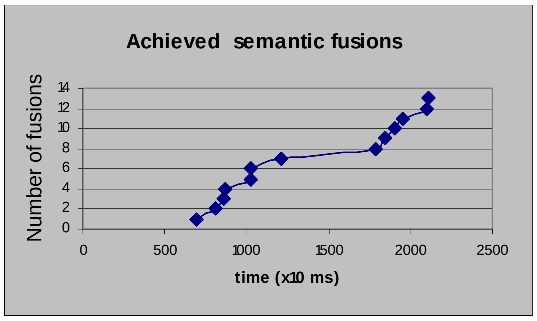

Fig. 8 (c): Achieved semantic fusions.

Figure 8 (c) presents the number of achieved fusions in the time (or the number of marks in the place 'FusionedMedia' of the CPN). In the same way, a command can be cancelled if the user says the word 'cancel' just after an achieved command (the proximity time between the two events: the command and the word 'cancel' is chosen below (ProxyTime/25)). Figure 8 (b) shows the accumulation of words in the corresponding thread (or the number of marks in the place 'ThreadofWords'). Figure 8 (a) shows the resulting cancelled commands in the time (or the number of marks arrived in the place 'CanceledCommand'). Figures 8 are obtained after the simulation of the network.

The results in Figures 8 (a), (b) and (c) quantify perceivable behavior of the architecture for random arrival time of inputs. This behavior depends on temporal proximity criterion. These results could vary according to the value of a proxymity time criterion used to achieve the fusion. The adjustment of this value should take into account the mean temporal behavior of users. This is done by a pertinent fine-tuning of the random generators with the function ExpLaw( ) [Jensen et al.1995]. It should also consider processing time, which is modeled by the values returned by the program of transition 'RecognitionSystem'.

The example of this section shows that the fusion engine works and performs semantic fusion (by combining results of commands to derive new results) as well as syntactic ones (by combining data to obtain a complete command).

The CPN example proposed in this section does not consider the problem of mark's accumulation in the multithreaded network. This important aspect could be easely resolved by adding new tasks to the distributed PCA or to an another network for error management.

5 CONCLUSION

In this paper an agent-based conversational model for multimodal fusion is proposed. The pipelined architecture of this model lead to new generic structures that unify applications based on multimedia multimodal dialog. They also offer to developers a framework specifying different functionalities used in multimodal software implementation. In a first phase, the main common requirements and constraints that multimodal dialogs need are gathered. Then the interaction types related to the early and late fusions are identified. The proposed fusion engine are modeled with a multithreaded timed Colored Petri Networks and supports both parallel and serial fusions. The quality attributes of the architecture are outlined to show the the genericity of our approach. An simulation example of a engine fusion is also presented.

ACKNOWLEDGMENTS

We wish to acknowledge the *Commission Permanente de Coopération Franco-Québécoise* 2003-2004, with the support of the ministry for the International Relations of Quebec and the Foreign Office of France (General Consulate of France in Quebec.) This project was also supported by the financial support of the Natural Sciences and Engineering Research Council (NSERC) of Canada.

REFERENCES

[Bolt 1980] Bolt, R.A., Put that there: “Voice and gesture at the graphics interface”, *ACM Computer Graphics 14,3,* 262-270, 1980.

[Crowley 1997] Crowley, J.L. and Bérard, F. “Multimodal tracking of faces for video communications”, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR’97)*, San Juan, IEEE Press, NY, June, 1997.

[Bellik 1994] Bellik Y., Burger D., “Multimodal Interfaces: New Solutions to the Problem of the Computer Accessibility for the Blind”. *Proc. CHI’94*, Boston, 24-28 April 1994.

[McGee 2000] McGee, D.R., Cohen, P.R., and Wu, L., “Something from nothing: Augmenting a paper-based work practice with multimodal interaction”, in *Proceedings of the Conference on Designing Augmented Reality*

Environments, ACM Press, Helsingor, Denmark, 71-80, April 12-14, 2000.

[Jensen 1997a] Jensen K., "Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use, Volume 1, Basic Concepts". *Monographs in Theoretical Computer Science*, Springer-Verlag, 2nd corrected printing, ISBN: 3-540-60943-1, 1997.

[Jensen 1997b] Jensen, K., "Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use, Volume 2, Analysis Methods". *Monographs in Theoretical Computer Science*, Springer-Verlag, 2nd corrected printing, ISBN: 3-540-58276-2, 1997.

[Jensen 1995] Jensen, K., Christensen, S., Huber, P and Holla, M., "Design/CPN Reference Manual", Department of Computer Science, University of Aarhus, Denmark, http://www.daimi.au.dk/designCPN/, 1995.

[Oviatt 2000a] Oviatt, S.L., "Multimodal Signal Processing in Naturalistic Noisy Environments", in B. Yuan, T. Huang and X. Tang Eds., *Proceedings of the International Conference on Spoken Language Processing (ICSLP'2000)*, Vol. 2, pp. 696-699, Beijing, China: Chinese Friendship Publishers, 2000.

[Oviatt 2000b] Oviatt, S.L., "Multimodal System Processing in Mobile environments", *Proceedings of the Thirteenth Annual ACM Symposium on User Interface Software Technology UIST'2000*, pp. 21-30, New York: ACM Press, 2000.

[Hutchins 1986] Hutchins, E. L., Holland, J. D. and Norman, D. A., "Direct manipulation interfaces", in Norman, D. A. and Draper, S. W. Eds., *User centred system design: new perspectives on human computer design*, Hillsdale, NJ, Lawrence Erlbaum, 1986.

[Oviatt 2000] Oviatt, S.L., Cohen, P.R., Wu, L., Vergo, J., Duncan, L., Suhm, B., Bers, J., Holzman, T., Winograd, T., Landay, J., Larson, J. and Ferro, D., "Designing the user interface for multimodal speech and gesture applications: State-of-the-art systems and research directions", *Human Computer Interaction*, vol. 15, no. 4, pp. 263-322, 2000.

[Bregler 1993] Bregler, C., Manke, S., Hild, H., and Waibel, A. "Improving connected letter recognition by lip reading", *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 1, pp. 557-560. IEEE Press, 1993.

[Project CNRS 1994] *Projet AMIBE*, Rapport d'activité, GDR |n° 9, GDR-PRC Communication Homme-Machine, CNRS, MESR, 1994, pp. 59-70 (Technical Report in french), 1994.

[Oviatt 1999] Oviatt, S. L., “Mutual disambiguation of recognition errors in a multimodal architecture”, *Proceedings of Conference on Human Factors in Computing Systems: CHI '99*, New York, N.Y., ACM Press, 576-583, 1999.

[Coutaz 1994] Coutaz, J., Nigay, L., “Les propriétés CARE dans les interfaces multimodales”, *IHM’94. Sixièmes journées sur l’ingénierie des Interfaces Homme-Machine*, Lilles, 8-9 Déc, (French paper), 1994.

[Jennings 1998] Jennings, N. R. and Wooldridge, M. J., “Applications of Intelligent Agents” in *Agent Technologies: Foundations, Applications and Markets*, Eds. N. R. Jennings and M. Wooldridge, 3-28, 1998.

[Weiss 1999] Weiss, G., *Multiagent Systems*, MIT-Press Ed., 1999.

[Bird 1993] Bird, S.D., “Toward taxonomy of multi-agents systems”, *International Journal of Man-Machine Studies*, 39, 689-704. 1993.

[Bond 1988] Bond, A.H. and Gasser, L., *Readings in Distributed Artificial Intelligence*, San Mateo, Calif.: Morgan Kaufmann, 1988.

[Ishida 1997] Ishida, T., *Real-Time Search for Learning Autonomous Agents*, Kluwer Academic Publishers, 1997.

[Muller 1996] Muller, H. J., “Negotiation principles”, in G. M. P. O’Hare and N. R. Jennings, eds, *Foundations of Distributed Artificial Intelligence*, pp. 211-229, Wiley, 1996.

[Cohen 1997] Cohen, P. R., Levesque, H. R., and Smith, I., “On team formation”, Hintikka, J. and Tuomela, R. (Eds.) *Contemporary Action Theory*. Synthesis, 1997.

[Ramdane] Ramdane-Cherif A. and Levy, N. “An Approach for Dynamic Reconfigurable Software Architectures”. in *IDPT’02: The Sixth World conference on Integrated Design and Process Technology*, Pasadena, California, USA, June 23-28, 2002.

[Abrial 1996] Abrial J.-R, *The B-Book: Assigning Programs to Meanings*, Cambridge University Press, 1996.

[Bellik 1995] Bellik Y.,: PHD Thesis of university of Paris XI (France). Thèse de Doctorat de l’Université de Paris XI, spécialité informatique. “Interfaces multimodales : concepts modèles architectures”, soutenue le 30 Mai 1995.

[Chen 1990] Chen, S.-C. and Kashyap, R. L., “Temporal and spatial semantic for multimedia presentations”, *International Symposium on Multimedia Information Processing*, pp. 441-446, Dec.11-13, 1997

[Kramer 1990] J. Kramer, J. Magee, "The Evolving Philosophers Problem: Dynamic Change Management", *IEEE Trans. On Software Eng.*, 16(11), pp. 1293-1306, Nov. 1990.

[Tabeling 2002] P. Tabeling, "Multilevel Modeling of Concurrent and Distributed Systems", in *SERP'02 International Conference*. P 94-100, 2002.

About the authors

Hicham Djenidi has been studying for his Ph.D. at PRISM laboratory, University of Versailles, France and Ecole de Technologie Supérieure (ETS) Canada, since 2003. His investigations and field interests concern, multimodal interactions, multi-agents architectures and software specifications. Hicham Djenidi is a student member of IEEE. E-Mail: djenidi@yahoo.com.

Amar Ramdane-Cherif received his Ph.D. degree from Pierre and Marie university of Paris in 1998 in neural networks and AI optimization for robotic applications. Since 2000, he has been associate Professor in the laboratory PRISM, University of Versailles Saint-Quentin en Yvelines, France. His main current research interests include: Software architecture and formal specification, dynamic architecture, architectural quality attributes, architectural styles and design patterns. E-Mail: rca@prism.uvsq.fr.

Chakib Tadj is a professor at ETS, University of Quebec at Montreal (Canada). He received his Ph.D. degree from ENST Paris in 1995. He is a member of Laboratory of Integration of Technologies of Information (LITI) at ETS. His main research interests are automatic recognition of speech and voice mark, word spotting, HMI, multimodal and neuronal systems. E-Mail: ctadj@ele.etsmtl.ca.

Nicole Levy is professor at the university of Versailles, Saint-Quentin en Yvelines, France. She has a doctorate of the university of Nancy. She directs an engineering school, the ISTY, and is responsible for the SFAL team (Formal Specification and software architecture) of PRISM, the laboratory of the University associated with the CNRS. Her main research interests are formal and semi-formal development methods, style and architectural models formalization, quality attributes of software architectures and distributed systems. E-Mail: nicole.levy@prism.uvsq.fr.