Appl Intell (2013) 38:586–599
DOI 10.1007/s10489-012-0387-3

Multi levels semantic architecture for multimodal interaction

Sébastien Dourlens · Amar Ramdane-Cherif · Eric Monacelli

Published online: 30 September 2012

Abstract This paper presents a semantic architecture for solving multimodal interaction. Our architecture is based on multi agent systems where agents are purely semantic using ontologies and inference system. Multi levels concepts and behavioural models are taken into account to bring a fast high level reasoning on a big amount of percepts and low level actions. We apply this architecture to make a system aware of different situations in a network like tracking object behaviours of the environment. As a proof of concept, we apply our architecture to an assistant robot helping blind or disabled people to cross a road in a virtual reality environment.

Keywords Knowledge representation language · Description logic · Ontologies · Multi-agent systems · Semantic memory · Multimodal interaction

1 Introduction

This is just the beginning of indoor and outdoor robotics assistance. However with actual computational power, ambient intelligence and ubiquitous connectivity, robot abilities begin to be reliable enough to be applicable to these kinds of human system applications. Many problems occur due to the amount of information being processed. Lots of events must be taken into account. Multimodal Interaction [1] systems try to solve these problems using two processes applied on events: multimodal fusion and multimodal fission [2]. The first composes higher level events from percepts. The latter takes composite events and splits them in low level actions control to act in the environment.

We developed a multimodal architecture for human-system interaction in the human environment. One main point of the architecture is the ability to understand what is happening in the environment in a situational context. We use a multi agent system (MAS) to build our architecture. We added to each agent a semantic memory composed of two ontologies: a domain ontology of concepts and an ontology of event models to store narrative and informative descriptions of facts. Another important point of our architecture is that it is fully compliant with Foundation for Intelligent Physically Agent (FIPA) from IEEE Computer Society and W3C consortium standards. Therefore all our services and agents are designed as networking webservices. It gives our architecture the ability to connect to ubiquitous and ambient sensors and actuators. Web services were designed by the W3C consortium with the goal of standardizing software services in a distributed and interoperable way on the World Wide Web. They are connected to the provider agents or consumer agents who need knowledge using the Simple Object Access Protocol (SOAP [3]) for Service Oriented Architecture (SOA) in the XML format. In our approach, Agent memory and communication messages require knowledge representation language (KRL). This language is defined by n-ary predicates. Each event is then represented by a predicate followed by a slot of several role-argument pairs. Each event is related to concepts which deeply link concepts ontology to events ontology. All events are stored in the agents' memory. According to the operation required by the fusion agents, they will query their memory for several previous events and create a composite event of a higher level of abstraction which will then be sent to other agents. The under-

S. Dourlens (✉) · A. Ramdane-Cherif · E. Monacelli
Laboratoire d'Ingénierie des Systèmes de Versailles (LISV),
Université de Versailles, Institut Universitaire de Technologie,
10-12 avenue de l'Europe, 78140 Vélizy, France
e-mail: sdourlens@lisv.uvsq.fr

standing will permit fission agents to send orders to services driving the actuators.

The objective of this paper is to present our architecture and show it is suitable to human-system interaction problems. For the system, multimodal awareness architecture consists in tracking and using some signals coming from the human environment and then to act accordingly and safely. For each different situation, we assume that values of sensors are valid in the simulation. The memory of agent stores the required knowledge about objects and the possible actions of these objects. For example, a car is a concept filled under mobile objects in the concept ontology and possible movements of a car on a road are stored under the *Move* or *Behave* predicates in the event models ontology. Depending of the abstraction level, *Move* predicate is used for short moves and *Behave* for more complex or composite movements.

In Sect. 2, we will present the related work. In Sect. 3, we will show our semantic multimodal architecture. In Sect. 4, we will study how to build a multimodal awareness system. We assume that input sensors outputs are correctly preprocessed by the services and are accompanied by a confidence value for agents error management on detected signal or failure on inputs. Finally, we will simulate our architecture to help people to cross a straight urban road where cars travel. In Sect. 5, we will conclude.

2 Related work

Lots of architectures have been designed in the aim of being embodied in a system, in a house, in the city or to simply bring an intelligent software component into a system. To realize multimodal interaction, there exist multiple types of architectures: dedicated HRI architectures like Situated Modules [4], C5 [5] and EICA [6, 7], cognitive architectures like updated ACT-R [8] or SOAR [9], and architectures with rational agents also called BDI agents [10]. The last 3 architectures listed previously have limited semantic reasoning capabilities and are often limited to First Order Logic (FOL). Excluding cognitive and rational agent architectures; there is to our knowledge no system capable of understanding the environment with a knowledge representation language as natural as can be the human language although Situated Modules and EICA can provide a more natural embodiment because of their inherent responsivity.

The systems mentioned can use models to recognize situations but are often limited to dedicated tasks; this is not the case for our architecture. Other systems [11] that use multimodal fusion and fission functions exist but they aren't semantic, they don't use any inference engine.

Gupta et al. present a promising method to observe human-object interactions [12]. Their fusion system on image and video uses spatial and functional compatibility for recognition. Graphical and Bayesian models are used to understand human scenes and events [13]. Object Classification is done by affecting stochastic values to relationships between objects in the database and Action or Activity Recognition [14] is done by affecting stochastic values to event models in the database; these values can be used to choose an object or an action and are often modified through learning. We will deeply improve these two operations by using a *contextual ontology* to store concepts, values for classification and predicates of models for actions. Coradeschi and Loufti propose a review of *Perceptual Anchoring*, a transversal domain where it is necessary to associate a symbol to signal sensor levels for the representation of a physical object by using context and higher level semantics information [15]. This is an important way to have symbolic information but this information is related only to intelligent sensor and not structured in the way to facilitate reasoning and understanding in the interaction system. Recent researches have been made to store low level data in database structures. In our approach, data, events and high level information like scenarios are stored in an ontology structure.

The advantages of using ontologies are:

- The ability to share and reuse common understanding and modelling of the exchanged information among people and software agents, to enable the reuse of domain knowledge (concepts and models), to make domain assumptions explicit, to separate domain knowledge (ontology of concepts) from the operational knowledge (ontology of models), and to analyze domain knowledge [16, 17];
- The use of natural language. Concepts in the ontology express real objects (physical or logical) and relationships in the domain of application. These are most likely to be nouns (objects) or verbs (relationships for concepts, predicates for models) in sentences that describe the domain.
- Hierarchically filled concepts by semantic relationships giving automatic expression of situational meanings. These semantic ontologies have a better expression of the relations interconnecting several classes of concepts. Instances are directly stored under the class following the concepts list.

The ideal scenario of Guarino [18] has been further investigated in domain-specific modelling research where it is generally accepted that an existing or newly developed ontology can act as a formal specifications of the domain of the language. It can afterwards be transformed in a domain-specific metamodel for the language. The ontologies used in this context are domain and task ontologies which specify, respectively, a conceptualization of a material domain (e.g. a software quality ontology) and a generic task (e.g. a measurement ontology) and which have themselves been created by specializing concepts of a meta ontology. Domain ontologies can play an important role in the quality assurance and

evaluation of models where quality is the degree to which a model respects the invariant conditions of a domain, as axiomatized in an ontology of concepts. This domain-specific quality is assured if schemas instantiate models which include the domain-specific constraints or rules.

The use of KRL with the ontology structure gives a powerful mechanism for fast reasoning and understanding of the environment. This permits solving issues such as large quantity of incoming events to manage from multiple sources, environment modelling or environment understanding.

More recently, MultiML [19] (not OWL compatible) and EMMA [20, 21] (OWL compatible) are the most advanced XML languages for Multimodal fusion and fission incorporating multiple modalities, contexts, locations and time but these languages don't deal with the interaction meaning however they bring a bridge to semantics interpretation, they want to represent modalities and recognize scenarios using their own XML tags. In addition, Giuliani and Knoll also present related work [19] like MURML [22], MMIL [11], Cogest [23] and MIND [24] languages or systems built for conversational agents without standard ontologies. These XML-based languages are limited to first order logic reasoners.

Predicates logic and Description logic bring a part of the solution to the lake of representation (events and concepts) but they are also limited to the FOL to ensure completeness. In reality, higher order logic is closer to the reality, unfortunately it cannot be fully proven in general but may be proven in the simplified context. That's why, we used KRL to use predicates in higher level logic and multimodal logic at different levels of abstractions. The objective is to reduce the reasoning time and memory space necessary to solve a problem. Our approach wants to provide an independent environment representative language compatible and close to natural language, and W3C standards like EMMA, it will allow the communication with web services controlling hardware parts. The meaning appears from the linked concepts using predicates to represent knowledge with context and narratives [25, 26] but without complex human common sense. In order to build our interaction architecture, we are looking for a design and realization of some cognitive or semantic components to achieve goals of multimodal interaction in the human environment [18, 27].

We focus on intelligent architecture by integrating semantic agents, semantic services as structural components, ontology as knowledge bases [28, 29], inference systems and KRL as communication protocols.

3 Multimodal architecture

3.1 Architecture

We have presented an architecture, based on semantic agents for multimodal interaction, capable of understanding its environment in several of our previous papers [30, 31]. In this paper, we have proposed a state of the art of the multiple technologies and standards that we have used. The environment is a set of systems where our architecture can be seen as a distributed system managing the interaction in a network in this environment. Figure 1 presents the fusion and fission operations realized by agents connected to input/output services that are used for perception and action (Fig. 1). The control is not centralized. Each agent has its own inference engine to reason on its own memory. Agents and services are parts of the environment which contains hardware parts and software parts. Some are embedded in robots; others are in a building or in servers. The hardware part contains sensors, actuators and the communication network. The software part however only contains networked web services. We can distinguish two types of web services: Semantic agents that take care of the cognitive capabilities of the system and the input/output services that control the hardware parts and communicate with the other agents (Fig. 2). Web services choice has been made for the interoperability of communication and execution. The input services send agents some events whose low level data has already been processed and most of the time they are accompanied by a confidence rate. The output services receive from agents and manage incoming orders upon reception or according to their planning.

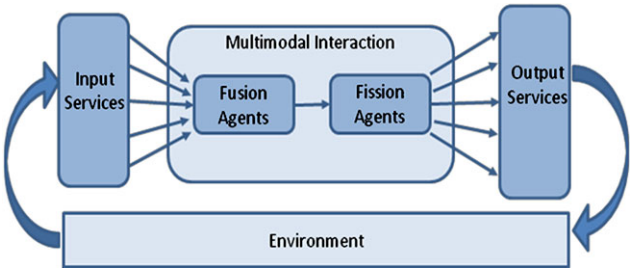

Fig. 1 Multimodal interaction architecture

We have also pointed out the operations that semantic agents are capable of realizing:

- modelling and memorisation (i.e. storage of events in a model ontology which is itself linked to a concept ontology),
- comprehension and reasoning for the extraction of new events with the support of an inference engine from query models or rule models, and
- communication with other agents or services [32].

To homogenize everything, the models, the facts, and in general, all the events that transit on the network are described in an Environment Knowledge Representation Language (EKRL) based on frames and not limited to FOL. We have also presented the roles of fusion and fission that those agents can take in the case of multimodal interactions.

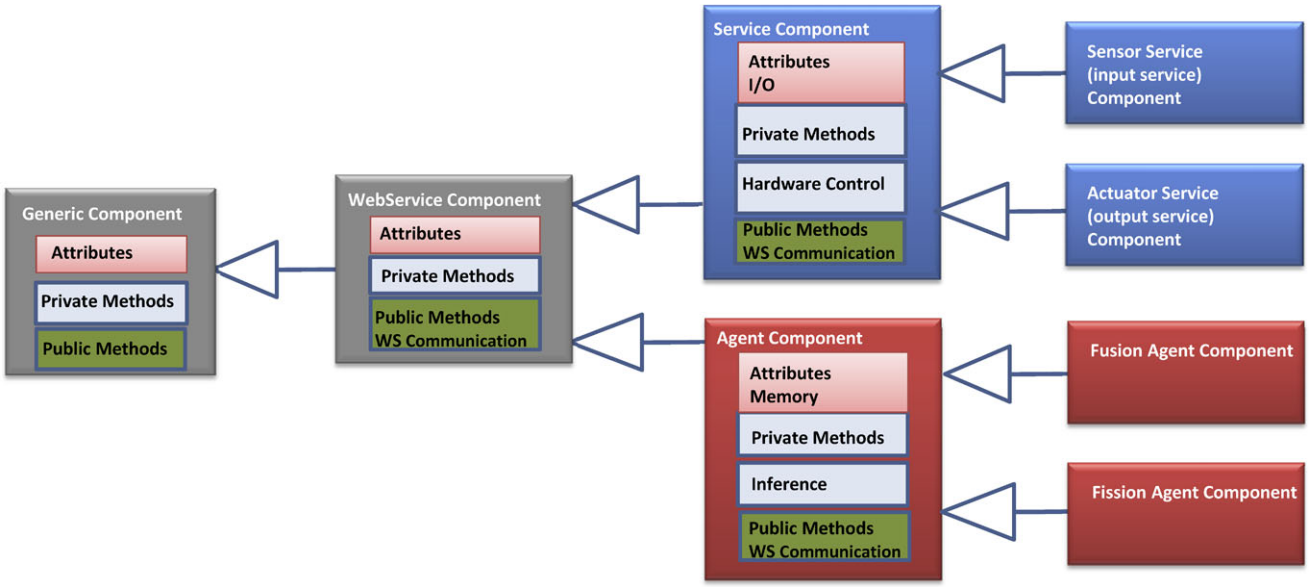

Fig. 2 UML components

Fusion is the process of composition of low-level abstractions (coming from input services or other agents) into an event of a higher level of abstraction (sent to other fusion agents). Fission is the process of decomposition of high level abstraction events (coming from other fusion or fission agents) into events of a lower abstraction level (sent to other fission agents) or in orders (specific events sent to output services). The fusion agents of the higher level of abstraction therefore have in memory all the information on observed contexts or situations in the environment. All of the event models in the semantic agents memories will permit:

- the storage of facts perceived (by the input services) and recognized;
- the understanding of these facts (thanks to semantic relationships);
- the fusion or fission of new events by reasoning with rule models on stored facts; and
- the reaction by sending orders to output services.

We have also standardized our components. The inference engine and communication algorithms of our semantic agents never change but the model ontology and the knowledge acquired with experience can evolve and be reused. In this paper, we will show how the inference engine can do the fusion or fission processes. We present our approach to apply a more generic and adaptable architecture able to manage awareness in parallel to other system tasks with a great understanding and disambiguation of the environment. We will be using models dedicated to the application that we developed.

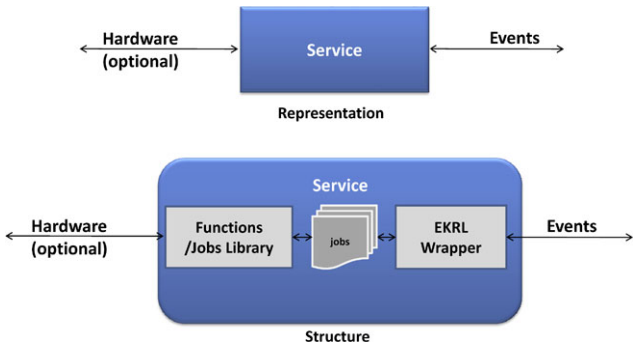

Fig. 3 I/O service

3.2 Semantic agents and services

Semantic agents and services are parts of our assistant system.

3.2.1 Services

Services are standard web services communicating with agents in EKRL events (Fig. 3). They may drive hardware parts or propose software functions. There are input services which can be connected to the environment with sensors to read data or send the results of software functions. And there are output services which can be connected to the environment with hardware actuators to execute orders or receive inputs to software functions.

3.2.2 Agents

Agents cannot act but just reason on EKRL events. They contain a knowledge base (we also called cognitive mem-

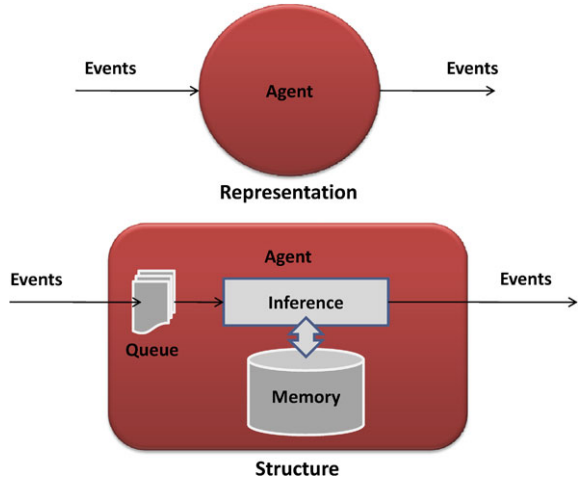

Fig. 4 Semantic agent

ory) composed of an ontology of concepts and an ontology of event models, an inference engine and a communication module (Fig. 4).

The inference engine is able to process the matching operation to answer a query and the aggregation function to process rule models. Most of the time, they receive new facts in their queue, they store the facts if they recognize them (i.e. a fact matches a known model). If they don't recognize them, they are simply ignored meaning that the agent is not concerned by this type of information. Once the new fact is added to the knowledge base, the inference engine processes an aggregation with rule models related to the model of the new fact. If rule models are fired, then new events are produced and sent to other agents or services (message is an order). We detail this process in the next section. Events, Scenarios and execution schemes of agents are stored in the knowledge base. All inserted facts coming from the network are linked to concepts and models. Event models instances (i.e. facts) are stored under classes of event models hierarchically. They compose the knowledge and meaning of what's happening in the environment (Fig. 5).

3.2.3 Inference engine

Knowledge representation language Our KRL is a semantic formal language L that can describe events in a narrative way very close to natural language. EKRL is fully used to build event messages and store facts directly in the models classes of the models ontology of the agent memory. The formal system is composed of the formal language based on variable arity relations in logic of predicates. It permits to realize semantic inference in order to extract the situational meaning from ontologies and match the instances of the ontologies. A predicate P is a semantic n-ary relationship between Roles R and Arguments A, and represents a simple

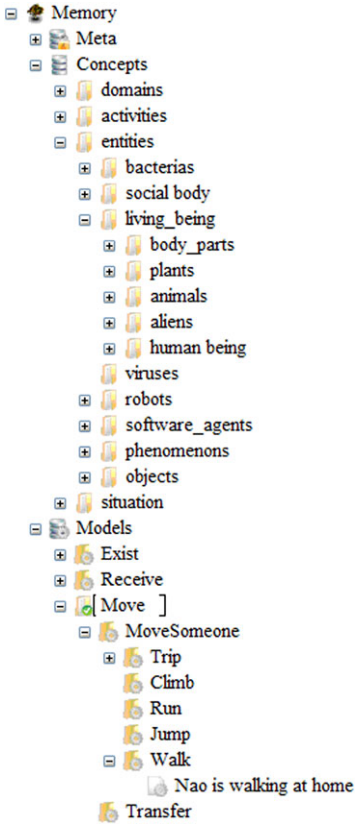

Fig. 5 Ontologies in memory

<RootPredicate>:<PredicateName> *(name)*
Natural language: '<Predicate Description>' *(optional)*
<Role1> <Arguments1>
<Role2> <Arguments2>
<Role3> <Arguments3>

Fig. 6 Event model description

Exist: Available Service
SUBJECT: composition
SENDER: services
DATE: date time
LOCATION: location

Fig. 7 "Exist:Available Service" event model

event or a composed event. It is denoted by the following formula: $P((R_1 A_1) \ldots (R_n A_n))$.

Where R are roles in the event (SUBJECT, OBJECT, SENDER, DATE, LOCATION, and so on) and arguments A are the possible values or instances of concepts in the stored facts. Figure 6 presents the writing of an EKRL event and Fig. 7 an example of an event model used for service composition. These models will be filled with a

combination of concepts to make an event instance (i.e. a fact) at several abstraction levels of behavioural description.

Matching Function This function performs matching operations between predicates and roles (Algorithm 1). *StoreRA()* and *ReadRA()* are SQL Insert and Select operations respectively in the Role-Arguments table filtered by ID arguments given to these two functions. *Modifier* is a role for modifying the sense of an event. Depending on modifiers, the matching result can vary. If the sense of the fact is negative, the event's sense is inverted so matching must take account of this too.

The function prototype is [EventsPredicateID]←Matching(RootPredicate, Predicate, [Roles], [Arguments]).

```
RootPredicateID← GetNodeID(RootPredicate)
PredicateID←GetNodeID (Predicate)
[RolesID] ← GetNodeID ([Roles])
[ArgsID] ← GetNodeID ([Arguments])
[EventsID]←ReadRA(RootPredicateID, PredicateID,[RolesID])
For Each EventID in [EventsPredicatesID]
    [[EventRolesID],[EventArgsID]] ←ReadRA(EventID)
    For Each ArgID in [ArgsID]
        EventArgID←EventsArgsID[rank(ArgID)]
        If not MatchArguments(ArgID,EventArgID) Then Next EventID
        Next ArgID
   If ApplyModifier(EventRolesID) != ApplyModifier(RoleID) Then
Next EventID
    [EventsPredicateID]← [EventsPredicateIDEventID]
Next EventID
Return [EventsPredicateID]
```

Algorithm 1. Events matching

MatchArguments function This function performs matching operations between two arguments of a role (Algorithm 2). *ReadConcept(QueryArgID)* is an SQL Select operation in the *nodes* table, where the node type is concept classes or instances and where these nodes are under the given node using the subsumption relationship of the *links* table. The SQL request gives all the subtree nodes sorted. The "date", "location", "context", "content" and "value" role arguments will be compared using specific meta operators like ">", "<", "<=", ">=", "AND", "OR" to check the given event models. This explains why the *CompareEvents*() function returns a boolean. In fact, any event calculations are made using this function. *Modifier*s is a role of predicate to change the meaning of the sentence (fact) and thus the applied logic (temporal, spatial, modal with necessity and obligation modalities, and so on). This function compares all arguments and applies a modifier to the final Boolean result. The function prototype is [*MatchingResult*]←*MatchArguments*(*QueryArgID*, *EventArgID*).

```
If QueryArgID=EventArgID Then Return True
[ArgsID]=ReadConcept(QueryArgID)
For all ArgId in ArgsID
    If ArgId=EventArgID then Return True
    Else Return CompareEvents(ArgId, EventArgID)
Next
Return False
```

Algorithm 2. Argument matching

Aggregation function The main code can be called at each time new facts are stored or at different time (interval specific to the agent triggered by a timer set by the designer). The given events, roles and arguments are compared to the rule models stored in the agent's memory. Rule Models set is equivalent to the program of the agent. They contain a precondition to be checked and if the rule model is fired (precondition is true) then all roles and arguments of facts in memory will be aggregated with the rule models (Algorithm 3). To be fast, the inference engine looks only at the facts of event models used in the rule model, this is an important optimization. If rule models match past and incoming facts (the precondition) then new events are produced (the postconditions). These events are obtained from the class of rule models. The arguments in the roles of all corresponding facts are aggregated to fill the roles of the rule models by using the matching algorithms previously given.

```
For all rule models r,
    For the k model events ek in r.precondition
        [ek,FactsID,k]←Matching(ek.rootpredicate,ek.predicate)
    Next
    [TrueFactsID]←Evaluate(r.precondition, ek, k, ReadRA(FactsID))
    For all tfid in TrueFactsID
        [NewEvents]←InsertArgs(r,tfid)
    Next
Next
Return [NewEvents]
```

Algorithm 3. Aggregation

For example, there is a very simple rule model "Behave:CrossTheRoad" (Fig. 8) which is composed of $k = 2$ facts named *f1* and *f2* (Table 1). The precondition is in the Object role of the model (Fig. 9). Depending on used roles in the precondition and postcondition, the matching will use temporal logic (working on dates), spatial logic (working on locations) and in general in multi modal logics corresponding to any roles of the model. The inference engine will also use the confidence value to validate the rule model to be aggregated. The composite confidence will be the mean of confidence values of different input events. Default confidence of an input event is 1 (1 for 100 % and 0 to 0 %). If this aggregated confidence value >0.5 then composite event is produced by aggregation with formulae of rule model. If the precondition is true, i.e. the arguments of facts match the

Table 1 Facts matching "Behave:CrossTheRoad" rule model

Fact	Event names	Roles	Arguments
f1	Move:Stand	user, content, date1, date2, location	"james", "speed = 0 m/s" 0 s, 1.7 s, sidewalk
f2	Move:Walk	user, content, date1, date2	"james", "speed = 0 m/s" 1.7 s, 3.5 s, street

Behave:CrossTheRoad
Object:*precondition*
User: *f1.User*
Source: *f2.position*
Recipient: *opposite sidewalk*
Content: *f1.value*
Date1: $inf(f_k.date1)$, for all f_k
Date2: $sup(f_k.date2)$, for all f_k
Location: *f1.location*

Fig. 8 "Behave: CrossTheRoad" rule model

precondition ←*Coord(f1.date2=f2.date1,f1.name="Stand", f2.name="Walk",f1.location="sidewalk",f2.location="street")*

(a)

Exist:ContradictoryEvents
Object:*Coord(f1.name=f2.name,f1.value<>,f2.value, f1.date1=f2.date1,f1.date2=f2.date2)*
Date1: *f1.date1*
Date2: *f1.date2*

(b)

Fig. 9 (**a**) "Behave: CrossTheRoad" precondition. (**b**) "Behave: CrossTheRoad" rule model

arguments of the *Object* role, and then the inference engine will produce and send the "Behave: CrossTheRoad" fact. This fact is the aggregation of the arguments of the facts *f1* to *f2* in the other roles of the model.

If two percepts or facts *f1* and *f2* contradict each other, the matching function still works without distinction because of the two following reasons: First, percepts or facts will be taken one after the other because of the presence of an input queue of each agent; And second, they will be also processed and filtered by the precondition embedded into the model of composite event (rule model).

The evaluation (Algorithm 3) of some contradictory values of some arguments in the precondition of a rule model can be false. Then the resulting composite event will be erroneous too and propagated to the system. Precondition must be checked by designer. To avoid any problem of this kind, these contradictory events can be filtered by a fusion agent with the addition of the "Exist:ContradictoryEvents" rule model event (Fig. 9). Two cases can be created: the events are propagated to other agents with a confidence of 50 % or can be simply ignored but it is a choice of a designer depending on the security of the realized application.

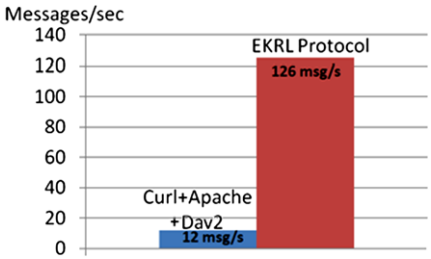

Fig. 10 Networking load—event message transfer rate

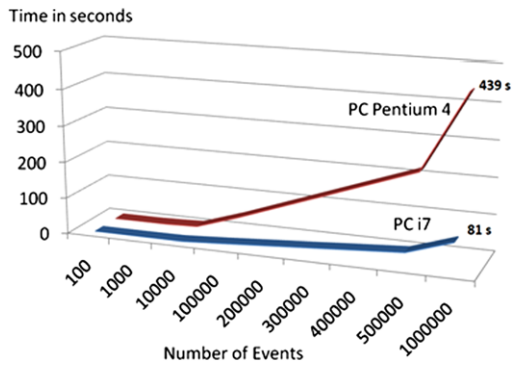

Fig. 11 Inference time needed to process the maximum number of events

3.3 Performance analysis

Reception and Emission of events are done by the communication module which is physically connected to the TCP/IP network.

To avoid potential bottlenecks, we have checked network performance (our architecture can manage about 100 events/second). As received and produced events are kept in memory of the agent, we have also checked the storage. A Temporal validation of events built by inference engine has been tested with CPNTools to formally analyze Coloured Petri Nets [33].

3.3.1 Network load

The networking load indicates the maximum number of messages per second that can be transmitted between two computers (Fig. 10).

3.3.2 Inference time

Inference time is a measurement of the maximum number of inferences performed per second. It depends on the number of data read from the database. We build a data set to insert the concepts and models into the ontologies of our agents (Figs. 11 and 12). Considering there is no queuing event, we send a query on facts to the agent.

Our query takes very little time, especially with the i7 processor, so we measure the required time for the agent to

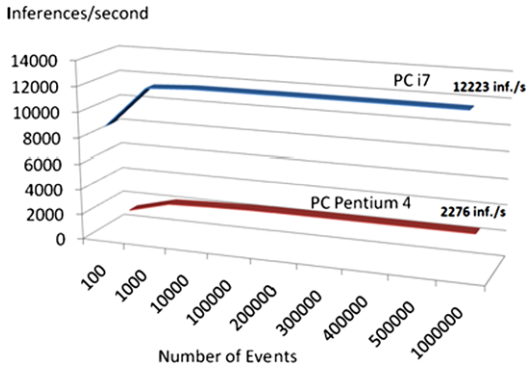

Fig. 12 Rate of inferences by second

Constraint	Result
Events Completeness	100%
Event arrivals order	100%
Temporal aspect	100%
Consistency	100%

Fig. 13 Validation results

execute *nbe* inferences, where *nbe* is the maximum number of events that can be returned by the database.

3.3.3 Temporal validation

With CPNTools, we validate event completeness, event arrivals order, temporal constraints (included in the event model) and the consistency (Fig. 13).

We may explain these results by the fact that unknown events are ignored and the validation can't continue. If the event arrivals order is not respected, the engine doesn't wait and restarts again to the first event. The temporal validation allows validating the event arrivals order and the start time and end time of the models are respected. Consistency is respected because of the presence of the concept and model in our ontologies, and because no generated events is altered by the network transmission or by a limitation of the reception queue in this program. If it is not the case, the event is ignored and the matching process must keep on. Inconsistency doesn't mean there is no repetition of the same output event. We managed this problem with an automatic deactivation parameters attached to the fact in the ontology. Our inference engine is validated for all scenarios because, if the rule model changes, only the comparators of the schema changes. Only the arguments of the roles of the incoming events will be matched with the rule model precondition.

4 Pedestrian crossing

In this section, we present how our semantic architecture of agents can help the pedestrian to cross a road.

Table 2 Pedestrians must wait on the sidewalk

Light color	No car present	Car stopped	Car not stopped	
		Speed null	Speed decreases	Speed increase
Green	Wait	Wait	Wait	Wait
Orange	Wait	Wait	Wait	Wait
Red	Walk	Walk	Wait	Wait

Table 3 Pedestrians are crossing

Light color	No car present	Car stopped	Car not stopped	
		Speed null	Speed decreases	Speed increase
Any color	Walk	Walk	Run if possible	Run if possible

4.1 Context

Assistant robot helps people to cross a road by checking the colour of the traffic light, presence of car on the road, position and speed of that car. The objective of our application is to keep a valid user aware of the situation or to protect disabled, blind or old people by making them avoid crossing in situations of danger. We will focus on people crossing the roads and to avoid being a victim of traffic. We use our multimodal architecture to solve this type of problem by applying it to only one crossroad. Obviously, some more actions could be performed, like the robot could assist disabled persons, control the pedestrian signal or car signal (keeping it "red" until arrival to the other side), make an emergency call in case someone wants to inform authorities about an accident.

Tables 1 and 2 present possible situations and required actions in order to protect pedestrians. Table 1 shows the case where pedestrians are waiting on the sidewalk and Table 3 shows the case where pedestrians are going across the road. For each situation evaluation of danger (matching of events), robot acts by speaking to humans for example.

4.2 Global architecture

The architecture is composed of

- a simulator in Microsoft Robotics Studio of the city street and cars circulating on that street;
- the Microsoft Kinect sensor linked through the network to a PC containing the service to manage the sensor;
- Semantic agents that can be embedded in a robot or a computer connected to the network;

and the Nao robot also linked to the network is capable of moving and talking. We don't use its camera to detect gesture because of the low computational power of its main board (Fig. 14). System architecture contains 3 layers: Hardware, Software Services and Software Agents. Hardware

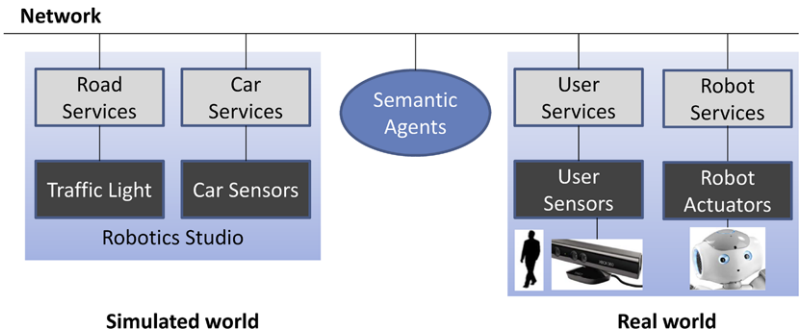

Fig. 14 Application architecture

layer consists of sensors and actuators in the robot or from a network of the system. Our Nao robot's body has 1 head with four mikes and two video cameras, 1 neck, 2 legs, 2 arms and 2 hands for a total of 25 DOF. It can listen, speak, walk and connect to a TCP-IP network. We added in the environment a MS Kinect to recognize human gestures and movements on the sidewalk. Software layer is a suitable composition of fusion and fission agents (internal or external mind) and embodied services as defined in the previous sections. Agents will manage input and output events to finally manage awareness and hardware controllers.

Car and Red Light are simulated web services into a MS Robotics Studio simulation and projected on a wall in a virtual reality room (Fig. 15). They send event about their own actions (car moves, car stops, light color changes), car speed (0 to 50 km/h) and color of light (green, orange, red) on Fig. 16. The user may walk in front of the simulation, Kinect service gets human gestures (skeleton) and send EKRL events to agents (Fig. 17).

4.3 Simulation preparation

This simulation is done in a virtual reality room with the simulation projected on a wall. Humans (one or two people) and Nao robot are real. A Kinect sends high level events of human gestures and postures to agents. When the humans are close to the wall of video-projection (less than 2 meters from the wall), we consider them on the pedestrian crossing else we consider them on the sidewalk (more than 2 meters from the wall). Cars and Red Light parts are web services that send events about themselves to agents. The car goes around the block with different speeds so that the arrival is random. Sometimes it stops at the red light, sometimes not, making it unpredictable. When car's position is further than 50 meters from the crossroad, we consider it is a situation where no car is present. Hardware and software parts of the system are controlled by services connected to agents and waiting for orders. In case of possible injury, robot will speak or play a sound to warn users by triggering an output event. Table 4 summarizes and presents the relationships between Hardware, Services and Event models used by agents to understand the situation in this application. We are now able to run our experimentations. We tried all situations of Tables 1 and 2 in the real world.

Fig. 15 Virtual reality projection on the wall of the virtual room

Fig. 16 Car and red light services view in simulation

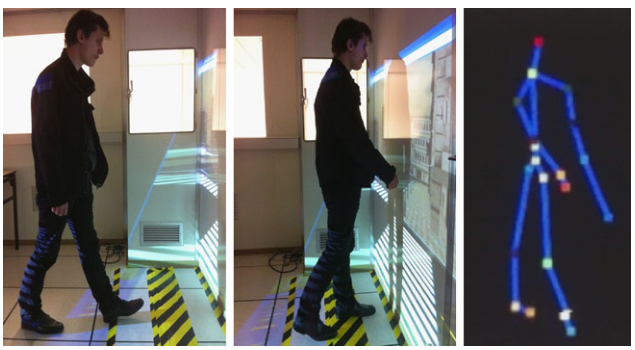

Fig. 17 Kinect "Move:Walk" event detection

Table 4 Hardware, services and event models relationships

Hardware	Service	Event model
Video Camera Sensor	Human Gesture Recognition (Kinect)	Exist:Human, Move:Walk
Speakers	Audio Output, Vocal Synthesis (Nao)	Behave:Play, Behave:Speak
Mike Sensors	Vocal Recognition, Sound Direction (Nao)	Exist:Entities, Move:Entities, Behave:HumanTalk, Exist:SoundLevel
Neck Actuator	Move Head (Nao)	Behave:Avareness, Behave:Look
Arms	Arms Gestures (Nao)	Move:MoveArm
Legs	Walk (Nao)	Move:Walk
Traffic Light	Signal Color (Pedestrian Signal)	Exist:SignalColor
Car	Move Speed (Car)	Move:MoveObject
UDDI.OWL-S Server	UDDI (input/output service), WSExecute (output)	Exist:AvailableServices, Exist:AvailableAgents, Behave:ExecuteService

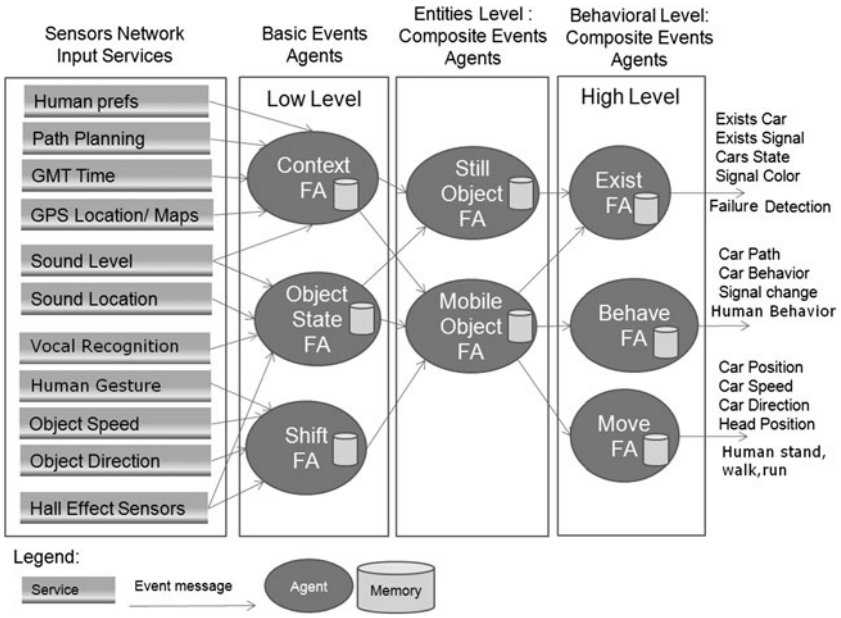

Fig. 18 Fusion agents dedicated to robot awareness

4.4 Architecture of agents

In this section, we develop two specialized semantic agents dedicated to our interaction problem.

4.4.1 Fusion agents

A *fusion agent* is a semantic agent that has the role of extracting a specific meaning from several events. The matching operation of the inference engine will use event models to extract the required meaning of a higher level of abstraction. Figure 18 shows a complete composition of input services and fusion agents to produce semantic events at different levels of abstraction from sensors input. Contextual events produced by fusion agents will be stored in memory under model events like "Exist:ExistsCar", "Exist:ExistSignal", "Exist:CarsState", "Exist:SignalColor", "Exist:FailureDetection", "Behave:CarPath", "Behave:CarBehaviour", "Behave:HumanBehaviour", "Move:CarPosition", "Move:CarDirection", and so on.

Exist: Signal Color
OBJECT: Pedestrian Signal4
VALUE: Green Color=1
ACCURACY: 100%
DATE: 10/05/2010 10:05
LOCATION: COORD(Hoche Street,5th Avenue)

Fig. 19 Composite event "Exist:Signal Color"

Object State agent, in charge of object detection, composes the following event from sensors events. Still Object fusion agent will receive and store this event concerning the signal. These events will be composed by next agents to start evaluating events concerning cars and signal. Figures 19, 20 and 21 are composite events made by fusion agents from events coming from Robotics studio simulation services and Kinect "Human Gesture" service.

```
Move: MoveObject
  OBJECT:      FourFourCar1
  VALUE:       Speed=20 km/h (deceleration)
  ACCURACY:    95%
  DATE:        10/05/2010 10:05
  LOCATION:    Hoche Street
```

Fig. 20 Composite event "Move:Object"

```
Move:Walk
  OBJECT:      Human
  SENDER:      MobileObjectFA
  DATE:        10/05/2010 10:05
  LOCATION:    Hoche Street, crossroad4
```

Fig. 21 Composite event "Move:Walk" made by fusion agents

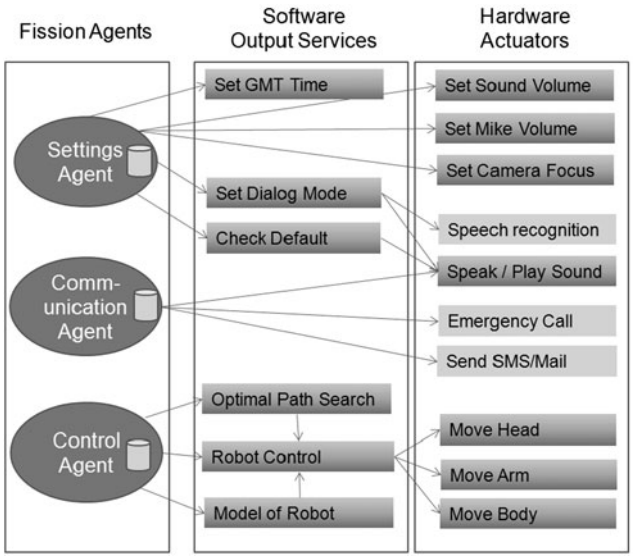

Fig. 22 Fission agents dedicated to robot awareness

4.4.2 Fission agents

A *fission agent* is a semantic agent that has the role of managing information to select and control actuator services. Fission agent will produce events sent to other agents and services to store current state and execute work in time. Figure 22 shows a composition of fission agents to act on hardware layer. Depending on input events, the objective is to prevent user to cross a road when danger is present. To achieve this goal, a set of actuators is available.

Figure 23 presents an example of output orders sent from Control Agent to a robot control service (LeftArmMotor) to move the left arm up to the 45° position. Figure 24 is an example of order to speak sent to the VocalSynthesis1 service embedded in a robot or elsewhere (speaker set in a wall or on the red light).

```
Move:MoveArm
  OBJECT: NaoRobot1
  VALUE: 45°
  SOURCE: ControlAgent1
  RECIPIENT: LeftArmMotor
  DATE: 25/04/2010 11:05:56
```

Fig. 23 "Move:MoveArm" event instance

```
Behave:Speak
  SOURCE: Coord(GreenColor,CarStop)
  CONTENT: "You can walk"
  SENDER: Communication Agent
  RECIPIENT: VocalSynthesis1
  DATE: 25/04/2010 11:05:56
```

Fig. 24 "Behave:Speak" event instance

```
Behave: SomeoneFallsDown
  OBJECT:      James
  SENDER:      FallingDetectorMobileService
  VALUE:       10 m/s
  ACCURACY:    56%
  DATE:        10/06/2011 10:05
  LOCATION:    COORD(Hoche Street,5th Avenue)
```

Fig. 25 "Behave: SomeoneFallsDown" fact

4.5 Application extensions

In the previous section, despite the number of sensors, actuators and agents, the application appears intentionally simple in order to be explicit. But the interest of this architecture, except the use of knowledge representation language close to natural language, can be easily improved by adding more components (services and agents) and knowledge in the memory of agents bearing more scalability, management of a large set of events and then consistency. So in this section, we propose a more complex application taking into account of more events with possible noisy facts (unknown and false data). We then consider integrating more sensors in the network, one for the weather (new "Exist:Weather" event and "Exist:WetRoad" rule model) and the others are mobile phone with a 3-axis gyroscope combined with the accelerometer sensor (6-axis measurement) carried by users to detect a possible falling on the pedestrian crossing using a new "Behave:SomeoneFallsdown" event model (Figs. 25, 26 and 27). [34] presents an example of application of accelerometer used for activity recognition.

The robot is one part of the network and is connected to any other information, even about what is happening to the 3 other pedestrian crossings. This gives the architecture the ability to manage all the crossings and even the traffic of the street and why not the citywide, if for example a police car or the presidential convoy needs to go through at high speed. Output services can control the lights using the new

```
Exist:Weather
   SUBJECT:     Weather
   SENDER:      CityMeteoService
   VALUE:       Rainy (3 mm)
   ACCURACY:  41%
   DATE:        10/06/2011 10:05
   LOCATION:   COORD(Hoche Street)
```

Fig. 26 "Behave: SomeoneFallsDown" fact

```
Exist:WetRoad
   SOURCE:      Coord(Exist:Weather,Behave:Behave:Sprinkle)
   PRECONDITON: Altern(Weather =Rainy,Sprinkle =on)
   VALUE:       Event.value
   ACCURACY:  Event.accuracy
   DATE:        Event.date
   LOCATION:   Event.location
```

Fig. 27 "Exist:WetRoad" rule model

```
Behave:ChangeLightColor
    SOURCE: Coord(GreenColor,CarStop)
    CONTENT: "You can walk"
    SENDER: Communication Agent
    RECIPIENT: VocalSynthesis1
    DATE: 25/04/2010 11:05:56
```

Fig. 28 "Behave:LightChangeColor" event instance

"Behave:ChangeLightColor" to avoid injuries, or if the road is wet (rainy weather), by delaying the red light if the light should be green and a person is falling or has fallen (Fig. 28).

In this experimentation, we store lots of events and prepare them to check the three following situations: the number of events is normal (less than 1000 events/s), overloaded if we send more events to agents, and noisy when the events have an accuracy lower than 50 %. The agents are stressed and they have to store all the events in memory for us to check consistency and robustness.

4.6 Results

Most of the knowledge about cars, signal and behavioural scenario were stored in the memory of agents. Evaluation of the situation consists to query this memory. Agents evaluate the meaning of the situation with its past recorded events. Different situations summarized in Tables 2 and 3 have been simulated with 100 tries. We have also tested the cases with several cars.

In the case "Pedestrians are waiting on the sidewalk" of Table 1, results are 100 % successful. Other results for the case "Pedestrians are crossing" with one car and with several cars are presented in Tables 5 and 6. And as it was expected, cars speed determines the dangerousness especially when pedestrians are on the walkway. In this clean environment (no sunlight, no other objects moving) and after a short

Table 5 Pedestrians are crossing with one car

Light color	No car present	Car stopped	Car not stopped	
		Speed null	Speed decreases	Speed increase
Any color	100 %	100 %	99 %	98 %

Table 6 Pedestrians are crossing with several cars

Light color	No car present	First car stopped	Car not stopped	
		Speed null	Speed decreases	Speed increase
Any color	100 %	100 %	86 %	79 %

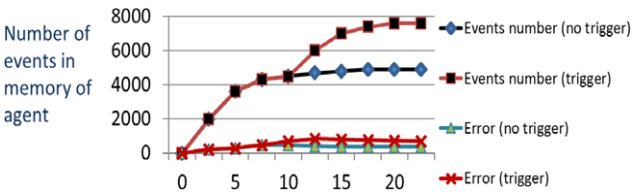

Fig. 29 Events in memory

time of Microsoft Kinect calibration, with sometimes some erroneous gesture recognition events, results are:

- at slow speeds 100 % of the cases are successful;
- at fast speeds with one car, we have 98 % of cases are successful;
- at high speed and several cars, 79 % of cases are successful.

These problems are due to the short distance between two cars that may occur like in true life when people drive when the orange light is on or don't see people crossing. If people are not on the pedestrian crossing, it is always possible to avoid danger but once on the pedestrian crossing, in most of the cases, it is impossible to avoid the accident. To the event that vocally allows pedestrians to go cross the road, we added an event model that checks if more than one car is coming and if the first car stops at the red light. With this new event, we have a success rate of 100 % in any case.

The next results concern the complication (Sect. 4.5) of the architecture and the next experimentation.

We focused our analysis on ignored events compared to well-composed events in order to check generated meaning. After validation of event models, we check the good correlation between expected output and inputs. We also check the robustness of the inference engine with noisy events and by increasing the number of events in time (maximum load). Results are presented in Fig. 29 and Table 7. Consistency decreases when too many events occur. Performance of agents decreases due to their processing speed but robustness is always good because, in this human situation, events are very redundant so there is no impact on outputs.

Table 7 Consistency in memory

Events	Consistency	Robustness
Normal <1000 events/s	100 %	100 %
Overload >1000 events/s	71 % 29 % ignored	100 %
Noisy 50 % unknown events	50 % 50 % events ignored	100 %
Noisy 50 % false data in known events	42 % 17 % events not in time so ignored	96 %

Agents are able to ignore events when they are not matching predicate name, time, location and other roles in event. Robustness remains good in cases of noisy data without taking into account uncertainty measures coming from sensors and hence corrupted data. Consistency is weak because false data in events impact correct events at 8 %.

5 Conclusion and future work

We have presented our architecture with multi levels of abstraction based on semantic agents suitable for interaction in the human environments like home and city. Our architecture brings a software part of the solution and manages multiples input and output modalities for interaction. A fast evaluation of possible scenarios is done by understanding and taking into account all past events and behaviours. It may be easily adapted to several different tasks and contexts by adding appropriate concepts and models in the memory. We have proved it works well for human assistance in virtual reality simulation (in a dedicated room or at home) but, in real world, we will certainly need more reliable services: better video camera managing sunlight, real-time communicative devices embedded in cars and in red lights. Future work will be the auto-reconfiguration of the architecture depending on other situational contexts and user preferences profiles.

References

1. Goodrich MA, Schultz AC Human-robot interaction: a survey. Found Trends Hum-Comput Interact
2. Landragin F (2007) Physical, semantic and pragmatics levels for multimodal fusion and fission. In: Seventh international workshop on computational semantics, vol 7, Tilburg, The Netherlands, pp 346–350
3. http://www.w3.org/TR/soap12-part0
4. Hiroshi I, Toshiyuki K, Katumi K, Toru I (1999) A robot architecture based on situated modules. IROS
5. Ioannidou A, Repenning A, Webb DC, Keyser D, Luhn L, Daetwyler C (2010) Mr. Vetro: a collective simulation for teaching health science. I. Comput-Support Collab Learn 5(2):141–166
6. Mohammad Y, Nishida T (2010) Controlling gaze with an embodied interactive control architecture. Appl Intell 32(2):148–163
7. Mohammad Y, Nishida T (2010) Using physiological signals to detect natural interactive behaviour. Appl Intell 33(1):79–92
8. Anderson JR, Lebiere C (eds) (1998) The atomic components of thought. Lawrence Erlbaum, Mahwah
9. Laird J, Newell A, Rosenbloom PS (1987) SOAR: an architecture for general intelligence. Artif Intell J 33(1):1–64. http://www.soartechnology.com
10. Rao AS, Georgeff MP (1991) Modelling rational agents within a BDI-architecture. In: Allen J, Fikes R, Sandells E (eds) Proceedings of the second international conference on principles of knowledge representation and reasoning. Morgan Kaufmann, San Mateo
11. Landragin F, Denis A, Ricci A, Romary L (2004) Multimodal meaning representation for generic dialogue systems architectures. In: Proc on language resources and evaluation, pp 521–524
12. Gupta A, Kembhavi A, Davis LS (2009) Observing human-object interactions: using spatial and functional compatibility for recognition. IEEE Trans Pattern Anal Mach Intell 31(10):1775–1789
13. Wang J, Byrnes J, Valtorta M, Huhns M (2012) On the combination of logical and probabilistic models for information analysis. Appl Intell 36(2):472–497
14. Sarkar J, Vinh LT, Lee Y-K, Lee S (2011) GPARS: a general-purpose activity recognition system. Appl Intell 35(2):242–259
15. Coradeschi S, Loutfi A (2008) A review of past and future trends in perceptual anchoring. In: Fritze P (ed) Tools in artificial intelligence. I-Tech Education and Publishing, Vienna
16. Gruber TR (1993) A translation approach to portable ontology specification. Knowl Acquis 5:199–220
17. Baumeister J, Reutelshoefer J, Puppe F (2011) KnowWE: a semantic wiki for knowledge engineering. Appl Intell 35(3):323–344
18. Guarino N (1995) Formal ontology, conceptual analysis and knowledge representation. Int J Hum-Comput Stud 43(5/6):625–640
19. Giuliani M, Knoll A (2008) MultiML—a general purpose representation language for multimodal human utterances. In: ICMI'08. Chania, Crete, Greece
20. Johnston M (2009) Building multimodal applications with EMMA. In: ICMI-MLMI'09, Cambridge, MA, USA, 2–4 November 2009. ACM, New York. ISBN 978-1-60558-772-1
21. Johnston M, Baggia P, Burnett DC, Carter J, Dahl DA, MacCobb G, Ragget D (2009) EMMA: extensible MultiModal annotation markup language. W3C Recommendation, 10 February 2009
22. Kranstedt A, Kopp S, Wachsmuth I (2002) Murml: A multimodal utterance representation markup language for conversational agents. In: Proc. of the AAMAS. Workshop on "Embodied conversational agents—let's specify and evaluate them"
23. Gibbon D, Gut U, Hell B, Looks K, Trippel ATT (2003) A computational model of arm gestures in conversation. In: EUROSPEECH-2003, pp 813–816
24. Chai J (2002) Semantics-based representation for multimodal interpretation in conversational systems. In: COLING 2002: the 19th international conference on computational linguistics
25. Sengers P (1998) Narrative intelligence. In: Dautenhahn K (ed) Human cognition and social agent technology, Advances in consciousness, vol. 19. John Benjamins, Amsterdam
26. Singh P (2005) EM-ONE: an architecture for reflective commonsense thinking. PhD Thesis, MIT
27. Obrst L (2003) Ontologies for semantically interoperable systems. In: Proceedings of the twelfth international conference on information and knowledge management, New Orleans, LA, USA. ACM Press, New York, pp 366–369. ISBN: 1-58113-723-0
28. Macal CM, North MJ (2006) Tutorial on agent-based modeling and simulation part 2: how to model with agents. In: Perrone LF,

Wieland FP, Liu J, Lawson BG, Nicol DM, Fujimoto RM (eds) Proceedings of the 2006 Winter simulation conference
29. Allan R (2009) Survey of agent based modeling and simulation tools. Technical report. http://epubs.cclrc.ac.uk/work-details?w=50398
30. Dourlens S, Ramdane-Cherif A (2010) Semantic memory for pervasive architecture. In: ICICA. LNCS, pp 94–102
31. Dourlens S, Ramdane-Cherif A (2010) Cognitive memory for semantic agents in robotic interaction. In: IEEE ICCI, pp 511–517
32. Dourlens S, Ramdane-Cherif A (2011) Semantic modeling & understanding of environment behaviours. In: SSCI 2011—IEEE symposium series on computational intelligence, symposium on intelligent agents, Paris, France, 11–15 April
33. Jensen K (1991) Coloured Petri nets: a high level language for system design and analysis. In: APN 90: proceedings on advances in Petri nets 1990. Springer, New York, pp 342–416
34. Vinh LT, Lee S, Le HX, Ngo HQ, Kim HI, Hanet M Lee Y-K (2011) Semi-Markov conditional random fields for accelerometer-based activity recognition. Appl Intell 35(2):226–241

Sébastien Dourlens holds a PhD thesis in Computer Engineering, Control and Signal Processing from the LISV Laboratory, Université de Versailles-Saint Quentin, France. He worked 15 years as Technical Manager where he leads teams which develop software, hardware, knowledge management and networking systems. His research is on Computer Science, Artificial Intelligence, Knowledge Representation & Reasoning and Engineering solutions for Human Assistance.

Amar Ramdane-Cherif is Professor at University of Versailles Saint Quentin en Yvelines (UVSQ). He has obtained his PhD in 1998 from Pierre and Marie Curie University (UPMC), Paris, France. He was an associate professor at the UVSQ (laboratory PRISM) from 2000–2009. He has obtained his HDR from UVSQ in 2007. Since 2009, he is a full professor at UVSQ (LISV laboratory). Its research activities focused on software architecture, multimodal interaction systems, semantic knowledge representation, ambient intelligence, Multi-Agent Systems, artificial intelligence (reasoning and learning). He has published around 25 research papers in international Journals and more than 100 international conference papers.

Eric Monacelli is chair of the Assistance and Interaction team in the LISV laboratory in Versailles University, since 2008. Objectives of this research team are the development of analysis methods and experimental devices adapted to the assistance of specific end users. Supervisor of 14 PhD students (7 of them are graduated) since 2000, Eric Monacelli is chair for several national projects. Those research projects incorporating issues of man-machine interface, robotics and ambient intelligence systems. Today's topic is mainly ambient assistance. Since 2008, he is president of the French Cluster for Mobility and Handicap: CEREMH. The CEREMH is a resource centre and innovation-oriented national whose mission is to promote mobility for all ages. The CEREMH has set five goals: providing a service to people, mobile aid innovation (from wheelchair to car), building accessibility evaluation, disseminating innovative structure and building a network.