Representing Context in an Agent Architecture for Context-Based Decision Making

Oana Bucur, Philippe Beaune, Olivier Boissier

Centre G2I/SMA, ENS des Mines de Saint-Etienne,
158 Cours Fauriel, Saint-Etienne Cedex 2, F-42023, France
{bucur, beaune, boissier}@emse.fr

Abstract. Ambient intelligence involves the convergence of several computing areas: ubiquitous computing, intelligent systems and context-awareness. Developing context-aware applications needs facilities for recognizing and representing context, reasoning on it and adapting to it accordingly. In what concerns context representation, the newest and most challenging representation is the ontological one. The problem is that current ontologies for context do not provide a standard for representing complex context attributes. In this paper, we propose a context definition and representation used to construct context-based agent architecture. The representation we propose combines the generality provided by ontologies with the complexity inspired by the object oriented models. The goal of the proposed architecture is to support the deployment of context-aware agents able to learn how to recognize the context of their decisions and to adapt to it. The use of this architecture is illustrated on a test MAS for agenda management, using the JADE-LEAP platform on PCs and PDAs.

1. Introduction

The rise of ambient intelligence has stressed the importance of *context* since building applications in this area requires the convergence of several computing areas such as: ubiquitous computing, intelligent systems and context-awareness. As defined in [5], context consists in "any information that can be used to characterize the situation of an entity" where an entity is "is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". Besides this intuitive definition, developing context-aware applications needs facilities for recognizing and representing context, reasoning on it and adapting to it accordingly. On the way to fulfill this purpose, the project including the work described in this paper, aims at building Multi-Agent Systems (MAS) composed of several "societies" of heterogeneous and situated agents sharing or not the same context that changes over time where agents are able to sense, model and reason using contextual information, but also to share their knowledge about this information in order to learn how to discern relevant from non-relevant context and how to make appropriate decisions based on it.

The focus of this paper deals with the crucial points of representation of contextual information on one hand and on the agent's architecture on the other hand. In what concerns context representation, the newest and most challenging representation is the ontological one (as in [4] or [20]). The problem is that current ontologies for context are quite simplistic and do not provide a standard for representing and reasoning on complex context elements (usually, they are represented as simple properties connected to a concept in the ontology). We propose an ontology-based representation for contextual information that combines the generality provided by ontologies with the complexity inspired by object-oriented models. Given the representation, there is few related work that focus on how to recognize the context of an agent's decision (in [6], when trying to learn "the models and the circumstances in which they work best"; in [22], when distinguishing between primary, contextual and irrelevant features in a classification or in [24], when doing the same thing about contextual clues). Usually, relevant context is assumed known in advance, and agents just learn how to adapt their decision to it. This is why, we propose an agent's architecture where we make an explicit separation between selection of relevant context from the decision making process of the agent.

Representation and architecture are validated on a case study of an open and interoperable context-aware Multi-Agent System (MAS) for agenda management, deployed on PCs and PDAs. The system is composed of several meeting-scheduling agents called mySAM (my Smart Agenda Manager), which can be executed both on PCs and PDAs. A mySAM agent assists its user in fixing meetings by negotiating them with other mySAM agents and by using context knowledge to decide whether to accept or to reject a meeting proposal made by another agent. Knowledge about how to choose the relevant context and how to use it to deal with a meeting proposal is acquired through individual agent learning and multi-agent knowledge sharing methods. We will not present the learning and knowledge sharing methods, as the goal of this article is to focus on the representation and management of context in MAS, with an architecture focused on context.

Before describing the proposed architecture (section 3), we will present the ontology-based representation for context (section 2). We then illustrate our work in the agenda management application (section 4). Before concluding, we will situate our approach in related work.

2. Context for MAS – definition and representation

Our approach is illustrated in Fig. 1. We propose a layered system's architecture in which the context management is not done by the agents themselves. Instead, they interact with a context-manager layer that is able to answer context-related queries. We thus separate all that is related to context acquisition and management (which will be done by the centralized context manager) from reasoning with context knowledge (done by the agents). The system architecture is structured along three layers (cf. Fig. 1): context sources (layer 0), context management (layer 1), a layer of agents that reason with context (layer 2). Over the last layer are dedicated applications, which use

agents' abilities to reason with context, or in which context can be used directly by the application itself.

The second layer is composed of several Context Managers (CM), one for each agent society. The CM is able to deal with the context information available in the area in which the corresponding group of agents is situated (for instance, we can imagine having one CM for each flour in a building). Agents, which are operating at the third layer, use the CM to retrieve context knowledge that will help them to decide and to solve problems. In order to assure a common understanding, the CM uses an ontology-based representation for contextual information. To improve their capabilities by using others' experiences, agents can share with other agents the way they use the context knowledge in solving similar problems. In this way, an agent can be sure that, no matter the society where it is located, it can use a specific way to ask the CM and other agents about context.

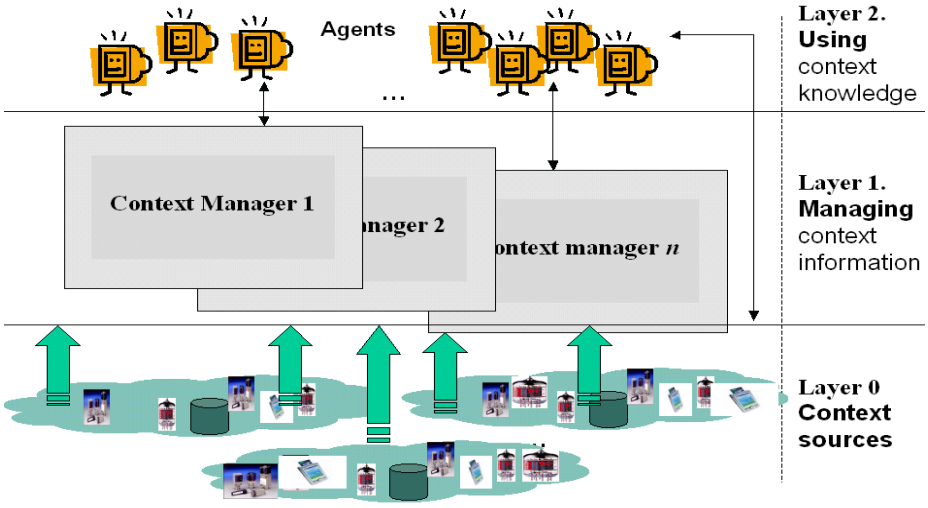

Fig. 1. Global architecture for Context-aware applications

An agent that moves around will be in contact with at least one CM at a time and will use the context that is available at that moment in the specific area. Our aim is to alleviate the task of the system's designer by not forcing him to create a huge ontology to define all imaginable context attributes. Each society can have a 'local' context defined and ready to be managed by the local CM.

Having drawn the global picture of our system, let's go now into its details: first by describing what the notion of "context" means when dealing with it in MAS, then by explaining how context is represented and used to design and implement our proposed context-based MAS.

2.1. Definition

Agents are situated entities, meaning that they sense their environment and act accordingly. Given the definition in the introduction, we can call the environment-related information as being the *environmental context* of an agent. Except their environment, agents are surrounded by and interact with other agents. Hence there is an *interaction* and an *organizational context* that should also be taken into account. Moreover, some agents are directly connected to user's needs and therefore should consider *user related context* when making decisions (for instance, the "role" of the user in a group).

From Dey's definition given in introduction, context may be further described as a set of attributes and a finality. The *finality*, f, is the goal for which the context is used at a given moment, the focus of the activity at hand. We can see the finality as being the information that is the most interesting for the agent at a given moment, for example: deciding what to do with a proposal, explaining an action, understanding a conversation, etc., are all finalities that determine the way the agent will act.

We saw that context is defined by the finality and the set of attributes that are relevant for that finality. In what follows we will define the structure that a context attribute has to fulfill. Usually, in the literature, a context element is either seen as a simple "label" that will have a value associated (like in [22], [1], [17]or [6]), or as a more complex structure, that also includes the specification of what is to be done when a certain value is reached (like in [7]). What we propose in this section is to construct a specific structure within which we will be able to define a context attribute. This structure's goal is to specify the pattern (the structure) of a context attribute, and not to mention which behavior should be adopted for a specific value of the context attribute. We extended the classical 'attribute-value' approach to cope with the necessity of describing any context element.

A *context attribute (a)* designates the information defining one element of context, e.g. "ActivityLocation", "NamePerson", and «ActivityDuration". Each context attribute has at least one value at a given moment, the value depending on several entities to which the attribute relates. An entity is an instance of a "person, object or place" (as mentioned in [5]), but can also be an activity, an organizational concept (role, group), etc. For instance, the context attribute "DevicesAvailableInRoom" defines the devices that are located in a room. When trying to instantiate this attribute, the needed parameter will be the specific room whose devices we are interested in. A "PersonIsMemberOf" context attribute will take a person as input, but will return (possibly) several groups in which that person takes part. Let's note V_a the definition domain of *a*, the set of possible values that *a* may take (example: V_{time} =[0,24[ ).

We can therefore associate to each context attribute an instantiation function called *valueOf*. We define *valueOf* for a context attribute as a function from A x P_a to $P(V_a)$, where A is the set of all attributes, $P(V_a)$ is the power set of V_a, and P_a is the set of parameters needed to compute the value of *a*. It is obvious that some context attributes (as "PersonIsMemberOf", "Supervises", "DevicesAvailableInRoom", etc.) can have as a value actually a set of values. This is why we allow the function *valueOf* to take values in $P(V_a)$, and not simply V_a.

Not all attributes are relevant for a finality. We define ***is_relevant(a,f),*** a predicate stating that attribute ***a*** is relevant for the finality ***f***. Let's call ***RAS***(*f*) the subset of ***A*** which defines the *Relevant Attribute Set* for the finality ***f***:

$$\boldsymbol{RAS}(f) = \{\ a \in \boldsymbol{A} \mid \boldsymbol{is\_relevant}(a,f)=\text{true}\ \}.$$

We will call an *instantiation of context attribute* $a \in \boldsymbol{A}$ the pair (a, v) where v is the set of values $v \in \boldsymbol{P}(\boldsymbol{V_a})$ of a at a given moment. For instance, (Day, {14}), (roleOfPersonInGroup, {Team Manager}), (PersonIsMemberOf, {MAS Group, Center_X, University_Y}) are instantiation of respective context attributes *Day*, *roleOfPersonInGroup*, *PersonIsMemberOf*. Let ***I*** be the set of instantiated context attributes as

$$\boldsymbol{I} = \{(a,v) \mid a \in \boldsymbol{A} \wedge \boldsymbol{valueOf}(a)=v\}.$$

We call *Instantiated Relevant Attribute Set* of a finality ***f***, ***IRAS*(f)**, the set of instantiated context attributes relevant to finality ***f***:

$$\boldsymbol{IRAS}(\text{f}) = \{(a,v) \mid a \in \boldsymbol{RAS}(\text{f}) \wedge (a,v) \in \boldsymbol{I}\}.$$

Let's notice that in related work ([12], [18], [19]), the notion of "context" is often understood as being what we defined as the IRAS. To explain the difference between RAS and IRAS and the reason to make this distinction, let's consider the following example. Given the finality f = "deciding whether to accept or not a meeting", RAS(f)={"RoleOfPersonInGroup", "ActivityScheduledInSlot"} is considered, i.e. role played by the person who made the proposal and if the receiver has something already planned for the proposed time slot. The resulting IRAS for a student may be $\text{IRAS}_{\text{student}}$(f)={(RoleOfPersonInGroup, {teacher}), (ActivityScheduledInSlot, {Activity001})} and for a teacher $\text{IRAS}_{\text{teacher}}$(f)={ (RoleOfPersonInGroup, {student}), (ActivityScheduledInSlot, {Activity255)}. As we can see, the difference between IRAS of student and teacher may lead to different rational decisions. Usually RAS used is almost the same for different users when needed to make decisions for the same finality, but the decision itself is IRAS-dependent. Taking into account the definitions that we proposed so far we now describe the representation that we defined.

2.2. Representing context

Our aim is to represent context in a general and suitable manner for all applications that need to represent and reason about it. What we focus on is to have a generic and complex representation for a context attribute, representation that will assure in this way the interoperability, but also the possibility to represent no matter what context attribute, not just the simple ones.

Several representations of context exist: contextual graphs ([1]), XML (used to define ConteXtML [17]), or object oriented models ([7]). All these representations have strengths and weaknesses. As stated in [8], lack of generality is the most frequent weakness: each representation is suited for a type of application and expresses a particular vision on context. There is also a lack of formal bases necessary to capture context in a consistent manner and to support reasoning on its different

properties. A tentative answer in [8] was the entity-association-attribute model, which is an extension of the "attribute-value" representation, where contextual information are structured around an entity, every entity representing a physical or conceptual object. We base our proposal on this idea and on ontologies as the explicit way to represent it. To take into account the need for generality, and also considering the fact that we aim at having several MAS, each of them dealing with different contexts (that we will need to correlate in some way), an ontology-based representation seems reasonable. This is not a novel idea; Chen *et al*. ([4]) defined context ontologies using OWL.

Usually, when context is represented using ontologies, what is said to be "context" are the properties associated to each concept (or entity, how we will call it further on). For example, when defining a meeting ontology, we can define the status of a meeting as a property of the entity Meeting. This property can also be considered a context attribute. This representation is simple and does not allow more complex context attributes to be represented in the same manner. For example, if we want to define an attribute that will have the value "true" if two meetings are taking place in the same time, we need to define this attribute not as a property (as it will be connected to two meeting entities – in order to check when they take place), but as an entity in itself, with three properties: meeting_1, meeting_2, value (true/false).

Considering this, what we propose is to define a concept called "context attribute", which will regroup all needed information for defining and instantiating a context attribute (corresponding to our definition of a context attribute as defined in section 2.1.). Instead of treating each attribute differently (based on its complexity), we define a class (called #ContextAttribute) that will always contain the same kind of information: the name of the attribute, the type of needed parameters (entities) for the instantiation, the V_a (values domain), if the attribute is allowed to have several values (or just one). Starting from this class, each context attribute will be characterized by these properties, with different restrictions: "RoleOfPersonInGroup" will need a #Person and a #Group as parameters and will give a #Role when instantiated, while "NamePerson" will need #Person as a parameter and will have a String as value, and so on. In our domain ontology, the class "#Entity" is the super class of all concepts, e.g. in MySAM, #Person, #Group, #Room, #Activity, etc. are subclasses of #Entity.

When we talk about an entity (and not about the class #Entity) we refer to an instance of a concept (class) defined in the domain ontology. For example, the class #Entity (defined in the context ontology) is the super class of all concepts that characterize our domain (in MySAM, such concepts are: #Person, #Activity, #Agenda, #Group, #Role, etc.). When we talk about an entity, that will be an instance of the class #Person, #Group or #Activity.

Table 1. Description of #ContextAttribute class

Property Name	Property Type	Domain	Range	Multiple values
name	Datatype	#ContextAttribute	String	No
noEntities	Datatype	#ContextAttribute	Integer	No
entitiesList	Object	#ContextAttribute	#Entity	Yes
valueType	Object	#ContextAttribute	#Entity	No
multipleValue	DataType	#ContextAttribute	Boolean	No

This way, our ontology is made of two components: one that defines the domain ontology (similar to all other "context" ontologies) and another one, which is the description of all context attributes that will be managed in the current system (by the available Context Manager).

In Table 1 is given the description of the #ContextAttribute class defined in MySAM ontology. As it can be seen, the description is a general description of the characteristics of all context attributes, without specifying anything about a specific attribute. Each attribute is afterwards represented as a sub class of #ContextAttribute, with restrictions on each property, in order to have a clear description of the specified attribute. For instance, the context attribute `RoleOfPersonInGroup` is described with the following restrictions on the properties of the defined class #ContextAttribute: Name = "RoleOfPersonInGroup"; NoEntities = 2 (we need to connect this attribute to a person and a group); valueType = #Role (value for this attribute is an instance of the class #Role); multipleValues = "false" (a person can only play one role in a group); entitiesList = { #Person; #Group} (connected entities are instances of class #Person and of class #Group).

Table 2. Some of the context attributes defined in MySAM ontology

Person – related	**Time-related**
IsSupervisorOf :(Person, Person)-> Boolean StatusPerson :(Person) -> String Supervises : (Person) -> Person* RoleOfPersonInGroup :(Person,Group)-> Role	TimeZone : (Time) -> Integer DayOfWeek : (Date) -> String TimeOfDay : (Time) -> String
Location - related	**Activity – related**
PersonIsInRoom : (Person, Room) -> Boolean PersonIsAtFloor : (Person, Floor) -> Boolean PersonIsInBuilding:(Person,Building)-> Boolean PersonIsInCity : (Person, City) -> Boolean	ActivityStartsAt:(Activity)->Time ActivityEndsAt :(Activity)-> Time AcivityGoal : (Activity) ->String ActivityDuration :(Activity) -> Integer ActivityParticipants: (Activity) -> Person*
Agenda - related	**Environment – related**
BusyMorning : (Agenda) -> Boolean BusyAfternoon : (Agenda) -> Boolean BusyEvening : (Agenda) -> Boolean	DevicesAvailableInRoom:(Room)->Device* DevicesAvailableAtFloor : (Floor) -> Device*

In Table 2. we give a small part of the list of context attributes defined in the OWL ontology used for MySAM application. The attributes are grouped based on their specificity: attributes that are connected to a person, attributes related to time management, etc. Each attribute is described by its name and then the list of parameters needed for its instantiation (given by their type) and the type of value it will have when instantiated. A star (*) near to the type of the value means that the attribute is allowed to have multiple values for an instantiation (for example, one person can belong to several groups at a time).

3. Architecture for a context-based learning MAS

The proposed layered architecture is composed of mySAM agents (Fig. 3), which assist a user, and several CM (Fig. 2)(each of them will manage a domain ontology and a set of context attributes). Being connected to the current state of the

environment, a CM provides agents with context related information. The CM (and not agents) has the responsibility to compute the values of context attributes in the environment. Agents learn how to recognize relevant context and how to act accordingly. We underline the fact that this architecture is suitable for developing context-aware agents that are light-weighted, therefore adapted to hand held devices. We continue by describing the CM and context-based agents that learn how to choose and how to use context.

3.1. Context manager (CM)

The main functionalities of CM are to let the agents know which is the context attributes set (defined in the ontology) that it manages and to compute IRAS corresponding to RAS given by the agents at some point of processing. When entering a society, an agent asks the corresponding CM to provide it with the context attributes that it manages. Acting as intermediary between agents and the environment, CM is able to answer requests regarding its managed context attributes.

The Context Knowledge Base contains the ontology of the domain, defined as a hierarchy with #Entity as root, on one hand, and all context attributes (defined as sub classes of the class #ContextAttribute) that will be managed by the CM, on the other hand. The *instantiation* module computes the IRAS(f) for a given RAS(f). The *dependencies* module computes the values for derived attributes by considering possible relations between context attributes concerning their relevance: if one attribute is relevant for a situation and it has a certain value, then another attribute could also be relevant for that situation.

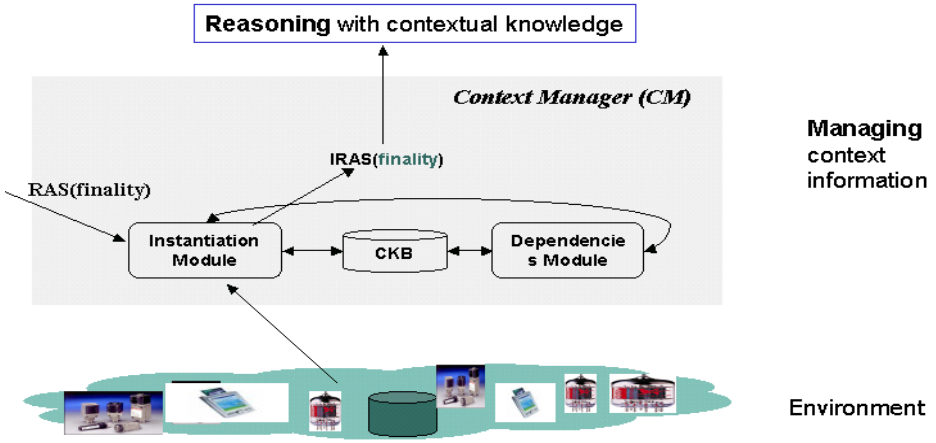

Fig. 2. Context manager architecture

3.2. Context-based learning agent

The context-based agent architecture that is the core of a MySAM agent is general and it is not restrained to the kind of application considered to illustrate our approach.

It has two main modules (Fig. 3): *selection* of relevant attributes for a certain finality f (RAS(f)) and *decision* based on instantiated attributes (IRAS(f)) provided by CM. For example, for a finality relative to deciding whether accepting or not a "2 participants" meeting, the *selection* module is supposed to build the RAS={"ActivityScheduled InSlot", "roleOfPersonInGroup"}; or, for a finality relative to a "seminar", the RAS can be {"ActivityParticipants", "Activity Description", "PersonInterests", etc}.

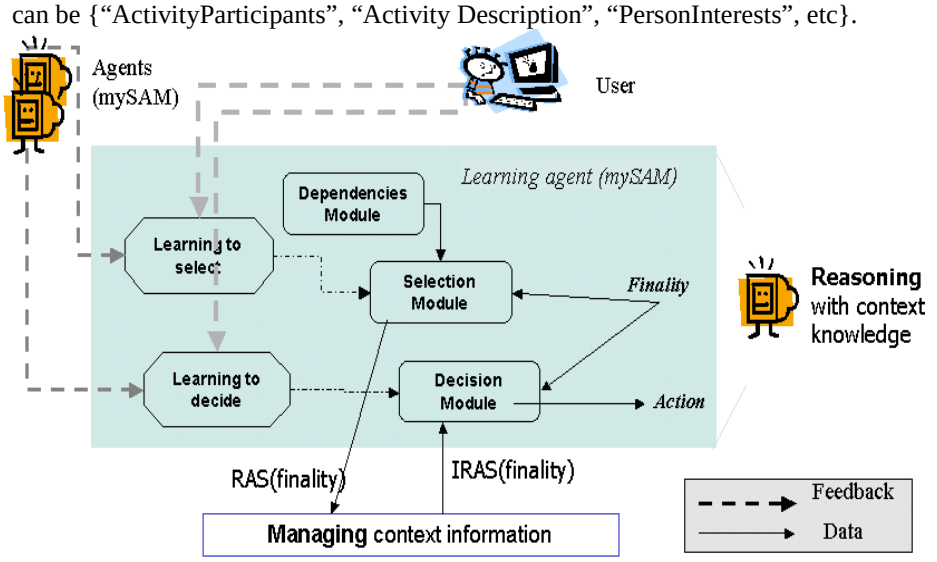

Fig. 3. Context-based agent architecture

The *decision* module knows how to accept or reject a meeting based on the IRAS resulted from the respective RAS instantiation. For example, if we have nothing planned for that period of time and if the person that demands this meeting is our chief, the decision module will decide to accept the meeting. Both the selection and the decision modules have associated learning modules (individual as well as multi-agent learning) that improve their abilities. Using individual learning and knowledge sharing techniques, agents are able to make a better selection of RAS sets for a given finality and to take better adapted decisions based on obtained IRAS (for a description of the learning methods see [2] and [3]).

4. Implementation

In order to validate our proposal, we developed the system proposed as a case study in section 2, a multi-agent system containing several mySAM agents and one CM Service. Agents were deployed with the JADE/LEAP platform ([9]) to run on handheld devices. Each mySAM agent is a JADE agent with a graphical interface that allows a user to manage her agenda. This graphical interface has been simplified to deploy mySAM agents on a HP iPAQ 5550 Pocket PC. Due to limited resources available on PDAs, mySAM agents deployed on them execute only the negotiation

task, without any learning methods. Learning methods are deployed on a proxy agent. Agents running on PDA and on proxy constitute a MySAM agent.

The CM Service is also implemented as a JADE agent. It is a special agent in the system that has access to the domain ontology that defines the context attributes that it will manage. It provides context knowledge to all agents that arrive in the system. The ontology was created using Protégé 2000 v.2.1.([16]) and the agent accesses the ontology using Jena ([10]), a Java library designed for ontology management. So far, the instantiation module is hard coded in CM, but as future work we intend to find a possibility to formally define the instantiation procedures, so that the CM implementation will not depend on the defined ontology.

When testing mySAM we were able to validate the fact the using a selection step to choose the RAS for a finality helps in having smaller and more significant rules. Using all attributes to describe a situation is not only difficult to deal with, but also unnecessary. Also, the fact that sharing with other agents the RAS for a finality and/or the solution for a situation described by an IRAS improved agent's abilities to make decisions, enforces our theory that the separation between RAS and IRAS and the use of sharing methods are important and necessary.

5. Related Work

In MAS, the notion of context is used to describe the factors that influence a certain decision. In similar applications (meeting schedulers), context means: type of event, number of attendees, etc. (Calendar Apprentice [13]), activity, participants, location, required resources (Personal Calendar Agent [12]), system load, organization size (Distributed Meeting Scheduler [19]), time, user's location, etc. (Electric elves [18]). None of these works mention the idea of "context" but they all use the context as "circumstances" or "environmental factors" that affect the decision to be made. In making Calendar Agent ([11]), Lashkari et al. use the notion of context, but they assume that the relevant context is known in advance, so that every contextual element that they have access to is considered relevant for the decision to be made.

We can mention here some definitions given by: Persson [15] ("context is […] the surrounding of a device and the history of its parameters"), Brezillon [1] ("the objective characteristics describing a situation, […] the mental image generated, […] the risk attitude" used to make decisions) or Turner [23] ("Any identifiable configuration of environmental, mission-related and agent-related features" used to predict a behavior). There are several other definitions in related work ([14], [21], etc.) and what we proposed in section 2.1. is a definition quite similar to all these in the sense that it is based on: (i) the elements that compose the context and (ii) its use, i.e. the finality (the goal we want to achieve) when using this context. The definition we proposed takes into account those two dimensions of context (its use and its elements); it also explains what each dimension is and how to properly define it when designing a context-based MAS.

This is the major problem with the way context is used: these approaches are not fitted for an application independent way of handling context, because they do not

provide a general representation of context knowledge and methods to choose between relevant and non-relevant context elements for a specific decision.

The main difference and contribution of our work is in the sense that we propose a MAS architecture based on an ontological representation of context and that can permit individual and multi-agent learning on how to choose and how to use context. It is not the choice of an application that generated this architecture, but MySAM is just a case study to validate our approach.

Mostly, when context is used to make behaviors context-adaptable, it is used in an ad-hoc manner, without trying to propose an approach suitable for other kind of applications. However, there is some research in proposing a general architecture on context-aware applications, like CoBrA (Context Broker Architecture), proposed by Chen et al.[4] or Socam, by Gu et al [20]. We based our proposed architecture on CoBrA and Socam, but we added the learning modules for finding relevant context and using it to decide. The context broker and context interpreter are very similar to our context manager, with the difference that our concern was not how to acquire certain information from heterogeneous sources, but mostly how to represent it in a semantic and generic manner and how to reason on context knowledge based on this representation.

6. Conclusions and perspectives

In this article, we have presented a definition of the notion of context, notion that is used in almost all systems without precisely and explicitly taking it into account. We have proposed an ontology-based representation for context elements and a context-based architecture for a learning multi-agent system that uses this representation. We then validated our approach by implementing a meeting scheduling MAS that uses this architecture and manages and learns context based on the definitions and representation we proposed.

Our definition and representation are not based on a specific application. The difference between our approach and the classical ontology approach is that we extended the representation of the attribute. In our ontology a context attribute is a class, not a property (in this way all attributes are represented in the same manner - as concepts).

The tests that have been realized confirm our hypotheses: a) that the distinction between RAS/IRAS is important in efficient decision making on one hand, and b) that limited sharing of knowledge is sufficient to learn to make better decisions.

As future work, we envisage adding to the framework for context-based MAS to be used for multiple application domains that consider context when adapting their behavior. Our future work will focus on representing and managing: (i) instantiation procedures for context attributes, (ii) dependencies that can exist between them, (iii) the importance of different attributes in a situation (make a more refined difference between relevant and non-relevant attributes). In what concerns learning agents, the framework will provide agents with several individual learning algorithm and all that is needed to communicate and share contextual knowledge.

References

[1] Brezillon, P. – "Context Dynamic and Explanation in Contextual Graphs", In: Modeling and Using Context (CONTEXT-03), LNAI 2680, Springer Verlag p. 94-106, 2003.

[2] Bucur O, Boissier O, Beaune P – "A Context-Based Agent Architecture for Learning How to Make Contextualized Decisions" (to appear in Proc. of First International Workshop on Managing Context Information in Mobile and Pervasive Environments 2005), may 2005.

[3] Bucur O – "Modeling context for a multi-agent learning of user preferences" – Master dissertation – Ecole des Mines de Saint-Etienne, July 2004.

[4] Chen H., Finin T., Anupam J. – "An Ontology for Context-Aware Pervasive Computing Environments", The Knowledge Engineering Review, p. 197–207, 2003.

[5] Dey A., Abowd, G.– "Towards a better understanding of Context and Context-Awareness", GVU Technical Report GIT-GVU-00-18, Georgia Institute of Technology, 1999.

[6] Edmonds B. – "Learning and exploiting context in agents", in proc. of AAMAS 2002, Bologna, Italy, p. 1231-1238.

[7] Gonzalez A., Ahlers R. – "Context based representation of intelligent behavior in training simulations", Transactions of the Society for Computer Simulation International, Vol. 15, No. 4, p. 153-166, 1999.

[8] Henricksen K., Indulska J., Rakotonirainy A – "Modeling Context Information in Pervasive Computing Systems", Proc. First International Conference on Pervasive Computing 2002, p. 167-180.

[9] JADE (Java Agent Development framework) : http://jade.cselt.it/

[10] Jena Semantic Web Framework - http://jena. sourceforge.net/

[11] Lashkari Y., Metral M., Maes P – "Collaborative Interface Agents", Proc. of CIKM'94, ACM Press.

[12] Lin S., J.Y.Hsu – "Learning User's Scheduling Criteria in a Personal Calendar Agent", Proc. of TAAI2000, Taipei.

[13] Mitchell T., Caruana R., Freitag D., McDermott J., Zabowski D.– "Experience with a learning personal assistant", Communications of the ACM, 1994.

[14] Pascoe J. – "Adding Generic Contextual Capabilities to Werable Computers". 2nd International Symposium on Wearable Computers (1998) 92-99.

[15] Persson P.– "Social Ubiquitous computing", Position paper to the workshop on 'Building the Ubiquitous Computing User Experience' at ACM/SIGCHI'01, Seattle.

[16] Protégé 2000 - http://protege.stanford.edu/.

[17] Ryan N.– "ConteXtML: Exchanging contextual information between a Mobile Client and the FieldNote Server", http://www.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html.

[18] Scerri, P., Pynadath D., Tambe M.– "Why the elf acted autonomously: Towards a theory of adjustable autonomy " , AAMAS 02, p. 857-964, 2002.

[19] Sen S., E.H. Durfee – "On the design of an adaptive meeting scheduler", in Proc. of the Tenth IEEE Conference on AI Applications, p. 40-46, 1994.

[20] Tao Gu, Xiao Hang W., Hung K.P., Da Quing Z – "An Ontology-based Context Model in Intelligent Environments", Proc. of Communication Networks and Distributed Systems Modeling and Simulation Conference, 2004.

[21] Thevenin D., J. Coutaz. – "Plasticity of User Interfaces: Framework and Research Agenda". In Proceedings of INTERACT'99, 1999, pp. 110-117.

[22] Turney,P. – "The identification of Context-Sensitive Features: A Formal Definition of context for Concept Learning", 13th International Conference on Machine Learning (ICML96), Workshop on Learning in Context-Sensitive Domains, p. 53-59.

[23] Turner, R. – "Context-Mediated Behaviour for Intelligent Agents", International Journal of Human-Computer Studies, vol. 48 no.3, March 1998, p. 307-330.

[24] Widmer G.– "Tracking context changes through meta-learning", Machine Learning, 27(3):259-286, Kluwer Academic Publisher.