# Knowledge Sharing on How to Recognize and Use Context to Make Decisions

Oana Bucur, Philippe Beaune, Olivier Boissier

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

**Abstract.** Multi-Agent Technologies propose a new paradigm of computation based on multiple autonomous agents situated in an environment, exhibiting reactive, pro-active and social behaviour. Context is an important dimension to take into account when building complex and open multi-level, multi-purpose systems. The increasing openness of applications strengthens the need for a common base grounded on an explicit specification of context in order to allow software units that enter a new system to adapt to their new context. In this paper we propose a context-management architecture to be used in open systems based on a multi-agent architecture. We show how agents are able to recognize the context of their decisions and to learn how to make decisions depending on it. Toward this aim, we propose en extended definition of the notion of context and use an ontology-based context representation. We validate our approach in a multi-agent system for the management of an agenda.

## 1 Introduction

Multi-agent Technologies aim at building multi-level, multi-unit and multi-purpose open systems, by transforming each software unit into an autonomous agent able to exhibit reactive, pro-active and social behaviour in the environment in which it is situated. Towards this aim, context gains an important place while being enlarged: agents have to take into account their physical but also their social environment to propose adapted services to users with which they interact. The focus on openness highlights the need on adaptation to new contexts since agents may leave/enter/move from one system to another.

Almost all applications in computer science are faced with context in the sense that, as defined in [5], it 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”. But they handle it in a more or less implicit way. It can be seen more particularly when dealing with the relation between context and the finality that makes us choose among all the available context information the ones that are relevant to make appropriate decisions according to the current situation.

As stated in classical decision support systems' (DSS) literature, decision making is the process of *choosing* among alternative course of action for the *purpose* of attaining a goal (our emphasis on "choosing" and "purpose") **[24]**. What motivates our research is to propose a highly adaptable decision support system, having as a central point the notion of context. There are several complex architectures that were proposed for decision support systems and we put into practice a very simple one. Our goal is to add an explicit context management and several learning modules in order to obtain a context-based DSS able to improve dynamically its capabilities to support user's decisions.

From the context-aware applications perspective, we are proposing a context-based multi-agent architecture, in order to add a reasoning and learning level to a classic context-aware application: agents reason about context information, learn how to select the relevant one and how to use it in order to make contextualized decisions on behalf of the users they assist.

In the following, we explain why and how context has to be taken into account when designing open and dynamic systems (section 2). In section 3 we briefly define context as we use it in MAS and the representation we chose to put into practice when constructing our system. We then describe the proposed multi-agent architecture (section 4) and learning methods for choosing and using context when making decisions (section 5) and illustrate its use in an agenda management application (section 6). Finally, we position our approach with respect to other definitions, representations and uses of context (section 7) and we draw some conclusions and present future directions for our research (section 8).

## 2 Vision

In an *open* multi-agent system, heterogeneous agents are supposed to enter and exit from the system at any time. To stress our point of view, let's draw the functional aim of the application that we describe in this paper. The application aims at helping users to manage their agendas. As in AgentCities project [27], several groups of users exist, each being supported by a different "society of agents". A society consists of agents that are situated in the same area and that try to help their users to manage their agenda.

Considering the possibility for users to move from one group to another one, agents can also move from one society to another. This way, an agent knowing how to manage the agenda of its user in a context may find itself in a new society where the context is totally different but where it may also have to solve the same problem and doesn't know how to do it, since the context is totally new. Our aim is to make this agent able to learn from other agents how to use the new context knowledge to help its user in the management of his agenda. This learning will be realized by sharing knowledge with other agents, in order to find equivalence relations. Thus, agents must be able to sense and to manage the available system-specific context information. They should also be able to communicate about context and to understand each other.

Given these motivations, our approach is the following: as in current context-aware application, the system architecture is structured along three levels: context sources retrieving context from different sensors, context management and dedicated application level in which context is used. The middle level 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 users is situated (for instance, we can imagine having one CM for each flour in a building). Agents which are operating at the application level, 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 the same protocol to ask the CM and other agents about context.

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. 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 an ontology of its own where the 'local' context is 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 how context is represented and then how it is used and learned into the multi-agent architecture.

## 3 Definition and representation

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. Let's note $\boldsymbol{F}$ the set of finalities.

An entity is an instance of a "person, object or place" (as mentioned in [5]), but also an activity, an organizational concept (role, group), etc.

A *context attribute (**a**)* designates the information defining an element of context, e.g. "ActivityLocation", "NamePerson", "ActivityDuration". Each context attribute has at least one value at a given moment, the value depending on several entities to which the attribute relates. For instance, the context attribute "NamePerson" defines the name of a person. When trying to instantiate this attribute, the needed parameter will be the specific person whose name interests us. The "PersonIsMemberOf" context attribute will also take a person as input, but will return (possibly) several groups in which that person takes part. Let's note $\boldsymbol{V_a}$ the definition domain of $\boldsymbol{a}$, the set of possible values that $\boldsymbol{a}$ may take (example: $\boldsymbol{V_{time}}$ =[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 $\boldsymbol{A}$ x $\boldsymbol{P_a}$ to $\boldsymbol{P(V_a)}$, where **A** is the set of all attributes, $\boldsymbol{P(V_a)}$ is the power set of $\boldsymbol{V_a}$, and $\boldsymbol{P_a}$ is the set of parameters needed to compute the value of $\boldsymbol{a}$ .

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 *Relevant Attribute Set* for the finality ***f***:

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

We will note an *instantiation of context attribute* $a\in \boldsymbol{A}$ as a 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's note ***I*** 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***, noted ***IRAS*(f)**, the set of instantiated context attributes relevant to finality ***f***:

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

## 3.1 Representing context with ontologies

Our aim is to represent context in a general and suitable manner for all applications that need to represent and reason about it. Several representations of context exist: contextual graphs ([1]), XML (used to define ConteXtML [19]), 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 express a particular vision on context. 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. To take into account the need for generality, and also considering the fact that we aim at having several MAS, each 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*. ([2]) defined context ontologies using OWL.

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.

## 4 General architecture for a context-based MAS

The proposed MAS architecture is composed of several learning agents (in our case study, we call them MySAM), each of them being the personal assistant of an user (see Figure 1). The agents can interact between them and can connect to a Context Manager that provides them with context knowledge by having access to the current state of the environment.

To make contextualized decisions, every agent needs to know the values of the relevant context attributes in a situation. It acquires them through the Context Manager whose main functionalities are to let the agents know which are the context attributes that it can manage and to compute the values to instantiate the context attributes that are relevant for an agent at some point.

### 4.1 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.

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.

### 4.2 Context-based learning agent

The context-based part of an agent is constituted of two main components: *selection* of relevant attributes of context for a certain purpose (the selection module) and *utilization* of context knowledge to make decisions (the module of decision). Even if we named the agent mySAM in reference to the application which we used as a modeling scenario, this architecture of a context-based learning agent is general and it is not restrained to the frame of application considered to illustrate this approach. Although mySAM has some negotiation modules (in order to establish meetings), we present here only the part of the agent that is relevant for the management and reasoning on context.

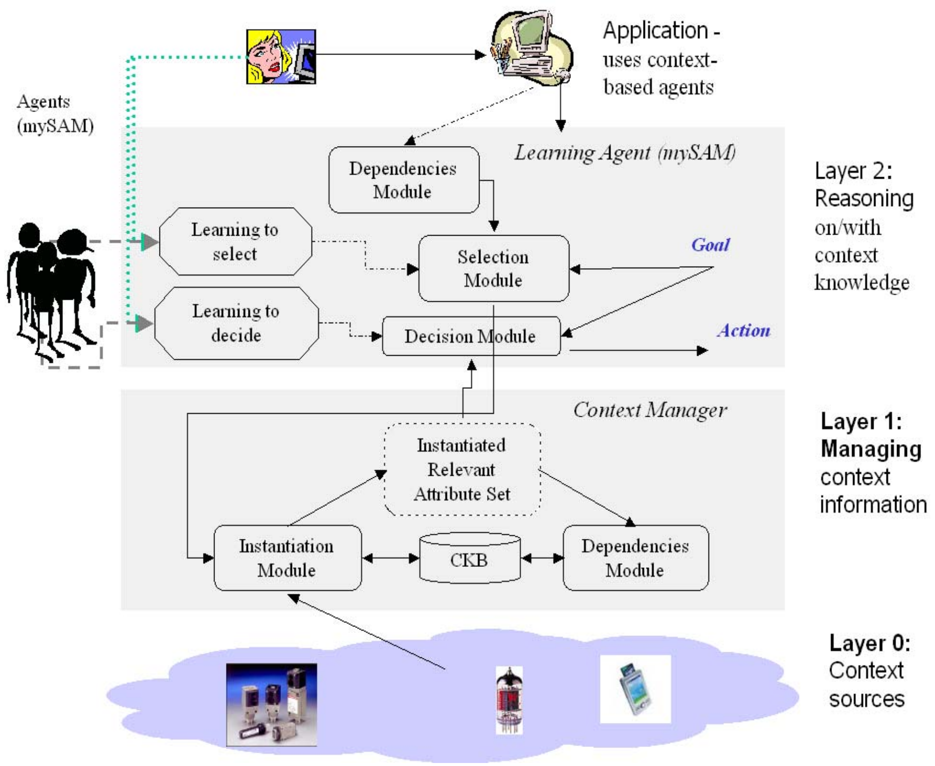

**Fig. 1. Context-aware MAS architecture**

In case of mySAM, the *selection* module provides the relevant attributes for a finality (RAS for this purpose). So, for instance, to decide if they should accept or not a ‘2 persons meeting’, the selection module is going to construct an RAS made of {the things already planned, the role of the person with whom they negotiate}; for a finality of type ‘seminar’ meeting, the RAS will be {the participants, the seminar’s theme, participants’ research fields, …}.

The *decision* module decides whether to accept or not a meeting taking into account the values computed for relevant context attributes (the context knowledge). This module is therefore going to know how to accept a meeting if we have nothing planned for that period of time and if the person that demands this meeting is our chief, for instance. It will also know that it should not accept otherwise. The purpose, the situation that we want to solve is going to help the agent choose relevant context. It is therefore the context knowledge set (IRAS) which is helping the agent to decide how to act in this situation.

The use of individual learning methods in MAS is already known as indispensable if we desire to have adaptable and efficient agents and multi-agent systems. Multi-agent learning is being considered for quite a while already, and different methods were already proposed. What we propose in this article are methods to collectively how to chose the ‘right’ context and what to do with it.

In what follows, we present the learning modules for the selection of relevant attributes (section 5.1) and for using context knowledge for decision-making (section 5.2). We want to underline that the presentation of these modules is made from a general point of view, without going into details in what concerns the specific learning method that is used for mono-agent learning (this choice is usually made taking into account the application). Our objective is to argue the necessity to use mono and/or multi-agent learning (as described in [21], [26]) when dealing with context and to imagine multi-agent learning mechanisms suitable for any type of context-aware application. We are not proposing a new learning method, but show how using existing methods can improve the decision making process in a context-aware system.

## 5 Learning how to reason with context to make decisions

### 5.1 Learning how to choose the relevant context for a decision (RAS)

Learning how to chose among context attributes which are the relevant ones could be very important in an application where the amount of available context information is too large and the effort needed to compute the values for all those attributes is not acceptable in term of efficiency. This step of the decision support process is the equivalent of the first phase of such a process as described in [24] or [6]: the *intelligence* phase, that involves several activities aimed at identifying problem situations or opportunities. We improve the agent’s capacity to choose the relevant information to assess this phase by an individual learning method (in connection to user’s feedback) and also by a collective one.

For *individual learning* (mono-agent learning) on how to choose among context attributes those who are relevant for a given situation, the agent uses the feedback from the user. For mySAM, the user chooses attributes which he considers as being relevant for the situation, before making a decision. When the user chose attributes as relevant for a type of meeting, the agent memorizes them. Next time the agent will have to deal with the same type of meeting, he will be able to propose to the user all known relevant attributes, so that the user adds or deletes attributes or uses them such as they are.

For example, the agent received a proposal for a “family meeting”. In the case of MySAM, the type of the proposed meeting is also the finality the agent will consider when choosing the RAS. Therefore, in this case, mySAM will know that to take a contextualized decision, it needs to find the RAS(“family-meeting”), then ask for the CM to instantiate it, in order to obtain the IRAS(“family-meeting”), and to make the decision based on this IRAS. Let’s suppose that, for now, the RAS(“family-meeting”)={“ActivityStartsAt”}. MySAM will propose the user this RAS and ask if it is enough to make the decision based only on this set of relevant ontext attributes. The user can further add to the RAS the attribute “ActivityImportance”, so that not only the start time of the meeting will be relevant. The new RAS(“family-meeting”) is now {“ActivityStartsAt”, “ActivityImportance”}. Next time a family meeting proposal will be received, MySAM will list this new RAS, so that the user can choose to remove one of the attributes or to add new ones.

An improvement for the method used in individual learning of relevant context attributes consists in using the attributes that other agents in the system have already learnt as relevant in that situation, therefore to make a knowledge sharing between agents(multi-agent learning). If an agent doesn’t know which attributes could be relevant for the situation in which he is (probably for the first time),it can ask other agents which are the attributes which they already know as being relevant. In the same way, if an agent had not had a lot of feedback on attributes in a specific situation, it can again try to improve its set of relevant attributes, by asking for others’ opinion. In the example given above, MySAM can choose to ask other agents for their RAS(“family-meeting”), then make the union of all RAS received as a response to its quest, and propose this enlarged set to its user for further modifications. This method improves the capabilities of mySAM with the ability to propose new relevant attributes. The user’s task becomes easier, in the way that he just has to check for the relevance of the proposed attributes, without going through the whole list of available context attributes to select possible interesting ones.

## 5.2 Learning how to use IRAS to make a decision

Using the IRAS retrieved from the CM is just a decision making process. The use an agent makes of the IRAS to propose a decision to be made in a specific situation corresponds to the *choice* phase of a DSS modeling process **[24]**.

To improve it, we used mono and multi agent learning to enhance the agent with the ability to evolve and to become more and more user-personalized.

Mono agent learning of how to use relevant context can be, therefore, any machine learning method developed in IA which is suitable to the type of application that we want to develop. For mySAM we chose the classification based on association (CBA) tool developed in the Data Mining II suite ([4]). In the “Implementation and Results” section we explain our choice and we give further details in what concerns this method.

The difficulty of multi-agent learning is due to the fact that agents have to have a common apprehension of the manner in which to use context attributes and knowledge. To be able to understand what other agents say, every agent must use the same ‘language’, the same manner of encoding information. They will therefore need to share an ontology of the domain, to make sure that the agents understand the contents of messages inter changed.

For multi-agent learning on how to use IRAS(f), we propose a very similar knowledge sharing method to that presented before. The protocol proposed for this distribution can be the same, the difference consists in the knowledge that we chose to share with others: we can chose to share only the solution to the problem and not divulge the way we reasoned to find that solution, or we can share directly our problem-solving method, so others can use it for themselves. The choice depends on the system. If agents should be cooperative and it is considered that no privacy matters should be taken into account, then the second solution is faster and more efficient. But if, as considered in mySAM, the agents should not share all their criteria for accepting or rejecting a meeting (for sure we don't want the boss' agent to know that we refuse the meeting because we don't feel like getting up that early…), then sharing just the solution (in mySAM, a "accept/reject" decision) could be preferable.

Using both individual and multi-agent learning for choosing and using relevant context in making decision makes our system open and distributed. Agents can come in and out of a society of agents with the context adjusting accordingly and without causing problems to the general use of the system. The decision making process is as well an individual process as a collective one. Agents try to solve the agenda managing problems on their own first, and then ask for opinions and propose a solution accepted by a majority of agents in the system, based on a voting procedure.

We underline the fact that, for choosing the RAS, as well for making decisions based on the IRAS, the user has the possibility to approve or reject the agent's proposal. What MySAM does is to propose solutions (actions), but it is always the user who makes the final decision. Of course, the goal would be to have an auto-customizing agent that will finally get the user's trust and will act on its own. In decision support system the idea of autonomy is not to be considered; the system is supposed to support the decision making process, not to bypass the user's approval. However, for multi-agent systems, one of the challenges is to develop autonomous agents that are able to accomplish tasks instead of its user by progressively reducing human involvement. For now, we propose a decision support agent (not an autonomous one), considering that giving the user the possibility to reject the agent's proposals helps increasing the acceptance of the user. In what follows, we detail implementation issues and give some results we obtained when testing MySAM.

## 6 Implementation and results

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 CMA. To deploy our agents we chose the JADE platform ([9]). Besides being the most used platform in MAS community at this time, JADE also proposes an add-on for deploying agents on hand-held devices, called LEAP.

JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same operating system) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one, as and when required. JADE is completely implemented in Java language and the minimal system requirement is the version 1.4 of Java (the run time environment or the JDK).

We used JADE to develop smaller mySAM agents, capable of handling the negotiation task for scheduling meetings on PDAs.

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. MySAM agents deployed on PDAs execute only the negotiation task, without any learning methods. Learning is done by a remote agent situated on a PC.

Learning mySAM agents use CBA (Classification Based on Association) algorithm to *learn how to use* relevant context (for acceptance or refusal of meeting proposals).

CBA (v2.1) has many unique features, the one that interests us being the classification and prediction using association rules. It builds accurate classifiers from relational data, where each record is described with a fixed number of attributes. This type of data is what traditional classification techniques use, e.g., decision tree, neural networks, and many others. It is proved to provide better classification accuracy (compared to CBA v1.0, C4.5, RIPPER, Naive Bayes) [4]. We used this method also because it generates behaviour rules comprehensive for both agents and humans. We have developed a module for the conversion of the rule from the CBA format:

```
Rule 6:   busyAfternoon = false
          durationEvent = 60
          groupType = work
                 -> class = yes
```

into a CLIPS format. CLIPS [3] is a productive development and delivery expert system tool which provides a complete environment for the construction of rule and/or object based expert systems. Although CLIPS also provides two other programming paradigms: object-oriented and procedural, we used the simple rule based one. Rule-based programming allows knowledge to be represented as heuristics, or "rules of thumb," which specify a set of actions to be performed for a given situation. A rule in CLIPS is specified in the following style:

```
     (defrule Rule6
          (busyAfternoon  false)
          (durationEvent  60)
          (groupType  work)
                 => (store CLASS yes))
```

“Class” specifies whether the agent should accept or refuse the proposed meeting. After transforming rules obtained with CBA in CLIPS rules, we could use Jess ([11]) inference engine. Jess was originally inspired by the CLIPS expert system shell, but has grown into a complete, distinct, dynamic environment of its own. Using Jess, one can build Java software that has the capacity to "reason" using knowledge supplied in the form of declarative rules. Jess is small, light, and one of the fastest rule engines available. The core Jess language is still compatible with CLIPS, in that many Jess scripts are valid CLIPS scripts and vice-versa.

Each agent has a module that deals with the rules generated by CBA in order to find out which rule can be applied in a specific context. The process is the following: giving a finality *f*, mySAM chooses the RAS(*f*), then asks for the IRAS associated with the RAS (the request is made to the CM). The obtained IRAS constitutes the current relevant context to be taken into account when making decisions. MySAM asserts as facts the current context in the form:

```
     (defrule startup  =>
          (assert ( ″attribute_1″  “value_attribute_1″))
      …  (assert ( ″attribute_n″  “value_attribute_n″)) )
```

Then, mySAM starts Jess inference engine to find the value of “CLASS”. The value can be ‘Yes’ (meaning that the meeting proposal should be accepted), ‘No’ (meaning that the meeting proposal should be refused) or ‘Unknown’ (no rule matched the specific context).

When no rule matches the specific context, mySAM can use *a multi-agent knowledge-sharing* session on how to use this specific context (IRAS) to find the solution. It starts a voting system: it asks all known agents in the system for their opinion on the situation (the situation being defined as (***f***, {(attribute_1, value_attribute_1),…})) and counts each opinion as a vote for “accept”, “reject” or “unknown”. The agent then proposes to the user the decision that has the most votes. Agents consider an “unknown” result as a “reject” (by default, an agent will propose the user to reject all meeting proposals that neither it, nor other agents know how to handle). We chose to use this “voting” procedure because it was faster than trying to share the rules for themselves and because, in this application, the privacy matter is important. Not all agents will want to share their decision-making techniques, but an “accept/reject/unknown” answer is reasonable. In this way, agents do not explain how they inferred that conclusion, but just what that conclusion is.

The context manager (CM) 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 concepts that characterize the domain and the context attributes that the CM will manage. CM provides context knowledge to all agents that are active in the system. The ontology was created using Protégé 2000 (Protégé is a free, open source ontology editor and knowledge-base framework [18]) and the agent accesses the ontology using Jena ([10]), a Java framework for building Semantic Web applications. Jena provides a programmatic environment for RDF, RDFS and OWL, including a rule-based inference engine.

Agents interactions in the system are quite simple: mySAM agents can query the CMA using a simple REQUEST/INFORM protocol, the meeting negotiations between mySAM agents are done in a simple PROPOSE/ACCEPT/REJECT manner.

When testing mySAM we were able to draw several conclusions:

- using a selection step to choose the RAS for a situation 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. We tested our hypothesis on a set of 100 examples. For 15 context attributes used, we obtained an overall classification error of 29.11% and more than 40 rules. When we split the example set on several finalities (“family-meeting”, “friends-meeting”, “work-meeting”), and for each situation we take into account a limited number of context attributes (7 for a meeting with family, 11 for others), the error becomes 7.59% and the number of obtained rules drops to an average of 15;
- sharing with other agents just the solution (accept/reject) for a situation is enough, because the agent that received the answer will then add this situation to its examples list, from where it will then learn the appropriate rule. Even if it will take longer to learn that rule than just having it immediately provided by others, the privacy problem is this way solved, because we share just the answer to a specific situation, and not the reasoning that produces such an answer.

## 7 Related work

Our study is centered on multi-agent systems, but our definition and vision on context is based more on approaches in other domains, like Artificial Intelligence, decision support systems, human-machine interaction, ubiquitous computing, etc.

We can mention here some definitions given by: Persson [17] ("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 [25] ("Any identifiable configuration of environmental, mission-related and agent-related features" used to predict a behaviour). There are several definitions DSS that are based on the same notions of relevance, without necessarily speaking of "context", but mentioning that the "situation should be identified", the choices are made based on some "criteria", etc. ([24]). We proposed in section 2.1. a definition that is quite similar to all these definitions 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.

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 [15]), activity, participants, location, required resources (Personal Calendar Agent [13]), system load, organization size (Distributed Meeting Scheduler [21]), time, user's location, etc. (Electric elves [20]). None of these works mention the idea of context but they all use "circumstances" or "environmental factors" that affect the decision making.

In making Calendar Agent ([12]), Lashkari et al. use the notion of context, but they assume that relevant context is known in advance, so that all contextual element that they have access to is considered relevant for the decision to be made. 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 behaviours 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.[2] or Socam, by Gu et al [23]. 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 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 how to represent it in a semantic and generic manner and how to reason on context knowledge based on this representation.

## 8 Conclusions and future work

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 a context-based architecture for a learning multi-agent system based on an ontology-based representation for context elements. 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, that in our ontology is a class, not a property (so is not restricted to an unique range). 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 enriching the framework for context-based MAS in order for it to be used for multiple application domains that consider context when adapting their behavior. Our future work will focus on taking into account a mode complex DSS, in order to highlight the use of context in each of the phases of a decision making process. Until now we focused on intelligence and choice phase to test how context can be used. Further more, we considered these phases as being atomic, and not decomposable into several sub-phases (sub-phases specified in **[24]**). We plan to extend the use of the notion of context also for the design phase and to consider each step of the four phases of decision making. We also consider the possibility to develop specialist agents, each for a specific task in the system (some first steps towards this have been already made in **[16]** or **[14]**) In what concerns learning, the framework will provide agents equipped with several individual learning algorithms and all needed modules to 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] Chen, H., Finin T., Anupam J. – “An Ontology for Context-Aware Pervasive Computing Environments”, The Knowledge Engineering Review Volume 18 , Issue 3, p. 197 – 207, 2003.

[3] CLIPS (C Language Integrated Production System) - http://www.ghg.net/clips/CLIPS.html

[4] Data Mining II – CBA - http://www.comp.nus.edu.sg/ ~dm2/

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

[6] Duvallet C. – “Des systèmes d’aide à la decision temps reel et distributes: modélisation par agents” – PhD thesis, Computer Science Laboratory of Havre, University of Havre, France, 05 October 2001, pp 7-17
[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, Zurich.
[9] JADE (Java Agent Development framework) : http://jade.cselt.it/
[10] Jena Semantic Web Framework - http://jena. sourceforge.net/
[11] Jess: http://herzberg.ca.sandia.gov/jess/index.shtml
[12] Lashkari Y., Metral M., Maes P.– “Collaborative Interface Agents”, Proc. of the Third International Conference on Information and Knowledge Management CIKM'94, ACM Press, 1994.
[13] Lin S., J.Y.Hsu – “Learning User’s Scheduling Criteria in a Personal Calendar Agent”, Proc. of TAAI2000, Taipei.
[14] Matsatsinis N.F., Moratis P., Psomatakis V., Spanoudakis N. – “An Intelligent Software Agent Framework for Decision Support Systems Development”, ESIT´99 (European Symposium on Intelligent Techniques).
[15] Mitchell T., Caruana R., Freitag D., McDermott J., Zabowski D.– “Experience with a learning personal assistant”, Communications of the ACM, 1994.
[16] Ossowski S., Fernandez A., Serrano J.M., Hernandez J.Z., Garcia-Serrano A.M., Perez-de-la-Cruz J.L., Belmonte M.V., Maseda J.M. – “Designing Multiagent Decision Support System. The Case of Transportation Management”, ACM/AAMAS 2004, New York, pp 1470-1471.
[17] Persson P.– “Social Ubiquitous computing”, Position paper to the workshop on ‘Building the Ubiquitous Computing User Experience’ at ACM/SIGCHI’01, Seattle.
[18] Protégé 2000 - http://protege.stanford.edu/
[19] 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
[20] Scerri, P., Pynadath D., Tambe M.– “Why the elf acted autonomously: Towards a theory of adjustable autonomy “ , First Autonomous Agents and Multi-agent Systems Conference (AAMAS02), p. 857-964, 2002.
[21] 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.
[22] Sian S. S. – “Adaptation Based on Cooperative Learning in Multi-Agent Systems”, Descentralized AI, Yves Demazeau & J.P. Muller, p. 257-272, 1991.
[23] 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 Conf., 2004.
[24] Turban E., Aronson J.E. – “Decision Support Systems and Intelligent Systems”, 6th edition, Prentice Hall, 2001, pp. 33-121.
[25] Turner, R. – “Context-Mediated Behaviour for Intelligent Agents”, International Journal of Human-Computer Studies, vol. 48 no.3, March 1998, p. 307-330.
[26] Weiss G., Dillenbourg P.– “What is “multi” in multi-agent learning?”, P. Dillenbourg (Ed) Collaborative-learning: Cognitive, and computational approaches, p. 64-80, 1999.
[27] Willmott S., Calisti M., Rollon E. –“Challenges in Large-Scale Open Agent Mediated Economie”, Proc. of Workshop on Agent Mediated Electronic Commerce AAMAS 02, july 2002.