A Context-Based Agent Architecture for Learning How to Make Contextualized Decisions

Oana Bucur, Olivier Boissier, Philippe Beaune

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

Abstract. Developing context-aware applications needs facilities for recognizing the context, reasoning on it and adapting accordingly. In this paper, we propose a context-based multi-agent architecture consisting of context aware agents able to learn how to discern relevant from non relevant context on one hand, and to make appropriate decisions based on it on the other hand. This multi-agent system interacts with a context manager layer, based on an ontological representation of context, which is able to answer context-related queries. 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 pervasive computing has stressed the importance of *context*. As defined in [4], this concept consists in "any information that can be used to characterize the situation of an entity". This intuitive definition does not specify how to choose among all the available context information the one that is relevant or how to deal with it to make contextualized decisions. Existing works handle this problem in an explicit or implicit manner. In this paper, our goal is to draw a common base for context-aware reasoning. We propose a general layered architecture composed of different Context Manager Services (CM Service) on which a context-based multi-agent architecture is defined. Since pervasive applications are inherently open, they may be composed of several "societies" of heterogeneous and situated agents sharing or not the same context that changes over time. Thus, agents must be able to sense and manage context but also to communicate and to understand each other about it. We propose an ontology-based representation for contextual information. The defined agents can learn how to discern relevant from non-relevant context and how to make appropriate decisions based on it. They are also able to share with other agents the way they use context knowledge in solving similar problems.

In this paper, we illustrate our proposal with a case study of an open and interoperable context-aware agenda management Multi-Agent System (MAS). It is composed of several meeting schedulers called mySAM (my Smart Agenda Manager). A mySAM agent assists its user in fixing meetings by negotiating them with other mySAM agents and by using context knowledge to decide to accept or 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 and multi-agent learning (knowledge sharing).

Before describing the proposed architecture (section 3), we will present the ontology-based representation for context (section 2). In section 3, we will also describe how agents are able to learn context for decision-making. We then illustrate our work in the agenda management application (section 4). Before concluding, we will situate our approach in related work.

2. Representation of context for MAS

In this section, we define "context" and describe how we represent it to design and implement our proposed context-based MAS.

2.1. Definition and classification

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 (e.g. to decide whether a proposal for an appointment should be accepted or not, to see whether the current situation is similar to another one or not, to understand a conversation, etc.). Let's note $\boldsymbol{F}$ the set of finalities.

A *context attribute* ***(a)*** designates the information defining context, e.g. "ActivityLocation", "NamePerson", "ActivityDuration". We consider a context attribute as a function, with one or more parameters, returning a value. For instance, context attribute "NamePerson" is a function defined on the set of Persons, returning a String value corresponding to the name of a person. 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 define ***valueOf*** as an application 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 $\boldsymbol{a}$ is relevant for the finality $\boldsymbol{f}$. Let's call $\boldsymbol{RAS}(f)$ the *Relevant Attribute Set* for the finality $\boldsymbol{f}$: $\boldsymbol{RAS}(f) = \{ a \in \boldsymbol{A} \mid \boldsymbol{is\_relevant}(a,f)=\text{true} \}$.

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 $\boldsymbol{I}$ the set of instantiated context attributes as $\boldsymbol{I} = \{(a,v) \mid a \in \boldsymbol{A} \wedge \boldsymbol{valueOf}(a)=v\}$. We call *Context Knowledge Set* of a finality $\boldsymbol{f}$, noted $\boldsymbol{CKS(f)}$, the set of instantiated context attributes relevant to finality $\boldsymbol{f}$: $\boldsymbol{CKS}(f) = \{(a,v) \mid a \in \boldsymbol{RAS}(f) \wedge (a,v) \in \boldsymbol{I}\}$.

Let's notice that in related work ([12], [18], [19]), the notion of "context" is often understood as being what we defined as the CKS. To explain the difference between RAS and CKS let's consider the following example. Given 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 CKS for a student may be $CKS_{student}$(f)={(RoleOfPersonInGroup, {teacher}), (ActivityScheduledInSlot, {Activity001})} and for a teacher $CKS_{teacher}$(f)={ (RoleOfPersonInGroup, {student}), (ActivityScheduledInSlot, {Activity255)}. As we can see, the difference between CKS 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 CKS-dependent. Taking into account the definitions that we proposed so far we now describe the representation that we defined.

2.2. Representing context elements

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 [17]), or object oriented models ([6]). All these representations have strengths and weaknesses. As stated in [7], lack of generality is the most frequent weakness: each representation is suited for a type of application and express 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 [7] 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 ([14]). In their model, each context attribute is represented as an OWL property (DataTypeProperty or ObjectProperty, depending on the range of values). We extended this representation due to the limitations it imposes when we need to represent more complex context attributes (like role, activities already planned, etc.).

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

Fig. 1. The description of the class #ContextAttribute

What we propose is to add to the ontology the class "#ContextAttribute" (see fig. 1.) corresponding to our definition of a context attribute as defined in section 2.1. This class is composed of the following properties: name, number and list of entities (parameters) it connects to, type of its value. Instances of that class will be the context attributes that are known and used in that system by the CM Service. 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. In Fig. 2 we give the list of context attributes that we defined in MySAM application. For instance, the context attribute `RoleOfPersonInGroup` is described with the following instance of 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>
<tr>
<td>Person - related
InterestsPerson
MailPerson
OriginPerson
IsSupervisorOf
StatusPerson
Supervises
RoleOfPersonInGroup
GenderPerson
DateOfBirth
PersonIsMemberOf</td>
<td colspan="2">Environment – related
DevicesAvailableInBuilding
DevicesAvailableInRoom
DevicesAvailableAtFloor
TransportAvailable
WeatherForecast</td>
<td>Time-related
Year
Month
Day
TimeZone
DayOfWeek
Seconds
Minutes
HHour
TimeOfDay</td>
</tr>
<tr>
<td colspan="3">Location - related
PersonIsInRoom
PersonIsAtFloor
PersonIsInBuilding
PersonIsInCountry
PersonIsInCity
RoomAtFloor
InCountry
InCity
InBuilding
InRoom
RoomInBuilding
AtFloor</td>
<td rowspan="2">Activity – related
ActivityStartsAt
ActivityEndsAt
ActivityScheduledInSlot
AcivityGoal
ActivityEmergency
ActivityImportance
ActivityDescription
ActivityParticipants
ActivityLocation
ActivityType
ActivityDuration</td>
</tr>
<tr>
<td>Agenda - related
BusyMorning
BusyAfternoon
BusyEvening
TimeSlotAvailable</td>
<td colspan="2">Agent – related
CurrentUser
StatusAgent</td>
</tr>
</table>

Fig. 2. Context attributes defined in MySAM ontology for agenda management

3. Architecture for a context-based learning MAS

The proposed layered architecture is composed of mySAM agents (Fig. 3), that assist a user. Agents interact with each other and with a context management layer composed of context manager services (CM Service). Being connected to the current state of the environment, a CM Service provides agents with context. The CM Service and not agents have 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 start by describing the CM service and continue by the details of the dedicated learning part of the agent's architecture.

3.1. Context manager service (CM Service)

The main functionalities of CM Service are to let the agents know which is the context attributes set (defined in the ontology) that it manages and to compute CKS

corresponding to RAS given by the agents at some point of processing. When entering a society, an agent asks the corresponding CM Service to provide it with the context attributes that it manages. Acting as intermediary between agents and the environment, CM Service is able to answer requests regarding its managed context attributes. This way, if, for instance, CM Service answers "Date" and "ActivityLocation" to an agent querying it about context attributes for managing rendez vous, even if the agent knows that other context attributes exist – e.g., "roleOfPersonInGroup"– it knows that it cannot ask CM Service for the value of this attribute since this latter is not able to compute it.

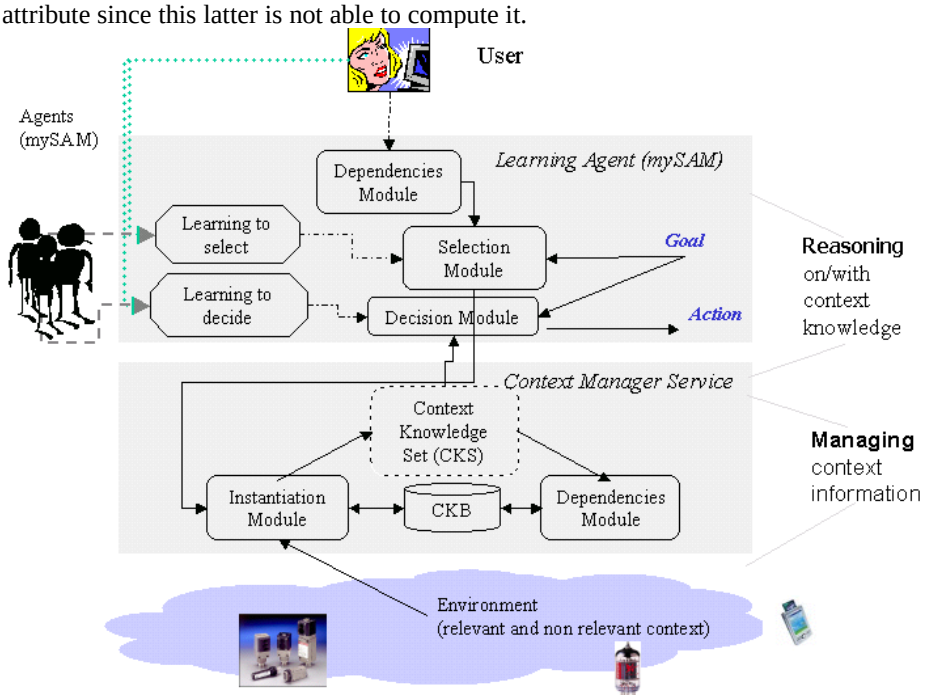

Figure 3. Context-based agent architecture

The Context Knowledge Base contains the ontology of the domain, defined as a hierarchy with #Entity as root, and all instances of class #ContextAttribute that will be managed by the CMService. The *instantiation* module computes the CKS(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.

3.2. Context-based learning agent

Although a mySAM agent has some negotiation modules (in order to establish meetings), we focus here on its management and reasoning on context modules. 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 (see Fig. 3): *selection* of relevant attributes for a certain finality f (RAS(f)) and *decision* based on instantiated attributes (CKS(f)) provided by CM Service.

For example, for a finality relative to deciding whether accepting or not a "2 participants" meeting, the RAS built by the selection module could be {"ActivityScheduledInSlot", "roleOfPersonInGroup"}; or, for a finality relative to a "several participants" type of meeting, the RAS could be {"ActivityParticipants", "ActivityDescription", "PersonInterests", etc}. The decision module knows 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.

Several approaches have been proposed [20], [23] in the recent years concerning multi-agent learning. Since the specific mono-agent learning method that is used for learning modules attached to the decision-making based on CKS is application dependent, we will not detail it here. We just highlight the necessity to add a multi-agent learning perspective and to point out what are the consequences.

Learning how to choose the RAS. Learning how to choose the relevant context attributes is important in our targeted applications since the amount of available context information is too large and the effort needed to compute the values for all those attributes rise efficiency problems. From an individual learning perspective, agents use the user's feedback to learn *how to choose* among context attributes those that are relevant for a given situation. In our application, mySAM memorizes the attributes chosen by the user as being relevant for that situation before making a decision. Next time the agent will have to deal with the same type of situation, it 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.

Using the context ontology defined in the previous section (3.1), agents are able to share a common understanding of the manner of using context attributes and knowledge. To improve the method used in individual learning of *how to choose* relevant context attributes, we made agents able to share knowledge, focusing on attributes that other agents in the system have already learnt as relevant in that situation. When an agent does not know which attributes are relevant for the considered situation f, it can ask other agents what are the attributes which they already know as being relevant in that situation (their RAS(f)). In the same way, if an agent needs more feedback on attributes in a specific situation, it can again try to improve its set of relevant attributes, by asking for others' opinion. The resulting RAS(f) is the union of the ancient RAS with the new relevant attributes proposed by other agents. Next time the agent will be in the situation f, it will propose the new obtained RAS to the user, so he can choose to keep the new attributes, to add some more or to delete some of them that seem not relevant for him. For example, when deciding about a meeting with a friend, the agent's RAS is {ActivityStartsAt, ActivityDuration}. The agent asks others what their RAS is and, at the end of the sharing session, its RAS will become {ActivityStartsAt, ActivityDuration, dayOfWeek, BusyEvening}. The user can then choose to keep the attribute "dayOfWeek" as relevant and to remove "BusyEvening" from the list of relevant attributes for this finality.

Learning how to use the CKS. Learning *how to use* relevant context may be realized by any machine learning method developed in AI, suited to the type of application that we develop. In our case, a mySAM agent uses a classification based on association (CBA) tool developed at School of Computing, University of Singapore, in the Data Mining II suite ([3]). We will show in the following section some results we obtained using this approach.

For multi-agent learning on *how to use* context knowledge, we modified the knowledge sharing method so that the agents can choose between (i) sharing only the solution to the problem, keeping for themselves the knowledge used to find that solution, or (ii) sharing the problem-solving method itself, so that others can use it for themselves. The choice depends on the application and more particularly on privacy matters. The second solution is more efficient in that it gives an agent the method to solve the problem, not just the answer to its problem. This way, next time the agent needs to solve the same type of situation, it will directly apply the method, without asking again for help from other agents. But if, as considered in mySAM, the agents should not share all their criteria for accepting or rejecting a meeting, then sharing just the solution (an "accept/reject" decision) should be preferable. We implemented the latter solution in our agenda management case study.

4. 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 CM Service. Agents were deployed with the JADE/LEAP platform ([8]) 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. Agent running on PDA and on proxy constitute a MySAM agent.

For learning *how to use* relevant context (for acceptance or refusal of meeting proposals), mySAM agents use CBA (Classification Based on Association) algorithm. That gives better results than C4.5 [3]. Another advantage is that it generates behaviour rules comprehensive for both agents and humans. The obtained rules have been used with Jess ([10]) inference engine.

In order to provide examples for learning algorithm, the system has been used (for meeting negotiations) by several members in our department for several weeks. Here is an example of the rules we obtained using CBA on the examples generated by using mySAM: *IF ActivityDuration = 120 AND BusyMorning = true AND BusyEvening = true THEN class = no* ("class" specifies whether the agent should accept or refuse the proposed meeting). When no rule matches the specific context, mySAM is constrained to use *a multi-agent knowledge-sharing* session on how to use this specific context (CKS) to find the solution. It starts a voting system: it asks all known agents in the system for their opinion on the situation, and counts each opinion as a vote for "accept", "reject" or "don't_know". The agent then adopts the decision that has the

most votes. Agents consider an "unknown" result as a "reject" (by default, an agent will 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 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 ([16]) and the agent accesses the ontology using Jena ([9]), a Java library designed for ontology management.

Agents interactions in the system are quite simple: mySAM agents can query the CM Service 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 ("meeting_with_family", "meeting_with_friends", "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.

5. Related Work

In this section we'll present a brief state of the art in context definition, context-aware MAS and context-aware architectures, in order to position our work relative to what has been done in this domain. We don't position our work relative to the individual and multi-agent learning domain, because our goal was not to propose a learning algorithm, but to use some already proposed methods, but for the specific goal of dealing with context matters [23].

Our definition of context is quite similar to definitions proposed by Persson [15], Brezillon [1] or Edmonds [5], 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 applications similar to our agenda management application, there are several works that adapt to context: Calendar Apprentice [13], Personal Calendar Agent [12], Distributed Meeting Scheduler [19], Electric elves [18], etc. Most of these works don't mention the idea of "context" but they all use the "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 context element that they have access to is considered relevant for the decision to be made. These approaches are not application-independent when handling context, because they do not provide neither a general representation of context knowledge nor methods to choose relevant context elements for a specific decision. This is the main difference and contribution of our work in the sense that we propose a MAS architecture based on an ontological representation of context and that can permit an individual and multi-agent learning of how to choose and use context. It is not the application that generated the 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, proposed by Chen et al.[2] or Socam, by Gu et al [21]. We based our architecture on CoBrA and Socam, but we added the learning modules for chosing relevant context and using it. The context broker and context interpreter are similar to our context manager, with the difference that our concern was not how to acquire information from heterogeneous sources, but mostly how to represent it and how to reason on context knowledge based on this representation.

6. Conclusions

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.

As future work, we envisage enriching this framework for context-based MAS to be used for any kind of applications that consider context when adapting their behavior. The context manager has to be able of dealing with all context-related tasks (including the computation of context attributes values) and to share all his context-related knowledge. In order to make this possible, our future work will focus on representing and managing: (i) how to compute the values for derived context attributes, (ii) dependencies that can exist between context attributes, (iii) the importance of different attributes in different situation (make a more refined difference between relevant and non relevant attributes, because there could be some attributes that are more important than others).

In what concerns learning agents, the framework will provide agents with one (or several) individual learning algorithm and all that is needed to communicate and share contextual knowledge (how to choose, compute and use context to make decisions).

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. et al. – "An Ontology for Context-Aware Pervasive Computing Environments", The Knowledge Engineering Review Volume 18 , Issue 3, p. 197 – 207, 2003

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

[4] 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

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

[6] 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

[7] Henricksen K. et al – "Modeling Context Information in Pervasive Computing Systems", Proc. First International Conference on Pervasive Computing 2002, p. 167-180

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

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

[10] Jess: http://herzberg.ca.sandia.gov/jess/index.shtml

[11] Lashkari Y. et al – "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. et al – "Experience with a learning personal assistant", Communications of the ACM, 1994

[14] OWL - http://www.w3.org/2004/OWL/

[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] Sian S. S. – "Adaptation Based on Cooperative Learning in Multi-Agent Systems", Descentralized AI, Yves Demazeau & J.P. Muller, p. 257-272, 1991

[21] Tao Gu et al – "An Ontology-based Context Model in Intelligent Environments", Proc. of Communication Networks and Distributed Systems Modeling and Simulation Conference, 2004

[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] Weiss G., Dillenbourg P.– "What is "multi" in multi-agent learning?", P. Dillenbourg (Ed) Collaborative-learning: Cognitive, and computational approaches, p. 64-80, 1999