What Is Context and How Can an Agent Learn to Find and Use it When Making Decisions?

Oana Bucur, Philippe Beaune, and 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. Developing context-aware applications needs facilities for recognizing 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 distinguish relevant from non relevant context and to make appropriate decisions based on it. 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 [5], this concept consists in "any information that can be used to characterize the situation of an entity". This definition does not indicate 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 layered architecture made of a Context Manager (CM) layer, on which a context-based multi-agent layer is defined. Since pervasive applications are inherently open, they may be contain several "societies" of heterogeneous and situated agents. Thus, agents must be able to sense and manage context but also to communicate 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.

In this paper, we demonstrate our proposal with a case study of an open and interoperable context-aware agenda management, implemented using Multi-Agent System (MAS). Our MAS is made of several meeting scheduler agents 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 seslect relevant context and how to use it to deal with a meeting proposal is acquired through individual and multi-agent learning (knowledge sharing).

M. Pěchouček, P. Petta, and L.Z. Varga (Eds.): CEEMAS 2005, pp. 112 – 121, 2005.

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

2 "Context" in MAS

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

2.1 What Is "Context" - Definition and Classification

From Dey's definition, 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* ($\boldsymbol{a}$) 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. We name $\boldsymbol{V_a}$ 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 ***isRelevant(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 $\boldsymbol{f}$: $\boldsymbol{RAS}(f) = \{ a \in \boldsymbol{A} \mid \boldsymbol{isRelevant}(a,f)=\text{true} \}$.

We call 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 *Instantiated Relevant Attribute Set* of a finality $\boldsymbol{f}$ - $\boldsymbol{IRAS}(\mathbf{f})$, the set of instantiated context attributes relevant for a finality $\boldsymbol{f}$: $\boldsymbol{IRAS}(\text{f}) = \{(a,v) \mid a \in \boldsymbol{RAS}(\text{f}) \wedge (a,v) \in \boldsymbol{I}\}$.

Let's notice that in related work ([13], [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 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 IRAS for a student may be $\text{IRAS}_{\text{student}}$(f)={(RoleOfPersonInGroup, {teacher}), (ActivityScheduledInSlot, {Activity001})} and for a teacher

$IRAS_{teacher}(f)$ = { (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 Attributes

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 ([7]). All these representations have strengths and weaknesses. As stated in [8], lack of generality is the most frequent weakness: usually, each representation is suited for a specific 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, is an extension of the "attribute-value" representation, contextual information being structured around an entity. An entity represents a physical or conceptual object. We based 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.* ([3]) defined context ontologies using OWL. 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.).

Table 1. The description of the class #ContextAttribute

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

What we did was to add to the ontology the class "#ContextAttribute" (see table 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 . 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 Table 2. we give some examples of context attributes that we defined for the 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 2. Some examples of context attributes defined in MySAM ontology

Person – related	**Time-related**
InterestsPerson :(Person)-> String 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	ActivityStartsAt:(Activity)->Time ActivityEndsAt :(Activity)-> Time AcivityGoal : (Activity) ->String ActivityParticipants: (Activity) -> Person*
Agenda - related	**Environment – related**
BusyMorning : (Agenda) -> Boolean BusyAfternoon : (Agenda) -> Boolean BusyEvening : (Agenda) -> Boolean	DevicesAvailableInBuilding : (Building) -> Device* DevicesAvailableInRoom:(Room)->Device* DevicesAvailableAtFloor : (Floor) -> Device*

3 Architecture for a Context-Based Learning MAS

The proposed layered architecture is composed of mySAM agents (Fig. 2), that assist a user. Agents interact with each other and with a context management layer composed of context managers (CM – Fig. 1). Being connected to the current state of the environment, a CM provides agents with context. The CM and not the 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 and continue by the details of the dedicated learning part of the agent's architecture.

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. This way, if, for instance, CM 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 for the value of this attribute since this latter is not able to compute it.

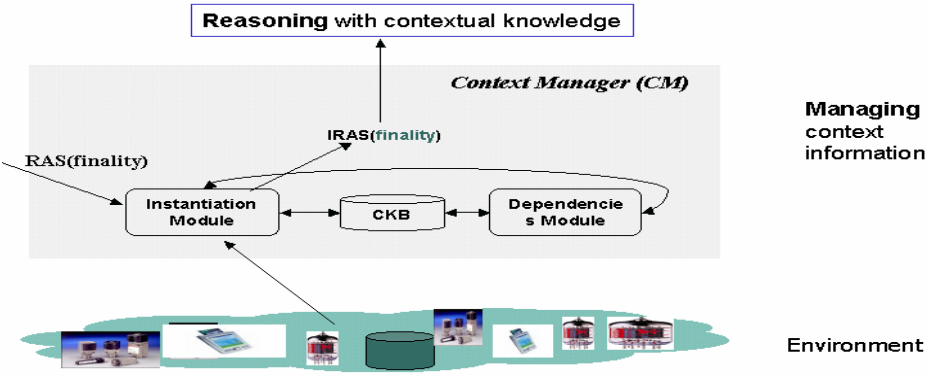

Fig. 1. Context manager architecture

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 CM. 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.

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. 2): *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 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], [26] recently concerning multi-agent learning. Since the specific mono-agent learning method that is used for learning modules attached to the decision-making based on IRAS 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.

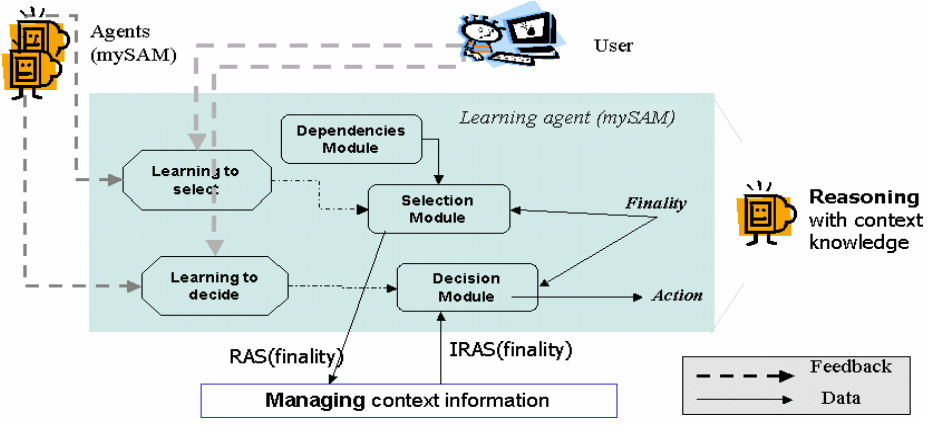

Fig. 2. Context-based agent architecture

Learning how to select RAS(f). 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 section 2.2, 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 make decisions based on IRAS. 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 ([4]). 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. For more details on learning methods, see [2].

4 Implementation and Results

In order to validate our proposal, we developed the system proposed as a case study in section 1, a multi-agent system containing several mySAM agents and one CM . 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.

For learning *how to use* relevant context (for acceptance or refusal of meeting proposals), mySAM agents use CBA (Classification Based on Association) algorithm. CBA gives better results than C4.5 [4] and it generates rules comprehensive for both agents and humans. The rules have been used with Jess ([11]) 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 (IRAS) to find the solution. 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 "unknown". The agent then proposes to its user 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 choose to use this "voting" procedure because not all agents will want to share their decision-making techniques, but an "accept/reject/unknown" answer is reasonable.

The 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 context attributes that it will manage. It answers to context-related queries from all agents that are in the system. The ontology was created using Protégé 2000 ([16]) and CM accesses the ontology using Jena ([10]), a Java library designed for ontology management.

Agents interactions in the system are as follows: mySAM agents can query the CM using a REQUEST/INFORM protocol, negotiations between mySAMs being done using a PROPOSE/ACCEPT/REJECT protocol.

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 decision (accept/reject) is preferable, 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 be slower than just sharing the specific rule, the privacy problem is this way addressed, because the agent shares just the answer to a specific situation, and not the reasoning that produced the 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 learning domain, because our goal was not to propose a learning algorithm, but to use some already proposed methods for the specific goal of dealing with context [26].

Our definition of context is quite similar to definitions proposed by Persson [15], Brezillon [1], Edmonds [6], or Thevenin and Coutaz ([22]) in the sense that it is based on: (i) elements that structure context and (ii) its use, i.e. the finality when using it. The definition we proposed takes into account those two dimensions of context; it also explains how to manage them when designing 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 [14], Personal Calendar Agent [13], 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 ([12]), 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. MySAM is just a case study to validate our approach.

Mostly, context 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.[3] or Socam, by Gu et al [21]. We based our architecture on CoBrA and Socam, but we added the learning modules for choosing relevant context and using it. The context broker and interpreter are similar to our CM, with the difference that our concern was not how to retrieve information from 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 context, notion that is used in almost all applications, without consistently and explicitly taking it into account. We have proposed an ontology-based representation for context and a context-based architecture for a learning MAS 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 will extend this framework for context-based MAS to be used for any kind of application that considers context to adapt. The CM will be able to deal with all context-related tasks (including the calculation of context attributes values) and to share all this context-related knowledge. In order to make this possible, our future work will focus on representing and managing how to calculate the values for context attributes, and the importance of different attributes in different situation (making 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 (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. Bucur O, Boissier O, Beaune P – “Knowledge Sharing on How to Recognize and Use Context to Make Decisions”, to appear in Proc. of Workshop “Context Modeling for Decision Support”, Vth International and Interdisciplinary Conference “Context 05”.
3. Chen H., Finin T., Anupam J. – “An Ontology for Context-Aware Pervasive Computing Environments”, The Knowledge Engineering Review, p. 197–207, 2003.
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. 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. Jess: http://herzberg.ca.sandia.gov/jess/index.shtml
12. Lashkari Y., Metral M., Maes P – "Collaborative Interface Agents", Proc. of CIKM'94, ACM Press.
13. Lin S., J.Y.Hsu – "Learning User's Scheduling Criteria in a Personal Calendar Agent", Proc. of TAAI2000, Taipei.
14. Mitchell T., Caruana R., Freitag D., McDermott J., Zabowski D.– "Experience with a learning personal assistant", Communications of the ACM, 1994.
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, 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.
22. Thevenin D., J. Coutaz. – "Plasticity of User Interfaces: Framework and Research Agenda". In Proceedings of INTERACT'99, 1999, pp. 110-117.
23. 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.
24. Turner, R. – "Context-Mediated Behaviour for Intelligent Agents", International Journal of Human-Computer Studies, vol. 48 no.3, March 1998, p. 307-330.
25. Widmer G.– "Tracking context changes through meta-learning", Machine Learning, 27(3):259-286, Kluwer Academic Publisher.
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.