An Agent-based Context Awareness Management Framework

Ionut Anghel, Tudor Cioara, Ioan Salomie, Mihaela Dinsoreanu

Computer Science Department, Technical University of Cluj-Napoca
Cluj-Napoca, Romania
{*Ionut.Anghel, Tudor.Cioara, Ioan.Salomie, Mihaela.Dinsoreanu*}*@cs.utcluj.ro*

***Abstract*—Context awareness drives adaptability of pervasive computing systems both in virtual and physical environments that change dynamically. This paper presents a context awareness management framework based on intelligent software agents. The fundamental element of this framework is our RAP (Resources, Actors, Policies) context model which defines two ways for context information representation: set based and ontology based. The RAP context model set based representation is used by the context management agents to evaluate the conditions under which management processes should be executed. The ontology based representation is used for reasoning and learning purposes, to infer new context related information. The context model management infrastructure is implemented using intelligent software agents that generate and administrate the context model artifacts at run time. The proposed context management framework is tested and validated in our DSRL (Distributed Systems Research Laboratory) smart environment infrastructure.**

***Keywords*-intelligent agents; context awareness; context management; smart environments.**

I. Introduction and Related Work

Context aware computing refers to the ability of software applications to detect and respond to changes in their execution environment [1]. The key elements that need to be incorporated into a context-aware application should solve the following issues [2]: (i) context identification and representation, (ii) context storing, sharing and retrieval and (iii) context information processing / reasoning.

One of the most important problems regarding developing context aware applications is to identify the properties and the elements that define the context. In [3], the authors take into consideration six attributes that describe the context: the identity of the user, the spatiotemporal characteristics (time and location), the facility attribute (the system devices and their capabilities), the prior activity of the user, the intrinsic / psychological properties of the user and the community. A similar classification can also be found in [4], where the authors describe the context in terms of identity, time, location and device capabilities.

For context representation, generic models that aim at accurately describing the system execution context in a programmatic manner are proposed [2]. *Key-Value models* represent context information using a set of attributes and their associated values [5]. *Markup models* enable structuring context information into a hierarchy where tags describe context attributes and associated values [6]. *Object models* structure context into object classes and their implicit relationships [7].

The current approach uses our RAP context model to represent the context information [8]. The RAP context model ontology based representation is used to infer new context related information using reasoning and learning algorithms. The use of ontology representations to model the context related information is also proposed in [9], [10]. In these approaches the context properties are represented as ontological concepts during design time and instantiated with run-time sensor captured values. The main disadvantage of these approaches is the high degree of inflexibility determined by the human intervention in the context representation phase. This problem is solved in our approach by defining an agent based solution that uses the RAP context model set based representation to evaluate the real world context and to automatically construct or update the context representation.

Another issue in dealing with context awareness is how the context information is stored, accessed and shared by the context aware system components and how these entities communicate. Three basic types of context consumption and sharing are described in the literature: *RPC-based* interaction (using client requests for particular context data) [11], *event-based* interaction (using notifications that are received by the client on specific events) [12] and *P2P interaction* (based on ad hoc composition of context sources) [13]. Our approach uses a web service-based loosely coupled architecture for the framework Sensor API component with the constraint that context providing web service operations must have a predefined description.

The use of mobile agents to manage and process the context information is underlined in [14]. The authors present the advantages of agents, such as their ability to communicate with each other and the ability to travel through heterogeneous networks and spawn across multiple distributed systems. For a context aware system to achieve intelligence, it has to benefit from the agents mobility in order to reduce the context processing computational complexity [15]. In [16], the authors present a detailed architecture that can be used for developing context aware applications. The architecture is based on four intelligent agents and is able to receive requests from the users and solve them using web service composition.

Using a layered approach instead of an agent-based one, the authors of [4] present a technique for responding to context aware requests. The bottom physical layer gathers data from the environment which is passed to the context management layer that can aggregate data from multiple sensors and attach semantics to the result. The service provisioning layer discovers, composes and executes services according to user requests.

Reasoning in context-aware applications is the focus of [17]. Here, fuzzy Petri nets are used to describe rules based on available context information. Data obtained from sensors together with user profiles and requests represent the input data for the reasoning mechanism. As output, a set of recommendations are returned to modify the system state. The use of reinforcement learning techniques to deduce new semantic knowledge information is proposed in [18].

The objective of this paper is to present a context management framework for smart spaces based on mobile and intelligent agents. The fundamental element of this framework is our RAP context model which represents environment context information using three sets: context resources, actors and policies. The context model management infrastructure is implemented by using intelligent software agents [19] that generate and administrate the context model artifacts at run time. The RAP context model set based representation is used by the management agents infrastructure to evaluate the conditions under which it should execute the context aware management processes. An ontology based representation is used for reasoning and learning purposes and in order to infer new context related information. The proposed context management framework is tested and validated in our DSRL smart environment infrastructure [23].

The rest of the paper is organized as follows: Section 2 presents a short overview of our RAP context model, Section 3 deals with the context management framework, Section 4 contains some aspects regarding the context management framework components development, Section 5 presents a simulation process and results evaluation while Section 6 concludes the paper and describes the future work.

II. The Context Model

To represent the context information in a programmatic manner we have used the RAP context model presented in [8]. This model defines to two types of context information representations: a set based and an ontology based. In the set based approach the context information is modeled as a triple:

$$C =< R, A, P >$$

where R is the set of context resources, A is the set of context actors and P is the set of context related policies.

A *context resource* has a unique identity, can be annotated with semantic information and is characterized by its properties, services and influence zone. The resource properties describes the set of relevant context information provided by the resource. The resource services specify its functionality (e.g. a service that locates / updates an object). The actors interact with a context resource through its attached services. The influence zone of a resource represents the physical or logical area in which the presence of that resource can be sensed (becomes visible for an actor or for another resource).

A *context actor* represents a physical or virtual entity that interacts directly with the context or uses the context resources to fulfill its needs. The actor is a context information generator, has a unique identity and can be annotated with semantic information. An actor is characterized by its specific resources, the context related request, its preferences and the actor-context contract.

A *context policy*, represents a set of rules that must be followed by the actors or resources located in the context influence zone.

In the ontology based representation the relationships between the context model sets are integrated in a general purpose context ontology core. The domain specific concepts are represented as sub trees of the core ontology by using is-a type relations. A system context situation is represented by the core ontology together with the domain specific sub trees of the concepts and their instances at a specific moment in time. The two ways of representing the context (set based and ontology based) are equivalent and need to be kept synchronized. The set based context model is used to evaluate the conditions under which the context agents should execute management processes. The ontology based model will be used by the context aware applications for reasoning and learning purposes.

In order to provide an accurate representation of the real world context, the following context representation artifacts are defined: specific context model, specific context model instance and context actor instance.

The specific context model is obtained by mapping the context model onto different real scenarios and populating the sets with real context specific elements:

$$C_S =< R_S, A_S, P_S >$$

A specific context model instance contains the set of context resources with which a pervasive application interacts, together with their values in a specific moment of time:

$$C_{SI} =< R_{SI}, A_{SI}, P_{SI} >$$

The specific context model instance represents the context situation to which a pervasive application must adapt.

The context actor instance contains the set of context resources with which the actor can interact, together with their values in at a specific moment in time:

$$CI_a^t =< R_a^t, a, P^t >$$

A context actor instance represents the projection of the specific context model instance onto a certain actor.

III. The Context Management Framework

The context management framework contains two major components (see Figure 1): (i) an agents infrastructure that is used to manage the context representation and the context interactions and (ii) a set of additional modules and resources used for gathering and representing the context information (the Sensor API and the RAP context model ontology).

The context model management infrastructure contains the following set of agents: Context Model Administrating Agents, Context Interpreting Agents, Request Processing Agents.

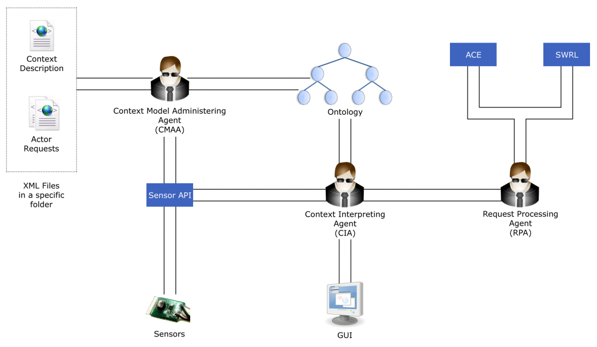

Context Description
Actor Requests
XML Files in a specific folder
Context Model Administering Agent (CMAA)
Ontology
ACE
SWRL
Sensor API
Context Interpreting Agent (CIA)
Request Processing Agent (RPA)
Sensors
GUI

Figure 1. The Context Management Framework Architecture

The *Context Model Administering Agent (CMAA)* is the specific context model manager. It initializes the context model and creates the other mobile agents. All its actions are triggered by outside events. The most important tasks of this agent are: keeping the context model synchronized with the real world and notifying the Context Interpreting Agent (CIA) and the Request Processing Agent (RPA) about events they are interested in. The CMAA agent is the only agent who has access to the real world interface, through which it looks for changes that should be reflected into the virtual world. Synchronization is achieved by creating, updating or deleting context elements from the RAP context model artifacts by considering the real world events. The inter-agent communication is achieved through a message passing technique, where each individual message contains all necessary information describing the event. There are two kinds of messages: CIA agent needs to be notified when the context model changes (i.e. a new context element is added or an existing one is modified or deleted) and RPA agent needs to receive messages with data regarding incoming requests.

The *Context Interpreting Agent (CIA)* semantically evaluates the information of a RAP specific context model instance and finds the instance meaning for the pervasive application. This agent represents the connection between the ontology individuals and the sensor data. Its main task is to periodically gather data from the sensors using the Sensor API and update the corresponding individuals from the ontology. The semantic value of a context instance is determined as a unique hyper-point in the hyper-space by projecting all the values of the resources from the context instance onto semantic axes. The semantic space and the semantic zones are constructed by the CIA agent using context policies, the context ontology and reasoning algorithms. The semantic values attached to the context instances that will determine the execution of the same actions form groups in the semantic space. From the communication perspective, CIA needs to react to two types of messages: (i) messages received from CMAA meaning that the real world has changed, so the semantic space must change too (e.g. a new sensor is added) and (ii) messages received from RPA meaning that a new request is available and it needs the semantic value of the current context instance in order to create a new action plan.

The *Request Processing Agent (RPA)* is in charge with executing various types of requests made by actors. This agent identifies and generates the action plans that must be executed for serving an incoming request. Upon identifying a valid request, RPA asks CIA for the current instance of the semantic space. RPA uses internal or external inference engines or reasoning algorithms to get the requested information.

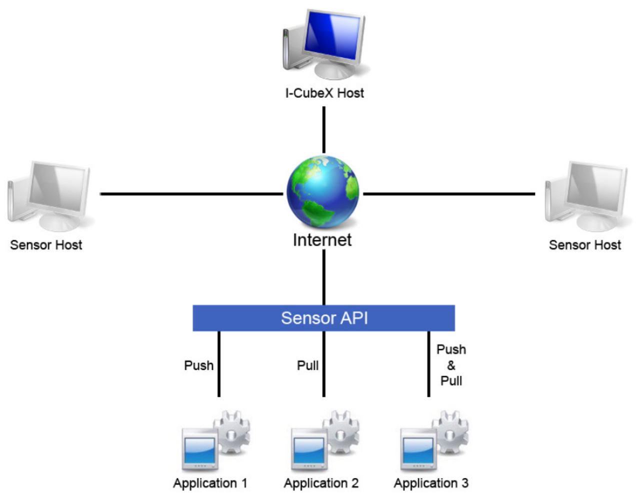


Figure 2. The Sensor API Architecture

The *Sensor API* monitors and captures information from the smart environment. This information is used by context agents for accurately representing the context. Our approach to the context information acquisition process provided by the Sensor API (see Figure 2) is based on defining both push and pull types of sensor information retrieval mechanisms. The push mechanism uses event based listeners to determine when information is available in order to make it visible to the management infrastructure. The pull mechanism is query based, allowing the sensor information to be provided on demand. Sensor information is made visible to the management infrastructure by exposing web services.

The *RAP context model ontology* is used to represent the context information within our management model. The concepts defined in the ontology can be seen as a tree-structure (see Figure 3).

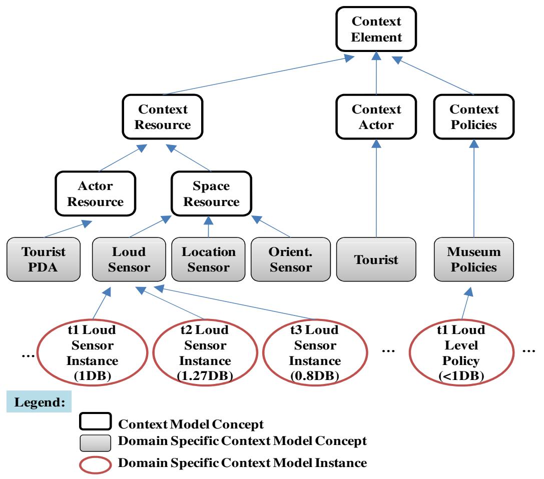


Figure 3. The Context Model Ontology Representation

An important aspect of designing the data structure is adding information precision. This is achieved in our ontology by adding details like: (i) Context source (specifies the service that provides the information), (ii) Quality attribute (provides the accuracy or confidence of the context source), (iii) Metric (defines the data unit of measurement) and (iv) Timestamp (captures temporal relationships). In order to separate resources related to the physical smart environment from those attached to actors, a further classification is made by defining the concepts Space-Resource and Actor-Resource. The Space-Resources need to be specialized according to their role: executing actions (i.e. Actuators) or detecting events (i.e. Sensors).

IV. The Context Management Framework Components Development

The current section details the development and implementation of the proposed agent based context management framework by focusing on the chosen technologies and comunication API's.

The *context management infrastructure agents* (CMAA, RPA and CIA) are implemented using the Java Agent Development Framework platform (JADE) [20]. The JADE framework simplifies the implementation of multi-agent systems through a middleware complying with the FIPA specifications. The agent platform can be distributed across machines which dont need to share the same operating system. The control of the framework is assured via a remote controller with visual overview over the agents, that can migrate from one container to another when required. The context agents tasks are modeled as specific behavior objects. The communication architecture offers flexible and efficient messaging. JADE creates and manages a queue of incoming ACL (Agent Communication Language) messages, private to each agent. The transport mechanism adapts to each situation, by transparently choosing the best available choice. User defined content languages and ontologies can be implemented and registered with agents. This way they can be automatically incorporated by the framework. In our case, we use only a few types of ACL Messages. Data is transported between agents by serialization, e.g. the CMAA sends *OwlIndividualEvent* objects to the CIA signaling a change in the context-model.

The *actor requests* can be specified in two independent ways, using two types of reasoning languages: SWRL (Semantic Web Rule Language) [21] and ACE (Attempto Controlled English) [22]. SWRL defines rules implicating an antecedent (body) and a consequent (head). Both parts of a rule must consist of a number of atoms. SWRL has roots in OWL, used to represent our ontology model, so it does not need a conversion. This is the reason which brings the advantage of achieving the best execution time. ACE is a controlled natural language, i.e. a subset of standard English designed for knowledge representation which allows users to express statements in plain English, with some limitations. The main advantage is that once written, ACE texts can be read and understood by anybody. Because our method of representing the ontology model is not native to ACE, a full conversion must be done before requests can be formulated implying that ACE slows down the execution in favor of an accessible, user-focused representation.

The *Sensor API* module is responsible for retrieving data from the sensors. The communication between the API and the sensors is realized through web services. The API simplifies the data gathering process and allows the information to be manipulated later on (e.g. build complex data structures upon it). The exact location of the sensors is transparent to the framework which treats them uniformly regardless of how they acquire and transmit data, how they connect to a computer, what operating system they support, etc. New types of sensors from various manufacturers can easily be added to the system by only implementing and exposing a web service on the computer that hosts them.

The *RAP context model ontology* development is based on the Protege-OWL API which provides classes and methods to load and save OWL files, to query and manipulate OWL data models and to perform reasoning using the ontology elements like concepts, properties and individuals.

V. Simulation and Results

In order to test our framework we have implemented a *Simulator* with the following features: (i) visualize in real time the ontology tree, with all the individuals and their properties, (ii) generate a 3D representation of the context, including the sensors, their influence zone and their value, (iii) draw a real time plot of the values returned by a set of sensors, (iv) allow the user to submit new requests to the framework and see the results and (v) management features (the agent platform status, memory usage, etc.).

As case study we have used the smart environment represented by our Distributed System Research Laboratory. In the laboratory the students are marked using RFID (Radio-Frequency Identification) tags and identified using a RFID reader. The students interact with the smart laboratory by means of wireless capable PDAs on which different laboratory provided services are executed (submit homework service, print services, information retrieval services, etc.). A sensor network captures information regarding students location or orientation and also ambient information like the temperature or humidity. The DSRL infrastructure contains a set of sensors through which the real context information is collected: two Hot and Humidity sensors that capture the air humidity and the temperature, four Orient sensors placed in the four corners of the laboratory that measure the orientation, one Loud sensor that detects sound loudness level and one Far Reach sensor that measures distances. The sensors are connected using a Wi-microSystem wireless network produced by Infusion Systems Ltd [24].

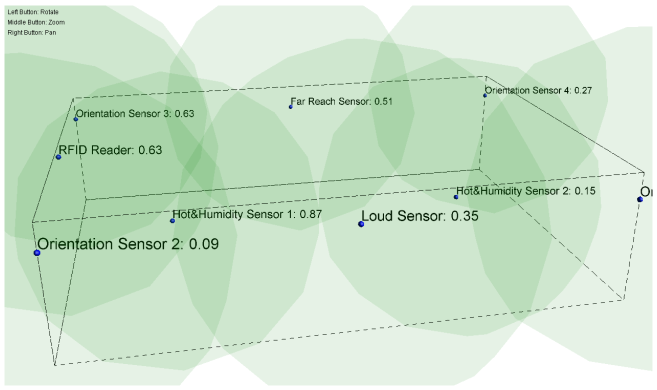
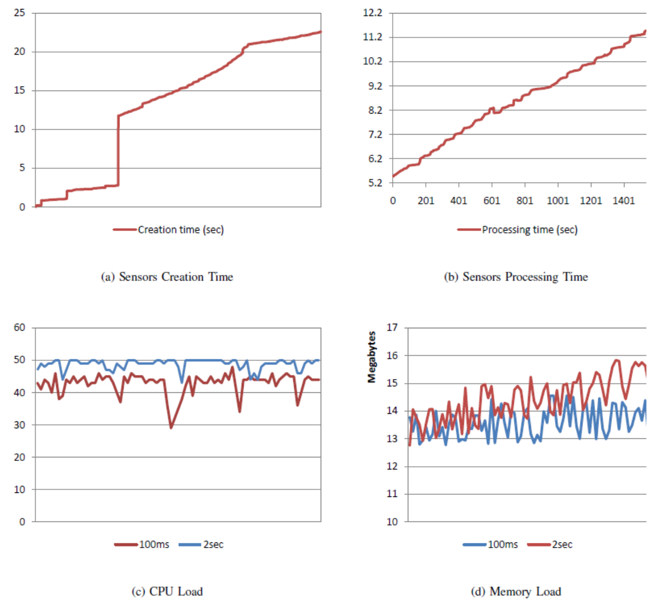

Left Button: Rotate
Middle Button: Zoom
Right Button: Pan
Far Reach Sensor: 0.51
Orientation Sensor 4: 0.27
Orientation Sensor 3: 0.63
RFID Reader: 0.63
Hot&Humidity Sensor 2: 0.15
Hot&Humidity Sensor 1: 0.87
Loud Sensor: 0.35
Orientation Sensor 2: 0.09

Figure 4. The 3D Representation of the DSRL Smart Environment

The middleware is deployed on an IBM Blade-based technology Server. The IBM Blade technology was chosen because its maintenance software offers autonomic features like self-configuring of its hardware resources. The context related data captured by sensors is collected through the Wi-microSystem that has an I-CubeX WimicroDig analog to digital encoder as its most important component. It is a configurable hardware device that encodes up to 8 analog sensor signals to MIDI messages which are transmitted in real-time through Bluetooth waves to the Blade Server for analysis and / or control purposes. The Bluetooth receiver located on the Blade Server is mapped as a Virtual Serial Port (VSP). Figure 4 shows our DSRL laboratory smart environment context in a 3D representation with sensors and their influence zone modeled using the Simulator.

For evaluating our framework capabilities we executed some performance tests (Figure 5). We simulate the sensors by using a web service that returns random numbers and a small program that periodically inserts new entries in the context. Figure 5.a shows how the creation time for a sensors depends on the number of sensors in the system. The creation time was measured from the moment in which CMAA considers the new sensor until all the changes were processed, i.e. the ontology was updated and CIA modified the semantic space. The chart from Figure 5.b displays the relation between the number of sensors and the time needed by the system to perform all the necessary operations when one sensor changes its value. The results show that the processing time grows linearly with the number of sensors in the system. In Figure 5.c and 5.d the load of the system with respect to the rate at which the sensors change their values is shown. We measured the CPU and memory load over one minute period, when the sensors change their values every 100 milliseconds or 2000 milliseconds. As a result, it can be noted that the difference in the system load for the mentioned scenarios is not significant.



(a) Sensors Creation Time

(b) Sensors Processing Time

(c) CPU Load

(d) Memory Load

Figure 5. The Context Management Framework Simulation Results

VI. Conclusion

This paper presents a smart space agent-based context awareness management solution that can be used as a basis for developing complex context-aware applications. The framework deals with two important research challenges: context representation and context management. For representing the context in a programmatic manner, we have used our RAP context model which gives us both set and ontology representations. The RAP context model set based representation is used by the context agents management infrastructure to evaluate the conditions under which the agents should execute the management processes while the ontology based representation is used for reasoning purposes to infer new context related information. To generate and administrate the context model artifacts at run time, we used an agent-based solution due to their mobility, reduced complexity and cross-platform properties. As future development, we intend to enhance the agent-based management framework with self-* autonomic capabilities in order to provide an efficient context awareness self-management solution.

REFERENCES

[1] Z. Maamar, D. Benslimane, and N.C. Narendra, "What can context do for web services", *Communications of the ACM*, Vol. 49 Issue 12, 2006, pp: 98 103, ISSN:0001-0782.

[2] D. Fournier, S.B. Mokhtar, N. Georgantas, and V. Issarny, "Towards ad hoc contextual services for pervasive computing", *Proceedings of the 1st workshop on Middleware for Service Oriented Computing*, Vol. 184, 2006, pp. 36 41, ISBN:1-59593-425-1.

[3] Y. K.Wang, "Context awareness and adaptation in mobile learning", *Proc. of the 2nd IEEE Int. Workshop on Wireless and Mobile Technologies in Education*, pp. 154 158, 2004, ISBN:0-7695-1989-X.

[4] S. Mostefaoui and B. Hirsbrunner, "Context aware service provisioning", *Proceedings of the The IEEE/ACS International Conference on Pervasive Services*, pp. 71 80, 2004, ISBN:0-7695-2535-0.

[5] K.M. Anderson, F.A. Hansen and N.O. Bouvin, "Templates and queries in contextual hypermedia", *In Proc. of the seventeenth conference on Hypertext and hypermedia*, Denmark, pp. 99 110, 2006.

[6] D. Raz, A. Tapani Juhola, J. Serrat-Fernandez and A. Galis, *Fast and Efficient Context-Aware Services*, Wiley Series on Comm. Networking and Distr. Systems, ISBN-13: 978-0470016688, pp. 5 - 25, 2006.

[7] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann and W. Retschitzegger, Context-awareness on mobile devices - the hydrogen approach, *In HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pp. 292, USA, 2003.

[8] I. Salomie, T. Cioara, I. Anghel and M. Dinsoreanu, RAP - A Basic Context Awareness Model, *Proc. Of 4th IEEE Int. Conf. on Intelligent Computer Communication and Processing*, ISBN: 978-1-4244-2673-7, pp. 315 - 318, 2008.

[9] G. Specht and T. Weithoner, Context-Aware Processing of Ontologies in Mobile Environments, *7th Int. Conf. on Mobile Data Manag.*, pp. 86, May 2006.

[10] K.C. Lee, J.H. Kim and J.H. Lee, Implementation of Ontology Based Context-Awareness Framework for Ubiquitous Environment, *Int. Conf. on Multimedia and Ubiquitous Engineering*, pp. 278 282, April 2007.

[11] Faqun Jiang, Jintao Li, Zhenmin ZhuA, "Service Provision Model Based on Context Awareness for Ubiquitous Computing", *Parallel and Distributed Computing, Applications and Technologies*, pp. 155 - 156, 2007, DOI 10.1109/PDCAT.2007.52

[12] Angel Nez, Jacques Noy, "An Event-Based Coordination Model for Context-Aware Applications", *Coordination Models and Languages*, pp. 232 - 248, 2008.

[13] Yang, S.J.H., "Context Aware Ubiquitous Learning Environments for Peer-to-Peer Collaborative Learning", *Educational Technology and Society*, vol. 9, pp. 188 - 201, 2006, ISSN 1436-4522.

[14] A. Zaslavsky, "Mobile agents: can they assist with context awareness?", *Proceedings of the IEEE International Conference on Mobile Data Management*, pp. 304 - 305, 2004, DOI 10.1109/MDM.2004.1263080.

[15] J.I. Hong and J.A. Landay, "An infrastructure approach to context-aware Computing", *Human-Computer Interaction*, vol. 16, pp. 287 - 303, 2001.

[16] S. Yang, B. Lan, and J.Y. Chung, "A new approach for context aware SOA", *Proc. of The 2005 IEEE Int. Conf. on e-Technology, e-Commerce and e-Service*, pp. 438 - 443, 2005, ISBN: 0-7695-2274-2.

[17] J. Pan, Z. Huang, G. Mao, and J. Dong, "A context awareness architecture for mobile learning based on Fuzzy Petri Nets", *16th International Conference on Artificial Reality and Telexistence*, pp. 552 - 557, 2006, DOI 10.1109/ICAT.2006.8.

[18] Mehdi Amoui, Mazeiar Salehie, Siavash Mirarab and Ladan Tahvildari, "Adaptive Action Selection in Autonomic Software Using Reinforcement Learning", *Proc. of the Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08)*, 2008, pp. 175-181.

[19] Anand S. Rao and Michael P. Georgeff, "BDI Agents: from Theory to Practice", *In Proceedings of the First International Conference on Multiagent Systems*, 1995, pp. 312-319.

[20] F. Bellifemine, G. Caire, T. Trucco, and G. Rimassa, "Jade programmers guide", free Library, 2007.

[21] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "Swrl: A semantic web rule language combining owl and ruleml", W3C, 2004.

[22] N.E. Fuchs, U. Schwertel, and R. Schwitter, "Attempto controlled English (ace) language manual, version 3.0", Department of Computer Science, University of Zurich, Tech. Rep. 99.03, August 1999.

[23] Distributed Systems Research Laboratory, Technical University of Cluj-Napoca. *http://dsrl.coned.utcluj.ro*.

[24] Infusion Systems Ltd, *http://www.infusionsystems.com*.