Hybrid Architecture for Pervasive Computing Supported Collaborative Work Application and Simulation

Kahina Hamadache
Orange Labs
Caen, France
Kahina.hamadache@orange-ftgroup.com

Luigi Lancieri
LIFL, University Lille 1
Lille, France
Luigi.lancieri@univ-lille1.fr

***Abstract*—In this paper we present an hybrid architecture for a computer supported collaborative work in pervasive computing environment simulator. This architecture supports pure simulation mode where all agents are simulated, real deployment where it seamlessly integrates pervasive computing to the computer supported collaborative work and finally an hybrid mode where real devices can interact with simulated ones. This specific architecture provides an original framework to develop and evaluate complex behaviours in pervasive computing environments by allowing scenario to be played in a semi-simulated way and avoid some of the constraints and difficulties of real conditions testing. In addition to theses obvious advantages it also provides a reliable base to capture and analyse real devices behaviour in pervasive computing environments.**

Keywords-pervasive computing; collaborative work; CSCW; simulation; hybrid architecture

I. INTRODUCTION

The computer supported collaborative work (CSCW) domain is probably one of the most active research fields of recent years. Indeed, due to the facilitations brought by computers and smart devices it is almost impossible to find people working without them. In this paper, instead of retelling the history of CSCW we will face next challenges and propose an original architecture based on promising perspectives.

Among the recent technologies and paradigms, one has a special interest for us: the pervasive computing. This concept describes the "simple" idea that devices of user's environment should be able to communicate and interact to adapt their behavior to user's needs.

Given this aspect our work has rapidly focused on the way we could integrate the pervasive computing within CSCW. Such integration could bring various advantages: resource and time saving for companies, work simplification and task automation for workers. In a "green" consideration it could also help reducing work's energetic impact by accompanying users' in using lighter devices and services.

On the long road toward this accomplishment we have already sowed some seeds. Hence as we will describe in the next sections we have proposed the *PCSCW* model (see below) which is designed to natively support pervasive computing for collaborative tasks. To validate this model and develop pervasive computing behaviors and rules we have defined the bases for a pervasive computing simulator relying on the *PCSCW* model. We made the choice to develop a simulator as it was the most efficient and reusable way to evaluate our model and could save us a useful time.

In this paper we present our next step on the road: an architecture to support both real and simulated devices. As we will show this specific approach provides us interesting aspects in term of evaluation and promising perspectives for the development of pervasive computing behaviors.

As we will go down in the depths of our proposed architecture we need to start the reminder of this paper by a presentation of our *PCSCW* model and technologies used to implement it and its simulation aspect. Then we will detail our hybrid architecture and talk about some interesting related works. We conclude by giving some promising perspectives.

II. PCSCW MODEL & SIMULATION

As our research interest has focused on the integration of the pervasive computing aspect in the computer supported collaborative work, we have proposed in a previous work [1] an original model which aims at making smart devices cooperate seamlessly to improve and facilitate the collaboration of users. This model, named PCSCW for Pervasive Computing Supported Collaborative Work, relies on some simple but essential "sub-models":

- A *Task Model* composed of mainly two concepts:
  - *Task*: represents a meaningful process to be performed by one or more users to achieve a specific goal, for instance "creating a webpage", can be composed of a set of subtasks or actions;
  - *Action*: describes an atomic step of a task, it has no discriminatory meaning as it can't be understood outside of a task. To illustrate it we can consider the action "opening a web browser" that doesn't convey any specific meaning but can be integrated in tasks such as "searching the web" or "checking mails".
- A *Role Model* built above the task model, it extends it by providing one more concept and some refinements about tasks:
  - *Role*: it defines a role to be played by one or more users by wrapping tasks into subsets: mandatory, allowed and forbidden tasks;

- Tasks can require one or more roles to be performed. Thus a single task may be shared among several roles and then becomes a "Collaborative Task".

- A *Resource Model* providing a common ground to represent:
  - *Required resources*: the set of resources required by an action to be performed. By describing these requirements in term of software, hardware, human and social resources at the action level we can efficiently describe resources required for a given task;
  - *Available resources*: the set of resources available in user's environment, it provides a structured representation of the context;
  - Device: the representation of a contextual device is merely a part of available resources but with a particular extent as it is considered as an active agent of the collaboration.

In addition to these sub-models the PCSCW Model includes *Device Collaboration Rules*. The main principle of these rules is the following: by comparison of resources required to perform a task or an action with available devices' resources we can trigger specific interactions between devices to make them cooperate to finally provide all required resources to the user. Going a little further these rules can even perform whole actions or tasks and prevent users from doing repetitive and thoughtless ones.

All these features create a model allowing smart devices of users' context to automatically and seamlessly cooperate to facilitate, channel and enhance the collaboration of users.

In order to validate this model we're currently developing a pervasive computing simulator based on a multi-agent system and an implementation of the PCSCW model. The purpose of this simulator is not only the validation of the model but also to develop devices collaboration rules. Then in addition to a tool to validate our model it also has to be considered as a laboratory to explore pervasive computing interactions and devices cooperation patterns.

In the next section we give an overview of the basic simulator's architecture and technologies used to develop it.

III. PURE SIMULATION ARCHITECTURE AND JXTA

The simulator we are developing relies on some choices and principles formulated according to its requirements. The first requirement is to represent a multi-agent system, to manage it we decided to build it with the well known JADE Framework [7] (Java Agent DEvelopment Framework). Second requirement: the simulation itself is based on the unfolding of collaboration scenario composed of an initial state and a set of scheduled events. The representation of such scenarios is ensured by an ontology that stores all information about it.

Third requirement we need to stick with the PCSCW Model thus we have to represent humans, devices, their interactions, other base concepts of the PCSCW model and allow devices to manipulate them. Then each human and device of the simulation is represented by its own JADE agent while interactions between agents are handled through the JADE agent messaging system.

Fourth requirement, as we were dealing with a simulation we needed to represent the environment in which the simulation runs. We decided not to represent it with a JADE agent but with a simple Java object dispatching, listening and integrating events.

Still, as agents haven't a perfect perception of their entire context, the fifth requirement is for them to store their own knowledge of the world individually.

To facilitate the representation of information within the simulator and to be able to reason efficiently over context information and as we already had good experiences using it in previous projects ([2] and [3]) we decided to rely on the Protégé [12] Framework developed at the Stanford University to manipulate the ontology and store it with the OWL [11] language. As we already mentioned, all information of a given scenario is stored in an ontology. It also implies that JADE agents are built according to their OWL representation and that their relative knowledge is extracted from scenario's data.

Sixth requirement, the need to represent devices collaboration rules was another point encouraging us to store information with OWL. Indeed, dealing with devices collaboration rules is merely reasoning over context information. In this perspective OWL has the perfect companion as the SWRL [13] language. SWRL provides a simple way to infer information on an ontology by writing rules made of a set of conditions to be checked and a set of consequences if conditions are met. For its simplicity and its efficient integration and use with OWL we chose to represent devices collaboration rules as SWRL Rules.

In addition to these "classical" multi-agent features we incorporate a behavior manager to each agent. This module has to handle the current behavior but also the start of new behaviors according to evolving context, changes due to an agent or to the environment. Besides these considerations, we had some constraints to handle: support for roles, actions, resources, human, groups and interactions.

Another technology we need to evoke is the JXTA peer-to-peer project [9] which was unveiled by Sun on April 25, 2001. The goal of JXTA is to provide a "general-purpose" network programming and computing infrastructure. JXTA framework provide a set of protocols and a series of services that let peers find each other, form groups and directly exchange messages. JXTA provides most of the basic services to build a P2P network. What you have to do is to develop advances and specific applications. One of the main advantages of JXTA is the fact than it aims to be network independent, meaning than you can connect a mobile device using mobile network and a computer using high bandwidth network. JXTA is based on three keys concepts: JXTA ID (which identifies every resource in the network), Peer (which can be any entity able to interpret JXTA protocols), Message (which is the basic unit of communication between peers, can be binary or XML), Peer Groups (which are self-organized team of peers, it can bet set up dynamically) and Pipes (which are virtual communication channel between

JXTA service and applications). Figure 1 gives an overview of JXTA's architecture.

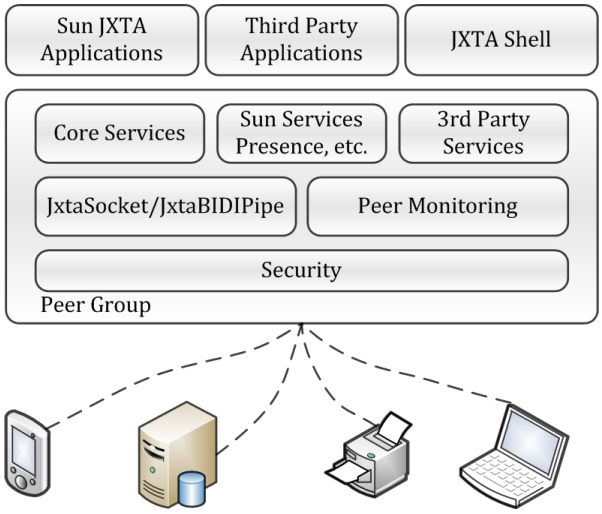

Figure 1. JXTA Architecture

As we'll describe in the next sections, we use JXTA to support and simplify real agents' communication and collaboration.

IV. HYBRID ARCHITECTURE

The evolution we propose for our simulator is to integrate real devices into the simulation and allow them to interact not only with other real devices but also with simulated ones. Even if it seems interesting and promising this approach requires some modifications of the existing simulator's and agents' design to be deployable. In this perspective we'll now describe our view of an hybrid architecture for a multi-agent pervasive computing supported collaborative work simulator allowing real and simulated agents to interact and run a collaboration scenario.

Figure 2 depicts the architecture of an hybrid agent running deployed PCSCW services and interacting with JADE simulation agents. The architecture itself is organized in three main layers:

- *PCSCW Core*: this layer is the real core of the agent, it is composed of the following components:
  - *Knowledge Manager*: as its name suggests, this first brick manages agents' knowledge. As we evoked in the previous section agents have two distinct but related sets of knowledge: the representation of their own characteristics and known context (as OWL triplets), and device collaboration rules (as SWRL rules);
  - Knowledge Interface: built on the top of the Knowledge Manager, this component works as a façade for all other components to access, wrap, update and delete agent knowledge;
  - Behavior Manager: last component of the PCSCW Core, it provides reasoning capabilities to the agent and task management. The task manager has in charge to handle the process of a task and to maintain the coherence of agent's behavior during its accomplishment. It avoids starting a new task implying conflicts with the current ones. The reasoner acts directly with semantic rules and cooperates with the task manager to know if a task can be or should be started or to know the current step of tasks. The combination of these two components allows agents to know how to adapt their behavior in the current context and then how to cooperate with other agents.

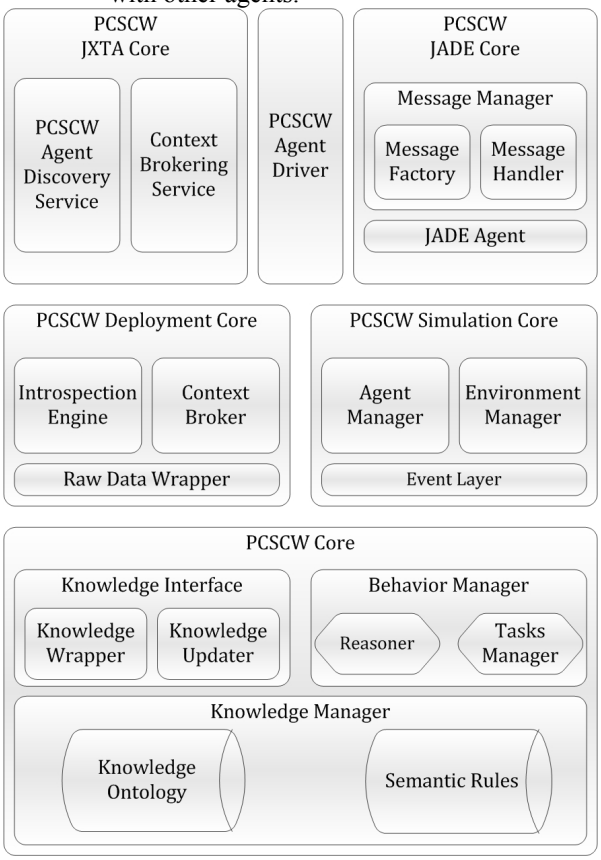

Figure 2. PCSCW Hybrid Agent Architecture

- *PCSCW Deployment Core*: this layer of the architecture contains base elements to deploy and use the PCSCW model on a real device. Indeed, even if the PCSCW Core is still required to manipulate agent's knowledge, it is not sufficient for a real deployment. Then, to be able to deploy it we need the following components:
  - *Context Broker*: this module has in charge to handle the collection of context data which are not directly available to the agent. For instance, it provides common services to exchange context information with other agents;
  - Introspection Engine: this component relies on information miners and collectors to give to the agent information about itself. Indeed, it is not

always obvious for a device to have access to its own characteristics, and it may require analysis, approximation and interpretation.

- *PCSCW Simulation Core*: this layer has the buildings blocks to enable the creation of a simulated agent. The reason why this part of the architecture is on the same level as the deployment core is that it shares a common goal: dealing with context information sources and feeding the *Knowledge Manager*. In this perspective, it is based on three modules:
  - *Agent Manager*: as we want to simulate an agent, we need a representation of this agent. Thus the purpose of this module is to represent an agent and furnish a shell to encapsulate and manipulate PCSCW features;
  - Environment Manager: as we have already evoked, the simulation uses a single object to represent the "environment" in which the simulation takes place. It stores, dispatches and updates information about the simulation. Information can be sent to agents when they "ask" for it. For instance when a device looks for wifi access points, it makes a request to the environment for access points in range. Then, given the agent's position, the environment can communicate to the asking device the set of wifi networks in range with their relative signal strength. Updates are done when an agent modifies its characteristics and notifies the environment of these changes;
  - *Event Layer*: to channel agent-environment interaction we've got an e*vent layer* which provides event listeners, event emitters and event factory. This part of the *simulation core* allows the environment to dispatch events to agents and agents to communicate their internal events to the environment. Here it is important to distinguish the event layer from the environment manager: environment manager handles and processes events inside the agent, while the event layer conveys them.
- *PCSCW JADE Core*: laying on top of the *simulation core* this part of the architecture represents how the JADE Framework is integrated within the simulator. It has not been integrated directly inside the simulation core as we want to leave us the capacity to implement a different version of the simulator using another framework or even a homemade multi-agent system.
  - *JADE Agent*: as each human and device of the collaboration is represented with an agent, we naturally used the JADE Agent class to represent them. This aspect of the agent comes in combination with the Agent Manager of the simulation core without replacing it;
  - *Message Manager*: the basic bricks of interaction between JADE agents are messages. In order to keep this mechanism we needed a component to handle messages. Then, the role of the message manager is to receive agents' messages, analyze and process them, but also to create messages for other agents (through the *message factory*) and send them.
- *PCSCW JXTA Core*: this "top" level layer contains JXTA services to facilitate the collaboration of devices and provides simple way to exchange context information. We've got two main services at this level:
  - *Context Brokering Service*: a service to simplify access to agent's context information and retrieval, it relies on the *deployment core* and improve it;
  - *PCSCW Agent Discovery Service*: similar to classical agent discovery, this module enables a device to find other PCSCW Agents (devices) in order to be able to cooperate with them; this specific part of the layer is particularly interesting as it can bring new devices and then new agents in the simulation.
- *PCSCW Agent Driver*: to enable the hybrid architecture we do not simply need to integrate real devices information and knowledge to the simulation, but we also need to allow real and simulated agents to communicate and interact. For us the natural way to do it is to integrate deployment and simulation features. Then, each real device in addition to the *PCSCW deployment core* has the PCSCW *simulation core.* As we described earlier in our implementation of the simulation we have used the JADE Framework; then to be able to communicate with other JADE agents (the simulated ones) we decided that each devices taking part in the simulation would have to use its own part of the JADE framework. In this perspective, a device will need to create its own JADE agent and make it communicate with the rest of the simulation. This can be done thanks to the capacity of JADE to interconnect multiple agents' containers. Still it is not sufficient to ensure a proper functioning and interactions of all agents in a scenario. Indeed, we still require making the real agent and its simulated alter ego interact, that's why we introduce the notion of *Agent Driver*. This part of the architecture provides to a real agent the necessary and sufficient methods and services to manipulate its JADE representation. This module is critical as the JADE agent is the entry point of the device to the simulation and has to give it the possibility to communicate with the simulated environment through the event layer and other agents through the messaging system.

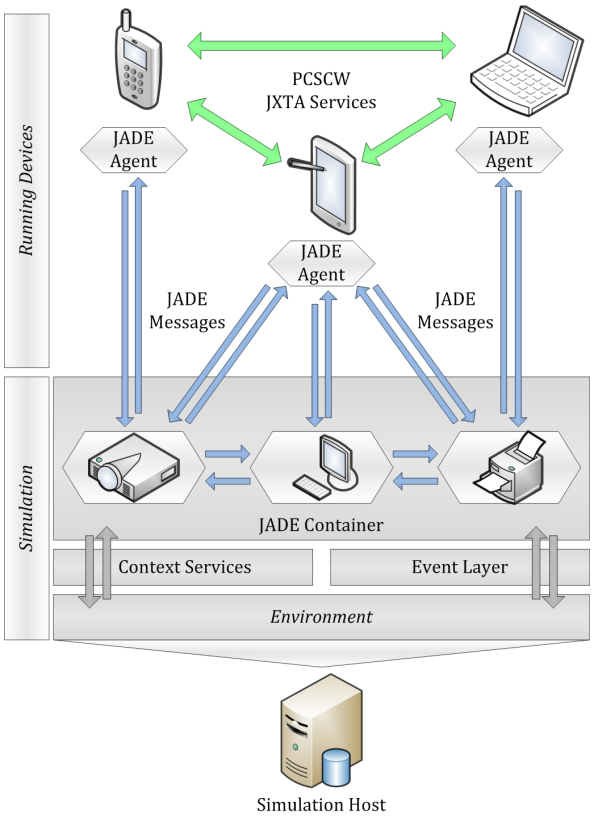

Figure 3. Agent's Interactions

Figure 3 summarizes and depicts how the various agents interact. At the bottom of the figure we've got the machine hosting the simulation, with the previously detailed environment and the JADE agents' container.

Interactions between these two parts of the simulation are ensured by the Event Layer and the Context Services. Interactions between JADE agents are provided by the JADE agent messaging system and the message manager of each agent. Interactions between "real" devices are channel through JXTA services. Finally, interactions between simulated and real devices are performed by making real devices driven agents interact with pure simulated ones.

V. RELATED WORK

Several researchers have investigated areas related to the model and design of pervasive computing systems. Simulation modeling for pervasive computing systems is a growing domain, motivated by the need for testing pervasive systems before they are deployed.

Huebscher et al [6] provide the design of a simulation model of contexts as a mean to test the context-logic of self-aware applications. This is achieved by allowing sensor data to be produced from a description of contexts, providing a highly flexible and configurable solution.

Miraoui et al [10] present the modelling and the simulation of multi-agent service oriented architecture for pervasive computing system. Their architecture is composed of two multi-agent systems: 1- the PCS multi-agent system composed of central agent of each device and 2- the device multi-agent system composed of the central agent and the service agents of the device. The PCS multi-agent system ensures the communication between devices by sending sensed contextual information to other central agent of the PCS and receiving acknowledgements.

Jouve et al [8] propose DiaSim, a simulator for pervasive computing applications. DiaSim is parameterized with respect to a description of a pervasive computing environment, relevant to a given area. This description automatically generates an emulation layer to run the application code unchanged. Also, a simulation programming framework is generated to allow the development of the simulation logic for primitive services. A graphical environment is provided to the user to define a simulated space, simulation scenarios, and to monitor and debug a simulated pervasive computing system.

As we can see there are already some really interesting an efficient researches about simulation of pervasive computing applications and services. However, only few of them focus on the collaboration between users. Besides, as far as we know, examples of hybrid architectures mixing simulation and real deployment are really scarce.

VI. DISCUSSION

From the CSCW evaluation point of view the hybrid architecture of this simulator has a specific interest if we put it in perspective with recent researches. Indeed, if we refer to [4] and [5] we know that evaluation of CSCW systems has to be organized in three phases:

1) *Formative laboratory*: to avoid main errors and misconceptions;
2) *Field methods*: with a part of users' context.
3) *Qualitative in real conditions*.

Considering these steps the hybrid architecture provides a single framework to conduct the evaluation of pervasive services. Indeed, we can handle laboratory evaluation with a pure simulation of agents, field methods by adding some real devices to the simulation and finally real conditions by replacing all simulated agents with real ones. This possibility can save resources and time by allowing simulations and "semi-simulations" to point out lacks and flaws at an earlier step of the development process.

From the pervasive computing perspective, the hybrid architecture we propose stands on the border of two worlds, simulated environments and real deployed applications. By putting a bridge between them we try to make them benefit from each other. We know that pervasive computing is a recent domain and we can't really imagine what "computing" will mean in a few decades. At present time smart devices are poorly linked and it almost always requires a human intervention. Hence developing new ways to make smart devices interact and cooperate automatically is a more and more interesting and promising domain. Considering this aspect our architecture proposes a real framework to develop such behaviors. Going a little further, an important point to notice is a new perspective brought to the simulator by the introduction of real devices. Indeed in the case of pure simulations, all types of agents are already known and their possible actions and constraints are well defined. By

enabling the possibility to interact with any kind of real device we extend the functioning of the simulator to make it accept completely the open world assumption. Hence, it is necessary for the simulation *Environment* and simulated agents to be able to update their knowledge and handle unexpected agents, resources and behaviors. Obviously we can't pretend that simulated agents will fully interact with real ones as there are still some insurmountable issues when you deal with a completely open world. However a really interesting perspective we found during our work on this architecture could give us a way to have a better management of unexpected and "exotic" devices. This perspective is the possible extension of the system to capture and trace behavior of a real device using the PCSCW model. This acquisition would provide us rich and contextualized data from which we would be able to extract knowledge about devices interactions and behaviors in pervasive computing environment. Thereby we should be able to integrate some of the new behaviors of these devices in our simulations. In addition to this integration of new behaviors and concepts, it would provide us useful information to refine, enhance and enrich the quality of simulation scenarios and simulated agents.

On the long range we can even imagine a more sophisticated system which could do more than just manage and simulate new devices behavior. Indeed we think that the analysis of new behaviors put in perspective with the PCSCW model could give us key information to allow the dynamic creation of new behaviors for devices. Besides, the hybrid architecture we propose would be a reliable and efficient way for devices to "automatically" evaluate the efficiency of generated behaviors by running their own simulations without the intervention of humans. In conclusion, if we think a little further ahead, this work could be a small brick toward the real breakthrough in ambient intelligence, where devices would be able to seamlessly communicate, interact, collaborate, learn from their context, learn from each other and optimally adapt their behavior.

REFERENCES

[1] K. Hamadache and L. Lancieri, "Role-Based Collaboration Extended to Pervasive Computing", Proceedings of the International Conference on Intelligent Networking and Collaborative Systems, Barcelona, Spain, Nov. 04-06, 2009.

[2] K. Hamadache, P. Manson and L. Lancieri, 'Pervasive services, brainstorming in situation of mobility'. In: 3rd International Conference on Pervasive Computing and Applications. Alexandria, Egypt, pp. 709-714. 2008.

[3] K. Hamadache, E. Bertin, A. Bouchacourt and I. Benyahia, "Context-aware communication services: an ontology based approach", 2nd International Conference on Digital Information Management (ICDIM'07). October, Lyon, France. 2007.

[4] K. Hamadache and L. Lancieri, 'Strategies and Taxonomy, Tailoring your CSCW Evaluation'. In: Nelson Baloian and Benjamim Fonseca (Eds): CRIWG 2009, LNCS 5784, pp. 206-221 © Springer-Verlag Berlin Heidelberg. 2009.

[5] V. Herskovic., J. A. Pino, S. F. Ochoa and P. Antunes, "Evaluation Methods for Groupware Systems". In: J.M. Haake, S.F. Ochoa, and A. Cechich (Eds.): CRIWG 2007, LNCS 4715, pp. 328–336. © Springer-Verlag Berlin Heidelberg. 2007.

[6] M. C. Huebscher and J. A. McCann, "Simulation model for self-adaptive applications in pervasive computing". In DEXA '04: Proceedings of the Database and Expert Systems Applications, 15th International Workshop on (DEXA'04), pages 694-698, Washington, DC, USA, 2004. IEEE Computer Society. 2004.

[7] JADE: http://jade.tilab.com/

[8] W. Jouve, J. Bruneau and C. Consel, "DiaSim: A Parameterized Simulator for Pervasive Computing Applications" Proceedings of The Sixth Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous'09), Toronto, Canada, July 2009.

[9] JXTA: https://jxta.dev.java.net/

[10] M. Miraoui, C. Tadj and C. Ben Amar, "Modeling and simulation of a multiagent service oriented architecture for pervasive computing systems". Proceedings of the 2009 Workshop on Middleware for Ubiquitous and Pervasive Systems, pp. 1-6, Dublin, Ireland, 2009.

[11] OWL: http://www.w3.org/TR/owl-features/

[12] Protégé: http://protege.stanford.edu/

[13] SWRL: http://www.w3.org/Submission/SWRL/