A Strategy for Cooperative Reasoning in Ubiquitous Computing Environments[*]

José Viterbo[†]
Pontifícia Universidade Católica
do Rio de Janeiro
R. Marquês de São Vicente, 225
Rio de Janeiro, Brazil
viterbo@inf.puc-rio.br

Markus Endler
Pontifícia Universidade Católica
do Rio de Janeiro
R. Marquês de São Vicente, 225
Rio de Janeiro, Brazil
viterbo@inf.puc-rio.br

ABSTRACT
In ubiquitous computing systems, heterogeneous applications must be capable of responding to dynamic changes in their environments with minimal human interference and strongly relying on context information. Reasoning is necessary in ubiquitous systems mainly for transforming raw context data into meaningful information and for infering new implicit context information that may be relevant for the applications. Besides that, reasoning is fundamental for triggering actions or adaptations according to specific situations described by rules. These rules typically depend on several context variables, which may originate from different distributed sources. Hence, we propose a strategy for cooperative context reasoning of complex situations involving a reasoner for the user side and a reasoner for the ambient side.

Categories and Subject Descriptors
C.2.4 [**Computer-Communication Networks**]: Distributed Systems; I.2.3 [**Deduction and Theorem Proving**]: Inference engines

General Terms
Design

Keywords
Middleware, Ubiquitous Computing, Cooperative Reasoning

1. INTRODUCTION
Ubiquitous computing features the seamless integration of computer systems into the lives of users to provide information and functionalities anytime and anywhere [12]. For that sake, as a fundamental requirement, ubiquitous applications must be capable of automatically responding to dynamic changes in the environment with none or minimal human interference. Therefore, ubiquitous applications are intrinsically context-aware, i.e. they have to strongly rely on context data to trigger adaptations at different levels, such as at communication protocols, middleware services, or the user interface.

In context-aware systems, reasoning is necessary for several purposes. First, it is useful to deal with the intrinsic imperfection and uncertainty of context data [4]. In this way, reasoning allows to detect possible errors, to estimate missing values, to determine the quality and validity of the context data. Second, reasoning may also be used for determining higher-level context information, i.e. to infer new, implicit context information, derived from other context data, and that may be meaningful and relevant for some applications. Third, reasoning is fundamental for triggering actions or adaptations according to specific situations, which are described by user-provided or learned rules that contain predicates based on different sorts of context variables.

Context reasoning in ubiquitous systems is very complex due to the dynamic, imprecise and ambiguous nature of context information and the need to process large volumes of data collected from a large number of context providers. This complexity is increased by the fact that in some conditions reasoning needs to be performed in a decentralized way involving several entities of the system, e.g. entities representing spaces, devices or users. Decentralization takes the form of physical distribution of computing and sensor devices, context providers and consumers, entities responsible for brokering and reasoning, and applications and users that may potentially engage into spontaneous collaboration. As a totally distributed reasoning is far too complex, this work proposes a strategy and a protocol to perform reasoning based on the cooperation of two parties: the user side, represented by the user and the mobile devices she carries, and the ambient side, represented by the fixed computational infrastructure.

In the next section we discuss some related works. Section 3 presents a motivating scenario. Section 4 describes the ontology and rules used in the knowledge representation. In the Section 5, we present our approach. Finally, Section 6 brings the conclusions about the proposed system.

[*]Work partially funded by CNPq research grants.
[†]Author supported by scholarship CNPq 140441/2005-0.

2. RELATED WORK

Middleware systems that give support to ubiquitous computing environments traditionally adopt a cetralized approach for their reasoning mechanisms. Such is the case for Gaia [6], CoBrA [3] and Semantic Space [10], for example. However, in ubiquitous systems, applications, rules and context information may be fully distributed. Thus in some circumstances a centralized approach may be inefficient and even unfeasible — e.g. if not all context information is available at the node in charge of reasoning.

There are, however, some proposals for distributed reasoning that try to overcome this limitation. A distributed solution for reasoning about context was proposed in [1]. This approach models the entities of an ubiquitous environment as nodes in a P2P system, in which each different node independently collects and processes the available context information. Specifically, it considers nodes that have exclusive knowledge, and that interact with neighbour nodes to exchange context information. The knowledge of each node is expressed in terms of rules, and knowledge is imported from other nodes through bridging rules. As each peer may not have direct access to all sources of information, they share their knowledge through messages with their neighbour nodes. The proposed reasoning algorithm (P2P-DR) models and reasons with potential conflicts that may arise during the integration of the distributed knowledge.

Distributed Reasoning Architecture for a Galaxy of Ontologies (DRAGO) is another approach for distributed reasoning [7]. It aggregates a web of ontologies distributed over a peer-to-peer network in which each participant may contain a set of different ontologies describing specific domains of interest (for example, ontologies describing different activities of users in a university). These ontologies may differ from a subjective perspective and level of granularity. At each peer there are also semantic mappings defining semantic relations between entities belonging to two different ontologies (described using C-OWL [2]). As the mappings establish a correlation between the local ontology and ontologies of other peers, a peer may also request reasoning services for other peers as part of a distributed reasoning task. The reasoning process may compare concepts in different ontologies to check concept satisfiability, determining if a concept subsumes the other (i.e. the latter is more specific than the former), based on the semantic mappings relating both ontologies. DRAGO supports three types of rules connecting atomic concepts in two different ontologies: *is equivalent*, *is subsumed* and *subsumes*.

While both P2P-DR and DRAGO tackle the problem of distributed knowledge, reasoning in DRAGO is dedicated solely to check the consistency across several ontologies, build classifications, verify concepts satisfiability and check entailment. In P2P-DR, however, as in our approach, the purpose is reasoning about comprehensive rules, i.e. describing relations involving sets of context variables.

As with P2P-DR and DRAGO, our approach lies in a decentralized reasoning process to cope with the intrinsic distribution of context information in ubiquitous systems. Nevertheless, while both works model the participant entities as equivalent peers, we model the interacting entities into two different categories, the user side and the ambient side, considering their different features, e.g. the resource-limited portable devices used at the user side. Therefore, by proposing that reasoning be performed cooperatively by these two different sides, we overcome some complexity limitations imposed by a fully distributed approach found in these two works.

3. SCENARIO

As a typical scenario to exemplify our approach, we consider a fictitious conference on Ubiquitous Computing where several researchers from different universities and companies in the world gather to present and discuss their most recent works. We called it *UbiConference*. It is divided in several technical sessions on subjects such as *Middleware*, *Smart Spaces* etc, and panel sessions on detached subjects (e.g. *Privacy*). It also comprises workshops on specific subjects as *Ubiquitous Systems in Healthcare* or *Context Modeling and Reasoning*.

Professor Silva is a lecturer and researcher affiliated with the Informatics Department of PUC-Rio. He is also participating in the next UbiConference in different ways: (a) he is a member of the Programme Committee; (b) he will chair the *Middleware* session; (c) he will present a paper in the *Context Modeling and Reasoning* workshop; and, of course, (d) he will also be a general attendee of others sessions in the event.

The conference takes place in a convention center with several meeting rooms equiped with a middleware infrastructure to support the organizing team and the attendees with ubiquitous services. An application called Conference Management Service (CMS) is part of this infrastructure and aims to provide context-aware functionalities. When registering at the event, Silva is asked to provide detailed information about his affiliation and research interests. He also downloads and intalls in his notebook and smart phone the Conference Companion (CC), a ubiquitous distributed application. This application not only helps him with his agenda during the event, but also helps to locate and recommend people with similar interests, thus stimulating the collaboration and social interaction with other researchers attending the event.

Let us assume that Silva agreed to provide his location and personal preferences to CMS, in order to take full advantdge of its recommendation and collaboration services. When he arrives at UbiConference, the CMS detects the presence of his device and registers it. By then, CC receives the updated schedule of sessions of that day, allowing Silva to set his preferences indicating which ones he would like to attend. Silva then configures CC (1) to ring an alarm whenever he is outside the room of a session that he wants to attend and it is about to start.

4. KNOWLEDGE REPRESENTATION

While users carrying mobile devices walk through different spaces and organizations, applications executing on their devices may have to interact with different ambient services, each having access to different context informations, such as exemplified in the scenario of Section 3. In[9] we proposed an ontology for describing context information involving not

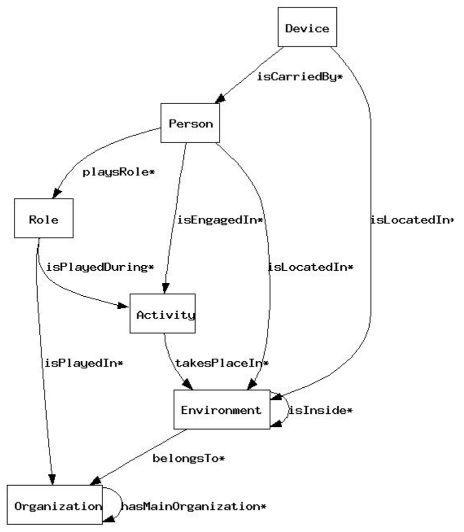


Figure 1: The context generic ontology

only aspects of the physical environment and computational resources but also social aspects, such as organizational features, roles, personal preferences and activities.

4.1 Context Ontology

The use of ontologies to represent context information in a ubiquitous system has not only the advantadge of enabling the reuse and sharing of common knowledge among several applications [8], but also of allowing the use of logic reasoning mechanisms to deduce high-level contextual information [11]. Our approach extends the support for ubiquitous computing so as to take into account not only the physical space (e.g., the modeling of rooms, locations) and the availability of resources (e.g., battery charge or network bandwidth), but also the social context [9], describing organizational aspects (e.g, a company, a department), roles (e.g. a manager), personal preferences (e.g. the preferred light intensity in a class room) and activities (e.g., a meeting). This ontology considers six basic concepts: Person, Device, Environment, Organization, Role and Activity, that represent separate contextual scopes. The Environment describes physical spaces, places such as buildings or rooms, and subclasses of Environment may describe specific kinds of spaces that are common to different organizations, such as a classroom, for example. Device describes the characteristcs of the computational devices. Its mandatory subclasses are Mobile Device, which may comprise subclasses such as PDA, Smartphone, etc, and Fixed Device, that may describe a stationary host. The Organization describes some social structure or institution, like a university os a company, that may have as subclass a department, for example. Role describes some social or professional function attached to a given individual while Person describes the personal characteristcs and preferences of an individual. Finally, Activity describes individual or group tasks in which a person may be engaged.

In each case, for describing a particular domain, the generic ontology has to be instantiated, with the definition of specific individuals for that domain, and may be extended with the definition of some appropriate subclasses. In our scenario, for example, Conference Center, Room A and Room B would be instances of Environment. We can think of University and Conference as subclasses of Organization, and PUC-Rio as an instance of subclass University while Ubi-Conference would be an instance of subclass Conference. Session would be a subclass of Activity and Middleware Session an instance of Session. Chair would be a subclass of Role and Middleware Session Chair an instance of Chair, and so on.

Figure 2(a) shows the entities and relations of the scenario described in Section 3 according to the adopted ontology. It shows Professor Silva (Person instance), a Lecturer (Role instance) in the Informatics department of PUC-Rio (Organization instance) and also a Programme Comitee Member (Role instance) at the UbiConference (Organization instance). In the Middleware Session (Activity instance), which is taking place in Room A (Environment instance), Silva is the Chair (Role instance). He carries with him his smart phone and his notebook (Device instances). In Fig. 2(b) we see the detailed description of Panel Session 1, which has Privacy as Subject, but where the target audience comprises also researchers that are interested in Security issues.

4.2 Inference Rules

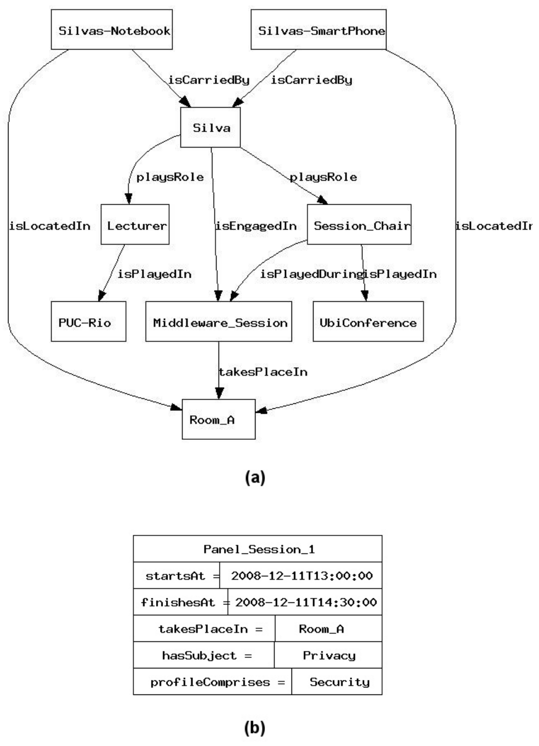
For representing the inference rules, we choose the Semantic Web Rule Language (SWRL) because, as it uses description logics, it allows a straightforward description of information derived from classes and properties of an ontology. Besides, it allows the definition of variables and has some pre-defined expressions (built-ins) that are particularly useful for describing context conditions [5]. SWRL allows users to write rules to reason about OWL individuals and to infer new knowledge about those individuals. It extends the set of OWL axioms to include Horn-like rules, that can then be combined with an OWL knowledge base. The proposed rules take the form of an implication between an antecedent (body) and a consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent holds, then the conditions specified in the consequent must also hold. Both the antecedent (body) and consequent (head) consist of zero or more atoms. Multiple atoms are treated as a conjunction. Atoms in these rules can be of the form C(x), P(x,y), sameAs(x,y) or different-From(x,y), where C is an OWL description, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values. SWRL does not support negated atoms or disjunction. Using a standard logic notation, in the form *"antecedent"* ⇒ *"consequent"*, an example of a rule asserting that when "Silva" is outside "Room A" it implies in Situation_A would be written as:

isLocated("Silva",?y) ∧ differentFrom(?y,"Room A") ⇒ Situation_A

In the above rule, variables are indicated using the standard convention of using a question mark prefix (e.g., ?x).

As a particularly useful feature, SWRL includes a set of built-in predicates that allow to describe the relation of variables with concrete domains, such as integers and strings.

These built-in predicates may occur in the body of a rule and have a predefined logical meaning (a fixed interpretation, such as comparison operators $<$, $>$,$=$,$\neq$, etc) and can be considered as dynamically evaluated predicates.



Panel_Session_1	
startsAt =	2008-12-11T13:00:00
finishesAt =	2008-12-11T14:30:00
takesPlaceIn =	Room_A
hasSubject =	Privacy
profileComprises =	Security

Figure 2: Ontology instances representing the proposed scenario

5. PROPOSED APPROACH

We are developing a middleware architecture to provide an infrastructure to create innovative context-aware applications that include mobile devices and mutiple context provider devices in ubiquitous environments. This architecture is intended as a configurable framework in which users can decide what services they want to enable in their environments. It is composed by a bottom layer, which responsible for providing context information and device discovery; an intermediary layer offering support for semantic interoperability; and a topmost layer that provides application specific and ambient services. In this third layer we are implementing our Cooperative Reasoning Service (CRS), which supports ubiquitous applications in triggering actions or adaptations based on situations described by rules.

5.1 The Cooperative Reasoning Service

Typically, for applications in ubiquitous systems, actions or adaptations are triggered when specific situations take place. These situations are described as a combination of different conditions based on context variablesevaluated by different (and distributed) sources. Therefore, a middleware for ubiquitous systems should provide a mechanism capable of inferring or detecting such complex situations when required by applications. We propose a *Cooperative Reasoning Service* (CRS) to infer and disseminate higher-level context information. It aggregates the context information obtained from context providers and reasons about rules submitted by applications that are being executed either on the users' devices or on the ambient infrastructure (e.g. an ambient service like the CMS of Section 3).

In a ubiquitous environment, not always all context information is available both for the users' mobile devices and for the ambient infrastructure. For several reasons, ranging from privacy to performance issues, some information may be available only on the users' side and some only on the ambient's side. Hence, the CRS is composed of two types of agents: one executing on the portable device — in a "trimmed" version —, and the other executing on a server of the infrastructure, where both agents run a protocol to perform the cooperative reasoning.

To perform the reasoning process, we need to have these two sides — the user side and the ambient side — cooperating according to a specific strategy. For explaining our approach, let us consider, for example, the situation *(1)* described in Section 3, in which Silva wants to be notified when a session is about to start and he is outside of the respective room. Such situation may be described by the inference rule:

Situation_A:

$$\textbf{wantsToAttend}(\text{“Silva”},?s) \wedge \textbf{isLocatedIn}(\text{“Silva”},?y) \wedge \\ \textbf{takesPlace}(?s,?x) \wedge \textbf{differentFrom}(?y,?x) \wedge \\ \textbf{isAboutToStart}(?s)$$

We can notice that the context information necessary for detecting this situation comprises data related to the user's preferences (sessions that Silva "wants to attend"), data collected from the device (where he "is located") and information about the event (room where a session "takes place" and the fact that it "is about to start"). While the first two pieces of information are originated at the mobile device, the latter two pieces are most likely managed by the ambient infrastructure.

To reason about the proposed rules, we devise three possible patterns of interaction: (a) the reasoning being performed solely by the device's cooperative reasoning agent (CRS/D) on the user side, (b) solely by the ambient cooperative reasoning agent (CRS/A) on the ambient side, or (c) cooperatively by both agents.

User Side Reasoning.. Applications running on the mobile devices may be interested in situations in which all involved context data is available in the device. In the previous example, assuming that the infrastructure sends notifications for all devices when a session is about to start, then the desired situation could be inferred solely by the CRS/D at the device. In this case, however, the reasoning computation may be too heavy to be performed by the resource-limited mobile device. In fact, an inference is more efficiently executed at the same device only when the corresponding rules refer exclusively to context information collected at the device itself, and not data provided by the ambient.

Ambient Side Reasoning.. As already mentioned, reasoning about situations whose descriptions involve data collected from ambient sensors or data stored in large OWL files is far too expensive to be executed on resource-limited mobile devices. If all data concerning the users and devices is also available at the ambient infrastructure, in a centralized way, however, the reasoning operation would be more efficiently executed at the CRS/A in the fixed infrastructure. Revisiting our previous example, let us assume that Silva provides his preferences and that his device periodically sends its location to the infrastructure, then Situation_A could be inferred by the ambient side. Privacy issues, however, may prevent the user from providing personal information such as his preferences or location for the ambient.

Cooperative Reasoning.. A specific condition may be described by a rule $R(C_1..C_n)$ — as the one presented before — that involve context variables that refer to data distributed over the devices and the ambient infrastructure. For the reasons that we previously discussed, in this case reasoning should not be performed solely by the device or by the ambient infrastructure, but rather executed cooperatively, involving both agents CRS/D and CRS/A. This reasoning strategy can be described by the algorithm shown in Fig. 3.

```
ON RECEIVING A NEW RULE (rule $R$)
1   parses $R$ identifying the split $R = <R_{in} + R_{out}>$
2   if $R_{out} \neq \emptyset$ then
3       Evaluates $R_{in}$ assining values to the variables
4       Subscribes at remote reasoner by forwarding $R_{out}$
5   else Check ($R_{in}$)
ON CONTEXT UPDATE (context_variable $C$)
6   for each $R_k$ which depends on $C$ do
7       if $R_{out} \neq \emptyset$ then
8           evaluates $R_k$ solving the variables
9           updates variables values for $R_{out}$
10      else Check ($R_k$)
ON RECEIVING UPDATE FROM PEER REASONER (rule* $R$, context_values $V[\,]$)
11  updates variables values in $R$
12  Check (*$R$)
ON RECEIVING NOTIFICATION FROM PEER REASONER (rule* $R$)
13  Check (*$R$)
CHECK (rule* $R$)
14  if $R$ is TRUE then
15      Notify (client)
16  else keeps $R$ in the rules database
```

Figure 3: Algorithm describing the reasoning strategy.

The interaction between the ubiquitous applications and the reasoning services for performing the cooperative reasoning may start at the ambient side (CRS/A) or at the user side (CRS/D), depending on where the client application is running. In practice, the application will submit the rule to be inferred to the local agent. It will then interpret the rule, and if all needed context information is available at the local knowledge base, it will perform the reasoning operation locally. If not, it will split the original rule in two parts R_{in} and R_{out}, one comprised by atoms that refer only to local context data (R_{in}), and the other comprised by the atoms that refer to remote context data (R_{out}). The local agent will subscribe the remote reasoner asking to evaluate R_{out}. Therefore, before submitting R_{out} to the remote agent, the local agent will evaluate R_{in} to determine values for variables that are solvable locally and have to be provided to the remote reasoner in R_{out}.

Each agent monitors context variables changes and checks if each rule is satisfied whenever applicable. If the remote reasoner verifies that R_{out} is TRUE, it will notify the local reasoner. By his turn, the local agent will notify the application if R_{in} becomes true. On the other hand, changes in context data in the local agent may cause the change of the variable values that were previously fulfilled in R_{out}. If that happens, the local reasoner updates such information with the remote agent. From this, we can conclude that this strategy will only be capable of inferring rules that can be split in two parts in which only the first part (local to the client) forwards results to the second part (the cooperative reasoner), but not the contrary.

For instance, considering the rule for Situation_A presented previously, the client application running on the user side (i.e. the device) would submit the rule to CRS/D. After interpreted, the rule would be decomposed in two parts. The first part would be evaluated and the variable "?y" would be assigned "Room A", while variable "?x" would be assigned "Smart Spaces Session". The following part of the rule would be forwarded to be inferred by the CRS/A:

Situation_A2:

takesPlace(?s,"Smart Spaces Session") ∧ differentFrom("Room A",?x) ∧ isAboutToStart(?s)

Table 1 describes the CRS protocol executed between the two cooperative agents.

Operation	Description
Check (R)	A client application sends a message to CRS agent with a XML rule R describing a situation to be verified. CRS will check the rule, independently or cooperatively (forwarding part of the rule), and reply TRUE or FALSE to the client.
Subscribe (R)	A client application (or a peer reasoner) sends a message to CRS containing a XML rule R, subscribing to be notified whenever R is satisfied.
Update (R, $V_1..V_n$)	A peer reasoner sends a message to CRS containing updated information about values of context variables $V_1..V_n$ that were previously informed in some rule.
Unsubscribe (R)	A client sends a message to CRS removing a subscription previously posted.

Table 1: The protocol executed between the CRS agents.

6. CONCLUSION

In ubiquitous systems, reasoning is necessary for triggering actions or adaptations when pre-defined situations happen. These situations are described by rules that typically depend on several context variables, which may originate from different distributed sources. In such conditions, reasoning may need to be performed in a decentralized way involving several entities of the system. Distributed reasoning has already been proposed in previuos works, but due to the high complexity of this task such reasoners usually perform only very simple inferences.

We proposed a strategy to perform reasoning of complex context rules through a cooperation of a user side and an ambient side agents. In this way, we expect to be able to efficiently execute more complex reasoning tasks. Up to now, we have implemented an ambient side reasoner capable of inferring facts based on SWRL rules. It was developed using the Jena API. The next step is to develop a light-weight reasoner to be executed in resource limited mobile devices and to fully implent the cooperative reasoning strategy and protocol described in this paper.

7. REFERENCES

[1] A. Bikakis and G. Antoniou. Distributed Reasoning with Conflicts in an Ambient Peer-to-Peer Setting. In R. Bergmann, K.-D. Althoff, U. Furbach, and K. Schmid, editors, *Proceedings of the Workshop "Artificial Intelligence Methods for Ambient Intelligence" at the European Conference on Ambient Intelligence (AmI'07)*, pages 25–34, November 2007.

[2] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing Ontologies. In *Proc. of the Second International Semantic Web Conference (ISWC-2003)*, volume 2870 of *Lecture Notes in Computer Science*, pages 164–179. Springer, 2003.

[3] H. Chen, T. Finin, and A. Joshi. A context broker for building smart meeting rooms. In *Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Symposium*, pages 53–60, 2004.

[4] K. Henricksen and J. Indulska. Modelling and using imperfect context information. In *Proc. Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*. IEEE Computer Society, 2004.

[5] 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 member submission, 2004.

[6] M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. A middleware infrastructure for active spaces. *Pervasive Computing Magazine*, 2002.

[7] L. Serafini and A. Tamilin. DRAGO: Distributed Reasoning Architecture for the Semantic Web. In A. Gómez-Pérez and J. Euzenat, editors, *ESWC*, volume 3532 of *Lecture Notes in Computer Science*, pages 361–376. Springer, 2005.

[8] A. Shehzad, H. Q. Ngo, K. A. Pham, and S. Y. Lee. Formal modeling in context aware systems. In *Proceedings of the First International Workshop on Modeling and Retrieval of Context*, September 2004.

[9] J. Viterbo, C. Felicíssimo, J.-P. Briot, M. Endler, and C. Lucena. Applying regulation to ubiquitous computing environments. In *Proceedings of the 2nd Workshop on Software Engineering for Agent-oriented Systems (SEAS 06)*, pages 107–118, Outubro 2006.

[10] X. Wang, J. Dong, C. Chin, S. Hettiarachchi, and D. Zhang. Semantic Space: An Infrastructure for Smart Spaces. *Pervasive Computing*, 3(3):32–39, July-September 2004.

[11] X. Wang, D. Zhang, T. Gu, and H. Pung. Ontology based context modeling and reasoning using OWL. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, pages 18–22, March 2004.

[12] M. Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):94–104, September 1991.