An ontology-based approach to context modeling and reasoning in pervasive computing

Dejene Ejigu — Marian Scuturici — Lionel Brunie
Laboratoire LIRIS-UMR-CNRS 5205, INSA de Lyon,
7 avenue Jean Capelle, 69621 Villeurbanne cedex, France
{dejene.ejigu ; marian.scuturici; lionel.brunie}@insa-lyon.fr

Abstract-*The behavior of pervasive applications should depend not only on their internal state and user interactions but also on the context sensed during their execution. In this paper, we propose an ontology-based generic context management (GCoM) model. The GCoM model facilitates context reasoning by providing structure for contexts, rules and their semantics. Context and context semantics in GCoM model are represented using the upper and the lower level ontology. Rules are represented using ontology compatible rule languages. Even though ontology data has static nature, GCoM model is designed to be dynamic and reusable in multiple domains of pervasive applications where resource limitation is a key issue. Initial prototype of the use of the model is created and the result obtained is promising.*

Keywords: *context modeling, context reasoning, context management, context-aware computing, pervasive computing*

1. Introduction

The emergence of a computing model for mobile ad-hoc networks in pervasive environments, the wide spread of pervasive enabling technologies and the availability of computing enabled handheld appliances like smart phones and personal data assistants make computing more distributed in such a way that computing could be with the user every where and every time. The growth of number of computing devices that interfere with our daily activities in our environment may be frustrating if they are not properly adapted to our situations and if they all require our attention. Hence, context and context awareness are the key components in pervasive computing.

Our conceptual framework that shows the basic elements of a pervasive computing environment is given in Figure 1. *Pervasive environment* as one of the elements is characterized by dynamicity, heterogeneity and ubiquity of users, devices and resources, ad-hoc connection among the devices and existence of hardware and software sensors. *Context modeling* deals with how contexts are collected, organized, represented, stored and presented. *Context awareness* performs reasoning about the context and passes decisions about the actions to be triggered.

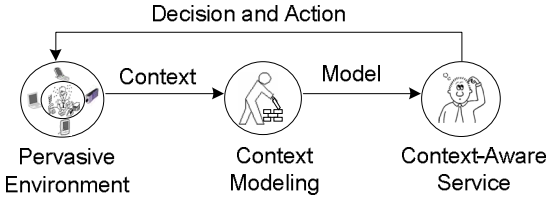

Figure 1. Conceptual framework of pervasiveness

How application programmers can effectively manage and use context information typically in the pervasive environments is still a challenge. Our objective in this work is to propose and investigate ontology based semantically rich, reusable and scalable context management model that supports collaborative reasoning in a multi-domain pervasive context-aware application.

The rest of the paper is organized into the following sections. In section 2, we discuss related works. Section 3 presents our innovative context model. Section 4 indicates case study on the use of the model. In section 5, we give concluding remarks and prospective.

2. Related works

Context-aware computing has been introduced as a key feature in different projects over the last decade and many works have been done so far that demonstrate the importance of context awareness in pervasive computing. Earlier works like the CoolTown [1] focus on the development of application specific context-aware systems.

Henricksen et al [2], [3] introduce a reusable context model and is used in the software engineering process for programming context-aware pervasive systems. It can be enhanced to support semantic reasoning if used with the ontology approach. CoBrA-ONT [4] is

Proceedings of the Fifth Annual IEEE International Conference
on Pervasive Computing and Communications Workshops(PerComW'07)
0-7695-2788-4/07 $20.00 © 2007 IEEE

architecture that enables distributed agents to control the access of their personal information in a context-aware environment. It provides a context model based on semantic web approach but depends on the assumption that there always exists a context-broker server that is known by all the participants. Other similar works include CONON [5] and CSCP [6]. CONON is based on ontology for reasoning and representation of contexts and CSCP is based on resource description framework for representation and manipulation of context data.

Strang et al. [7] present a survey of six context modeling approaches: Key-value modeling, markup scheme modeling, object oriented modeling, graphical modeling, logic based modeling and ontology based modeling approaches. Their analysis favors ontology based context modeling.

In this paper, we propose a comprehensive data independent ontology based semantically rich context management model that insures reusability of context resources and reasoning axioms and rules.

3. Context modeling

Computational entities in pervasive environments need to be context-aware so that they can adapt themselves to changing situations. This requires domain independent context models for context representation, context management and semantic interoperability. In this section, we show our ontology based approach to generic context modeling.

3.1 What is context?

The most widely referenced definition of context is given by Dey et al [8] and states that context *is "Any information that can be used to characterize the situation of an entity. An entity is a user, a place, or a physical or computational object that is considered relevant to the interaction between a user and an application, including the user and application themselves."* Using Dey's definition and our conception about context in relation to its descriptors, we consider the term context as an operational term whose definition depends not only on the inherent characteristics of the entity but also on the interpretation of the operations involved on an entity at a particular time and space.

We classify source of context into computing entity classes. This classification is important in our context modeling process where context representation depends on these entities and the relationships created between them. These entities can be organized into hierarchy (Figure 2) where the root of the hierarchy is the term *ContextEntity*. The listing of generic context entities as sub entities of the root term *ContextEntity* indicates that all descriptors have some properties to inherit from the root. The lower sub classification indicates domain dependant views of context where each component can be defined depending on the specific domain of application (hospital, home, car, tourism, etc).

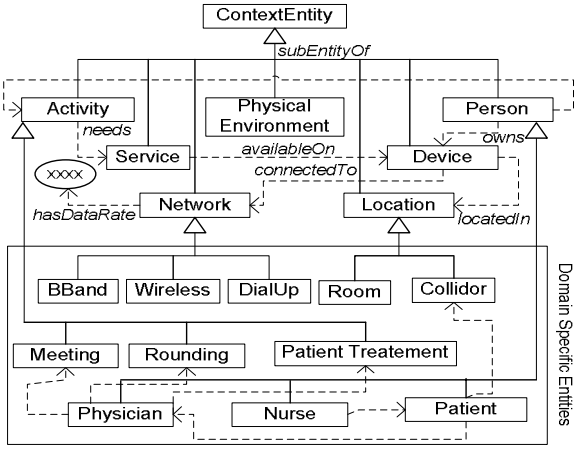

Figure 2. Hierarchy of context entities and relations

Generic context entities in the figure include person, device, network, physical environment, activity, location and service. Personal entity provides contexts like person's identity, address, service preference, device ownership, activity, location, etc. Device entity provides contexts like hardware properties, software properties, display properties, device capabilities, etc. Network entity contexts are expressed in terms of properties like delay and error characteristics, data rate, transport protocols, etc. Physical environment entity provides contexts like illumination, noise level, humidity, temperature, etc. Activity entity contexts tell if an activity is scheduled or not, if it needs special location or not, type of the activity, starting time, etc. Location entity provides contexts about its containment and situation with respect to other entities. Service entity provides context about where the service is available, type of the service, who is permitted to use the service, etc. This listing of entities can by no means be a complete list and therefore we need to have a scalable model to accommodate additions of new components.

The primary characteristic of a context is that it possesses an actor or a *subject*. The type and value of the context is expressed in terms of multiple properties. In our subsequent discussion, we use the terms *predicate* and *object* to represent the situation of the subject with respect to a specific property. This convention goes with the RDF-triple naming style which we intend to use for modeling context using Ontology. This gives the basic RDF triple <*subject predicate object*>. Additional metadata about the basic context triple can also be included as part of the context data.

Proceedings of the Fifth Annual IEEE International Conference
on Pervasive Computing and Communications Workshops(PerComW'07)
0-7695-2788-4/07 $20.00 © 2007 IEEE

3.2 Using reification in context modeling

In addition to the subject, predicate and object triples, context modeling requires context attributes like source, time, place, validity, claims, doubts, proofs, etc. to describe the context itself and to extend the context model towards probabilistic, or confidence-carrying models. Such attributes are meaningful only when thought of as referring to a particular instance of the triple, not to each individual element. To describe this situation we use the method called RDF reification [9]. A reified RDF data contains each original statement as a resource and the other additional statements made about it. The four properties used to model the original statement as the RDF resource are: *subject, predicate, object and type*. A new resource with these four properties represents the original statement and can be used as the subject or object of other statements with additional statements made about it.

We can demonstrate this using an example triple statement: *"Bob is located in the Library"*. This statement can be reified by additional meta-information like *"is reported by sensor #5", "has accuracy of 88%", "has occurred at 11:40 today",* etc. Figure 3 shows an abridged example of RDF data model for this context metadata. The RDF/OWL reification principle can, therefore, be used to represent additional context attributes to the basic context triple.

```
//Reification
<ns:Bob ns:isA ns:Student/> //original statement
<ns:Bob ns:isLocatedIn ns:Library/> //original statement
<ns:Library ns:willBeClosedAt "11:50"/> //original statement
<ns:XX rdf:type resource=rdf:Statement/> //reification starts
  <ns:XX rdf:subject resource= ns:Bob/>
  <ns:XX rdf:predicate resource=ns:isLocatedIn/>
  <ns:XX rdf:object resource= ns:Library/> //reification ends
<ns:XX ns:isReportedBy "Sensor#5"/> // using reified XX
<ns:XX ns:hasTimeStamp "1140"/> // using reified XX
<ns:XX ns:hasAccuracyOf "88%"/>// using reified XX
```

Figure 3. RDF data model on context reification

3.3 The need for semantic context model

Considering the situation of staff members' (Ben, Dan and Rita) tea break scenario in Table 1, a simple query (*select Subject from table.context where predicate= "isLocatedIn" and Object= "Room-305"*) selects "Ben" as an output. But in reality, if the information in the table is given to a human assistant who knows, by common sense, that the terms "Office" and "Room" are synonymous in the domain of interest, s/he will respond "Ben" and "Rita" to the query. Moreover, a human assistant can also deduce that Ben and Rita are now together. But incorporating such semantic interpretation of data using standard database schema is not a straight forward task.

Table 1. Scenario for demonstration of semantics

Subject	Predicate	Object	Time
Ben	isLocatedIn	Room-305	2006022310
Dan	isLocatedIn	Room-301	2006022310
Rita	isLocatedIn	Office-305	2006022310
…	…	…	…

This simple example demonstrates the need for a context model that describes concepts, concept hierarchies and their relationships.

We chose OWL for our context modeling for several reasons. It is a W3C recommendation that employs web standards for information representation such as RDF and XML Schema. It also provides a high degree of inference making by providing additional vocabulary along with a formal semantics to define classes, properties, relations and axioms. For the class concepts, *Office* and *Room,* in the above table, for example, we can use the *owl:sameAs* property that defines them as the same concepts. Similarly, the property concepts, *together* and *coLocatedWith,* can also be defined as the same concepts using OWL as follows:

```
//similarity between classes and properties
<rdf:Description rdf:about= "#Office">
  <owl:sameAs rdf: resource = "#Room">
</rdf:Description>
<rdf:Description rdf:about= "#together">
  <owl:sameAs rdf: resource = "#coLocatedWith">
</rdf:Description>
```

We can also define the concept that co*LocatedWith* is symmetric, which means if "*X is coLocatedWith Y*" then we can say that *"Y is coLocated with X"*.

We can enhance the potential of semantic context reasoning using additional user defined rules like, for example, *"if* $user_1$ *is located in a roomN and* $user_2$ *is also located in roomN then conclude that* they *are coLocatedWith each other or according to the above similarity definition they are together"*. This rule can be represented as follows using a generic rule language:

```
//User defined Rule
  [ruleR: ?user1 nsp:locatedIn ?roomN)
  (?user2 nsp:locatedIn ?roomN)
  ->?user1 nsp:coLocatedWith ?user2)]
```

3.4 Ontology based context management model

We now present our ontology based approach for modeling context and context management. The expressive power, hierarchical organization, formality, standard, support for efficient reasoning, support for programming abstraction and interoperability are among the attractive features of ontology in context modeling. As partly demonstrated in our earlier paper [10], hierarchy of ontology classes are used to represent context entities, concept hierarchies and relationships.

For capturing, interpretation, representation and management of context data, we propose a Generic Context Management (GCoM) model. GCoM model consists of three basic components (Figure 4); context ontology, context data and context related rules.

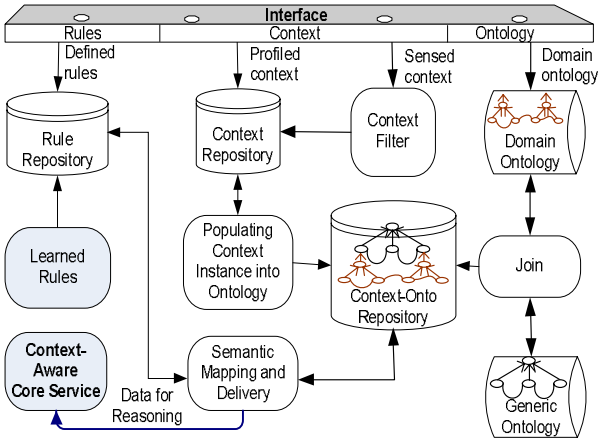

Figure 4. Components of the GCoM model

Ontology represents semantics, concepts and relationships in the context data. It is formed by the merger of ontology that describes domain independent generic contexts and domain specific contexts. *Context data* represents instances of context that may exist in the form of profiled data stored on a disk file or in the form of context instances obtained from the sensors. *Rules* represent derivation axioms that are used by context-aware systems to reason out and derive decisions about the actions that follow. These rules have two sources; rules that are explicitly given by the users through the user interface and rules that are implicitly learnt by the system itself. *Semantic mapping and delivery* module is responsible for mapping and conversion between rules and context-onto so as to deliver a data that is ready for reasoning.

Demonstration on the GCoM model components can be given using a cell phone ringing tone management service example based on a scenario of a university regulation. To comply with the regulation, students must have their cell phones set to non disturbing modes during different activities: attending lectures, consultation with their professors, in libraries, etc. Students therefore need to have their phones automatically switched to silent mode or vibrating mode while in the library, attending lectures, or discussing with their professors and switch back to ringing mode when they are engaged in none of these activities. They also like to use a decent ringing tone when in the vicinity of the campus and a musical ringing tone when outside the campus.

Figure 5 is OWL representation of part of the ontology for the phone ringing management scenario.

Persistent data about static contexts (e.g. ownership relationship of persons to devices like telephone) can be stored in any standard database format that can be selectively populated as context instances into the ontology structure at runtime. Sensed context is to be communicated to GCoM using RDF/XML triple representation format and is then converted to the indicated representation (Figure 6) to make the data ready for reasoning using the Jena generic rule language.

Ontology representation

```
<rdf:RDF …….
  <owl:Class rdf:ID="Student">
    <rdfs:subClassOf> <owl:Class rdf:ID="User"/> </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Library">
    <rdfs:subClassOf> <owl:Class rdf:about="#Location"/> </rdfs:subClassOf>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="ownedBy">
    <rdfs:range rdf:resource="#User"/>
    <rdfs:domain rdf:resource="#Device"/>
    <rdf:type rdf:resource="http://www.w3.org/.../owl#FunctionalProperty"/>
     <owl:inverseOf> <owl:ObjectProperty rdf:ID="ownerOf"/> </owl:inverseOf>
  </owl:ObjectProperty>
  <Student rdf:ID="Bob">
    <ownerOf>
      <PDA rdf:ID="PDA001">
        <hasScreenSize
            rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Medium
        </hasScreenSize>
      </PDA>
    </ownerOf>
    <ownerOf rdf:resource="#Cellphone001"/>
  </Student>
…...
</rdf:RDF>
```

Figure 5. Part of context ontology for the scenario

Context data representation

Profiled context defined in the Ontology

```
->(Bob sys:type gcom:Student). //type = instance of
->(CellPhone001 sys:type gcom:Phone).
->(PDA001 sys:type gcom:PDA).
->(Bob gcom:owns PDA001).
->(Bob gcom:owns CellPhone001).
->(ClassRoom001 sys:type gcom:ClassRoom).
->(Semantic-Theory sys:type gcom:Class).
```

Case 1: Bob, according to his schedule, has just entered in ClassRoom001 to attend a lecture

```
->((PDA001 gcom:locatedIn ClassRoom001) gcom:hasTime 200603251002).
        //new context
->(Bob gcom:hasSchedule Semantic-Theory).
->(Semantic-Theory gcom:scheduledIn ClassRoom001).
->(Semantic-Theory gcom:startTime 200603251000).
->(Semantic-Theory gcom:endTime 200603251100).
```

Case 2: Bob has finished his activity of the day and is just getting out of the campus.

```
->(PDA001 gcom:locatedIn gcom:OutSideCampus). //new context
```

Case 3: Bob has just entered in the library reading room

```
->(ns:PDA-01 gcom:locatedIn ns:DocINSA). //new context
```

Figure 6. Context representation for the scenario

Domain specific rules for students' explicit wishes in the scenario and context data expressed using Jena generic rule are given in Figure 7.

Rules representation

Rules derived or imported from ontology (implicit rules defined in the ontology)

[OntoRule1: (?a gcom:locatedIn ?b) (?b gcom:locatedIn ?c) -> (?a gcom:locatedIn ?c)] //transitive

[OntoRule2: (?a gcom:ownerOf ?b) -> (?b gcom:ownedBy ?a)] //inverse

…..

Defined Rules

```
[locatedRule:(?device gcom:locatedIn ?location)
         (?device gcom:ownedBy ?person)
         -> (?person gcom:locatedIn ?location)
]
[libraryRule:(?student gcom:locatedIn gcom:Library)
         (?student gcom:owns ?phone)
         -> (?phone "switchMode" "silent")
]
[classRule:(?student gcom:hasSchedule ?class)
        (?class gcom:isScheduledIn ?classRoom)
        (?class gcom:startTime ?t1)
        (?class gcom:endTime ?t2)
        ((?Student gcom:locatedIn  ?classRoom) gcom:hasTime ?t)

        (?t sys:greaterThan ?t1)(?t sys:lessThan ?t2)
        (?student gcom:owns ?phone)
        → (?phone "switchMode" "Vibrating")
]
[meetingRule:(?student gcom:hasSchedule ?meeting)
        (?meeting gcom:scheduledIn ?meetingRoom)
        (?meeting gcom:startTime ?t1)
        (?meeting gcom:endTime ?t2)
        ((?student gcom:locatedIn ?meetingRoom) gcom:hasTime ?t)
        (?t sys:greaterThan ?t1) (?t gcom:lessThan ?t2)
        (?student gcom:owns ?phone)
        →(?phone "switchMode" "Silent")
]
[campusRule:(?student gcom:locatedIn gcom:InCampus)
        (not classRule) (not meetingRule)( not libraryRule)
        //because InCampus subsumes ClassRooms, MeetingRooms and
        Library
        (?student gcom:ownerOf ?phone)
        →(?phone "switchMode" "DecentRingingTone")
        ]
[xcampusRule:(?Student gcom:locatedIn OutSideCampus)
        (?student gcom:owns ?phone)
        → (?phone "switchMode" "MusicRingingTone")
]
```

Figure 7. Rule representation for the scenario

4. Case study on reasoning

Figure 8 shows a context-aware service platform classified into four functional groups; the interface, the basic data source (GCoM model), the core service (context awareness) and the supplementary service.

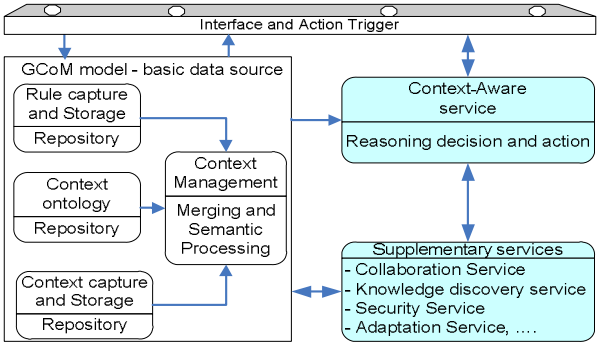

Figure 8. GCoM in a context-aware platform

Interface Manager: Manages a user interface and interface between the platform and other modules specific to domains of applications like context sensors and action triggers.

GCoM model (Basic Data Source): Components in this group are responsible to provide the data necessary to provide proactive or reactive context-aware service. It consists of three basic elements; context capture, context ontology and rule capture. Context capture is the interface to the context sources either in the user interface, context fetching modules, location detection module in our previous work [11] or other devices. Context ontology consists of domain dependant ontology and the generic domain independent ontology combined in to one. Rule capture is an interface to the rule sources.

Rules play an important role in the process of reasoning about contexts. Rules sited in the demonstration example of the campus mobile phone management scenario, in previous section, exist in two forms: implicit rules derived from the ontology and explicit rules defined by the user in the specific domain of application. Table 2 shows an abridged form of some of these rules grouped into two categories: ontology based rules and user defined rules.

Table 2. Sample rules and their category

Rule	Category
(?a property ?b) (?b property ?c) → (?a property ?c) E.g. locatedIn, subClassOf, contains… *(transitive property)*	Ontology
(?a property1 ?b) → (?b property2 ?a) E.g. ownerOf and OwnedBy, locatedIn and contains …*(inverse property)*	
(?a property ?b) → (?b property ?a) *coLocatedWith, friendOf …(symetric)*	
(?device locatedIn ?location) (?device ownedBy ?person) → (?person locatedIn ?location)	User defined
(?student locatedIn Library) (?student owns ?phone) → (?phone "switchMode" "Silent")	

Context-Aware Service: Responsible to provide the core context-aware services like reasoning, decision and action.

Supplementary Service: Consists of knowledge discovery service that adds features to enhance self learning, collaboration service that adds features for collaboration between peers in the neighborhood space, security service, adaptation service, etc.

Figure 9 shows a java code that put together all the major components of the service platform for reasoning, inferences and decisions. A demonstration of the implementation of the reasoning engine in the platform is given using the campus cell phone ringing tone management service scenario. The three GCoM elements use text files named: *cellphone.owl* Figure 5), *cellphone.ctxt* (Figure 6) and *cellphone.rules* (Figure 7).

```
//Code listing for part of reasoning and decision engine in the campus scenario
//imports …
public class OwlReasoner {
 public static void main(String[] args) {
   //Reasoning setup
   List rules = Rule.rulesFromURL("file:cellphone.rules");
   OntModelSpec customInfSpec = new OntModelSpec(OntModelSpec.OWL_MEM);
   GenericRuleReasoner reasoner = new  GenericRuleReasoner(rules,
                                    customInfSpec.getReasonerFactory());
   List context = Rule.rulesFromURL("file:cellphone.ctxt ");
   reasoner.addRules(context);
   customInfSpec.setReasoner(reasoner);
   OntModel model= ModelFactory.createOntologyModel(customInfSpec, null);
   model.read("file:cellphone.owl");
   //Example usage
   String queryString = "PREFIX coca: <http://www.owl-
     ontologies.com/unnamed.owl#> "+
   "SELECT ?phone WHERE {?phone coca:setRingTone coca:Silent.}";
   Query query = QueryFactory.create(queryString) ;
   QueryExecution qexec = QueryExecutionFactory.create(query,model) ;
   ResultSet results = qexec.execSelect() ;
   for ( ; results.hasNext() ; )
   {
     QuerySolution res = results.nextSolution() ;
     RDFNode phone = res.get("phone") ;
     System.out.println("Setting ringing tone of "+ phone +" to silent");
     fireProactiveAction("RingingTone", phone,"silent"); //Module Call
   }
   qexec.close();
 }
}  .. ..
```

Figure 9. Reasoning and decisions using GCoM

After reasoning, we can draw parameters for the action to be taken. In this example, we use the RDQL/SPARQL query tool to draw parameter for the *setRingingTone* action.

5. Conclusions and future work

We have proposed the GCoM model that is based on ontology representation of context data, its semantics that exist as part of context ontology structure and context instances and rules. Rules are either derived or defined by users based on the requirements and policies of a specific application domain.

Breaking down of context into semi-independent components (static and dynamic context instance,, context semantic, context rules) is a unique contribution of GCoM that makes it dynamic and reusable in a pervasive computing environment where individual resource limitation is a key concern. The phone ringing tone management example in the prototype demonstrates how this is achieved.

We are aiming to incorporate and use this context model into our ongoing work on the CoCA service platform [12]. We will also continue to work on some benchmark issues that will help to evaluate the performance of our proposed model and platform, and perform a comparative study with other works in the area. Scalability of this model can be enhanced by separating what we call relevant context data from the entire context data. This may involve some hybrid approach to context modeling that will be considered as a continuation to this work.

6. Reference

[1] Kindberg T., Barton J. “A web-based nomadic computing system”, Computer NW, 35(4):443–456, 2001.

[2] Henricksen K., Indulska J., Rakotonirainy A. “Modeling Context Information in Pervasive Computing Systems”, Proceedings Pervasive 2002 -Zurich August 2002.

[3] Henricksen K. and Indulska J., "Developing context-aware pervasive computing applications: Models and approach," Pervasive and Mobile Computing, Elsevier, 2005.

[4] Chen H., Finin T., Joshi A. “An ontology for context-aware pervasive computing environments”, Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, Acapulco MX, August 2003.

[5] Wang X., Zhang D. Q., Gu T., Pung H. K. “Ontology Based Context Modeling and Reasoning using OWL”, workshop on context modeling and reasoning at IEEE International Conference on Pervasive Computing and Communication , Orlando, Florida, March 2004.

[6] Held A., Buchholz S., Schill A. “Modeling of Context Information for Pervasive Computing Applications”, Proc. of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SC12002), Orlando, FL, USA, Jul 2002.

[7] Strang T., Linnhoff-Popien C. “A Context Modeling Survey”, Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management, Sixth International Conference on UbiComp2004. Nottingham, England, 2004.

[8] Dey A. K., Salber D., Abowd G. D. “A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications”, Context-Aware Computing: A Special Triple Issue of Human-Computer Interaction, Lawrence-Erlbaum March 2002.

[9] Staab S., Erdmann M., Maedche A., Decker S. “An Extensible Approach for Modeling Ontologies in RDF(S)” Proceedings of ECDL 2000 Workshop on the Semantic Web, Lisbon, Portugal, 2000.

[10] Chaari T., Ejigu D., Laforest F., Scuturici M. “Modeling and Using Context in Adapting Applications to Pervasive Environments”, In the Proceedings of the IEEE International Conference on Pervasive Services (ICPS'06), Pages 111-120, Lyon, France, June 2006.

[11] Scuturici M., Ejigu D., “Positioning Support in Pervasive Environments”, In the Proceedings of the IEEE International Conference on Pervasive Services (ICPS'06), Pages 19-26, Lyon, France, June 2006.

[12] Ejigu D., Scuturici M., Lionel B. “CoCA: A Collaborative Context-Aware Service Platform for Pervasive Computing”, IEEE/CS International Conference on Information Technology: New Generations, ITNG 2007, Las Vegas, Nevada, USA (to appear).