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

Dejene Ejigu, Vasile-Marian Scuturici, Lionel Brunie

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 – **A growing interest for pervasive applications and an increasing diversity of pervasive computing devices integrated in our surroundings demand incorporating context-awareness in such applications in order to protect users from being disturbed by such services while on their regular duty. The behavior of these applications should depend not only on their internal state and user interactions but also on the context sensed during their execution. Context and context awareness, therefore, are the key components of pervasive computing so as to perform tasks on behalf of users. In this paper, we propose ontology based reusable context model. The model facilitates the context reasoning by providing structure for contexts, rules and their semantics. Initial prototype of the use of the model in a multi-domain platform 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 assistances 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.

The conceptual framework in our work that shows the basic elements of a pervasive computing environment is given in Figure 1. The arrows running from and to the nodes show the relationship that exists between the elements.

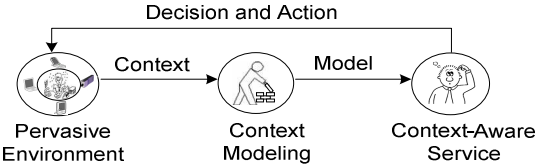

Figure 1-Our conceptual framework showing basic elements of context-aware pervasive computing

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.

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 CoolTown [1] focus on the development of application specific context-aware systems.

Henricksen et al [2], [3] introduce a reusable context model and is use in the software engineering process for programming context-aware pervasive systems. It can be enhanced to support semantic reasoning if used with ontology approach. CoBrA-ONT [4] is architecture to enable 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 neighborhood based and data independent ontology based semantically rich context management model that inures 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 on the interpretation of the operations involved on an entity at a particular time and space rather than the inherent characteristics of the entity.

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. The classes are:

-User context: identity, preference, activity, location…

-Device context: processor speed, screen size, location...

-Application context: version, availability…

-Physical environment context: illumination, humidity…

-Resource context: availability, size, type, etc.

-Network context: minimum speed, maximum speed…

-Location context: contents, where it is subsumed...

- Activity context: start time, end time, actor, etc.

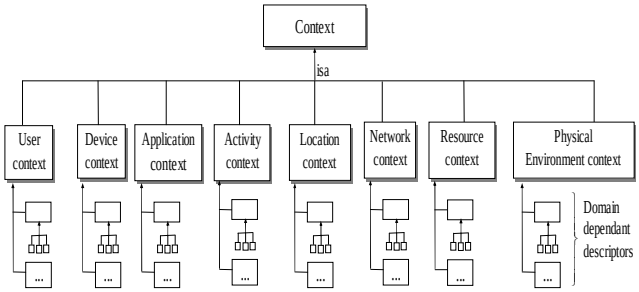

Figure 2-Context entities and a view of their domain dependant components

These entities can be organized into a class hierarchy (Figure 2) where the root of the hierarchy is the term context itself. The listing of basic context entities as subclasses of the root term context indicates that all descriptors have some common 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, truism, 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, therefore, 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 naming convention goes directly 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 context metadata information about the basic triple like time of occurrence, accuracy and source from which the context is captured can also be included as part of the context representation model.

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 applied to the entire reified triple, which are meaningful only when thought of as referring to a particular instance of the triple. To realize this principle and include these parameters into the context model, we need to introduce a higher-order RDF statement that helps us to make statement about another statement. This can be achieved by building a model of the original statement, and this model is a new resource to which we can attach additional properties. This process is called reification [9] and a reified RDF database 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 and have additional statements made about it.

Figure 3 shows an example RDF data model of a context data. The RDF/OWL reification principle is, therefore, an ideal solution to represent additional number of context attributes to the basic context triple.

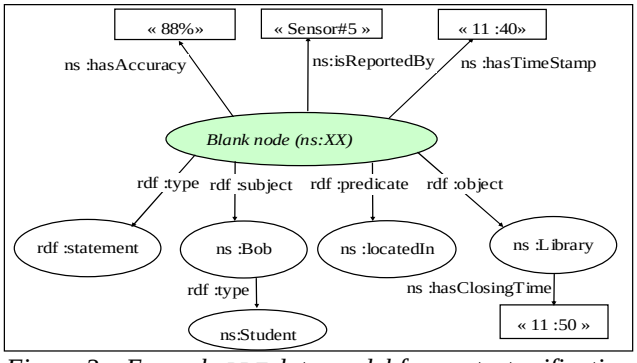

Figure 3 – Example RDF data model for context reification

The *RDF* triple to show reification on this example using abridged RDF/XML syntax can be given as follows.

```
<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
```

3.3 The need for semantically rich context model

Considering the situation of staff members' (Ben, Dan and Rita) tea break scenario in the table below, a simple query (select Subject from context_table 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. In addition to this, a human assistance 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.

Subject	Predicate	Object	Time
Ben	isLocatedIn	Room-305	200602231030
Dan	isLocatedIn	Room-3001	200602231035
Rita	isLocatedIn	Office-305	200602231030
…	…	…	…

This simple example demonstrates the need for a context model that describes concepts, concept hierarchies and their relationships. A web ontology language, OWL, is used when the information contained in documents needs to be processed by applications, as opposed to situations where they are presented to humans as shown in the above query.

We chose OWL for our context modeling due to several reasons. It is a W3C recommendation that employs web standards for information representation such as RDF and XML Schema. OWL allows the necessary semantic interoperability between context-aware systems. 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 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 concepts *together* and *coLocatedWith* can also be defined as the same concepts using OWL as follows:

Similarity

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

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

Similarly, we can 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 and vice versa.

We can also define a rule that states *"if user1 is located in a room and user2 is also located in the same room then conclude that* they *are coLocatedWith each other or according to the above similarity definition they are together"*. This rule can for example be represented using the generic rule languages in Jena (from sourceforge.net) reasoner which we intend to use in our prototype:

```
[rule1: ?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 its 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 **G**eneric **Co**ntext **M**anagement **M**odel (GCoMM). GCoMM (Figure 4) consists of three basic components; context semantics (ontology), context instance data and context related rules.

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 represent instances of contexts. Contexts may exist in the form of stored data on a disk file (context database) or in the form of context instances obtained from the sensors. *Rules* represent derivation axioms that are used by context-aware systems to derive decisions and conclusions 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.

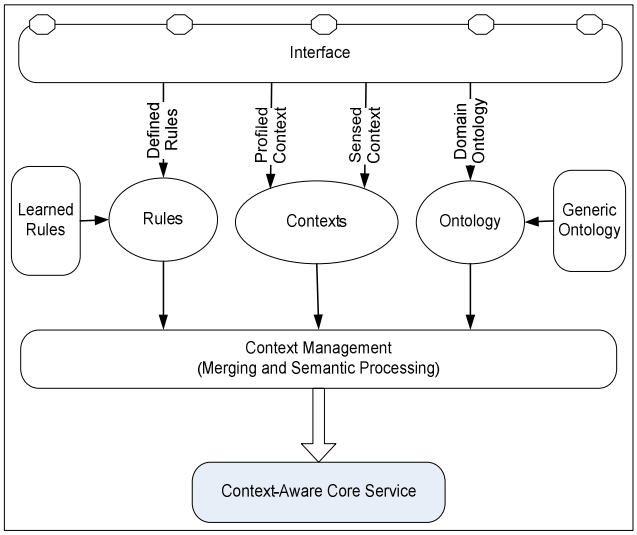

Figure 4-GCoMM structure with its functional components

In GCoMM, the base ontology part is derived from our context descriptors while domain ontology part is dependent on domain specific sub descriptors.

Demonstration on the GCoMM components can be given using a cell phone ringing tone management service example based on the scenario of a university regulation on the use of cell phone for students. 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 would also like to

use a decent ringing tone when in the vicinity of the university campus and a hot musical ringing tone when outside the university campus.

A small portion of the OWL representation of part of the context ontology for the campus telephone ringing tone management scenario is given in Figure 5.

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 campus scenario

Persistent data about static contexts (e.g. ownership relationship of persons to devices like telephone or PDA) can be stored in any standard database format which can be linked by using libraries for the database connectivity and can then be selectively populated as context instances into the ontology structure at runtime. Sensed context is to be communicated to GCoMM using XML or RDF triples representation format and is then converted to the indicated representation (Figure 6) to make the data ready for reasoning, interpretation, aggregation and decision. This representation is the Jena generic rule format.

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).
```

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
->(ClassRoom001 sys:type gcom:ClassRoom).
->(Semantic-Theory sys:type gcom:Class).
->(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 campus scenario

Rules for students' explicit wishes in the scenario and context data expressed again using Jena generic rule (this time rule with preposition) are given in Figure 7.

Rules representation

Rules derived from ontology (just to show what type of implicit rules we have 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 "setRingTone" "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:owns ?phone)
        →(?phone "switchMode" "DecentRingingTone")
        ]
[xcampusRule:(?Student gcom:locatedIn OutSideCampus)
        (?student gcom:owns ?phone)
        → (?phone "switchMode" "MusicRingingTone")
]
```

Figure 7- Rule representation for the campus scenario

4. Case Study on Context Reasoning

Figure 8 shows a context-aware service platform classified into four functional groups; Interface, data source, core service and supplementary service.

Interface Manager: Manages a user interface and interface between the platform and other modules specific to domains of applications. It also hosts action triggering process depending on the specific application domain in which the platform is used.

GCoMM (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 or other devices. It filters and sends useful contexts to the context database. Context Ontology consists of domain dependant ontology and the generic domain independent ontology combined in to one as context ontology. They are fetched into the reasoning engine for further use. Rule capture is an interface to the rule sources either in the user interface or the datamining tools. It keeps the rules in the rule database.

Context-Aware Service: Responsible to provide the core context-aware service after reasoning on the context.

Supplementary Service: Consists of the knowledge discovery service that adds features to enhance self learning, and the collaboration service that adds features for collaboration between peers in the neighborhood space. Components in this group can be extended to

accommodate other services like security and adaptation that are important to enhance the core service.

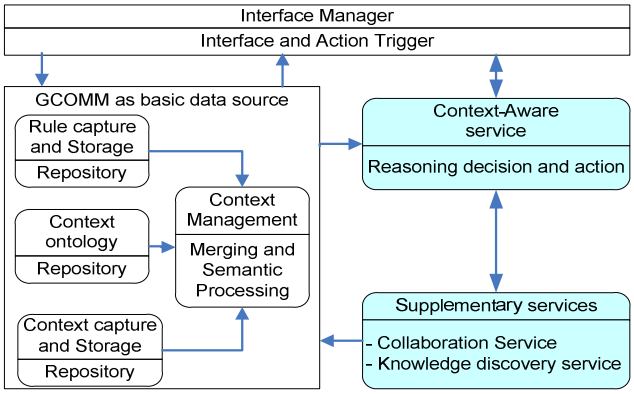

Figure 8-View of GCoMM in a context-aware platform

Figure 9 shows a java code that uses Jena API to put together all the major components of the service platform for reasoning, inferences and decisions using the GCoMM as a source of data. A demonstration of the implementation of the reasoning engine in the platform is given using the campus cell phone ringing tone management service scenario data that is stored on a disk file. Ontology data is represented using OWL while context and rule data can be represented using text files or any other Jena compatible database format like MySQL, PostgresSQL or Oracle. Our example in this scenario uses text files named "cellphone.rules", "cellphone.ctxt" and "cellphone.owl".

```
//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- A portion of code t for reasoning and decisions in a campus scenario based on the GCoMM model

After combining these together by the reasoner, we can draw parameters for the action. 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 ontology based generic context management model, the GCoMM. Initial prototype of the use of GCoMM in a multi-domain context-aware platform is created. The ontology based context model with the parsing and interfacing mechanism of rules and context instances play an important role for reasoning and decisions involved to provide context-aware services.

The run-through example about the campus scenario on cell phone ringing tone management is implemented in the prototype. We have also tested this same module with data from a hospital scenario on patient monitoring and follow up service.

Prototyping modules are developed as independent components and the interface part is missing in this initial implementation. As a continuation to this work, we are aiming to develop a complete context-aware platform that uses the GCoMM. 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.

6. Reference

[1]Kindberg T., Barton J. "A web-based nomadic computing system", Computer Networks, 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 14-18, 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.