Ontology based Context Alignment for Heterogeneous Context Aware Services

Seungkeun Lee

INRIA Rhône-Alpes
Montbonnot Saint-Martin, France
Seung-Keun.Lee@inrialpes.fr

Abstract. In a pervasive environment, context aware services are necessary to enable collaborate and communicate with others in order to provide proper service to users. Each service can be designed with individual context information which represents different perspective over the context. This paper proposes an advanced context management model with context alignment among heterogeneous context aware services. This model enable the interaction between context information producer devices and context information consumer devices and as well as their insertion in an open environment. And this paper show how this model can be used in context aware middleware.

Keywords: Context Aware, Context Alignment, Ubiquitous Computing

1 Introduction

The context aware computing systems are being studied and many prototypes have been implemented [1,2,3,4,5]. The popularity of context aware computing research indicates that systems that can identify the user's context and that of the surrounding environment have the profound potential to provide services that are much more user specific. Context information can be retrieved from a wide diversity of sources such as user profiles, location system, sensors, devices.

However, conventional context modeling approaches based on ontology bear several problems. First, a context awareness service must share the context ontology with the context awareness system in the designing phase. In turn, when a context awareness service is dynamically added to or deleted from the system, the new context awareness service cannot share the context ontology with the system under the circumstances where other context awareness services are not affected. In addition, the problem of context uncertainty, which can arise in the process of deducing the data acquired from the sensor into the context information based on the dynamic modification of the context ontology, must be resolved. This study proposes the dynamic context management by context alignment between the context awareness services.

We have built a context alignment manager for test the proposed model. The designed manager can align between context aware services and manage the alignment relation. This alignment relation is used for context aware service to cooperate with others.

2 Relative Works

In this dynamic pervasive environment, each context manager manages context information of its device. To express its context model, its needs or its capabilities, we use semantic web languages described below. They ensure interoperability between these heterogeneous devices. The ground language for the semantic web is RDF (Resource Description Framework [5]). It enables expressing assertions of the form subject-predicate-object. The strength of RDF is that the names of entities (subjects, predicates or objects) are URIs (the identifiers of the web that can be seen as a generalization of URLs: http://www.w3c.org/sw). This opens the possibility for different RDF documents to refer precisely to an entity (it is reasonable to think that a URI denotes the same thing for all of its users).
The OWL language [6], has been designed for expressing « ontologies » or conceptual models of a domain of knowledge. It constrains the interpretation of RDF graphs concerning this domain. OWL defines classes of objects and predicates and makes it possible to declare constraints applying to them (i.e., that the « output » of a « thermometer » is a « temperature »). The context model that we will use at that stage is very simple: a context is a set of RDF assertions. Interoperability is guaranteed through considering that context-aware devices are consumer and producer of RDF. However, this is not precise enough and devices might want to extract only the relevant information from context sources. For that purpose, a language like RDQL [7] is useful for querying or subscribing to context sources. In order to post the relevant queries to the adequate components, it is necessary that components publish the OWL classes of objects and properties on which they can answer.

3 Context Alignment Model

A context awareness service must be able to exchange context information based on identical understanding of the content among the user, devices and services. This chapter presents a method for dynamically modifying the context information required by the context awareness services operated from the context awareness middleware, as well as the hierarchical context ontology management and context uncertainty resolution methods in order to deliver the changes in the middleware so that the corresponding context information can be received from the middleware.

3.1 Hierarchical Context Ontology

The context information managed by the middleware is constructed into the domain context information that is delivered to all services in the middleware and the individual context information defined for each service. All of the context information is defined by ontology. If a service is newly allocated in the middleware, the middleware must be able to integrate the existing domain context information and the individual context information separately defined in the service for operation. Here,

the correlation between the two sets of context information must be guaranteed, for which purpose the common context information layer is located between the two context information layers. The common context information defines the basic elements required for the context awareness application as the Person, CompEntity, Location, Environment and Activity, and the domain context information and the individual context information are designed by inheriting the common context information. By providing correlation among sets of context information inherited from an identical parent class, the information is used to integrate the two sets of context information in the middleware. Fig 1 displays the hierarchical context ontology designed in this paper.

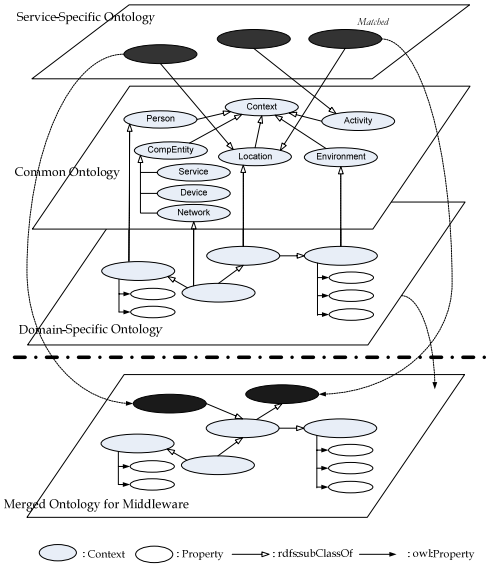

Fig. 1. Hierarchical Context Information

Every type of ontology is inherited from the context class for creation. The common ontology comprises Person, CompEntity, Location and Environment and Activity Class inherited from the highest Context class, as well as Service, Device and Network inherited from CompEntity. Context may include multiple attributes to describe the corresponding situation, as well as other situations as properties. For example, in home network ontology, if there is a Room context inherited from the Location class and a Temperature context inherited from the Environment class, the Temperature context can be used as the property of the Room context.

3.2 Context Alignment

We identify one-to-one context alignment between two context aware services using lexical resemblance between concept names and then inference. The lexical alignment

identifies shared concepts across context information based on lexical similarity between concept names. Both preferred concept names and synonyms are used in the lexical alignment process. Lexical similarity is assessed through exact match. Concepts exhibiting similarity at the lexical level across context are called alignment, as they could be be used as reference concepts in the structural validation and for comparing associative relationship. Additional alignment provided a definition of the alignment structure so as to be able to use and reuse it in various situations. Given two contexts C_1 and C_2, alignments are made of a set of correspondences (called mappings when the relation is oriented) between pairs of (simple or complex) entities e_1, e_2 belonging to C_1 and C_2 respectively. A correspondence is described as a quadruple:

$$< e_1, e_2, R >$$

e_1, e_2 are the entities (e.g., formulas, terms, classes, individuals) between which a relation is asserted by the correspondence. R is the relation, between e_1 and e_2, asserted by the correspondence. For instance, this relation can be a simple set-theoretic relation (applied to entities seen as sets or their interpretation seen as sets), a fuzzy relation, a probabilistic distribution over a complete set of relations, a similarity measure, etc. These relationships are *subClassOf, TransitiveProperty, subPropertyOf,* disjointWith and inverseOf

Having extracted the relations explicitly represented in the ontologies, we then normalize the representation of the relations in each ontology in order to facilitate structural comparisons across contexts. We first complement the hierarchical relations represented explicitly with their inverses as necessary. Implicit semantic relations are then extracted from various combinations of hierarchical relations (inference). Inference generates additional semantic Inference generates additional semantic relations by applying inference rules to the existing relations in order to facilitate the comparison of paths between anchors across ontologies. These inference rules, specific to this alignment, are listed in Table 1.

Table 1. Infence Rule

Relation	Inference Rule
subClassOf	(?A rdfs:subClassOf ?B),(?B ?rdfs:subClassOf ?C) -> (?A rdfs:subClassOf ?C)
TransitiveProperty	(?P rdf:type owl:TransitiveProperty), (?A ?P ?B),(?B ?P ?C) ->(?A ?P ?C)
subPropertyOf	(?A rdfs:subPropertyOf ?B) ∧ (?B rdfs:subPropertyOf ?C) -> (?A rdfs:subPropertyOf ?C)
disjointWith	(?A owl:disjointWith ?B) ∧ (?X rdf:type ?C) ∧ (?Y rdf:type ?D) -> (?X owl:differentFrom ?Y)
inverseOf	(?A owl:inverseOf ?B) ∧ (?A ?X ?Y) -> (?Y ?B ?X)

4 Experiment

This chapter explains the process of conducting tests by implementing a smart home network to evaluate the functions and performance of the context awareness middleware proposed in this paper. The server used for the home network service was IBM eServcer X206, 2.8GHz, 512MB RAM, and it was operated with Windows Server 2003 using the OSGi framework Knopflerfish 1.3.3 and HP Semantic web toolkit Jena2[8,9]. Fig 2 displays the home network environment implemented for the test.

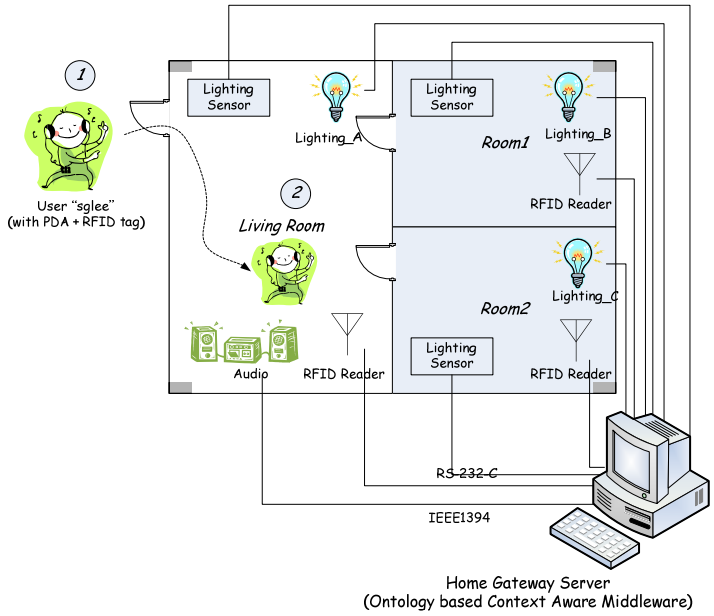

Fig. 2. Prototype of Smart HomeNetwork

The test scenario involved creating context information that is generated when a user comes into the house, and the information was delivered to the context awareness service. The light control service is a context awareness service that automatically turns on the light nearest to the user if the lighting is too dim. The light control service uses the information acquired by the illumination intensity sensor to recognize the brightness of each location. In order to assess the level of illumination in the house, light sensors are installed in Room1, Room2 and LivingRoom.

The sensor in each location detects the light intensity and expresses it in 10 bits, creating a value between 0 and 1023. The context is deduced to be "dim" if the value is below 512, and "bright" otherwise. The light control service defines the independent contexts of the "user location dim" and the "light must be turned on" as in List 1.

List 1. The Rule for Context Inference

```
(?LightSensor locateIn ?place), (?LightSensorValue
bigger 512) -> (?place lightLevel "Bright")

(?LightSensor locateIn ?place), (?LightSensorValue
smaller 512) -> (?place lightLevel "Dim")

(?user locatedIn ?place), (?place lightLevel "Dim") ->
(?place needed "lighting")
```

When the user "sglee" moves from location 1 to 2 in Fig 4, the RFID reader 3 reads the user ID from the RFID tag attached to the user. The RFID reader delivers the data to the RFID management service, which parses the corresponding data to forward the user ID value to the middleware. Then the basic context generator creates the information ("sglee" locatedIn "LivingRoom"). The generated basic context information is delivered to the ontology deduction engine, as well as the light control context service.

The illumination intensity in each room and the living room is regularly checked using the sensors, and the values are delivered to the middleware. According to List 2, the middleware generates the context "bright" for a sensor value acquired in each location greater than 512 and "dim" otherwise. Fig 5 describes the process of deducing the complex context information using the data obtained from the light sensors.

The values acquired by the sensors were ("Room1" lightingValue 284), ("Room2" lightingValue 653) and ("LivingRoom" lightingValue 327), which created the basic contexts ("Room1" lightLevel "Dim"), ("Room2" lightLevel "Bright") and ("LivingRoom" lightLevel "Dim"). The contexts are delivered to the deduction engine, which used the knowledge ("sglee" locatedIn "LivingRoom"), (?user locatedIn ?place) and (?place lightLevel "Dim") -> (?place needed "lighting") to deduce the complex context information ("LivingRoomg" needed "lighting"). The deduced complex context is delivered to the light control service that has registered the context through the event broker.

Since the context awareness service can dynamically register the context ontology, the middleware proposed in this paper has the ontology integration overhead in addition to the time required for loading the ontology to the memory. As a result of measuring the overhead, the integration time was not a significant burden compared to the loading time as indicated in Fig 5. Moreover, the amount of increase in the integration time was not substantial compared to the amount of increase of the domain context ontology value and the developed context ontology value. Therefore, due to the time required for integrating the context information, it can be suggested that the middleware proposed in this study is not adequate for the real-time service platform, but useful for general application services.

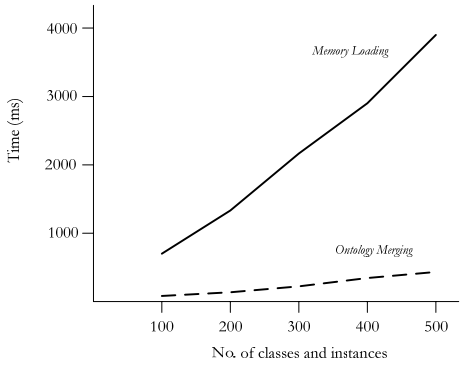

Fig. 3. Overhead of Merging Context Ontology

6 Conclusion

We specifically addressed the problem of adaptability of context management to an ever-evolving world. This is achieved by providing distributed component based architecture and by using semantic web technologies. Components enable the addition, at any moment, of new devices that can provide information about the context of applications. The use of RDF and OWL ensures interoperability between components developed independently by taking advantage of the open character of these technologies. Moreover, using ontology alignment modules allows dealing with the necessary heterogeneity between components. The proposed approach relies on a minimal commitment on basic technologies: RDF, OWL, and some identification protocol. Functions and performance of the middleware were evaluated through the test, and it was confirmed that the overhead of the proposed hierarchical context ontology management model makes the middleware unfit for the hard real-time ubiquitous computing environment. However, it was determined that the model can be applied to the general ubiquitous computing environment. It can be concluded that the ontology based context awareness middleware proposed in this paper can be applied in the service gateway for various ubiquitous environments such as the home network, telematics and smart office for providing context awareness services.

Acknowledgement

This work was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD)" (KRF-2006- D00163)

References

1. S. Lee, J. Lee, “Dynamic Context Aware System for Ubiquitous Computing Environment,” PRIMA 2006, LNCS4088, Springer, 2006.
2. N. Q. Hung, S. Y. Lee, and L. X. Hung, "A Middleware Framework for Context Acquisition in Ubiquitous Computing Systems", Proceedings of the Second International Conference on Computer Applications, 2004.
3. T. Gu, H. K. Pung, and D. Q. Zhang, “An Ontology-based Context Model in Intelligent Environments,” Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, pp. 270-275, 2004.
4. J.Euzenat, J.Pierson and F. Ramparany. A contex information manager for pervasive computing environments, in: Proc. 2nd ECAI workshop on contexts and ontologies (C&O), Riva del Garda (IT), pp25-29, 2006.
5. G. Klyne, J. Carroll, Eds., Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, 2004 http://www.w3.org/TR/rdf-concepts/
6. M. Dean, G. Schreiber Eds, OWL Web Ontology Language: Reference, W3C Recommendation, 2004. http://www.w3.org/TR/owl-ref/
7. A. Seaborne, RDQL — A Query Language for RDF, W3C Member submission, 2004. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/
8. Open Services Gateway Initiative. http://www.osgi.org
9. Knopflerfish. http://www.knopflerfish.org