# Context Modeling and Inference System for Heterogeneous Context Aware Service

Seungkeun Lee

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

**Abstract.** Context can be utilized as an effective source of information for supporting the user system interface under the ubiquitous environments. The context awareness function forms the basis of the studies regarding the ubiquitous computing environment for creating numerous smart spaces, and a context awareness service is usually based on the context awareness system or middleware. However, conventional context modeling approaches based on ontology bear several problems. First, a context awareness service must share the context information with other context awareness services in the designing phase. Second problem is a 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 context management system based on the dynamic context ontology management method, which involves the hierarchical context management method using common context and the paper proposes a method for resolving the context uncertainty problem of interpreting the data acquired from the sensors as two or more types of context information.

**Keywords:** Context Aware, Context Modeling, Ubiquitous Computing.

## 1 Introduction

The context awareness function forms the basis of the studies regarding the ubiquitous computing environment for creating numerous smart spaces, and there are several studies under way in universities and research institutes[1,2,3,4]. A context awareness service is usually based on the context awareness system or middleware. A con-text awareness system has a structure for generating the context information using the data acquired from the sensors and forwarding the data to the context awareness service. The context awareness services provide specific services to the user according to the context information received from the context awareness system. Therefore, the context awareness service must be able to clearly understand the significance of the context information conveyed from the context awareness system. There have been various studies on the techniques of context modeling, and the ontology modeling approach is receiving much attention, thanks to its capability to enable significance exchange among systems.

M.J. Smith, G. Salvendy (Eds.): Human Interface, Part II, HCII 2007, LNCS 4558, pp. 413–422, 2007.

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[5]. 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[6,7,8].

This study proposes the context awareness middleware based on the hierarchical con-text ontology management method, which involves designing of the common ontology hierarchy for the context ontology to be dynamically shared between the context awareness service and the middleware. The context awareness service dynamically defines the context ontology that it needs using the common ontology, which is forwarded to the middleware. The middleware integrates the context ontology received from the context awareness service with the domain ontology that is managed by the middleware for management. The context awareness middleware does not transfer all the deduced context information to all context awareness services, and only the context information that the context awareness services take interests is forwarded. Furthermore, the paper proposes a method for resolving the context uncertainty problem of interpreting the data acquired from the sensors as two or more types of context information. The context awareness middleware is designed using the proposed hierarchical context ontology management approach. The designed middleware uses the data acquired from the sensors to deduce the context information, which is delivered to the requesting context awareness service using the event delivery method. Compared to the existing studies, the designed context awareness system allows dynamic management of the context ontology according to service addition and deletion, and also has the advantage of resolving the context uncertainty issue.

## 2 Hierarchical Context Management 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.

### 2.1 Hierarchical Context Ontology Structure

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. 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.

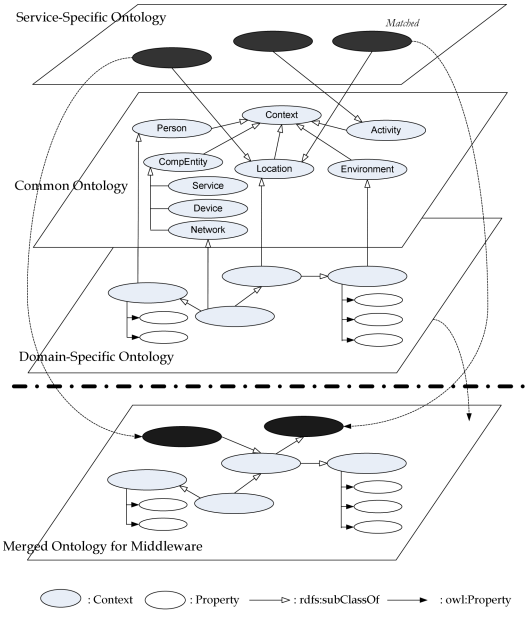


**Fig. 1.** Hierarchical Context Management bundle

The domain ontology author inherits the common ontology to define the context in-formation that is identically analyzed in all services of the middleware. Each context awareness service developer inherits the common ontology and defines the

individual ontology that can be separately interpreted in each service. Here, the context awareness system assesses the correlation of the context ontology between the two ontology layers using the common ontology in the middle layer and manages it.

### 2.2 Uncertainty Problem Management

A context is determined according to whether various attributes collected from the sensors at a specific time are included in a particular realm. For example, if the user's body temperature received from the temperature sensor is from 36.021°C to 36.521°C, the user can be regarded as being in the "healthy state", and in the "ill state" otherwise. Hence, the system must have a method of analyzing the situation based on the data values received from the sensors. Eq 1 expresses the set of attributes delivered at a particular time*(t)* from the sensors linked to the middleware designed in this paper.

$$S(t) = \sum_{i=1}^{n} A_i(t) \tag{1}$$

In Eq 1, $A_i(t)$ indicates the attribute value at a particular time t. Context attributes are used in describing the elements that can be utilized in deducing the situation. Context attributes are linked to virtual or physical sensors. A context space is defined with the range of permissible values of the attributes that can induce a particular situation. A context space is expressed with the attribute values acquired from the sensors that correspond to the predefined situation. Eq 2 expresses the context space.

$$C_i = \sum_{i=1}^{n} S(t) \quad \mathrm{S}(t) = \{A | P(A)\} \tag{2}$$

P is the Predicate function that determines whether the attribute value from the sensor is permissible, and Ai is defined as a set of elements that satisfy the function P. Context(*C*) is expressed as a sum of a particular range of attribute values(*A*) received from the sensors. $A_i$, which is the permissible range of the values received from the sensors, comprises a set of particular values.

Although the permissible values of the attributes and the context information are linked through C(*t*), they cannot be directly mapped to a specific context information through the attribute values delivered from the sensors. This is due to the fact that there can be overlapping regions for $C_A$ describing Context A and $C_B$ for Context B. The following Fig 2 displays the circumstance where two contexts overlap. Since $S_A$ is included in $C_A$ in Fig 2, it can be mapped to Context A. Since $S_B$ is not included in $C_A$ or $C_B$, it cannot be mapped to any context. On the other hand, $S_B$ is located in the overlapping region of $C_A$ and $C_B$. In this case, since the attribute values recognized by the sensors are located in more than one context space, a filtering process is necessary to determine which context space it belongs to.

If the set of attribute values received from the sensors is included in various context spaces as in Eq 3, they are compared with the inclusive context spaces to deduce an adequate context. In order to do so, this paper defines a context space

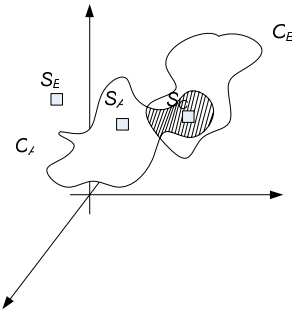


**Fig. 2.** The Overlap of Context Information

variation function to determine the difference among the attribute values and the context space. Using the space variation function, the variation values are mapped to the small context space. The context space variation function is defined as in Eq 3.

$$ContextSpaceDerivation = \sqrt{\sum (A_i - A_r)}$$
$$A_i : \text{Attributes from Sensors}, A_r = \frac{MaxValue(A)+MinValue(A)}{2} \quad (3)$$

## 3 System Design

The context awareness middleware deduces the context information using the data received from the sensors and delivers the information to the desired context awareness service. In the development process of the context awareness service, the context awareness middleware collects, manages, combines the context information and for-wards it to the context awareness services.

In order to deduce the context creator for creating the basic context and the basic context into complex contexts, the context awareness middleware is constructed into the ontology deduction engine, the deduction engine interface for interaction, the context awareness service, the event broker for delivering events in the supply/register format, and the translator for analyzing the commands received from the context author and the context awareness service. The author defines the context ontology required for the domain using the common context ontology defined in Chapter 3. The defined ontology is constructed with OWL files, which are sent to the translator. The translator provides a method for registering the context information types that the context awareness service needs. The context ontology defined by the context author and the rules for deducing the complex context are delivered to the context deduction engine through the deduction engine interface, and the context space information is stored in the database of the basic context generator. The basic context generator creates an appropriate basic context using the data collected from the sensors as well as the context space information stored in the database and the context conflict processor. The basic context information is delivered to the event deliverer through the deduction engine interface, converted into a fact for complex

context deduction, and forwarded to the deduction engine. The basic context delivered to the deduction engine goes through the consistency test and the complex context deduction process using the rules and ontology defined by the context author. The generated complex context is again sent to the event deliverer through the deduction engine interface and finally to the context awareness service that requires the corresponding context information. Fig 3 is the system configuration of the overall context manager.

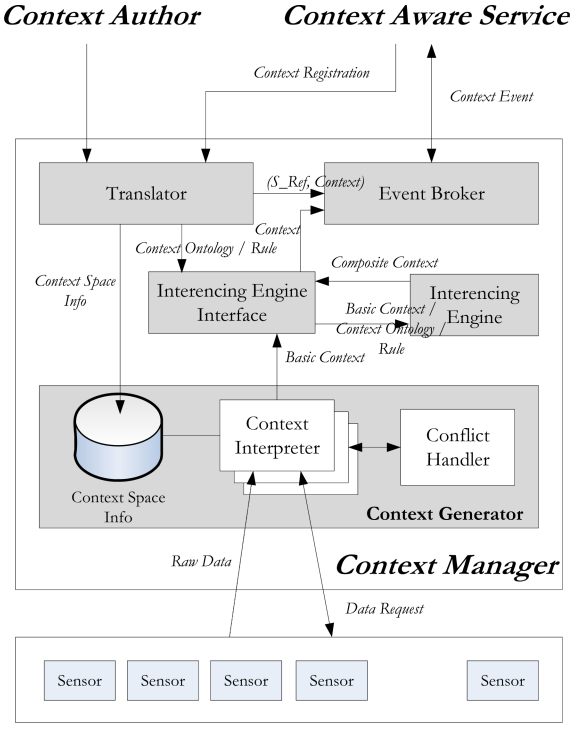


**Fig. 3.** An Overview of Context Awareness Middleware

The context awareness middleware was designed as a bundle on the OSGi framework foundation. Therefore, the middleware inherits the BundleActivator interface of the OSGi framework and uses the Jena.jar file for ontology deduction. The translator receives the domain context information and rules from the ontology author, and the readFile() method that allows reception of individual context information in a file format from the context awareness service, internally parses the OWL file, which is delivered to the ontology deduction interface and the translator. The basic context translator compares the information acquired from the sensors and the context space information to generate the basic context. If there is a conflict among the context spaces, an appropriate context space is sought using resolveConflict() of the conflict processor.

The context ontology information input by the middleware manager in the OWL file format and the basic context information generated by the context generator are transferred to Jena through the deduction engine interface. The deduction engine interface induces the complex context deduction using Jena, which is the ontology deduction engine. Here, insertContext() and removeContext() are for adding and deleting the basic context created by the basic context generator to and from the deduction engine in the Fact form. The complex context deduction rules defined by the middleware manager are managed using insertRule() and removeRule(). The deduction engine interface is constructed into six methods for adding and removing facts, rules and ontology. Each element receives an ID when being added to the deduction engine, and the ID is used to manage the corresponding element.

The basic context generator receives the data from the sensors and compares them with the context space information to deduce the appropriate basic context. The context information is given a context ID to be managed by the middleware, converted into a Fact through the deduction engine interface, and retransmitted to the deduction engine. The deduction engine checks consistency of the context information and generates complex context through ontology deduction and user rule-based deduction using the facts and the rules. The generated context information is delivered to the event broker, which forwards the information to the context awareness service that has registered the context information.

## 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[9,10]. Fig 4 displays the home network environment implemented for the test.

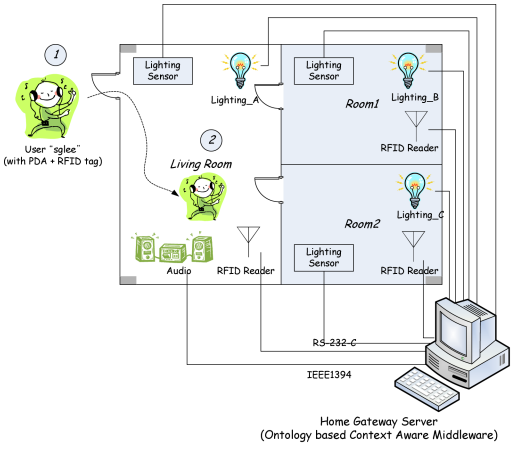


**Fig. 4.** Prototype of Smart HomeNetwork

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 2].

**List 2.** 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.

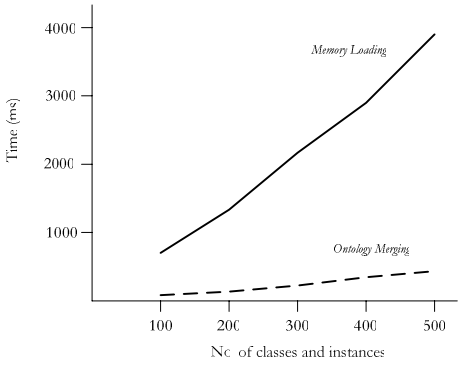


**Fig. 5.** Overhead of Merging Context Ontology

## 5 Conclusion

This paper proposed an ontology-based context awareness service management model with hierarchical context management. This hierarchical context management provides the resolution method for context uncertainty. And This study proposes the context management system based on the dynamic context ontology management method, which involves the hierarchical context management method using common context and the paper proposes a method for resolving the context uncertainty problem of interpreting the data acquired from the sensors as two or more types of context information. The context awareness service also can be registered to system dynamically. The designed middleware was implemented under the OSGi framework environment to connect with physical devices such as sensors and appliances. Finally, the home network context awareness service was implemented to validate the feasibility of the middleware designed and proposed in this paper. 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. Lee, S., Lee, J.: Dynamic Context Aware System for Ubiquitous Computing Environment. In: Shi, Z.-Z., Sadananda, R. (eds.) PRIMA 2006. LNCS (LNAI), vol. 4088, Springer, Heidelberg (2006)
2. Loke, S.: Context Aware Artifacts: Two Development Approaches. IEEE Pervasive Computing (2006)
3. Hung, N.Q., Lee, S.Y., Hung, L.X.: A Middleware Framework for Context Acquisition in Ubiquitous Computing Systems. In: Proceedings of the Second International Conference on Computer Applications (2004)
4. Gu, T., Pung, H.K., Zhang, D.Q.: An Ontology-based Context Model in Intelligent Environments. In: Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, pp. 270–275 (2004)
5. Euzenat, J., Pierson, J., Ramparany, F.: A contex information manager for pervasive computing environments. In: Proc. 2nd ECAI workshop on contexts and ontologies (C&O), Riva del Garda (IT), pp. 25–29 (2006)
6. Truong, B., Lee, Y., Lee, S.: Modeling Uncertainty in Context-Aware Computing. In: Proceeding of the 4th Annual ACIS International Conference on Computer and Information Science (2005)
7. Padovitz, A., Loke, S., Zaslavsky, A.: On Uncertainty in Context-Aware Computing: Appealing to High-Level and Same-Level Context for Low-Level Context Verification. In: Proceeding of the 1st International Workshop on Ubiquitous Computing (2004)
8. Gu, T., Pung, H., Zhang, D.: A Bayesian approach for dealing with uncertain contexts. In: The Proceeding of the Second International Conference on Pervasive Computing (2004)
9. Open Services Gateway Initiative. http://www.osgi.org
10. Knopflerfish. http://www.knopflerfish.org