International Journal of Smart Home
Vol. 11, No. 5 (2017), pp. 33-40
http://dx.doi.org/10.14257/ijsh/2017.11.5.03

Development of a Cooperative Middleware to Provide Context-Aware Service in Smart Home

M. Robiul Hoque, M. Humayun Kabir and Sung-Hyun Yang

*Department of Electronic Engineering,
Kwangwoon University, 139-701 Seoul, Republic of Korea
shyang@kw.ac.kr*

Abstract

To realize smart home services, several context-aware applications should be deployed which adapt their behavior depend on context information of home occupants. For context-aware service, context providing, processing, reasoning, delivery and designing context-aware applications are complex tasks. Although every application uses its registered contexts to provide services but some of the contexts are common for different services. So, it is wise to compute contexts at once and distribute to each application cooperatively rather than compute the same context for different application separately. Here the context reusable mechanism can significantly reduce the context computing cost. For this purpose, middleware-based implementation is a good alternative. Middleware uses raw data through sensors then generates context using these data, and finally conveys context to applications. In this paper, we present a context-aware middleware which provides above tasks and supports context sharing in cooperative manner. Context reasoning is an important task in context-aware systems. A formal context model based on ontology can play a vital role in facilitating reasoning by formally representing domain knowledge. We model domain context using ontology and adopt FOL which helps the context reasoning.

Keywords: *Context-aware, Cooperative middleware, Ontology, service, Smart home*

1. Introduction

Smart home is an intelligent home which is capable of sensing the home's occupants and their current state, and providing appropriate services to them [1]. The main target of smart home is to provide the proper services by determining users' situation with their context information. By context, we refer to any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or physical or computational object [2]. An appropriate middleware could support most of the tasks involved in dealing with contexts such as acquiring context information from various sources and producing context, processing and reasoning of context, and carrying out dissemination of context to interested parties.

Several context-aware systems have been developed to demonstrate its usefulness. In [3], [4], and [5] proposed middleware system to provide context-aware services in smart home. Another similar research [6] implements smart classroom service. Based on context dissemination, authors of [4] address cooperative characteristic of middleware system, but they did not consider any formal context modeling or representation. As they implemented using modular approach, it is very difficult to extend their middleware for multiple-domain pervasive computing. On other side, they developed context inferring module embedding with context provider on low resource device, it may increase service providing time delay.

To overcome those limitations, we have proposed a middleware system which supports cooperative context sharing. This middleware is implemented using OSGi-based service oriented approach where each middleware component is published as a service, and other components can use as web service. So, it will help the system extension in multiple-domain. We have developed context reasoner on rich resource device *e.g.*, residential gateway, as a result, service providing delay will be decrease significantly by reducing context computational time. Moreover, context is modeled based on ontology [7] using OWL- Web Ontology Language [8] which enables logic inferring, and ensure context consistency.

The rest of this paper is organized as follows. In Sections 2 and 3, Ontology-based context model and reasoning technique are presented respectively. In Section 4, cooperative middleware architecture is presented in details. In Section 5, implementation and evaluation of this middleware is presented, and finally, in Section 6, authors conclude this paper and indicate future plan.

2. Context Modeling

Context is any information that can be used to characterize the situation of entities that are considered relevant to the interaction between a user and an application, including the user and the application themselves [2]. Context could be classified in two types such as low-level context and high-level context. The low-level context is directly detected from context provider, and regards as explicit context. The high-level context is derived from low-level contexts, and regards as implicit context. To make high-level context semantically, it is need to context reasoning where ontology based context modeling could help this reasoning.

Context modeling is the specification of all entities and relations between these entities which are needed to describe the context as a whole. Among other, we adopted ontology-based context modeling which has several benefits, such as, if any context-aware system uses ontology to generate a high-level context, the system could make a semantically perfect high-level context for context-aware service. The use of ontology in the context domain is becoming more and more important. By using the ontology approach, we obtain a formal context representation and reasoning methods on contextual data.

Ontology can be defined as a formal explicit definition of a shared conceptualization [9], and has many benefits like as knowledge sharing, logic inference, and knowledge reuse. It provides a vocabulary for representing domain knowledge, and for describing specific situations in a domain. We propose an ontology-based approach for context modeling that describe contexts semantically, and share common understanding of the structure of contexts among users, devices, and services. The main benefit of this model is that it enables a formal analysis of the domain knowledge, such as performing context reasoning using first order predicate calculus.

There are many ontology language *e.g.*, RDF(S)[10], OIL[11], DAML+OIL[12], OWL[13]. OWL (Web Ontology Language) is a key to the Semantic Web Ontology language, proposed by W3C’s Web Ontology Working Group. OWL ontology language is a language for defining and instantiating web ontology. OWL is much more expressive than other ontology languages such as RDFS (Resource Description Framework (Schema) [14]. OWL can describe ontology representation like Class, subClassOf, individuals, ObjectProperty, DatatypeProperty, constraints, etc. OWL is divided into 3 languages - OWL Lite, OWL DL, and OWL full in accordance with representation power. We adopt OWL DL to represent context ontology. It is sufficient to describe our smart home context modeling using OWL DL.

Some researcher has been constructed ontology for context in a specific domain to reach the goals of knowledge sharing across distributed system. CONON [15] introduced extensible context ontology for pervasive computing environments. CONON define four basic concepts. They don’t define the environment and time concepts. COBRA [16], proposed by Chen et al., used an ontology to describe person, places, and intentions.

COBRA does not have any general or upper level ontology, so it will be difficult to ensure knowledge reuse and knowledge sharing. Obviously our ontology model focuses on smart home and could be satisfied required queries related to smart home services.

We proposed two layered ontology model for contextual information: upper level and lower level ontology. Where, upper ontology provides definitions for general-purpose terms and acts as a foundation for specific domain ontology. Lower ontology, other hands, provides the domain specific knowledge to context-aware applications.

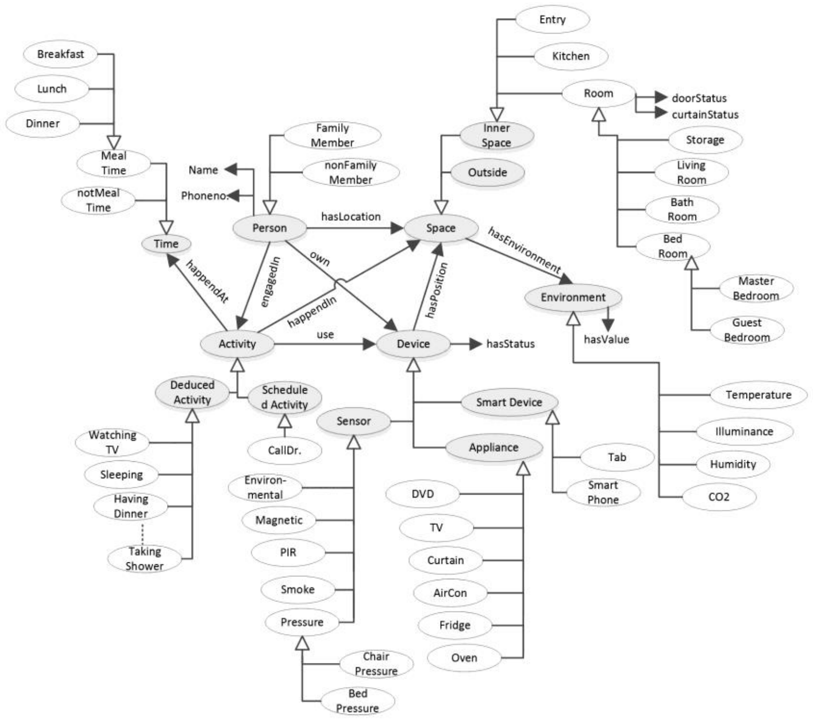

Figure 1. A Part Of Details Ontology

Context entities, in case of smart home, range from various kinds of devices *e.g.*, smart phone, laptop, ambient displays; to various environment conditions *e.g.*, illuminance, temperature, CO_2, humidity; and user, service and so on. All these entities are present and functioning at different times at different places in home. We categorize these entities, in our model, mainly into *person*, *device*, *space*, *environment* and *time* classes which are defined upper ontology. Each class may divide in several sub-classes. For example, space emphasize on the spatial aspects of our smart home domain. Space has two sub-classes *Inner* space and *outside*. Again, inner spaces of our home are divided into *room*, *kitchen* and *entry* and represents lower ontology for home domain. Furthermore, room is divided into *bed room*, *living room*, *bath room* and *storage room*. We have also defined some objectProperties such as *hasLocation*, *hasEnvironment*, *use*, *own*, *engagedIn*, *hppendIn etc.*, which present relationship between classes. We have defined some inverse properties, *e.g.*, *isLocationOf* is an inverse property of *hasLocation*. In Figure 1, we present partial ontology of our context model written under the Protégé editor. Each entity is associated with its attributes and relations with other entities. OWL object properties represent relation and data properties represent attributes of entities. Hence, gray oval and white oval represent upper ontology class and lower ontology class respectively.

In our model, we represent contexts in first-order predicate calculus. The basic model has the form *predicate (subject, value)* or *predicate (subject)* in which.

- *subject* $\in S^*$: the set of subject names (for example, a person, space, device, *etc.*,)
- *predicate* $\in V^*$: the set of predicate names (for example, haslocation, hasStatus, *etc.*,)
- *value* $\in O^*$: the set of all values of subjects in S* (for example, living room, On, false, *etc.*,)

To illustrate,

- *haslocation (MrHong, bathroom)* means that Mr. Hong is located in the bathroom.
- *hasTemperature(kitchen, warm)* means that the temperature in the kitchen is greater than a comfortable temperature.

3. Context Reasoning

Reasoning is a process to reason about various types of contexts and their properties. Our system has the ability to support automated context reasoning. It can reason deduced context from other contexts. We adopt two kinds of reasoning: Rule-based approach based on first order predicate calculus for inferring high-level context, and Ontology-based reasoning for retrieving relation among context entities, and compute subsumption.

3.1. Ontology Reasoning

The ontology reasoning is responsible for checking class consistency and implied relationships among entities. Our ontology's reasoning mechanism supports RDF Schema and OWL DL. Ontology reasoning is useful in this system. For example, if x and y are two context entities, ontology reasoning retrieves the relation between them r, then the system makes the context in the format of r(x,y). For computing subsumption, let an indoor location provider provides a context that Mr. Hong is currently located in his Living Room, the context interpreter can conclude that Mr. Hong is located in his home since Living Room is a part of Home, shown in Eq. (1).

$$(?a\ rdfs{:}subClassOf\ ?b) \wedge (?b\ rdfs{:}subClassOf\ ?c) \rightarrow (?a\ rdfs{:}subClassOf\ ?c) \quad \ldots\ldots \quad (1)$$

3.2. User-defined rule-based Reasoning

Rule-based reasoning is done by a context reasoner. The need for reasoner arises because not all contexts can be gathered from sensors. For rule-based reasoning, we have defined a set of rules to assert high-level context from low-level context. This reasoning is provided using forward chaining algorithm. Equation (2) shows a rule to deduce a person's current status where, Mr. Hong is a person as well as a *FamilyMember*, his location is in *MasterBed* and *BedPressure* sensor is ON, and there is no motion. For a simple example, using these information reasoner can deduced Mr. Hong is sleeping using Eq. (2).

$$Person(?x) \wedge BedRoom(?y) \wedge PIR(?r) \wedge BedPressure(?p) \wedge hasLocation(?x,?y) \wedge isPositionOf(?y,?r) \wedge isPositionOf(?y,?p) \wedge hasMotion(?r,false) \wedge hasStatus(?p,ON) \rightarrow engagedIn(?x,Sleeping) \quad \ldots\ldots\ldots \quad (2)$$

4. Cooperative Middleware Architecture

Our context-aware middleware architecture is shown in Figure 2. There are three main layers in hierarchical design which are context providing layer, middleware layer and application layer. Context provider and application can be reside on any computational devices *i.e.*, smart phone, laptop, pc or server, but middleware layer resides, in this case, only on a residential gateway.

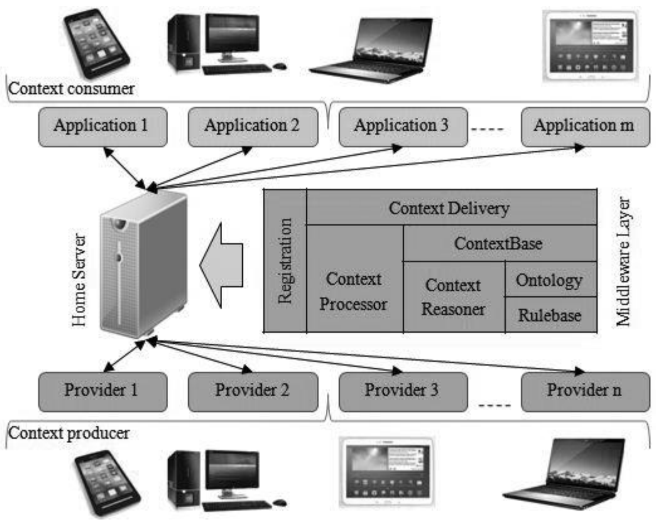

Figure 2. Cooperative Middleware Architecture

Context provider performs the task of context acquisition, mapping and representation. It produces low-level context entity from sensing raw data using mapping technique. To get more precise symbolic context entity, we have used fuzzy function in mapping technique. It also represents context in FOL format and send to registered services i.e., application or context processor. Sensors may provides information with its own format, e.g. location sensor provides location using specific coordinate system, but in real world we need more high level information e.g. instead of coordinate we need information like bedroom, livingroom, home or office etc. To make these high level information, context provider uses mapping technique. Similarly, for temperature, we need whether it is cold, worm or hot. Here fuzzy function decides more precise information from several overlapping temperature group. Using these high level information, context provider generate simple context with the help of context ontology. For example, ontology provides a relation between location and temperature which is '*hasTemperature*' and context provider produces simple context: *hasTemperature (bedroom, cold).*

Context processor decides whether a context is needed to combine with other to make high-level context. High-level context may be computed from combining only low-level contexts or both low-level and high-level contexts in a hierarchy fashion.

Context reasoner is an inference engine that is responsible for computing high-level context from low-level contexts using rule base reasoning. It supports two types of reasoning such as ontology-based reasoning and user define rule-based reasoning. Ontology reasoning retrieves relation between context entities and maintains class consistency which is used by the context provider. On the other hand, rule-based reasoning asserts new context from other contexts. In this middleware, rule-based reasoning is implemented using forward chain algorithm based on RETE algorithm [17]. Here context processor with context reasoner acts as a context provider.

Registration service provides facilities to declare capabilities of context providers and context processor. Registration service stores entity name, context type and access point of each registered context provider in context base. It is also able to track and adapt to the dynamic changes of context providers based on adding or removing sensors in smart home.

Application generates smart home services using context with IF-THEN condition. Application gets access point of context provider through context delivery. That means, context delivery gets query from applications and, after processing that query, returns required context provider access point. Then application can access context provider directly.

5. Implementation and Evaluation

Middleware components have been implemented as services using Java. Context provider and application are developed as the distributed independent services where Java RMI is used as communication mechanism. At time of system initialization, applications get requested context providers' access point to receive context from that context providers directly.

The Open Service Gateway Initiative (OSGi) is an independent, non-profit corporation working group to define and promote open specifications for the delivery of managed services to networked environments, such as homes. The OSGi service platform consists of a framework and multiple bundles, and provides a sharing mechanism for packages and services among bundles executed in the framework [18-19]. In this paper we adopt OSGi platform which provides bundle management mechanism, so that we can easily control and manage several services.

The OSGi service platform refers to the software stack embedded in the OSGi-compliant residential gateway as shown in Fig. 3. It provides a service-hosting environment as well as a set of common APIs to develop application bundles.

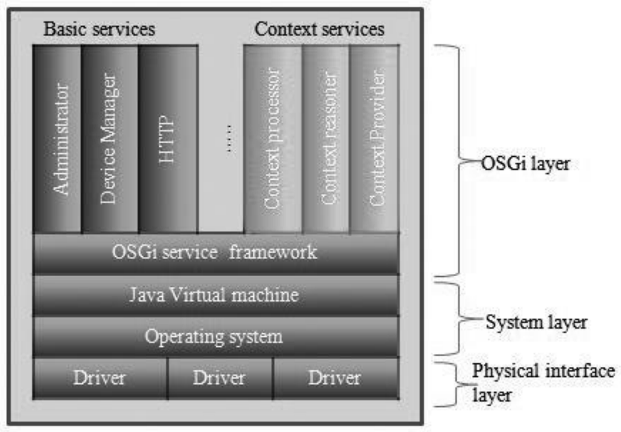

Figure 3. Software Architecture For OSGi Service Platform

To evaluate our middleware performance, we have developed a test bed smart home which is equipped with different types of smart home appliances, smart devices and various sensors. Sensors are grouped into residential, security and human sensors. Residential group sensors are temperature, humidity, CO_2 and illuminance sensor. Intrusion (door open) and fire (smoke) detection sensor are fallen in security sensor group. User identification (RFID Tag), user motion detection (PIR sensor) and user's presence on bed (pressure sensor) detection sensors are fallen in last group. Home appliances (*e. g.*, aircon, DVD, TV, refrigerator, curtain, washing machine, *etc.*,) are placed in different position in the home. They are connected with internet via various wired and wireless network.

We can describe cooperative characteristic with the example of smart home applications. Suppose, a home user enters into his bedroom, RFID tag will provide his location and light adjustment application turns on room light based on location context. Temperature control service needs user location and current temperature context in that location to adjust temperature based on user preference. In the case of sleeping service, application needs some additional context, such as motion context provider provides no motion in room and pressure detection provider provides pressure sensor on bed is ON. Then, application provides sleeping service (turn off light, close window curtain and adjust room temperature) combining previous location, temperature context and newly received motion and pressure context. Here, same location context is shared among three applications and temperature context is shared between two different applications. It is obvious, if application is increase, shared contexts also will be increased and context computation cost will be reduced significantly. Hence, context providers and applications of the middleware cooperate among them. So, we called our middleware supports context sharing by cooperative manner.

6. Conclusions

In this paper we have presented context-aware middleware design and its implementation issues together for smart home services. This middleware provides service for user by inferring current context and distributing cooperative manner. The cooperative property facilitates middleware to provide service in time by reducing the context computation cost. Moreover, we have presented a formal context model based on OWL to support middleware functionalities. This model is sufficiently expressive to satisfy required queries about the smart home context. In this paper, we used ontology reasoning for only retrieving relation properties among context entities, computing subsumption and checking class consistency. In the future, we will extend our middleware for multiple-domain as well as we will improve ontology reasoning to infer complex context *e.g.*, social context /group activity context.

Acknowledgements

This work was supported by the Industrial Strategic Technology Development Program (10041788, Development of Smart Home Service based on Advanced Context-Awareness) funded by the Ministry of Trade, Industry & Energy (MI, Korea). The present research has been conducted by the Research Grant of Kwangwoon University in 2016.

References

[1] J. Sei, Research activities on smart environment, The Institute of electronics engineers of Korea Journal, pp. 1359-1371, **(2001)**
[2] A. K. Dey, and G. D. Abowd, Towards a better understanding of context and context-awareness, in Proc. Workshop on the what, who, where, when and how of context-awareness, **(2000)**
[3] T. Gu, H. K. Pung, and D. Q. Zhang, A service-oriented middleware for building context-aware services, Journal of Network and Computer Applications, vol. 28(1), pp. 1-18, **(2005)**
[4] V. N. Hamed, Z. Kamran, and N. Nasser, Context-aware middleware architecture for smart home environment, International Journal of Smart Home, vol.7 (1), pp. 77-86, **(2013)**
[5] M. R. A. Jose, and W. Johnny, Service-oriented middleware for smart home applications, in Proc. Wirelss HIVE Networks Conference, pp. 1-4, **(2008)**
[6] W. Win, Y. Shi, and Y. Suo, Ontology-based context-aware middleware for smart spaces, Tsinghua Science and Technology, vol. 12(6), pp. 707-713, **(2007)**
[7] D. Fensel, Ontologies: A silver bullet for knowledge management and electronic commerce, Springer, **(2003)**
[8] M. Smith, C. Welty, and D. McGuinness, Web ontology language (OWL) guide, http://www.w3.org/TR/owl-guide/, **(2003)**
[9] D. fensel, Ontologies: A silver bullet for Knowledge Management and Electronic Commerce, ISBN 3-540-41602-1, **(2000)**
[10] W3C, RDFS(RDF Vocabulary Description Language1.0: RDF Schema), Recommendation, http://www.w3.org/TR/rdf-schema/ , **(2004)**
[11] OIL, http://www.ontoknowledge.org/oil
[12] DAML+OIL, http://www.daml.org
[13] W3C, OWL Web Ontology Language Overview, Recommendation, http://www.w3.org/TR/2004/REC-owl-features-20040210/, **(2005)**
[14] T. Gu, A service-oriented middleware for building context-aware services, Journal of Network and Computer Applications, Vol. 28(1), pp. 1-18, **(2005)**
[15] X. Wang and et al., Ontology based context modeling and reasoning using owl, in Proc. Workshop on Context Modelling and Reasoning at IEEE Int'l Conference on Pervasive Computing and Communication, USA, **(2004)**
[16] H. Chen, T. Finin, and A. Joshi, An ontology for context-aware pervasive computing environments, Special Issue on Ontologies for Distributed Systems, **(2003)**

[17] C. L. Forgy, RETE: a fast algorithm for the many pattern/many object pattern match problem, artificial intelligence,**(1982)**
[18] Open Services Gateway Initiative, OSGi Service Platform, http://www.osgi.org
[19] K. Kang, Dynamic Service Management Strategy in an OSGi-Based Operation, in Proc. ITC-CSCC, **(2005)**