Enabling Context-aware Smart Home with Semantic Web Technologies

Daqing Zhang, Tao Gu, Xiaohang Wang

Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613
{daqing, tgu, xwang}@i2r.a-star.edu.sg

Abstract. Context-awareness has emerged as a key enabling technology for future smart homes. In this paper, we propose a generic layered model (the Context Stack) to guide the context-aware system design. We investigate the use of Semantic Web technologies in context modeling and reasoning. A Web Ontology based context model (CONON) is defined to facilitate explicit context representation, semantic context sharing, and logic reasoning. We show how the use of logic reasoning enables context-aware services to deduce high-level, implicit context from low-level, explicit context. In particular, we integrate the temporal ontology into our ontology model to facilitate annotating and reasoning about temporal context information. Following the general layered model, we present an OSGi-based context-aware service architecture for smart homes. We also build the smart home prototype with context-aware services.

Index Terms:. Context-awareness, smart home, semantic web, ontology

1 Introduction

Recent years have witnessed rapid advances in the enabling technologies for smart homes, such as the increasingly mature networked appliances, pervasive sensor/actuator technologies and various kinds of wired and wireless communication protocols. Home users are progressively offered more and more personalized services, ranging from home automation, security and monitoring, entertainment, to healthcare. In particular, many research projects have targeted to build smart homes for ageing and disabled people, to prolong their stay at home for independent living and assist them in their everyday life [1]-[3]. It is widely acknowledged that an important step in ubiquitous computing is context-awareness. Services in pervasive and mobile environments need to be context-aware so that they can adapt themselves to rapidly changing situations. By context, we refer to any physical or conceptual information about the execution environment of a service.

Even though a lot of context-aware systems have been built in the past years, there lacks, however, a general model to guide the system design. This fact has caused the duplicate of huge effort and great interoperability issue in the community. Inspired by

the seven-layer OSI model, in this paper, we first propose a generic five-layer model for guiding the design and implementation of context-aware systems. Our layered model abstracts the functional elements of context-aware systems, i.e., context acquisition, context representation, context aggregation, context interpretation, and context utilization. In order to address the interoperability issue among context-aware systems, we leverage on our previous ontology-based context model [4] to represent and reason context information. As compared to previous context representation such as in [5]-[7], our context model has great advantages in terms of expressiveness, knowledge reuse, scalability and enabling logic reasoning. Knowledge reuse requires heterogeneous computational entities such as devices, services, applications, and agents to have a shared understanding of context information. The use of ontology enables different context-aware systems to have common concepts about context information to achieve better interoperability. With context ontology, we are also able to apply various logic inference engines such as Jena2 [8], JTP [9], FaCT [10] to deduce high-level context from low-level context. In this paper, we further enhance our context model by defining a temporal ontology and demonstrate the use of temporal reasoning to infer time-related context information. We also present the design of layered model based on OSGi infrastructure in smart home environments. We show how the functionalities of each layer can be implemented on top of OSGi platform, and demonstrate typical context-aware services to enhance everyone's daily life.

The rest of this paper is structured into five sections. In the next section, we present the generic layered model: context stack. In Section 3, we briefly describe our context ontology with the enhancement of temporal ontology. In Section 4, we describe the use of logic reasoning in our architecture via some use cases, with special focus on temporal reasoning. In Section 5, we present the design of the OSGi-based service infrastructure for smart homes with our layered model and describe the implementation of the smart home prototype with several context-aware services. Section 6 concludes the paper.

2 The Context Stack: A Layered Model for Context-Aware Systems

Network oriented systems are easily implemented largely because of the ISO Open System Interconnection (OSI) layered model which attributes different functionalities to independent layers. Much the same way, a generic layered model is needed to guide the development of context-aware systems.

In this section, we present a generic layered model, namely Context Stack, as a reference model for context-aware systems. Our model is similar in spirit to the seven-layer OSI model for computer networks. The layered model combines the functional elements of context-aware systems (i.e., context acquisition, context model and representation, context aggregation, context interpretation, context utilization), and melds them into a coherent, generic architecture that most current context-aware systems can mapped onto. The Context Stack provides context-aware systems with

robust separation of concerns. Fig. 1 shows the proposed five-layer Context Stack and the illustration of a context-aware smart phone service mapping on to the layered model as an example.

2.1 Context Acquisition Layer

The lowest level is the context acquisition layer, in which context in raw format may be acquired from a wide variety of ubiquitous context sensors. For example, the indoor location of a user can be obtained from a RFID location sensor system, which detects the presence of a RFID tag to conclude the location of the user wearing it, the light level is sensed by X.10 light sensors, and the noise level is obtained from noise sensors. The acquired sensor data might not provide meaningful context that can be directly understood and utilized by context-aware services, thus need to be passed to upper layers for further processing.

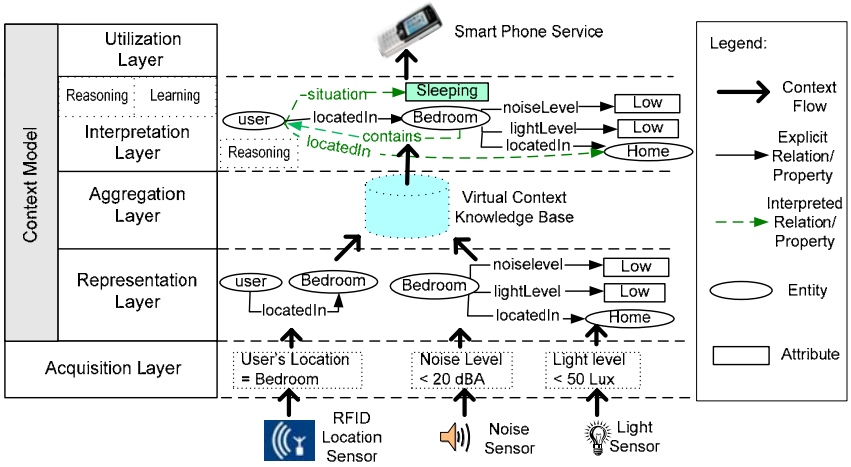


Fig. 1 The Context Stack, a reference model for context aware systems

2.2 Context Representation Layer

The upper layers (i.e., representation, aggregation, interpretation, and utilization layers) rely on a common context model, which forms the basis for context management to facilitate expressive representation, semantic sharing and interoperability of heterogeneous computational entities. Existing context models vary in the type of context they represent and the expressiveness they provide. As shown in Fig. 1, we choose to use a general object-oriented (or entity-relation oriented) context model to illustrate the role of context model in context-aware systems. In such context model, context information is structured around a set of entities, each describing a physical or conceptual object such as a person or an activity, and these contextual entities are linked to their attributes and other entities with relations.

At the context representation layer, raw sensor data is represented into a machine-readable format following the adopted context model. This is actually an abstraction layer that acquires sensor data from context sources, and then annotates sensor-driven context with semantics that are structured around a set of contextual entities (e.g., '*user*', '*location*' and '*device*'.) and the relations (e.g., '*locatedIn*') that hold between them.

2.3 Context Aggregation Layer

The context aggregation layer aggregates and relates contexts from distributed context sensors to form a centralized context database. Context aggregation helps simplify the context query procedure of computational entities through a centralized manner instead of distributed approach, and provides a basis for further interpretation over related contextual knowledge. Therefore, context aggregation by its nature is to provide the functionalities of a context knowledge base. Another function of context aggregation is to store historical context to support further interpretation based on both current and past contexts.

2.4 Context Interpretation Layer

When taking a context model approach to represent and manage context, context can be interpreted with various kinds of reasoning and/or learning mechanisms to derive high-level additional context. The context interpretation layer leverages reasoning/learning techniques to deduce high-level, implicit context needed by intelligent services from related low-level, explicit context. For example, the rule-based reasoning engine can deduce user's current situation based on his location and environmental contexts. The inferred context might suggest that the user might be sleeping currently, since the time is 11 pm and he is staying at a dark, quiet '*Bedroom*'. Another example of context interpretation could be machine learning based behavior prediction. The intelligent system could learn the pattern of user's actions from historical sequences of context data and then use this learned pattern to predict next event. For example, it could be predicted that once the user finished showering (turn off the electronic water heater) after 10:30 pm, he will check emails using the hand phone, and then go to bed after reading them.

2.5 Context Utilization Layer

Finally, At the uppermost level (context utilization layer), context-aware services utilize both low-level and high-level context to adjust their behaviors. The smart phone service then queries user's context (sleeping) and decides to forward all phone calls to the voice message box. It also could take account of predicted context to trigger certain actions.

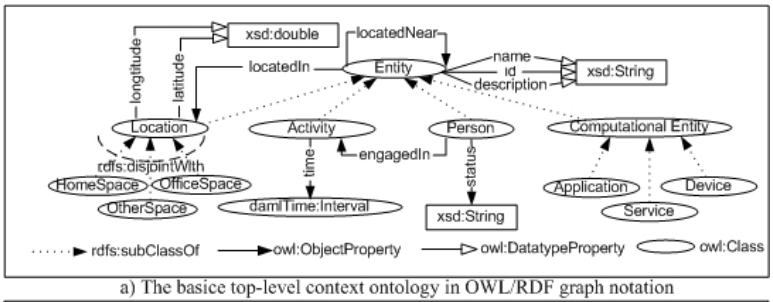
Obviously, the Context Stack provides a concise layered architecture for the specification of context-aware systems design. In the following sections, we will investigate how to use Semantic Web technologies to model and reason context, and show how the Context Stack is adopted as the design abstraction to develop a context-aware service infrastructure for smart homes.

3 The Context Ontology

In this section, a Web ontology based context model (**CONON**) is defined to facilitate explicit context representation, semantic context sharing, and logic reasoning [4]. We further integrate the temporal ontology to facilitate the annotation and reasoning about temporal relationships between different context events and activities.

3.1 The Ontology for Contextual Entities

To completely formalize all contexts in pervasive service environments is an insurmountable task; the objectives of designing a reasonable context ontology is as follows:
Modeling a set of top-level contextual entities and their interrelationships.
Providing flexible future extensions to add specific ontologies in different service domains.
Fig. 2 (a) shows the top-level context ontology. The context model is structured around a set of contextual entities, each describing a physical or conceptual object such as a user or an activity. Each entity is associated with its attributes (represented in *owl:DatatypProperty*) or relations with other entities (represented in *owl:ObjectProperty*). The built-in OWL property *owl:subClassOf* allows for hierarchically structuring additional entities, thus provides future extensions to add new type of contextual entities that are required in specific services. Each subclass entity inherits common attributes and relations from its superclass, and is associated with a number of additional properties.
The context ontology provides a shared vocabulary for describing basic concepts including users, locations, activities as well as a set of computational entities (i.e., services, applications and devices). It also defines a set of attributes (e.g., user's status) and relationships (e.g., spatial containment between different locations) hold between these contextual entities.



a) The basice top-level context ontology in OWL/RDF graph notation

```
<owl:Class rdf:ID="Entity">
</owl:Class>
<owl:DatetypeProperty rdf:ID="name">
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>
. . .
<owl:Class rdf:ID="Location">
  <rdfs:subClassOf rdf:resource="#Entity"/>
  <owl:equivalentClass>
     <owl:Class>
       <owl:unionOf rdf:parseType="Collection">
         <owl:Class rdf:about="#HomeSpace"/>
         <owl:Class rdf:about="#OfficeSpace"/>
         <owl:Class rdf:about="#OtherSpace"/>
       </owl:unionOf>
     </owl:Class>
   </owl:equivalentClass>
</owl:Class>
. . .
```

```
<owl:Class rdf:ID="Person">
  <rdfs:subClassOf rdf:resource="#Entity"/>
</owl:Class>
<owl:Class rdf:ID="Activity">
  <rdfs:subClassOf rdf:resource=":#Entity"/>
</owl:Class>
. . .
<owl:ObjectProperty rdf:ID="locatedIn">
  <rdf:type rdf:resource="TransitiveProperty" />
  < rdfs owl:domain rdf:resource="#Entity"/>
  < rdfs owl:range rdf:resource="#Location"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="locatedNear">
  <rdf:type rdf:resource="TransitiveProperty" />
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="lontitude">
  <rdf:type rdf:resource="FunctionalProperty" />
  < rdfs owl:domain rdf:resource="ConOnt:#Location"/>
  < rdfs owl:range rdf:resource=" &xsd:double"/>
</owl:DatatypeProperty>
```

b) A partial OWL/XML serialization

Fig. 2 The context ontology and a partial serialization

3.2 The Temporal Ontology

Context-aware services are often interested in more than the current state of context, as temporal context plays an important role in analyzing historical information and predicting future trends. Capturing the temporal information requires the annotation of temporal information including both instantaneous time point of events, duration of activities and ordering between events, among other things.

OWL-Time [11], a part of OWL project has defined an ontology of temporal concepts, for describing the temporal characteristics of Web resources. It is built on well-defined model of temporal representation, Interval Temporal Logic [12], and is formulated as a set of first-order predicate calculus axioms. We adopt this temporal ontology to annotate and reason about temporal contexts.

Fig. 3. The annotation and reasoning about temporal contexts based on the temporal ontology.

We associate time to temporal relationships by tagging properties with time-stamps or time intervals. For example, the OWL segment shown in Fig. 3 describes a piece of sensed context “Ray is located in the meeting room now (e.g., the current time is 16:10)”, and a piece of profiled context “a meeting is scheduled to be held in the meeting room during 16:00 to 17:00”. Through temporal reasoning, the inference service can deduce the additional temporal relation “the time point (16:10) is inside of the time interval (16:00-17:00)” according to the interval temporal logic, and this inferred context along with other related contexts might be used to further deduce more complex upper level contexts such as “Ray is in a meeting now”.

4 Context Reasoning

The goal of context reasoning is to deduce high-level, implicit context from low-level, explicit context. In this section, we show the reasoning power based the proposed context model CONON. We choose to implement context-reasoning by using first-

order logic. For example, the physical location context "Ray is located in the bedroom" can be described as: *(Ray, locatedIn, BedRoom)*. The structure of the context predicate has tree fields - a subject (i.e., *Ray*) that is a contextual entity, an object (i.e., *BedRoom*) that is contextual entity or a datatype value, and a verb (i.e., *locatedIn*) that describes the attributes of the subject or the relationship between the subject and the object. We believe that logics are powerful tools for representing and reasoning context, they are sufficient for a large number of context-aware smart home applications.

4.1 Relational Reasoning

In this section, we demonstrate relational reasoning via a number of use cases. These use cases focuse on the inference over relationships between related entities.

Use Case 1: Deducing user's Location
A user's location is an important customization criterion for context-aware computing. Location contexts are quite useful in location-based services, for example, when a user is near his/her friends, or to find out nearby restaurants or automatic teller machines.
In the context ontology, the property *locatedIn* is specified as transitive property, which implicitly defines the spatial containment relationships between different spaces. The following inference rule can be used to determine "whether Ray is located at any room inside of the building":

$$rdf{:}type(locatedIn, owl{:}TransitiveProperty) \wedge locatedIn(Ray, BedRoom) \wedge locatedIn(BedRoom, RayHome) \Rightarrow locatedIn(Ray, RayHome)$$

This rule demonstrates that if Ray is *locatedIn* the BedRoom, which is *locatedIn* the RayHome, then Ray must be also *locatedIn* RayHome, since *locatedIn* is a transitive relationship".
Another example concerning location contexts is to determine the proximity of different entities by examining their spatial locations:

$$locatedIn(Ray, RayHome) \wedge locatedIn(John, RayHome) \Rightarrow locatedNear(Ray, John)$$

This rule demonstrates that if "Ray and John are both *locatedIn* RayHome, we can conclude that "Ray is *locatedNear* John".

Use Case 2: Deducing User's Status
High-level contexts such as "what the user is doing" are often deduced from relevant environmental contexts. In this section, we demonstrate several example rules which are used to derive a user's status. These rules have been built in a smart phone application. By using these rules, users can customize the behaviors of the augmented mobile phone. Examples of the rules are:

IF $status(Ray, Shower) \vee status(Ray, Sleep)$
THEN *forward the incoming call to voice message box*
IF $status(Ray, Lunch) \vee status(Ray, Dinner)$
THEN *set the mobile phone to vibrate*
IF $status(Ray, WatchingTV) \vee status(Ray, LisentingMusic)$
THEN *turn up the ring volume*

The following relational reasoning rules show parts of the deductions of these high-level contexts. Other contexts used in the rules above are deduced from temporal contexts, which will be discussed in use case 3.

locatedIn(Ray, Bedroom) $\wedge$ *drapeStatus(Bedroom, ON)* $\wedge$
lightLevel(Bedroom, LOW) $\Rightarrow$ *status(Ray, Sleeping)*

locatedIn(Ray, Bathroom) $\wedge$ *status(WaterHeater, ON)* $\Rightarrow$ *status(Ray, Shower)*

locatedNear(Ray,TV) $\wedge$ *status (TV, ON)* $\Rightarrow$ *status(Ray, WatchingTV)*

locatedNear(Ray,CDPlayer) $\wedge$ *status (CDPlayer, ON)*
$\Rightarrow$ *status(Ray, ListeningMusic)*

4.2 Temporal Reasoning

A major advantage of the context ontology is to allow the combination of relational reasoning and temporal reasoning about contexts to describe a dynamic changing environment. The temporal ontology and temporal contexts are also formulated as first-order predicates: *Instant (e, t)* represents that an event *e* occurs at time instant *t*; *Interval (e, T)* represents that an activity *e* happens during a time interval *T*.

Note that *e* could be either a contextual entity (e.g., the birthday party will be hold during 16:00 to 17:00) or a temporal relationship between two entities (e.g., Ray stayed at RayHome during 2001-01-01 to 2002-10-01, Ray entered his BedRoom at 16:10).

Use Case 3: Deducing temporal context

Precisely determining a user's status often relies on various kinds of contextual knowledge, including environmental contexts and their temporal information.

For example, in a smart home environment, location sensors periodically report the presence of a user, and this information is associated with a time-stamp. An online calendar service provides the information about scheduled activities, including time, location, etc. From these original contexts, the inference service concludes the temporal relationships between different context activities and events based on temporal interval logic. This temporal context along with other related environmental contexts can be used to further deduce more complex high-level context. For example, the following temporal reasoning rule helps to determine "whether Ray is really in a meeting":

locatedIn(BirthdayParty, RayHome) $\wedge$
Interval (BirthdayParty,, Scheduled-birthdayparty-time) $\wedge$
Instant (locatedIn(Ray, RayHome), NOW) $\wedge$
insideOf (NOW, Scheduled-birthdayparty-time)
$\Rightarrow$ status(Ray, BirthdayParty)

Another similar rule can be used to determine the opposite context- "whether Ray is not really in a birthday party":

locatedIn(BirthdayParty,, RayHome) $\wedge$
Interval (BirthdayParty,, Scheduled-birthdayparty-time) $\wedge$
Instant (locatedIn(Ray, RayHome), NOW) $\wedge$

(before(NOW, Scheduled-birthdayparty-time) $\vee$ *after(NOW, Scheduled-birthdayparty-time))*

$\Rightarrow \neg$ *status(Ray, BirthdayParty,)*

From the use case described above, we see the capability of our context ontology to support advanced relational and temporal reasoning about context.

5 Service Architecture for Context-aware Smart Home

In this section, we describe the design and implementation of service-oriented architecture for context-aware smart homes.

5.1 Design Objective

The design of the service infrastructure adopts the generic layered model - Context Stack as the design abstraction to provide reusable service middleware. It demonstrate the following features:

- the component based middleware architecture aims to combine independent functional elements of context-aware systems (namely context acquisition, aggregation, interpretation, utilization, and discovery), and melt all the elements together into coherent, reusable middle layers.
- the design explores the Semantic Web infrastructure to acquire rich context information. Semantic Web Ontology (CONON), Semantic Web query and inference mechanisms are incorporated into this infrastructure to facilitate formal context modeling, knowledge sharing, semantic query, context reasoning and semantic context discovery.
- we leverage open standards including OSGi and UPnP to simplify the programming of reusable service components. It aims to provide a set of API standards and implement a number of deliverable context-aware services in a smart home environment.

5.2 Overview of OSGi-based Service Infrastructure for Smart Home

The OSGi-based service architecture is shown in Fig. 4 and has been described in [13][14]. Central to the architecture is an OSGi residential service gateway that is responsible for connecting home appliances, mobile terminals and sensor/ actuator devices to Internet through broadband access technologies. Various home networking technologies such as Ethernet, X.10, IEEE 1394, Lonworks, WLAN or Bluetooth can be used to interconnect various appliances devices. The OSGi framework standardizes the way for secure and reliable home service delivery and provisioning, for remote service lifecycle management as well as for bridging between different home networking standards.

In order to achieve the knowledge sharing and logical inference about context information, the following four key functional modules are incorporated into the service architecture in form of OSGi service bundles:

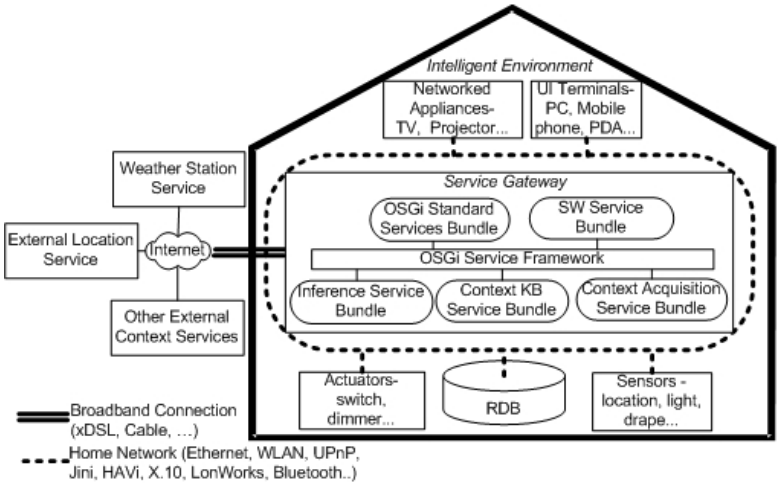


Fig. 4 OSGi-based context-aware service architecture

- SW (Semantic Web) Service: The SW service bundle provides Semantic Web APIs for accessing and manipulating OWL-encoded contexts, for persisting and querying semantic models of contexts in relational databases and for inferring additional contexts from existing contexts in KB. The current implementation is based on Jena2 Semantic Web Toolkit, however, alternative Semantic Web engines and KB tools should be wrapped in a generic interface so that they can be plugged into the service architecture. The SW service bundle provides fundamental for the following three services.
- Context Acquisition Service: The context acquisition service is responsible for collecting context information from a wide variety of context sources and transforming the *ad hoc* context description to semantic representation using context ontology. This service provides the functionalities of context acquisition layer and semantic representation layer in our generic layered model described in Section 2.
- Context KB (Knowledge Base) Service: The context KB service provides a persistent storage for context knowledge through the use of relational database. It stores and manages both static context ontology (the ontological descriptions of classes or types of interrelated entities) and dynamic context information (situational instances or individuals represented based on the context ontology), thus supports for reasoning about both context ontology and situational context information. The KB service enables the service architecture to aggregate and relate dynamic contexts from highly distributed context sources, therefore provides the functionalities of semantic aggregation layer.

- Context Inference Service: The context inference service is a packaged reasoning engine that can be used by context-aware services to deduce high-level contexts from low-level contexts. The inference service is based on Jena2 APIs, which provides rule-based inference support for RDFS and OWL through pluggable reasoners. This service provides the functionalities of context inference layer.

5.3 Implementation of the Context-aware Smart Home

The OSGi-based context-aware smart home prototype has been implemented as shown in Fig. 5. The prototype system consists of an OSGi-compliant residential gateway (RG) and different kinds of devices and sensors connecting to the RG. The RG is Intel Celeron based with embedded Linux (kernel 2.4.17), developed by our team. The OSGi framework adopted is ProSyst's mBeded Server.

Figure 5. Prototype system set-up and experimental results

Context is acquired from a wide variety of context sources including sensors, profiles, devices, internal and external context services, as well as the context inference engine. The acquisition of users' location information is accomplished by indoor wireless LAN and RFID based solutions. Physical contexts are provided by embedded sensor devices (temperature sensor, light sensor, X.10 drape controller etc.). Runtime status of networked appliances (digital TV, CD player, electric water heater, projector, etc.) is collected by a device monitoring service hosted by OSGi service gateway. High-level conceptual context needed by context-aware services is either provided by user-defined profiles (e.g., user calendar or activity schedule table) or inferred from relevant context (e.g., the situational context that one is taking shower can be determined from the physical context that he is in bathroom and the water heater is turned on). Other external contexts such as future weather report are provided by external services of certain domain service providers (e.g., weather information servers).

Using the same prototype, several context-aware services are developed and tested. One is the home monitoring service: when a child is detected falling down and crying at home using audio and video analysis technique, a SMS message is sent to you and you can use remote monitoring service to see what is happening in your home. Another service is the situation-aware phone service: when the house owner is sleeping, the phone is put in silent mode automatically and all the phone calls are forwarded to the voice message box.

6 Conclusion

Context-awareness is critical to achieving the vision of future smart homes. To date, the development of context-aware systems lacks a generic model to avoid duplicate of effort and interoperability issue. In this paper, we remove this obstacle by proposing a five-layer reference model to guide the context-aware system design and deploying Semantic Web technology for context representation and reasoning. As a part of our long term research project - Context-Aware Smart Home, we are incorporating different new ideas into the OSGi-based service infrastructure for smart homes. Over time, we envision the context-aware services will significantly improve the quality of our life, benefiting particularly the elderly and disabled people in their everyday life.

Reference

1. Z. Zenn Bien. Human-friendly Man-Machine Interaction in Smart Home. Keynote speech at the 3rd International Conference On Smart homes and health Telematic, Sherbrooke, Québec, Canada, July 4-6, 2005.
2. A. Helal, W. Mann, H. Elzabadani, J. King, Y. Kaddourah and E. Jansen, "Gator Tech Smart House: A Programmable Pervasive Space", IEEE Computer magazine, March 2005, pp 64-74.
3. Mohamed Ali Feki, Stéphane Renouard, Bessam Abdulrazak, Gérard Chollet, Mounir Mokhtari. Coupling Context Awareness and Multimodality in Smart Homes Concept. The 10th International Conference on Computers Helping People with Special Needs. Lecture Notes in Computer Science 3118 Springer 2004, ISBN 3-540-22334-7. Paris, France, July, 2004.
4. Xiaohang Wang, Tao Gu, Daqing Zhang, Jinsong Dong, Hung Keng Pung: Ontology Based Context Modeling and Reasoning using OWL. To appear in Workshop on Context Modeling and Reasoning at 2nd IEEE International Conference on Pervasive Computing and Communication (PerCom'04), March 14, 2004, Orlando, Florida
5. Kindberg, T., et al.: People, Places, Things: Web Presence for The Real World. Technical Report HPL-2000-16, Hewlett-Packard Labs, 2000
6. Dey, A.K., Salber, D. Abowd, G.D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, Human-Computer Interaction (HCI) Journal, Vol. 16(2-4), pp. 97-166, 2001.
7. Anand Ranganathan, Roy H. Campbell, Arathi Ravi and Anupama Mahajan: ConChat: A Context-Aware Chat Program. IEEE Pervasive Computing, pp. 52-58, July-Sept 2002.
8. Jena2, May. 2003. http://www.hpl.hp.com/semweb/jena2.htm.

9. Fikes, R., Jenkins, J., & Frank, G.: JTP: A System Architecture and Component Library for Hybrid Reasoning. Knowledge Systems Laboratory, 2003.
10. Horrocks, I.: The FaCT System. Automated Reasoing with Analytic Tableaux and Related Methods, 1998
11. OWL-Time - http://www.isi.edu/~pan/OWL-Time.html
12. Allen, J.F.: Towards a General Theory of Action and Time. Artificial Intelligence 23, pp. 123-154, 1984.
13. Daqing Zhang, Xiaohang Wang: OSGi Based Service Infrastructure for Context Aware Connected Home. Proceeding of 1st International Conference On Smart Homes and Health Telematics (ICOST2003), IOS Press, Paris, France.
14. T. Gu, H. K. Pung, D. Q. Zhang, “Towards an OSGi-Based Infrastructure for Context-Aware Applications in Smart Homes”, IEEE Pervasive Computing, Vol. 3, Issue 4, 2004.