A Study of Service Ontology for Service Search in Ubiquitous Environment

Je-Min Kim*, Young-Tack Park

*Dept of Computer Science, Soongsil University
kimjemins@hotmail.com, park@Computing.ssu.ac.kr

Abstract

Now, computing is moving toward ubiquitous computing environments. Form of context is different each ubiquitous service system to handle it. In ubiquitous environment, it happens frequently context change between systems. So, all system must have parsers that properly change form of context. Ontology is supposed to ease shared understanding about contexts between different systems. Add to this, OWL-S will enable users and software agents to automatically discover, invoke, compose, and monitor web resources offering services, under specified constraints.

Ubiquitous service ontologies based on OWL-S are supposed to automatically discover, invoke, compose, and monitor device to provide ubiquitous service. In this paper, we propose ubiquitous service ontologies to define service that device offer in ubiquitous environment. The idea comes from using ubiquitous service ontology in model ubiquitous device service. Ubiquitous service ontologies can be used in ubiquitous service system to facilitate service device discovering and service device execution and service device composition.

Key words: Semantic Contexts, Ontology, OWL-S

1. Introduction

Ubiquitous computing environments consist of a large number of autonomous service systems that work together to transform physical spaces into smart and interactive environments. In order for an service system to function effectively in these environments, they need to perform two kinds of tasks – they need to sense and reason about the current context of the environment; and they need to interact smoothly with other service system [1].

The role of context has recently gained great importance in the field of ubiquitous computing. "Context" is any information about the circumstances, objects, or conditions by which a user is surrounded that is considered relevant to the interaction between the user and the ubiquitous computing environment [2]. Ubiquitous Computing environments are characterized by many sensors that can sense a variety of different contexts. The type of context include physical contexts (like location. time), environmental contexts (weather. light and sound level), informational contexts (stock quotes, spots scores), personal contexts (health, mood, schedule, activity), social contexts (group activity, social relation, other people in a room), application contexts (email received, website visited) and system contexts (network traffic, status of printers) [3]. Ubiquitous service systems execute suitable service to user through context awareness and context reasoning.

Form of context is different each ubiquitous service system to handle it. In ubiquitous environment, it happens frequently context change between systems. So, all system must have parsers that properly change form of context. To share knowledge between humans and software agent, ontology is model to be standardization and to have express. Indeed, service systems share context through context ontology in ubiquitous computing system.(Figure 1)

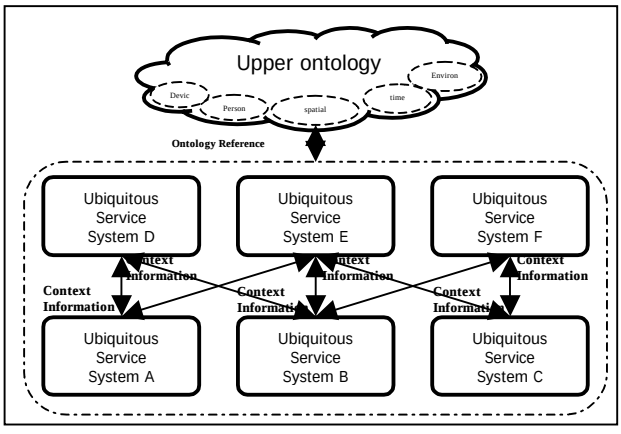

Figure 1: Ubiquitous service systems based on context ontology

The OWL language is a Semantic Web language standard backed by the W3C. As oppose to the use of

other knowledge representation scheme, the OWL language is more suitable for expressing information that is to be exchanged and shared by service systems in ubiquitous computing environment. And it has rich expressive power for defining complex ontology, and it has standard language syntaxes for computer systems to process and manipulate represented information.

There is CoBrA (Context Broker Architecture) of UMBC that is research to share context through ontology. But constructed CoBrA ontology only defines words to express context of objects. Therefore, Service systems did not recognize service that various devices provide, condition and result to execute service though this ontology in ubiquitous computing environment. For example, there is service - "if an owner entered in the room and temperature is more than 30°C, then air conditioner operates." CoBrA dose not recognize service that air conditioner provides (cooling service), condition (owner, temperature) and result to operate air conditioner. CoBrA makes use of rule based system to solve this problem. Indeed, if inputted contexts and rule condition coincide (contexts that an owner entered in the room and temperature is more than 30°C), CoBrA execute service to operate air conditioner. But, rules are bulky in ubiquitous computing environment.

Therefore, in this paper, we propose ubiquitous service ontologies to define service that device offer in ubiquitous environment. The idea comes from using ubiquitous service ontology in model ubiquitous device service. Ubiquitous service ontologies can be used in ubiquitous service system to facilitate service device discovering and service device execution and service device composition.

In this paper, we have identified several features. These are (1) standardization of contexts that various ubiquitous service systems handle - Enable semantic interoperability between different ubiquitous service systems. (2) In ubiquitous environment, define services that various devices provide - Enable automatic device discovery. (3) In ubiquitous environment, define service conditions and service results - Enable automatic device composition. (4) In ubiquitous environment, define situation information of device - Enable automatic device execution monitoring.

In the rest of the paper, we describe method to share context between ubiquitous service systems and to recognize service that devices offer in ubiquitous computing environment. In section 2, we present OWL to describe ontology and OWL-S that is service ontology to construct by OWL. In section 3, we describe the shortcomings and structure of CoBrA which is developed in UMBC. In section 4, we present ubiquitous service ontology. In section 5, we state our conclusions.

2. Related Works

2.1 The Web Ontology Language OWL

The OWL language is a Semantic Web language for use by computer applications that need to process the content of information instead of just presenting information to humans [6]. This language is developed in part of the Semantic Web initiatives sponsored by the World Wide Web Consortium (W3C).

The current human-centered web is largely encoded in HTML, which focuses largely on how text and images would be rendered for human viewing. Over the past few years we have seen a rapid increase in the use of XML as an alternative encoding, one that is intended primarily for machine processing. The machine which process XML documents can be the end consumers of the information, or they can be used to transform the information into a form appropriate for human understands.

As a representation language, XML provides essentially a mechanism to declare and use simple data structures, and thus it leaves much to be desired as a language for expressing complex knowledge. Enhancements to the basic XML, such as XML Schemas, address some of the shortcomings, but still do not result in an adequate language for representing and reasoning about the kind of knowledge essential to realizing the Semantic Web vision.

OWL is a knowledge representation language for defining and instantiating ontologies. An ontology is a formal explicit description of concepts in a domain of discourse (or classes), properties of each class describing various features and attributes of the class, and restrictions on properties [7].

The normative OWL exchange syntax is RDF/XML. Ontologies expressed in OWL are usually placed on web servers as web documents, which can be referenced by other ontologies and downloaded by applications that use ontologies.

In this paper, we described context ontology to share contexts between ubiquitous service systems through CoBrA project to construct context ontology based on OWL. The CoBrA ontology for enabling knowledge sharing and ontology reasoning based on OWL language. These ontologies play an important role in CoBrA, helping the context broker to share contextual knowledge with other agents and enabling it to reason about context.

2.2 OWL-S: Semantic Markup for Web Services

OWL-S is an ontology, within the OWL-based framework of the Semantic Web, for describing Web service. It will enable users and software agents to automatically discover, invoke, compose, and monitor web resources offering services, under specified constraints. OWL-S is including its subontologies for profiles, processes, and groundings. The ontology is still evolving, and making connections to other development efforts, such as those building ontologies

of time and resources.

The service profile tells ``what the service does''; that is, it gives the types of information needed by a service-seeking agent or system to determine whether the service meets its needs. In addition to representing the capabilities of a service, the profile can be used to express the needs of the service-seeking agent.

The service model tells ``how the service works''; that is, it describes what happens when the service is carried out. For nontrivial services (those composed of several steps over time), this description may be used by a service-seeking agent in at least four different ways: to perform a more in-depth analysis of whether the service meets its needs; (2) to compose service descriptions from multiple services to perform a specific task; (3) during the course of the service enactment, to coordinate the activities of the different participants; and (4) to monitor the execution of the service.

The service grounding specifies the details of how an agent can access a service. Typically a grounding will specify a communication protocol, message formats, and other service-specific details such as port numbers used in contacting the service.

In this paper, to use feature of OWL-S, we proposed ubiquitous service ontology to automatically discover, invoke, compose, and monitor services that every devices provide in ubiquitous environment. Indeed, to execute correct service in situation, ubiquitous service system (1) discover devices to provide suitable service, (2) operate devices to satisfy input condition(input, precondition) by user, (3) compound services to provide various devices by service planner and execute suitable service to user.

If a rule based system handle these processes, ubiquitous service system have bulky rules to discover, execute, compound service. If there are service ontologies to provide service definition, service execution-condition, service result, service systems execute efficient ubiquitous service.

3. CoBrA Ontology

CoBrA is a broker-centric agent architecture for supporting context-aware systems in smart spaces [8]. Central to the architecture is the presence of a Context Broker, an intelligent agent that runs on a resource-rich stationary computer in the space. It's responsible for acquiring and maintaining context knowledge, reasoning about the information that cannot be directly acquired from sensors (e.g., intentions, roles, temporal and spatial relations), detecting and resolving inconsistent knowledge that is stored in the shared model of context.

Harry Chen et al [9] uses the CoBrA ontology for enabling knowledge sharing and ontology reasoning based on OWL language. These Ontologies play an important role in CoBrA, helping the context broker to share contextual knowledge with other agents and enabling it to reason about context. CoBrAONT is a collection of ontologies expressed in the Web Ontology Language OWL for describing information in an intelligent meeting room environment.

The CoBrA ontologies can be used in CoBrA to facilitate knowledge sharing and ontology reasoning. The following use case describes typical uses of ontologies in CoBrA:

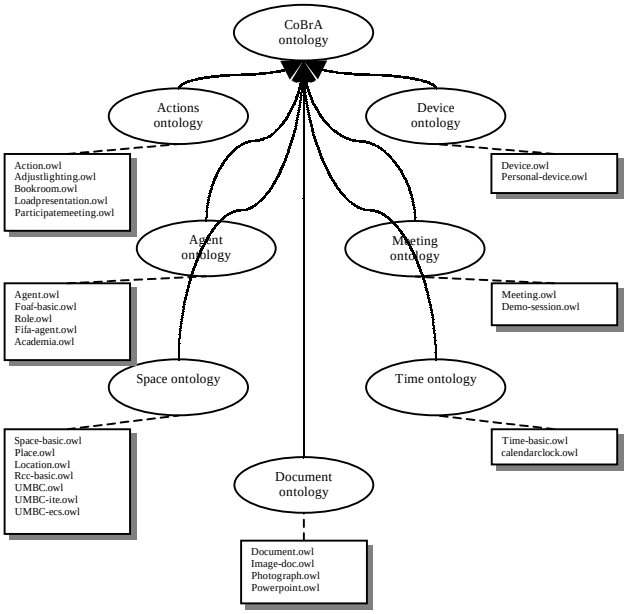

Figure 2: The structure layout of CoBrA ontologies for intelligent meeting room. Ontologies are expressed using the OWL-DL subset of the OWL language.

A sensor agent detects the presence of a Bluetooth-enabled cell phone in Room 210. It composes a description of this sensed event using COBRA-ONT, which then is sent to the context broker in the associated space. Without having any evidence to the contrary, the broker asserts that the owner the device is also in present in Room 210. Based on a physical location ontology predefined in COBRA-ONT, knowing Room 210 is a part of the Computer Science Building which in turn is a part of the UMBC campus, the context broker concludes the device owner is in school today.

But constructed CoBrA ontology only defines words to express context of objects. Therefore, Service systems did not recognize service that various devices provide, condition and result to execute service though this ontology in ubiquitous computing environment. To solve these shortcomings, ubiquitous service systems need to ubiquitous service ontology to describe service that devices provide, input-condition to use service, service

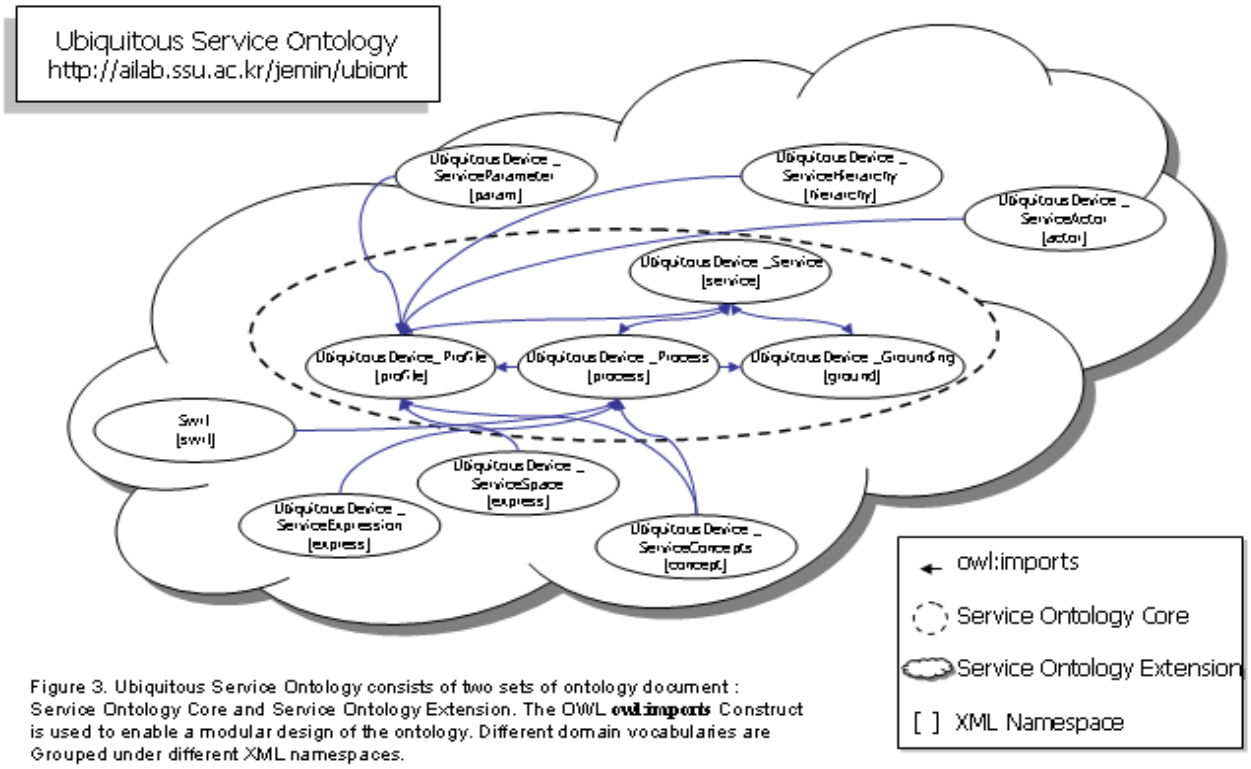

Figure 3. Ubiquitous Service Ontology consists of two sets of ontology document : Service Ontology Core and Service Ontology Extension. The OWL **owl:imports** Construct is used to enable a modular design of the ontology. Different domain vocabularies are Grouped under different XML namespaces.

result. In section 4, we explain this ubiquitous service ontology.

4. Ubiquitous Service Ontologies

In this paper, we proposed ubiquitous service ontology that based OWL-S. Ubiquitous service ontologies can be used in ubiquitous service system to facilitate service device discovering and service device execution and service device composition. The following use case describes typical uses of ubiquitous service ontologies:

In the summer, Mr. Kim arrived at home. Current temperature is 30 ℃. Mr. Kim's room installed the air conditioner and the electric fan. In situation, ubiquitous service system must to discover devices-information to provide cooling service through ubiquitous service yellow page. Then ubiquitous service system discovers information of the air conditioner and the electric fan. And, it decides service devices by used to service input-information and service precondition to need service execution. Indeed, ubiquitous service system discovers air condition and electric fan to provide cooling service. Input information and precondition of an air condition are userID, powerON, in_room(?person), (<= (?currentTem 30)). But Input information and precondition of an electric fan are userID, powerON, in_room(?person), (<=(?currentTem 28)). Ubiquitous service system operates an air condition because current temperature is 30 ℃.

Ubiquitous service ontologies consist of two distinctive related set of ontologies: Ubiquitous service core and ubiquitous service extension. The set of Ubiquitous service core ontologies attempts to define generic ubiquitous device services that are universal for different pervasive computing applications in ubiquitous environment. The set of ubiquitous service extension ontologies, extended from the core ontologies, define additional concepts for supporting specific types of devices. Figure 3 shows a diagram of the ubiquitous service ontologies and their associated relations.

4.1 Ubiquitous Service Ontology Core

This set of ontologies consists of 'Ubiquitous service profile', 'Ubiquitous service process,' and 'Ubiquitous service grounding'. The Ubiquitous service profile tells ``what the service does this device provides?"; that is, it gives the types of information needed by a ubiquitous service-seeking agent. The Ubiquitous service process tells "how the service provides"; that is, it describes what happens when the ubiquitous service is carried out. The Ubiquitous service grounding specifies the details of how an agent can access a ubiquitous service. In this paper, we described the ubiquitous service ontology, the ubiquitous service profile ontology, the ubiquitous service process ontology.

Figure 4 shows ubiquitous service based on service ontology. Service ontologies only describe overall class, property to execute service, to compound service, to discover service. To execute suitable device by ubiquitous service system, information (provided service, input information to execute device, service output, service precondition) of every device should be made service instance file. Service instance built by service provider (service manager, service coordinator) though service upper ontology.

Figure 4: Ubiquitous service based on service ontology

4.1.1 Ubiquitous service ontology

The class "Ubiquitous Device_Service" provides an organizational point of reference for declaring services that every device provide. One instance of Ubiquitous Device_Service will exist for each distinct published service. The properties presents, describedBy, and supports are properties of Ubiquitous Device_Service. The classes "Ubiquitous Device_Profile", "Ubiquitous Device_Process", and "Ubiquitous Device_Grounding" are the respective ranges of those properties. Each instance of Ubiquitous Device_Service will present a descendant class of Ubiquitous Device_Profile, be describedBy a descendant class of Ubiquitous Device_Process, and support a descendant class of Ubiquitous Device_Grounding.

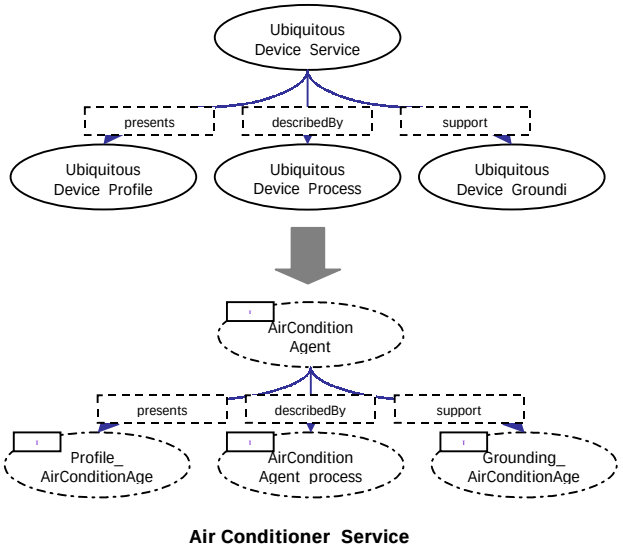

Figure 5: Top level of the service instance

Therefore, structuring of the ontology of ubiquitous service is motivated by the need to provide three essential types of knowledge about a ubiquitous service (shown in figure 5). The following shows a ubiquitous service ontology description of an air conditioner service agent. The OWL-s class 'service:Service' is define to represent a set of all device to provide service in ubiquitous environment. A service class can be described by a set of properties, which include basic profile information (Profile_AirConditionAgent), process information (AirConditionAgent_process), grounding information (Grounding_AirConditionAgent).

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-
                                        s/1.1B/Service.owl">
  <!ENTITY air_profile "http://ailab.ssu.ac.kr/JeMIn/owl-
                                        s/AirCondition_Profile.owl">
  <!ENTITY air_process "http://ailab.ssu.ac.kr/JeMIn/owl-s/
                                        AirCondition_Process.owl">
  <!ENTITY air_grounding "http://ailab.ssu.ac.kr/JeMIn/owl-s/
                                        AirCondition_Grounding.owl">
  <!ENTITY DEFAULT "http://ailab.ssu.ac.kr/JeMIn/owl-
                                        s/AirCondition_Service.owl">
]>
…
…
<owl:Ontology rdf:about="">
   <owl:versionInfo>
     $Id: AirCondition_Service.owl,v 1.10 2004/10/10 05:37:25
      Kim, Je-Min $
   </owl:versionInfo>
   <rdfs:comment>
     This ontology represents the OWL-S service description for the
     Air Conditioner service example.
   </rdfs:comment>
   <owl:imports rdf:resource="&service;" />
   <owl:imports rdf:resource="&air_profile;" />
   <owl:imports rdf:resource="&air_process;" />
   <owl:imports rdf:resource="&air_grounding;" />
 </owl:Ontology>
<service:Service rdf:ID="AirCondition_ServiceAgent">
    <service:presents
          rdf:resource="&air_profile;#Profile_AirConditionAgent"/>
    <service:describedBy
          rdf:resource="&air_process;#AirConditionAgent_Process"/>
    <service:supports

rdf:resource="&air_grounding;#Grounding_AirConditionAgent"/>
 </service:Service>
```

4.1.2 Ubiquitous service profile ontology

Ubiquitous service profile ontology gives the types of information needed by a service-seeking agent. Ubiquitous service profile does not mandate any representation of services to provide by devices, but it mandates the basic information to link any instance of ubiquitous service profile with an instance of ubiquitous service. Ubiquitous service profile consists of two set of properties. First, a ubiquitous service profile instance may have at most one service name and text description, but as many items of device information. Second, the ubiquitous service profile represents two aspects of the functionality of the ubiquitous service: the context transformation (represented by contextInputs and context Outputs) and the state of ubiquitous environment change produced by the execution of the service (represented by servicePreconditions and

serviceEffects). The following shows a ubiquitous service profile ontology description of an air conditioner service agent. The OWL-S class 'newProfileHierarchy:AirConditioner' is define to all air conditioner to provide cooling service in ubiquitous environment.

```
<owl:Ontology rdf:about="">
   <owl:versionInfo>
    $Id: AirCondition_Profile.owl,v 1.0 2004/10/15 23:53:35 Kim, Je-Min
$
   </owl:versionInfo>
   …
</owl:Ontology>
<newProfileHierarchy:AirConditioner rdf:ID="
Profile_AirConditionAgent ">
   <service:presentedBy rdf:resource="&air_service;#
                                AirCondition_ServiceAgent "/>
   <profile:has_process rdf:resource="&air_process;#
                                AirConditionAgent_Process"/>

<profile:serviceName>AirCondition_ControlAgent</profile:serviceName>
   <profile:textDescription>
         This  service control air conditioner based on the specification
         of a request.
   </profile:textDescription>
   …
<profile:serviceCategory>
     <newAddParam:Ubiq rdf:ID="Ubiquitous service category">
           <profile:value>
               Air Conditioner Control Service
           </profile:value>
           <profile:code> 561599 </profile:code>
    </newAddParam:Ubiq>
   </profile:serviceCategory>
  …
   <profile:contextInput rdf:resource="&air_process;#userID"/>
   <profile:contextInput rdf:resource="&air_process;#switchON"/>
   <profile:contextOutput
rdf:resource="&air_process;#spout_ColdWind"/>
   <profile:hasResult rdf:resource="&air_process;#cold_Air"/>
  </newProfileHierarchy:AirConditioner>
</rdf:RDF>
```

4.1.3 Ubiquitous service process ontology

Ubiquitous service processes consist of atomic service processes and composite service processes. Atomic service processes correspond to the actions a service can perform by engaging it in a single interaction between user and service systems. Atomic service processes have no sub- processes and execute in a single step, as far as the user is concerned. On the other hand, composite service processes correspond to actions that require multi-step. Composite service processes are decomposable into other (atomic service process or composite service process) processes. Their decomposition can be specified by using control constructs such as Sequence and If-Then-Else.

A ubiquitous service process can have any number of contextInputs (including zero), representing the context that is required for execute. It can have any number of contextOutputs, the context that the service process provides returns. There can be any number of servicePreconditions, which must all hold in order for the service process to be invoked. Finally, a ubiquitous service process can have any number of seviceEffects. contextOutputs and seviceEffects can depend on serviceConditions that hold true of the state of ubiquitous environment at the time the service process is performed.

```
<owl:Ontology rdf:about="">
   <owl:versionInfo>
    $Id: AirCondition_Process.owl,v 1.0 2004/10/15 00:14:57 Kim,Je-Min
$
   </owl:versionInfo>
   …
 </owl:Ontology>
<process:CompositeProcess rdf:ID= "air conditioner Process">
  <process:composedOf>
  <process:If-Then-Else>
     <process:ifCondition>
      …
          <expr:SWRL-Condition>
          <expr:expressionBody rdf:parseType="Literal">
              <swrlx:Variable rdf:ID="X1"/>
                <swrlx:IndividualPropertyAtom>
                <swrlx:propertyPredicate
rdf:resource="#UserVerification"/>
               <swrlx:argument1 rdf:resource="#X1"/>
                </swrlx:IndividualPropertyAtom>
           </expr:expressionBody>
           </expr:SWRL-Condition>
     …
     </process:ifCondition>
     <process:then>
       <process:Perform>
         <process:process
               rdf:resource="#Air_conditioner_Operation_process"/>
       </process:Perform>
     </process:then>
   </process:If-Then-Else>
  </process:composedOf>
</process:CompositeProcess>
```

We use the term 'result' to refer to a coupled contextOutput and serviceEffect. Having declared a result, a ubiquitous service process can then describe it in terms of four properties. The subCondition property specifies the condition under which this result (and not another) occurs. The withContextOutput and hasServiceEffect properties then state what ensues when the condition is true. The hasResultVar property declares variables that are bound in the subCondition. The following shows a process that provides cooling service. Cooling service execute through if current temperature is higher then criterion temperature.

```
<process:AtomicProcess rdf:ID=" Air_conditioner_Operation_process ">
 <process:contextInput/>
     <process:ContextInput rdf:ID="userID"/>
 </process:contextInput>
 <process:contextInput/>
     <process:ConTextInput rdf:ID="switchON"/>
 </process:contextInput>
 <process:contextOutput/>
      <process:ContextOutput rdf:ID=" spout_ColdWind"/>
 </process:contextOutput>
 <process:hasResult>
      <process:Result>
          <process:hasResultVar>
              <process:ResultVar rdf:ID="CurrentTemp">
                  <process:parameterType
                              rdf:resource="&temp;#Temperature"/>
              </process:ResultVar>
          </process:hasResultVar>
          <process:subCondition expressionLanguage="&expr;#KIF"
                                          rdf:dataType="&xsd;#string">
              (and (current_Temp ?CurrentTemp)
              (>= ?CurrentTemp ?criterionTemp))
          </process:subCondition>
          <process:hasServiceEffect expressionLanguage="&expr;#KIF"
                                          rdf:dataType="&xsd;#string">
              (decrease ?CurrentTemp)
          </process:hasServiceEffect>
      </process:Result>
 </process:hasResult>
</process:AtomicProcess>
```

4.2 Ubiquitous Service Ontology Extension

Ubiquitous service ontology extension are defined with a propose: define a set of concepts for supporting specific service types in ubiquitous environment. At present, ubiquitous service ontology extension consists of experimental service ontologies for supporting ubiquitous service system: light service, screen service, sound service, and temperature service. Due to space limitation, in this section, we briefly describe the existing ubiquitous service ontology extention.

UbiquitousDevice_ServiceHierarchy - This ontology provides a basic ontology of services to support the ubiquitous service examples for light service, screen service, sound service, and temperature service (shown in figure 6). It provides a example of how such an ubiquitous service ontology can be built and how it interacts with the ubiquitous service profile to define ubiquitous services.

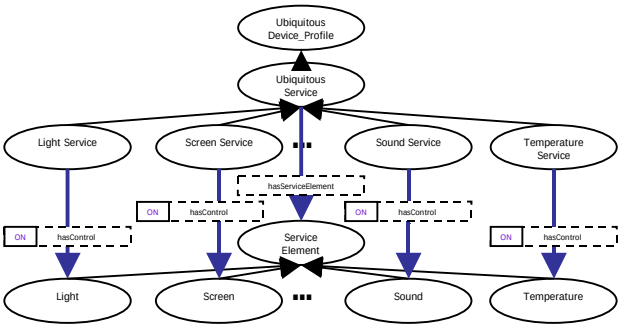

Figure 6: structure of the UbiquitousDevice_ServiceHierarchy

UbiquitousDevice_ServiceConcepts - These concepts could be defined locally, but typically they should correspond to known concepts within public service ontologies to allow for general access and usage without prior knowledge of the light service, screen service, sound service, and temperature service.

```
<owl:Ontology rdf:about="">
   <owl:versionInfo>
     $Id: NewConcepts.owl,v 1.0 2004/10/15 01:17:35 Kim,Je-Min $
   </owl:versionInfo>
   <rdfs:comment>
     DAML-S Coalition: NewConcepts used by Ubiquitous Service
Example
     for OWL-S Process Model
   </rdfs:comment>
 </owl:Ontology>
<owl:Class rdf:ID="userID">
 </owl:Class>
 <owl:Class rdf:ID="switchON">
 </owl:Class>
 <owl:Class rdf:ID=" switchOFF">
 </owl:Class>
 <owl:Class rdf:ID=" spout_ColdWind ">
 </owl:Class>
 <owl:Class rdf:ID="Bright">
 </owl:Class>
 <owl:Class rdf:ID="Dark">
 </owl:Class>
 <owl:Class rdf:ID="ScreenON">
 </owl:Class>
 <owl:Class rdf:ID="ScreenOFF">
 </owl:Class>
…
```

UbiquitousDevice_ServiceSpace – This ontology defined space to execution ubiquitous service. At present, UbiquitousDevice_ServiceSpace include bedroom, living room, dining room, powder room, office. There is a more substantive space ontology file, but it draws on very large geography ontology. We will reintroduce the more substantive space ontology file.

UbiquitousDevice_ServiceActor - This ontology adds the range element to the deviceInformation property as defined in the ubiquitous service profile. The class deviceActor contains details about the individual device that offers a service: deviceName, installSpace, providService, productNumber, manufacturingCompany, ipAddress.

UbiquitousDevice_ServiceExpression – Preconditions and effects are represented as logical formulas. An instance of this ontology represents a particular logical formalism, such as KIF, SWRL, or DRS.

5. Conclusions and future works

We proposed ontology to define service that device offer in ubiquitous environment and to define parameter to use service. There is CoBrA (Context Broker Architecture) of UMBC that is research to share context through ontology. But constructed CoBrA ontology only defines words to express context of objects. Therefore, service systems did not recognize service that various devices provide, condition and result to execute service though this ontology in ubiquitous

computing environment. So, to use feature of OWL-S, we suggested ubiquitous service ontology to automatically discover, invoke, compose, and monitor services that every devices provide in ubiquitous environment. Ubiquitous service ontologies can be used in ubiquitous service system to facilitate service device discovering and service device execution and service device composition. Ubiquitous service ontologies consist of two distinctive related set of ontologies: Ubiquitous service Core and Ubiquitous service Extension. The most important ontologies are ubiquitous service core in ubiquitous service ontologies. Ubiquitous service core consist of ubiquitous service profile, ubiquitous service process, ubiquitous service grounding. In this paper, we described the ubiquitous service ontology, the ubiquitous service profile ontology, the ubiquitous service process ontology.

References

[1] Anand Ranganathan, Roy H. Campbell, "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments", In ACM/IFIP/USENIX International Middleware Conference, 2004, Rio de Janeiro, Brazil, June 16-20, 2004

[2] Dey, A.K., et al. "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", anchor article of a special issue on Context-Aware Computing, Human-Computer Interaction(HCI) Journal, Vol.16,2001

[3] Korkea-aho, M. "Context-Aware Applications survey", http://www.hut.fi/~mkorkeaa/doc/context-aware.html

[4] Harry Chen, Tim Finin, Anupam Joshi. "An Intelligent Broker for Context-Aware Systems", Adjunct Proceedings of Ubicomp 2003, Seattle, Washington, USA, October 2003

[5] "OWL-S: Semantic Markup for Web Service", http://www.daml.org/service

[6] D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. In *Proposed Recommendation(PR) for OWL*. Web Ontology Working Group, W3C, 2003.

[7] N. F. Noy and D. L. McGuinness. Ontology development101: A guide to creating your first ontology. Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory, 2001.

[8] H. Chen, T. Finin, and A. Joshi. A context broker for building smart meeting rooms. In *Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Symposium, 2004 AAAI Spring Symposium*. AAAI, March 2004.

[9] Harry Chen AND Tim Finin AND Anupam Joshi. "An Ontology for Context-Aware Pervasive Computing Environments." In *Workshop on Ontologies and Distributed Systems*, August 2003.

[10] Harry Chen, Filip Perich, Tim Finin, "SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications", International Conference on Mobile and Ubiquitous Systems: Networking and Services, August 22, 2004