Evaluating Service Description to Guarantee Quality of U-service Ontology

Meeyeon Lee*, Jung-Won Lee**, Kyung-Ah Kim*** and Seung Soo Park*

Abstract—Efficient service description and modeling methodologies are essential for dynamic service composition to provide autonomous services for users in ubiquitous computing environments. In our previous research, we proposed a 'u-service' ontology which is an abstract and structured concept for device operations in ubiquitous environments. One of the problems that we faced during the design process was that there are not enough standards to analyze the effectiveness of a u-service ontology. In this paper, we propose a quality evaluation model to facilitate the design process of a u-service ontology. We extract modeling goals and evaluation indicators based on the u-service description specification. We also present quality metrics to quantify each of the design properties. The experiment result of the proposed quality model shows that we can use it to analyze the design of u-service ontology from various angles. Also, it shows that the model can provide a guideline, and offer appropriate recommendations for improvements.

Keywords—Quality Model, Quality Metric, Dynamic Service Composition, U-service (Ubiquitous-Service) Ontology, U-service Description Specification

1. INTRODUCTION

In dynamic ubiquitous computing environments where user location and the available devices change constantly, it is crucial to find the appropriate service depending on the run-time circumstances. To provide appropriate services in a ubiquitous environment, an effective method is required to describe and model service/context information. In our previous work [1, 2], we introduced the concept of 'u-service', u-service ontology, and u-service description specification. However, we did not offer any assessment indicators or modeling guides which should be considered during the design process. Without them, we cannot confirm the efficiency of u-service ontology constructed by an individual developer until actually testing it under all possible situations. The overall evaluation of u-service ontology is unfeasible because it is impossible to predict and test all situations which can occur in ubiquitous environments.

※ This research is supported by the ubiquitous Computing and Network (UCN) Project, the Ministry of Knowledge and Economy (MKE) Knowledge and Economy Frontier R&D Program in Korea and a result of subproject UCN 11C3-T3-10M.

※ This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0005305).

Manuscript received October 1, 2010 ; accepted January 7, 2011.

Corresponding Author: Jung-Won Lee

* Dept. of Computer Science and Engineering, Ewha Womans University, Seoul, Korea (ailmy@ewhain.net, sspark@ewha.ac.kr)

** Dept. of Electrical and Computer Engineering, Ajou University, Suwon, Korea (jungwony@ajou.ac.kr)

***Dept. of Computer Science and Information, Myongji College, Seoul, Korea (kakim@mail.mjc.ac.kr)

Most research on 'ontology evaluation' has been carried out for domain ontologies. There are few studies on design quality evaluation of service ontology, even though it is an essential module to select the best service automatically. Furthermore, there are limitations in directly applying SOA (Service-Oriented Architecture)'s design principles, which are for services in the business domain.

In this paper, we develop a quality evaluation model specifically for u-service ontology. We define the design goal for u-service modeling, and suggest design properties and measurement metrics. With the proposed model, we can evaluate the effectiveness of u-service ontology design. We also can get improvement directions from this model.

The rest of this paper is organized as follows: Section 2 briefly introduces the u-service ontology proposed in our previous works. In Section 3, we explain the proposed quality model with design properties and their metrics. Section 4 reports experiment results with representative service ontologies and our u-service ontologies, and Section 5 concludes the paper.

2. Related Work

U-service ontology proposed in our previous work [1, 2] includes information about properties of an individual u-service. U-service is the abstracted concept of operations of various devices in ubiquitous environments. The ontology organizes u-services in a hierarchical structure. Features of each u-service for dynamic discovery and composition are represented with 14 factors of description specification - ServiceName, AbsLevel, Child, Object, Precondition, Environmental/Functional Effect, and so on. All u-services are classified into three levels - ABSTRACT, COMPOSITE, and ATOMIC - according to their degrees of abstraction. U-service ontology is given a hierarchical structure by grouping them based on Environmental and Functional Effect such as 'TemperatureUP' and 'NoiseDOWN'. Fig. 1 shows a part of our u-service ontology in a tree-like structure.

ISO/IEC 9126 [3] is a standard model for evaluating quality of software products with six

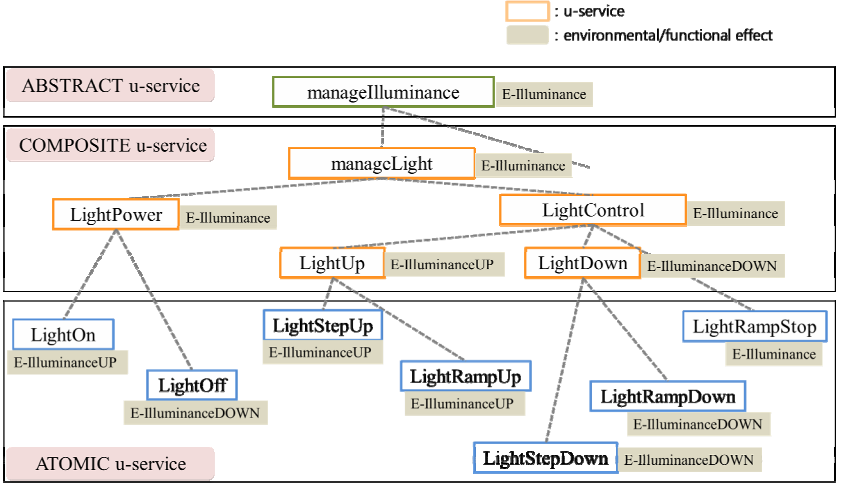

Fig. 1. A part of u-service ontology

characteristics, 27 sub-characteristics, and their metrics. Based on this model, quality models for object-oriented systems [4], SOA systems [5], and ubiquitous computing systems [6] have been proposed by analyzing their own features. Masri and Mahmoud [7] proposed a model to check performance of an individual web service regarding its execution efficiency e.g. its response time, throughput, and cost. However, these models are specialized in unique characteristics of their targets, so we cannot directly use them to assess quality of u-service ontology.

The model by Tartir et al [8] targets domain (declarative) ontology, but it is inadequate for evaluating service ontology because it considers just domain-specific factors like class, property, and relation. From these, we conclude that we need a new quality model to evaluate u-service ontology.

3. QUALITY MODEL FOR U-SERVICE ONTOLOGY

Our quality model is developed based on the development methodology of the quality model used in QMOOD [4]. Fig. 2 shows the sketched process with four steps explained in detail in the following sub-sections.

'Good' u-service ontology should be a knowledge base to support service discovery, alternative service discovery, and u-service overloading [1]. As shown in Fig. 2, we set 'dynamic service composition' as the goal of our u-service ontology, because it is the most significant feature among the qualities such as expressivity of an individual u-service and quantitative richness. We believe u-service ontology should serve as the basis to select the appropriate u-service and/or alternative ones according to the run-time context dynamically.

3.1 Design Components and Properties of U-service Ontology

The design components are identifiable elements which define the architecture of a target design [4]. They can be identified from our u-service description specification since it defines the principal elements that u-service ontology should involve and represent. With the u-service description specification [1, 2], u-service ontology can describe properties of each u-service required to check whether it is suitable for the run-time context.

Design properties are tangible concepts that can be directly assessed by examining the internal and external structure, relationship, and functionality of the design components [4]. In order to define design properties, we choose important components from 14 of the specification that

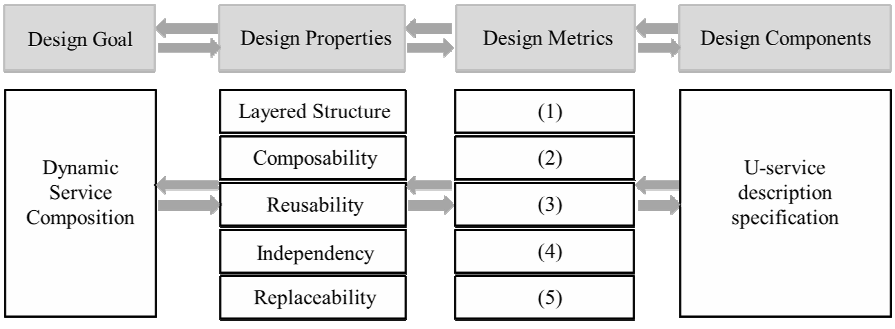

Fig. 2. The development process of a quality model for u-service ontology

are essential to describe u-services and to form the structure of a u-service ontology.

- AbsLevel: the abstraction levels of u-services. As shown in Fig. 1, there are three levels – ABSTRACT, COMPOSITE, ATOMIC. With this factor, u-service ontology has a hierarchical structure capable of searching the most appropriate u-service.
- Child: sub-u-services of the u-service in the high abstraction levels.
- Object: the platforms or devices which can perform the u-service.
- Functional/Environmental Effect: effects occurring from performing the u-service. For example, u-services in Fig. 1 can control illumination, so their Effect will be 'E-Illuminance' or 'E-IlluminanceUP' or 'E-IlluminanceDOWN'.

Design properties are derived based on these components and assess whether the u-service ontology properly represents information about them or makes full use of them to support the design goal – dynamic service composition. Fig. 3 shows the selected design components and relationships with design properties. Table 1 summarizes the definitions of design properties, and the details will be presented in the following section with their metrics.

Table 1. Design property definitions

Design Property	Definition
Layered Structure	A measure of the number of hierarchical layers formed in a design.
Composability	A measure of the number of compositions used in a design.
Reusability	A measure of the number of reused u-services in a design.
Independency	A measure of the ability to independently perform u-services on devices.
Replaceability	A measure of the ability to dynamically substitute u-services with matching Effects.

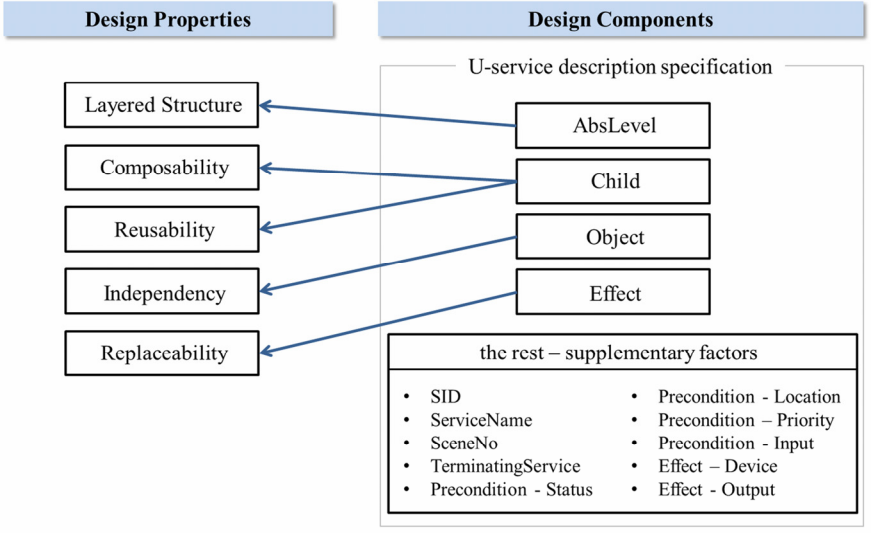

Fig. 3. Our design properties derived from design components

3.2 Design Metrics of U-service Ontology

Based on the above properties, we defined the metrics to quantify the ability to support dynamic service composition (Table 2).

Since our goal is to establish a quality model to evaluate effectiveness of u-service ontology in the design process, we excluded performance measures of the run-time such as execution times, execution success rates, execution costs, and so on. Table 2 shows the proposed quality model – design properties and their metrics. For all metrics, a high score means good quality.

We chose five properties and their quality metrics so that they constitute a plausible model to evaluate the effectiveness of u-service ontology. We now discuss the justification and the rationale behind these choices in detail.

3.2.1 Layered Structure

We adopt this property from SOA which categorizes business services into levels according to their granularity and roles. This may look similar to 'composability' in the next section, but 'layered structure' focuses on the number of hierarchical layers of ontology, while 'composability' deals with the type of compositions. All the u-services are classified into the three levels of ABSTRACT, COMPOSITE, and ATOMIC. The ABSTRACT and ATOMIC levels have a single layer, and the COMPOSITE level can have multiple layers. A multiple-layered structure like SOA is a fundamental basis for dynamic service composition. With the metric (1) in Table 2, this property can be calculated as the average number of layers for all compositions which form layers. For example, the u-service ontology in Fig. 1 has the value (5+4+2+3+2+2)/6 = 3.

3.2.2 Composability

Since required u-services keep changing depending on the run-time context, we cannot predict and predefine procedural workflows for composite u-services. In u-service ontology, the structure to discover the optimum u-service according to the context is more useful than the static structure of sub-u-services and their execution order. Therefore, we re-defined the concept of 'composite u-service' as 'a group of u-services related to a specific functional or environmental Effect'. It includes two types of u-services - ABSTRACT and COMPOSITE. Depending on the

Table 2. Quality model: design properties and design metrics

Design Goal	Design Property	Design Metric	
Dynamic Service Composition	Layered Structure	$\frac{\sum \#\ of\ layers\ of\ each\ composition}{total\ number\ of\ 'compositions(composite\ u-services)'}$	(1)
	Composability	$\frac{\#\ of\ 'non-procedural\ compositions'}{total\ number\ of\ 'compositions'}$	(2)
	Reusability	$\frac{\#\ of\ \text{re-used u-services}}{total\ number\ of\ 'u-services'}$	(3)
	Independency	$\frac{\#\ of\ 'dynamic\ \text{u-services}'}{total\ number\ of\ 'u-services'}$	(4)
	Replaceability	$\frac{\#\ of\ 'dynamic\ \text{effects}'}{total\ number\ of\ 'effects'}$	(5)

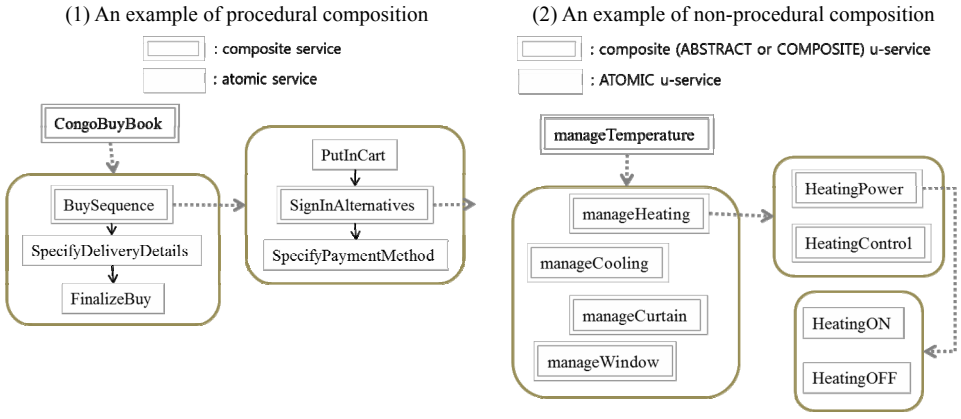

Fig. 4. Two types of service composition

run-time context, the appropriate one among its sub-u-services will be selected, rather than executing all sub-u-services in sequence. To have high 'composability', we have to design compositions of the 'non-procedural' type that do not define the execution order of sub-u-services.

As an example, the diagram in Fig. 4 (1) shows a part of the 'Congo.com' service (an online book purchasing service) [9] in OWL-S (Web Ontology Language for Services) [10]. A composite service 'CongoBuyBook' is completed with the sequential/procedural execution of three sub-services. On the other hand, a composite u-service 'manageTemperature' in Fig. 4 (2) has a group of u-services which can contribute to the 'controlling temperature' Effect. With these composite u-services, it can select and provide the pertinent ones among sub-u-services based on the real-time circumstances such as the season, the user location, the user preference, the status of devices, and so on.

3.2.3 Reusability

'Reusability' means that we can reuse predefined u-services to design a new composition. Good u-service ontology should be flexible enough to facilitate composition of services, and enhance the reusability of its sub-u-services. The more relations each u-service has with others, the more able we are to achieve meaningful abstraction, and to support dynamic actions such as service replacement. So, it is recommended not to just create many independent u-services. In the case of Fig. 5 (1), all the u-services are exclusively divided into two groups. However, in Fig. 5 (2), 'CS-2' can be a sub-u-service of 'CS-6', which enhances the reusability of the ontology. To calculate reusability, as in Table 2, we propose the proportion of re-used u-services to total u-services. The reusability value of (1) is 0/15 = 0, and (2) has 7/15 = 0.47.

3.2.4 Independency

U-services designed to be dependent on specific spaces or devices are not very useful for dynamic service composition. If u-services are designed to be provided by various platforms or devices, then it can decrease the failure rate of the system due to device problems such as their already having been used by other services or malfunction. For example, u-services 'AS-1' and 'AS-3' in Fig. 6 can be executed by a single device ('D-1'), and they cannot be served successfully if the device 'D-1' is not available. On the other hand, 'AS-4' and 'AS-5' are independent

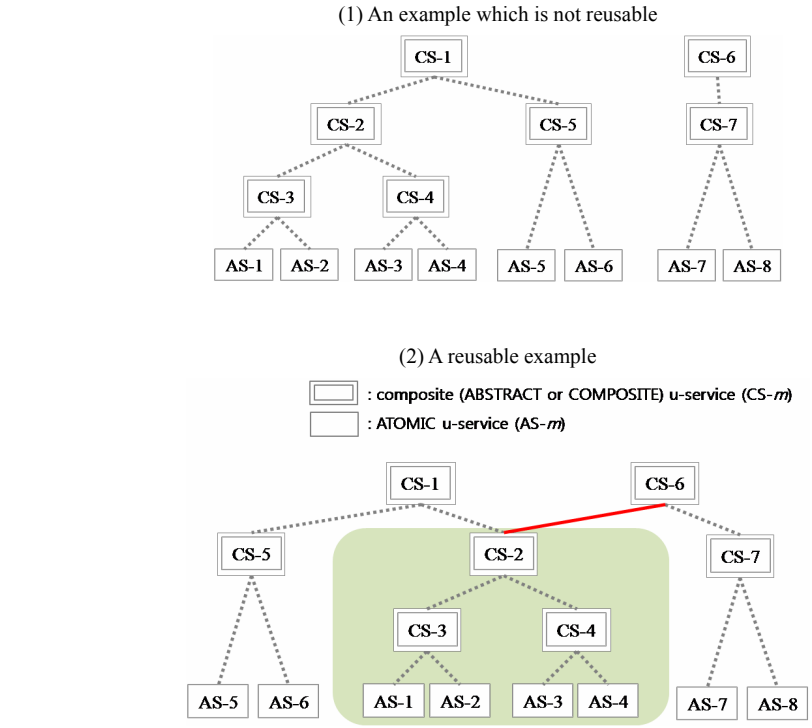

Fig. 5. An example of u-service ontology for reusability

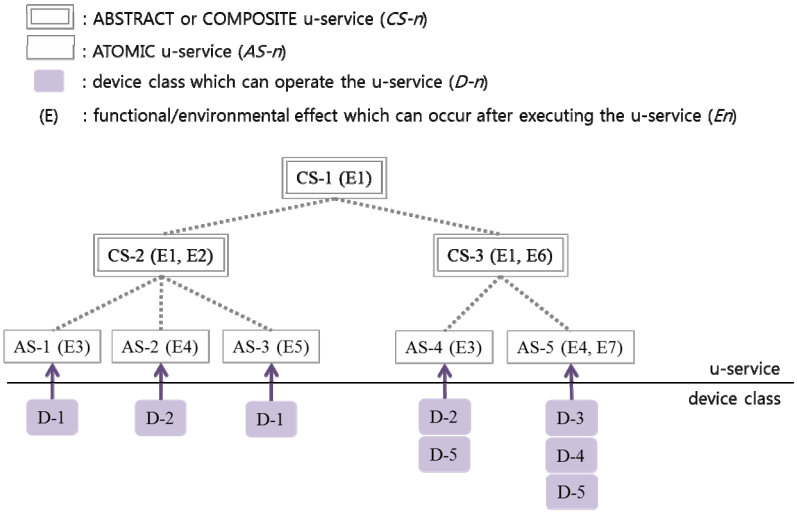

Fig. 6. An example of u-service ontology for independency and replaceability

of a specific platform, since they can be handled by other device classes. We call these u-services 'dynamic u-services' i.e. a 'u-service which can be operated by multiple (more than two) device classes or platforms'.

3.2.5 Replaceability

Replaceability means that an unavailable u-service can be replaced with other similar ones by referring to the structure of the u-service ontology. We need to design u-service ontology by structuring u-services according to Effect, so that it can identify u-services generating the same Effect. As an example, the u-service ontology in Fig. 6 represents information of each u-service, and u-services are linked and grouped together based on Effect. So, the substitution between 'AS-1' and 'AS-4' is feasible, because they can generate the same Effect 'E3'. However, if the u-service ontology has excessive 1:1 mappings between u-services and their Effects as with 'E5' and 'E7', we can say that it has bad replaceability. We define 'dynamic Effect' in the metric in Table 2 as 'Effect which can be achieved by multiple (more than two) u-services'. Under this definition, Effects 'E1', 'E3', and 'E4' are dynamic Effects.

4. EXPERIMENT

We applied the quality model to various service ontologies. The subjects of the experiments were four service ontologies: two representative OWL-S examples - Bravo Air service [11] and Cogo.com service [9] – and our two u-service ontologies. Table 3 shows their brief summary.

As a part of our research, our u-service ontology was developed and updated for u-Home and public safety domain from 2007 with Protégé [12]. For the u-Home domain, u-service ontology supports services such as managing a user's sleep environments and managing indoor temperature. And, for the public safety domain, it enables comparatively procedural services – tracing unauthorized humans in apartments, managing emergency situations (when a stranger approaches children, etc.).

The experiment results are shown in Table 4.

The Bravo Air and Congo service ontology are procedural business services comprised of a simple list of each process which is grounded in just one web service, so they showed bad quality for all design properties compared with u-service ontology. Hence, the results for two u-service ontologies are discussed and displayed in Fig. 7.

1. Layered Structure: both u-service ontologies have an average of about three layers.
2. Composability: the u-service ontology for public domain includes some composite u-services of the procedural type. For example, a composite u-service 'managing emergency situations' is achieved with execution of all sub-u-services such as 'detecting situations', 'reporting to

Table 3. Comparison of four service ontologies

target / comparison	BravoAir Service	Congo Service	U-service ontology for u-Home	U-service ontology for public
description	Service ontology for an online flight reservation service.	Service ontology for an online book buying service.	U-service ontology for u-Home domain.	U-service ontology for public safety management domain.
# of sample situations	0	0	12	52
# of u-services	7	14	151	99
# of Effects	0	0	80	86

Table 4. Experiment results

	BravoAir Service	Congo Service	U-service ontology for u-Home	U-service ontology for public
Layered Structure	10/3 = 3.33	22/6 = 3.67	164/63 = 2.60	92/34 = 2.71
Composability	0/3 = 0	1/6 = 0.17	63/63 = 1	30/34 = 0.88
Reusability	0/7 = 0	1/16 = 0.06	37/151 = 0.25	26/99 = 0.26
Independency	0/7 = 0	0/16 = 0	115/151 = 0.76	58/99 = 0.59
Replaceability	0	0	30/80 = 0.38	7/86 = 0.08

guards', 'reporting to a center', 'turning warning lights on' in sequence. So, we can tell that composite u-services for the u-Home domain are designed more efficiently.

3. Reusability: two ontologies show poor performance, which means that they include a lot of u-services that cannot be shared during service composition in the design phase.
4. Independency: from the results above, we know that a large portion of u-services for the public domain have a 1:1 relation with an execution device or platform. It implies that we might just add a lot of operations of devices without abstracting them as u-services properly. Or, in the public domain, 'Object' which can provide a specific function is unique, so it cannot be abstracted to u-services.

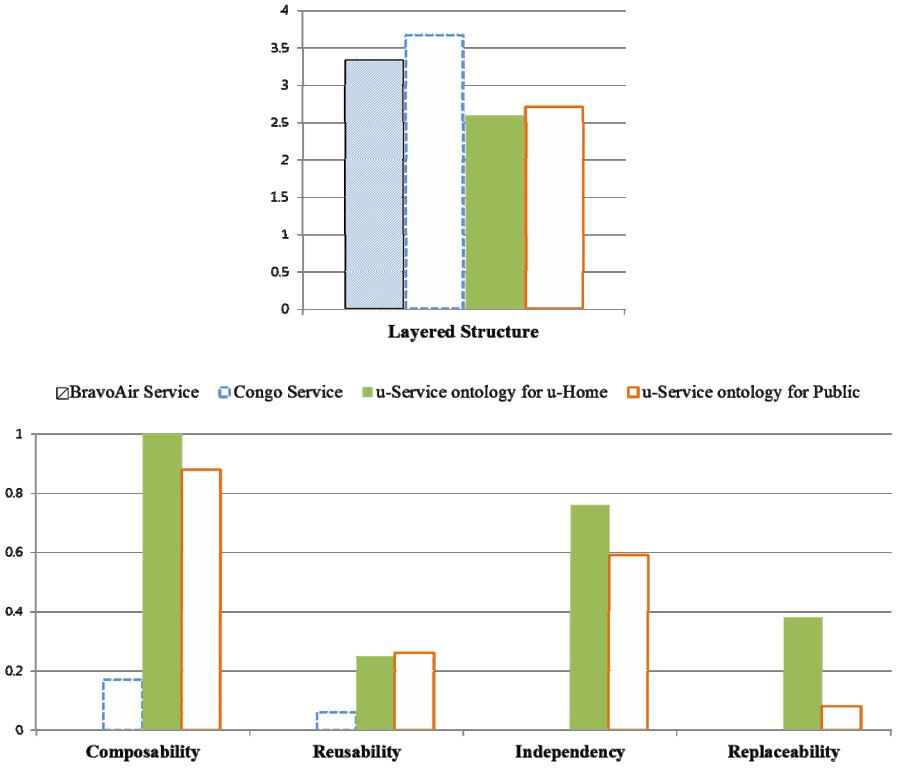

Fig. 7. Experiment results

5. Replaceability: the evaluation value for u-service ontology for the public domain is extremely low, because the number of total u-services is similar to the number of Effects. In the u-service ontology building process, we extract operations from some sample situations and abstract them as u-services, and generate/assign Effects for the u-services. When constructing it, we did not expand or enrich u-services semantically, and did not analyze u-services which can generate similar Effects. As a result, it contains a lot of 1:1 relations between u-services and Effects, which is not desirable.

In summary, among four ontologies, the u-service ontology for u-Home has mostly good quality. Since there are many situations in which services should be executed in order in the public safety management domain, the u-service ontology for the public proved to be bad u-service ontology. However, this is because of the characteristics of the domain. To comply with abstraction and reusability properties which show relatively low quality for the two u-service ontologies, we have to focus on abstracting u-services and recognizing the semantic relations of them when creating each u-service.

5. CONCLUSION

In this paper, we proposed a quality evaluation model to check efficiency of u-service ontology in the design process. Our quality model reflects the essential characteristics of u-service ontology, and can present modeling standards which can help developers to extract u-services and structure them. We selected 'dynamic service composition' as the design goal of a u-service ontology, and proposed five design properties as well as their metrics to quantify them. Using this model, we can evaluate u-service ontology and identify design factors that need to be improved.

REFERENCE

[1] M. Lee, S. Park and J. Lee, *"Ontology-based Service Layering for Facilitating Alternative Service Discovery", Proceedings of the Second International Conference on Ubiquitous Information Management and Communication (ICUIMC2008)*, Suwon, Korea, January, 2008, pp.482-487.

[2] M. Lee, J. Lee, S. Park and W. Cho, "*Ontology-based Service Description and Overloading Method for Ubiquitous Computing*", Journal of Korea Information Processing Society, Vol.15-B, No.5, 2008, pp.465-476.

[3] *ISO/IEC 9126-3 Software-Product quality-part 3: Internal Metrics*, 2003, ISO/IEC Technical Report TR 9126-3.

[4] J. Bansiya and C.G. Davis, "*A Hierarchical Model for Object-Oriented Design Quality Assessment*", IEEE Transactions on Software Engineering, Vol.28, No.1, 2002, pp.4-17.

[5] B. Shim, S. Choue, S. Kim and S. Park, "*A Design Quality Model for Service-Oriented Architecture", Proceedings of the 15th Asia-Pacific Software Engineering Conference*, Beijing, China, December, 2008, pp.403-410.

[6] S. Oh, S. Kim and S. Rhew, "UCQM: *A Quality Model for Practical Evaluation of Ubiquitous Computing Systems*", Journal of KIISE: Software and Applications, vol. 34, no. 4, 2007, pp.342-358.

[7] E. Al-Masri and Q.H. Mahmoud, "*Toward Quality-Driven Web Service Discovery*", IT Professional, Vol.10, issue 3, 2008, pp.24-28.

[8] S. Tartir, I. B. Arpinar and A.P. Sheth, "*Ontological Evaluation and Validation*", In Theory and Applications of Ontology (TAO), Vol.2, Springer-Berlin, 2008.

[9] CongoService.owl, http://www.daml.org/services/owl-s/1.1/examples.html
[10] D. Martin, et al., "OWL-S: Semantic Markup for Web Services", W3C Member Submission, 2004.
[11] BravoAirService.owl, http://www.daml.org/services/owl-s/1.1/examples.html
[12] Protégé, http://protege.stanford.edu/

Meeyeon Lee

She received BS and MS degrees in Computer Science and Engineering from Ewha Womans Univ. in 2003 and 2005, respectively. And now she is undertaking a doctorate course as a member of the artificial intelligence lab at Ewha Womans Univ. Her research interests include Ubiquitous computing, AI, Knowledge Representation, Ontology, and Quality Evaluation.

Jung-Won Lee

She received her Ph.D. in Computer Science and Engineering from Ewha Womans Univ., Seoul, Korea, in 2003. She was a researcher at LG Electronics and did an internship at the IBM Almaden Research Center, USA. Currently, she is an assistant professor of the Dept. of Electrical and Computer Engineering of Ajou University, Korea. Her areas of research include SOA and Ubiquitous Computing.

Kyung-Ah Kim

She received her Ph.D. degree in Computer Science and Engineering from Ewha Womans Univ. in 2001. She is an associate professor of the Department of Computer Science and Information at Myongji College, Seoul, Korea. Her main research interests are Ubiquitous Computing and Recommendation Systems.

Seung Soo Park

He received his Ph.D. in Computer Science from the Univ. of Texas in Austin, USA in 1988. He is a professor of the Department of Computer Science and Engineering at Ewha Womans Univ., Korea. His main research interests are Ubiquitous Computing, Semantic Web, AI, and Data Mining.