

An Evaluation of Semantic Service Discovery of a U-city Middleware

CHANG HO YUN*, YONG WOO LEE(Corresponding Author)*, HAE SUN JUNG**

U-city Consortium, The University of Seoul, Seoul, Korea

*The University of Seoul, Seoul, Korea

** Korea University, Seoul, Korea

Corresponding Author: ywlee@uos.ac.kr

Abstract— In ubiquitous-city, a middleware is essential since various kinds of unified services are required in many different applications and flexibility of service is required. It should dynamically discover and invoke services to exploit the flexibility and the power of the middleware and the service discovery is very important in the middleware. In this paper, we present a mechanism and elements of semantic service discovery which have been implemented and used in a u-city middleware which is the ever first research in the world. We evaluated the performance of the semantic service discovery in a u-city middleware and present the performance result in this paper. It is also the ever first research in the world.

Keywords—U-city, Middleware, Semantic Service Discovery, Performance Evaluation

I. INTRODUCTION

Ubiquitous computing aims to provide human-beings with IT services anytime and anywhere [1]. Context-aware computing enriches ubiquitous computing by providing intelligence: "Context-aware computing is commonly understood by those working in ubiquitous computing, where it is felt that context is a key in their efforts to disperse and enmesh computation into our lives" [2].

The u-city middleware requires the service discovery to find the appropriate service to user, since there are a lot of services in the u-city service list. Conventional service discovery cannot meet to the complex situation of the u-city. There have been many researches on semantic service discovery in the area of World Wide Web but there is no research on semantic service discovery in a u-city middleware. This is the world first research in the semantic service discovery in a u-city middleware. In this paper, we propose a semantic service discovery mechanism and elements in a u-city middleware.

The outline of the paper is organized as follows: Section 2 briefly analyses related works. Section 3 and section 4 describes the architecture of semantic service discovery in a u-city middleware, called SmartUM, and the operation (implements) respectively. Section 5 evaluates the

performance of the semantic service discovery in SmartUM. Finally, section 6 gives the conclusion.

II. RELATE WORKS

Cooltown project [3], by Hewlett Packard, tries to expand the view of physical entities to a virtual world of web content, where people, places and things are web-present. Cooltown does not have middleware. Its service discovery is done in the manner of being location aware but it is not semantic service discovery.

Easyliving [4], by Microsoft, aims to support the mobile computing technology for intelligent environment. Easyliving has the inconcert that is a middleware for distributed computing not a middleware for u-cities. Easyliving provides a facility of service abstraction for division among device control, internal logic, and user interface but does not provide any semantic service discovery.

Gaia [5], developed by University of Illinois at Urbana-Champaign, is pursuing to realize the Active Space that merges Physical Space and Software Infra. GAIA infers with many kinds of context information, and components in GAIA provide context-aware service with one's functional skill through interactions each other. It supports framework to enable users to utilize resource-aware and context-aware services with current context. However, GAIA does not support semantic service discovery. Although it is a kind of middleware to build active space it covers far smaller space and far less applications than those that a u-city requires. Thus we do not count this as a u-city middleware. Yet there is no research on service discovery in the u-city middleware except our one.

III. DESIGN DETAILS

SmartUM [6] is a prototype ubiquitous computing middleware for u-city as shown in Figure 1. SmartUM is able to adapt operations of application to the current context without explicit user intervention. SmartUM has four characteristics. First, it has the common device interface to collect the data from heterogeneous sensors and devices. Second, it derives the high level context through the intelligent inference engine based on ontology: the Context Converter, the Context

Analysers, the Context Repository and the Context Provider. Third, it performs the service discovery semantically and service execution using the inferred context information: the Service Discoverer and the Service Executor. Fourth, it has the common application interface for u-City applications.

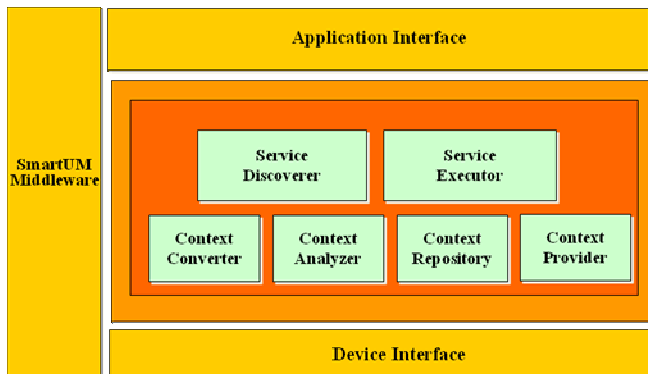


Figure 1. The architecture of SmartUM

Context Convert converts data of sensors into the context instance of OWL format based on domain ontology for inference. Context Analyser reasons the context instance based on domain ontology and user-defined rule with the pellets reasoner [7]. It provides the context information to search appropriate services. It uses data that is changed into ontology instances. Inference rules are specified with first-order logic for a variety of application [8]. The rule is loaded at inference engine and deals with ontology instances.

Context Provider manages all of the contexts in SmartUM. When context information is requested, context provider generates query statement and sends it to context repository. Context Repository executes query statement and then returns context information to the context provider. The data of sensor, context instances and reasoned contexts are saved in the context repository.

Service Discoverer searches services and Service Executor deploys and executes them according to inferred context information. Service Discoverer receives the inferred context from Context Provider and changes its format into OWL-S format to do semantic matchmaking.

WSDL2OWLS [9] in Service Discoverer collects the information of the service providers and changes it into service ontology. Service requester and service providers are described in form of OWL-S that is one of the service ontology languages. Then, the Semantic Matchmaker in the Service Discoverer performs semantic matchmaking with the domain ontology which provides common understanding.

The Service Discoverer carries out semantic matchmaking with the Silkroad that is based on greedy algorithm [10]. It uses the domain ontology that includes a variety of classified services. In order to semantically match the requester's concept of 'serviceType', 'parameters' and 'location' to the provider's concept, our service ontology is extended to include 'serviceType' and 'location' of 'profile:Profile' part. Silkroad provides ranked list according to the location between service requesters and service providers.

IV. CONTEXT-AWARE SEMANTIC DISCOVERY

We use a user-defined rule, specified in first-order logic, which is used to reason contexts. Figure 2 shows the part of the rule used in the application of the underground fire accident management. We use the reasoning facility provided by Jena API [11]. The necessary and sufficient condition to reason fire occurrence is as follows. First of all, the temperature is higher than the predefined value, for example, 120. Secondly, the value from the spark sensor should be one (true), which means that the spark is detected. Finally, the value from the smog sensor should be one (true), which means that the smoke is detected. All conditions must be satisfied at the same time.

```
[r1:  (?d rdf:type ucity:Room)
      (?d ucity:hasSensorID ?s)
      (?d ucity:hasEvent 'UnKnown'^xsd:string)
      (?s ucity:hasTempSensorValue ?num1)
      (?s ucity:hasSparkSensorValue ?num2)
      (?s ucity:hasSmogSensorValue ?num3)
      (?s ucity:hasSensorType ?t)
      equal(?t, ucity:Fire)
      greaterThan(?num1, 120)
      equal(?num2, 1)
      equal(?num3, 1)
      ->
      remove(2)
      (?d ucity:hasEvent 'Fire'^xsd:string)
]
```

Figure 2. A part of the rule for the application of the underground fire accident management

We extend the profile of OWL-S to include the 'ServiceType'. The 'ServiceType' predefines the type of services in order to help the service discoverer to classify the services before the service matching is performed.

The 'ServiceType' in the domain ontology shown in Figure 3 is classified into 'Fire', 'Environment' and 'Traffic' class. The subclasses of 'Fire' are 'FireStation', 'Hospital' and 'Police', and the subclasses of 'Environment' are 'Noise', 'Air Pollution', 'Water quality' and 'Water quantity'.

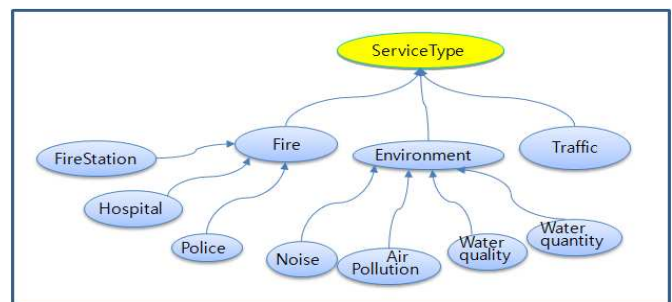


Figure 3. The 'ServiceType' domain ontology

We implement semantic service discovery which is composed of five steps; how services are described semantically, how services are advertised and their contextual information is published, how users or software agents request services, how the semantic service discovery locates services that satisfy the request, and how it uses the contextual information to rank the matching services [12]. Figure 4 describes the five steps.

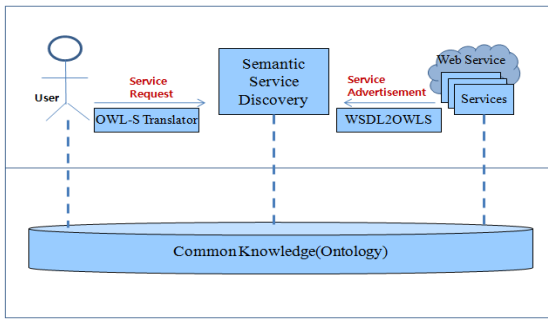


Figure 4. The mechanism of the semantic service discovery

The service provider advertises services with WSDL: 'ServiceType' and 'location' are specified by the service provider. Since the WSDL is described syntactically, the service is not semantically discovered. Hence, we have to change WSDL into service ontology in OWL-S format with WSDL2OWLS method to do semantic service discovery.

The format of the user request is converted into OWL-S format with XML2OWLS converter in order to be used by the semantic matchmaker. Now, semantic matchmaker or Silkroad compares the request with the advertisement to do matchmaking. The comparison can be performed with the content of the profile such as input, output and 'ServiceType'. After the service matchmaking is performed, as the result, it returns the matched service list.

Now, we want to rank the matched services and the context attributes such as 'location' are used to do it. The service matchmaker refers to the value of 'location' in the Context Repository.

V. PERFORMANCE EVALUATION

For the performance evaluation, we use the performance index of the precision and recall that are useful in the performance evaluation of semantic web service discovery in World Wide Web. A precision (1) is defined as the number of relevant and retrieved advertisements over the number of retrieved advertisements.

$$precision = \frac{|\{relevant_documents\} \cap \{retrieved_documents\}|}{|\{retrieved_documents\}|} \quad (1)$$

A Recall (2) is defined as the number of relevant and retrieved advertisements over the number of relevant advertisements [13].

$$recall = \frac{|\{relevant_documents\} \cap \{retrieved_documents\}|}{|\{relevant_documents\}|} \quad (2)$$

In order to evaluate semantic service discovery of the SmartUM, we used owls-tc2 [14] that provided 1007 semantic web services written in owl-s 1.1 and 20 test queries which were associated with relevance sets to conduct

performance evaluation experiments. We also used modified owls-mx [15] that was hybrid semantic web service matchmaker. Figure 5 shows the result of the performance evaluation experiments.

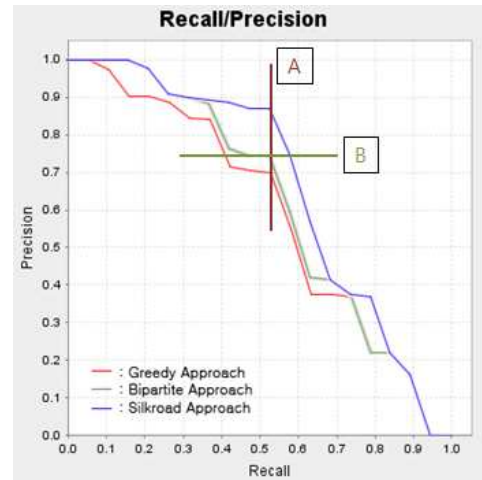


Figure 5. Performance of the semantic service discovery

In the Figure 5, the line A is used to check the value of the "precision". In line A, the value of the "recall" is fixed, that is, all curves have the same "recall" value. Therefore, we can find which curve has better "precision" when the "recall" value is fixed to a value or same each other. The line B is used to check the value of the "recall". In line B, the value of the "precision" is fixed, that is, all curves have the same "precision" value. Therefore, we can find which curve has better "recall" when the "precision" value is fixed to a value or same each other. We can see our semantic service discovery based on Silkroad shows better performance in terms of the "precision" and the "recall" than those which use Greedy matching or Bipartite graph based matching.

Greedy matchmaking algorithm has four matching degrees. It has false positives and false negatives [16]. Bipartite graph based matching regulates false positives and false negatives. However, like the previous suggestion, it does not provide ranking of the matched results. [17] We use the service type to classify further.

VI. CONCLUSIONS

In this paper, we have described the mechanism and the elements of semantic service discovery in a u-city middleware which is the ever world first in the area of u-city. Semantic service discovery has been actively studied and used in the area of World Wide Web. We applied the essence of semantic service discovery mechanism to the u-city middleware and showed what and how we had done in this paper. Since there has been no u-city middleware except our one, there is no well-established methodology and index for the performance

evaluation of the semantic service discovery in u-city before our work. Here in this paper, we introduced the performance result when we applied the performance index which was used for the evaluation of semantic service discovery in World Wide Web. In the performance result, we showed that our approach produced good performance when we compared our performance result with the performance results of other reference cases. Since our approach used in the u-city and used for the performance evaluation are just the first step in the area, we think it is just the beginning and have to do many things in our further research.

ACKNOWLEDGMENT

This study was supported by the Seoul Research and Business Development Program (10561), Smart (Ubiquitous) City Consortium and Seoul Grid Center. We would like to give thanks to, Miss Ji-Sung Kim, Mr. Jong-Won Park, Mr. Seung-Woo Rho, Mr. Sun-Cheol Paik, Mr. Hyun-Kyu Park, Mr. Cheol Sang Yoon, Mr. Kyoung Kyu LEE, Mr. Phil Doo Hong, Mr. Chang-Won LEE and the staffs of Seoul Grid Center and the members of Smart (Ubiquitous) City Consortium for their contribution to this research.

REFERENCES

- [1] M. Weiser, "Some computer science issues in ubiquitous computing," CACM, vol. 36, pp. 74-83, July 1993.
- [2] T. Moran and P. Dourish, "Introduction to this special issue on context-aware computing," Human-Computer Interaction, vol. 16, pp. 87-95, 2001.
- [3] Location Awareness in HP's CoolTown, [Online]. Available: <http://www.w3.org/Mobile/posdep/HPw3cwapref.html>
- [4] B. Brumitt, et al., *Easyliving: Technologies for intelligent environments*, Lecture notes in computer science, p. 12-29, 2000.
- [5] M. Roman, et al., *Gaia: A middleware infrastructure to enable active spaces*, IEEE Pervasive Computing, vol. 1, no. 4, p. 74-83, 2002.
- [6] H. S. Jung, C. S. Jeong, Y. W. Lee, and P. D. Hong, *An Intelligent Ubiquitous Middleware for U-City: SmartUM*, Journal of Information Science and Engineering, vol. 25, pp. 375-388, 2009.
- [7] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical OWL-DL reasoner," Journal of Web Semantics, vol. 5, pp. 51-53, 2007.
- [8] A. Ranganathan and R. Campbell, *An infrastructure for context-awareness based on first order logic*, Personal and Ubiquitous Computing, vol. 7, pp. 353-364, 2003.
- [9] WSDL2OWLS. [Online]. Available: <http://www.mindswap.org/2004/owl-s/api/doc/javadoc/examples/WSDL2OWLS.html>
- [10] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, *Semantic matching of web services capabilities*, Lecture notes in computer science, p. 333-347, 2002.
- [11] Jena API framework. [Online]. Available: <http://jena.sourceforge.net/inference/index.html>
- [12] A. El-Sayed and J. Black, "Semantic-based context-aware service discovery in pervasive-computing environments," Proc. of the 1st WS on SIPE at IEEE ICPS, 2006.
- [13] U. Kuster, H. Lausen, and B. Konig-Ries, *Evaluation of semantic service discovery-a survey and directions for future research*, Emerging Web Services Technology, p. 41-58, 2008.
- [14] OWLS-TC. [Online]. Available: <http://projects.semwebcentral.org/projects/owls-tc/>
- [15] M. Klusch, B. Fries, and K. Sycara, *OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services*, Web Semantics: Science, Services and Agents on the World Wide Web, 7(2), p. 121-133, 2009.
- [16] Witold Bednorz, *Advances in Greedy Algorithms*, I-Tech, Vienna, Austria, ISBN 978-953-7619-27-5, p. 586, Nov. 2008.
- [17] B. Umesh and R. Kulkarni, *Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching*, IEEE International Conference on Web Service 2007, p. 86-93, 2007.