Service Composition Techniques Using Data Mining for Ubiquitous Computing Environments

Sun Young Lee[†], Jong Yun Lee[†], and Byung Il Lee[††]

[†]Department of Computer Education, Chungbuk National University, South Korea
[††] Department of Computer Engineering, Chungbuk National University, South Korea

Summary
When users want to have a suitable application in ubiquitous computing environments, it is necessary to discover and compose primitive services considering context information. However, previous works focused on only service discovery for user queries and lack the consideration of context information such as location, time, network, computing environment, and preference for users. They also do not use the service history information on service composition. Therefore, we present a framework for a service provisioning middleware system that can discover primitive services and compose dynamic complex services according to the context information. We also describe an algorithm of service composition which uses the service history information and an ontology engine with data mining. Finally, we show that our experiments enhance the possibility of provisioning services considering user's preference and thus provide users with newly composed optimal services.

Key words:
Service composition, ubiquitous computing, service provisioning Data mining

1. Introduction

In ubiquitous computing environments, available resources and services changes over the mobility for users and devices. It is also important to provide users with a best-fit service considering the context information because users want to have an optimal service in the surrounding circumstance. If the suitable services for user requests do not exist in primitive service databases (i.e. service databases), it is necessary to compose primitive services. A service composition is the construction of complex services from basic services and the connection of services implementing a logic that depends on the application domain and on the control flow [1]. The service composition enables users to utilize services in the environment to solve complex queries [2].

1.1 Research Motivation

There have been studied on many service discovery protocols such as JINI [3], UPnP [4], [5], [6], [7], and [8]. JINI [3] was developed to support the federation between a user and resources by Sun Microsystems Corporation and UPnP [4] was proposed by Microsoft Corporation using existent IP and HTTP protocols. However, the previous works focused on only the service discovery and did not consider context information. Therefore, we need a service provisioning mechanism that discovers and composes services according to the context information.

In addition, existing service composition techniques [1, 9, 10, 11, 12] are only a discovery of primitive services and do not consider the usage historical information for services. In [1], a service composition module manages service retrievals and composes stored primitive services when a user requests an application query. For example, USON [9, 10] searches STs (Service Template) at service composition step based on a query and location information and selects a candidate among the STs. The detection of SE (Service Entity) requests to fix to the ST. Service composition has a feature that services make a group with correlation. If there is not a direct correlation among services, however, we can discover a newly composed service using the service history information. Therefore, we need to provide a new service composition method using service history rules.

1.2 Contributions

To solve these problems, we present a service composition technique with data mining considering the context information and the service usage history information. We can summarize our results as follows. First, we design a framework for context-aware service provisioning system, called COSEP, which dynamically provides services according to context information. Second, we present an algorithm of service composition using an ontology engine with data mining function, where it can discover primitive services and compose new complex services considering service historical information. Finally, we believe that our results enhance the possibility for service provision and thus provide users with newly composed optimal services.
The rest of this paper is organized as follows: Section 2 discusses related work and presents challenges. Section 3

Manuscript received September 5, 2006.
Manuscript revised September 25, 2006.

describes the structure of context-aware service provisioning middleware system. Section 4 presents our ontology engine with data mining and Section 5 shows simulation results. Section 6, we conclude briefly. From this section, input the body of your manuscript according to the constitution that you had. For detailed information for authors, please refer to [1].

2. Related Work

This section reviews previous works for a framework in ubiquitous computing environment and for service composition techniques.

2.1 Frameworks

Jini[3] was developed by Sun Microsystems and implemented using Java. All members of the network are known as services. A service proxy object is registered with a service registry, called a lookup service. Once the lookup service of a suitable group has been located, the service provider will upload its service proxy object. A client service in search of a certain service type will contact the lookup service to download the proxy object. Another strong aspect of the Jini is independent from any platform since it is based on Java and uses Java RMI for communication. This makes Jini dependant on a programming language Java as well as requires a java virtual machine JVM which consumes a large part of the device's memory and resources. Furthermore, a detailed standardization process is necessary if services are to be represented in Java classes.

UPnP [4] is a middleware solution proposed by Microsoft Corporation using existent IP and HTTP protocols. It is designed to bring easy-to-use, flexible, and standards-based connectivity to ad-hoc or unmanaged networks whether in the home, in a small business, public spaces, or attached to the Internet. SLP [5] is an enterprise protocol to search public resource in a large network generated by IETF. In addition, there have been proposed service discovery protocols such as Bluetooth SDP [7], Salutation [6], HAVi [13], etc. There have been studied on several frameworks for provisioning services using the context information, such as UbiCOSM [14], reggie [15], and COPS-SD [16]. The UbiCOSM is a context-based access control framework for service design and deployment and the reggie is context-awareness service discovery. In a corporate network with various levels of users' authorization or in an operator network specifying several classes of service level agreements, COPS-SD is a protocol which can discover and use authorized services depending on a user level.

2.2 Service Composition

Service composition refers to the technique of creating complex services with the help of smaller, simpler and easily executable services or component [12]. These techniques can be categorized into two types: static service composition and dynamic service composition. The static service composition is an approach in which application designers implement a new application manually by designing a workflow or a state chart describing the interaction pattern among components. On the other hand, the dynamic service composition composes an application autonomously when a user queries for an application.
UBIDEV [17] proposes a context centric management of the environment. This allows applications to automatically reconfigure themselves according to context changes and represents a unified management model for resources, services and context information at an application level as a homogeneous coordination space. In this model, *the Coordination Manager* at *the Ubiquitous Access layer* decomposes complex queries coming from *the Application layer* in terms of composing primitive services. *The Service Manager* is responsible for instantiating and monitoring services. It also analyzes the resource classification to bind resources and services depending on the application request. *The Context Manger* classifies the context information into context types according to the application ontology.

CB-SeC[1] was proposed to increase the functions of service discovery and composition considering the contextual information in order to satisfy user requests regardless of position and resource in pervasive computing environment. The system hierarchically consists of four layers, such as *Application layer*, *Service Provisioning layer*, *Context Management layer*, and *Physical Entities layer*. *The Service Provisioning layer* is responsible for carrying out the process of managing the discovery of services to yield a composite service. Indeed when a client queries an application service, it is necessary to compose a complex service from primitive services. The composite service that satisfies all the user preference is chosen and the corresponding capsules are sent to t*he Service Execution module*. Both UBIDEV and CB-SeC provide a set of basic services for context information, but they do not consider service history information.

In USON [9, 10], a service is provided through the combination of SEs (Service Entity). A SE is built by an ST (Service Template) which acts as a meta-design for the service. Service composition is achieved through the actions of the USON engine as follows. First, the user's USON engine requests to discover STS for the USON network. This request is based on information on a user including what he wants, where he is located at, etc. Second, the USON network handles STs and SEs according to their advertising qualities. Third, after one or more STs obtained, the USON engine selects a candidate from among them and requests the USON network to discover the SEs that are suitable for the candidates. Next, we can successfully obtain SEs and the USON engine invokes the methods of the SEs. Then the actions of SEs provide the services for the user request. During these steps, the information on the usage history of the STs and SEs is stored in the USON network and this information is used for service emergence.

In [2], a broker-based distributed service composition protocol is proposed for pervasive environments. The protocol enhances the flexibility of service discovery by using GSD discovery protocol [18] and the efficiency of service discovery by using a broadcast-based service discovery mechanism. This distributed broker-based composition architecture performs better than the centralized solution in terms of composition efficiency, broker arbitration efficiency and composition radius.

The semantics-based dynamic service composition architecture [12, 19 and 20] assumes that a user requests a service in an intuitive manner and dynamically composes the requested service based on its semantics. The proposed architecture consists of a semantics-aware component model CoSMoS, a middleware CoRE, and a semantics-based service composition mechanism SeGSeC. The dynamic service composition has the potential to realize flexible and adaptable applications by properly selecting and combining components based on the user request and context. For service composition, however, it is important to discover primitive services simply as well as to compose complex queries from the primitive services considering the context information and the service usage history for users.

3. A Framework for Service Provisioning Middleware System

This section presents a service provisioning middleware framework that can discover and compose services considering the context information and service usage pattern history for users. In ubiquitous environments, a platform consists of several hardware and software components to provide services effectively (Fig. 1). That is, a platform can consist of service providers, service consumers, and a service mediator (i.e. COSEP server) and the peer-to-peer communication between service provider and service consumer (Fig. 2).

In Fig. 2, *the Service Provider* registers available services to a COSEP server and receives service requests from the server. *The Service Consumer* sends an application queries to the COSEP. After *the Service Providers* and *Service Consumer* identify available services from COSEP, *the Service Consumer* communicates with *Service Providers* directly through the peer-to-peer communication. The service mediator is a middleware system that identifies service providers and service consumers like a network server and shall be divided into four layers (Fig. 1): An application layer, a service provisioning layer, a context awareness layer, and a physical layer.

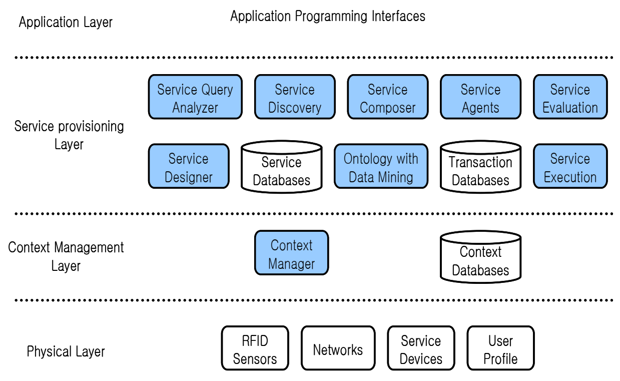


Fig. 1. Components of service provisioning middleware system at COSEP

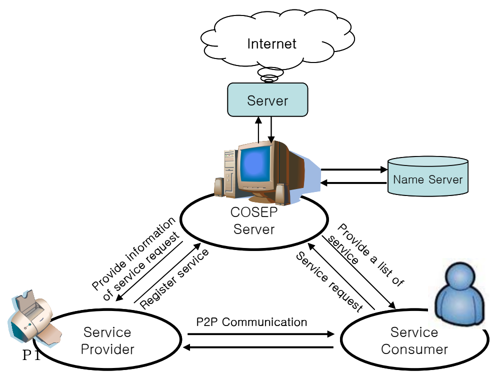


Fig. 2. Service provisioning platform

First, the physical layer represents a federation of physical computing devices and sensors such as RFID sensor, mobile and online networks, databases, service devices (PC, notebook, printer, WiBro phone, DMB, etc.) and user profiles. Second, the context management layer is responsible for gathering, processing, representing context information from hardware and software sensors at the physical layer. It also classifies this information and stores into the context databases, where the context information needs to be structured according to the application conceptual model. Thus, the context management layer consists of a context manager and context databases. Third, the service provisioning layer is responsible for analyzing, discovering, composing, executing the requested services and caching the serviced results. It also includes service agents for automatically designing, registering, and gathering the services, in addition to storing requested queries into transactional databases for service usage history and analyzing them. Fourth, the application layer works as the application programming interfaces (API) between service providers and service consumers and embodies a generalization of application results.

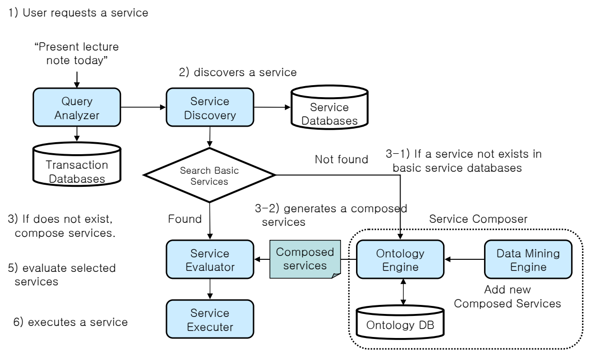


Fig. 3. A process of handling a query for the service provisioning layer

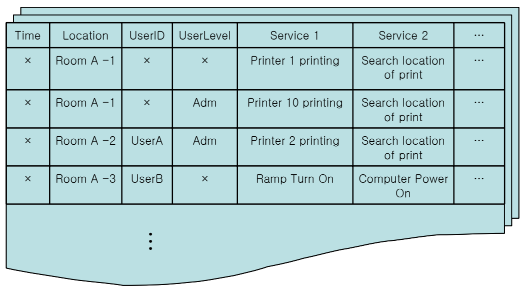
Fig. 3 shows a process of handling a query for the service provisioning layer at COSEP. In Fig. 3, when a user requests a service query, *the Query Analyzer* parses and stores it into *the Transactional Databases* as the service historical information. *The Service Discovery* receives the parsing query from *the Query Analyzer* and discovers available resource types for the user request from *the Service Databases*. If the available services exist as basic components on *the Service Databases*, they are passed into *the Service Evaluation*; otherwise, *the Service Composer* builds a new complex service based on context information and registers it into *the Transactional Databases* as well. Upon composing a complicate service, we use the context information such as location, time, network, databases, devices, and user's preference. Upon building a new service, *the Service Composer* uses an *Ontology Engine* with data mining function. After discovering the services, *the Service Evaluation* examines whether the service semantics is compatible with the user request and it is an optimal service or not. Then, the evaluated services are executed immediately.

4. Service Composition Using Ontology Engine with Data Mining

This section presents a service composition technique for context-aware service provisioning middleware system. *The Service Composer* builds new services based on context information for users and registers them into *the Transactional Databases*. *The Service Composer* also consists of an ontology engine, a data mining engine and ontology databases. *The Ontology Engine* dynamically composes complex services by using predefined rules in ontology. For transactions, *the Data Mining Engine* describes the discovered usage patterns for services and store the usage association rules into *the Ontology Databases*. For the detailed information on *the Ontology Engine*, we describe below.

4.1 The Ontology Engine

The Ontology Engine composes new complex services by using predefined rules in ontology. For example, Fig. 4 shows an example of *Ontology Engine* at the context-aware service provisioning middleware system.



Time	Location	UserID	UserLevel	Service 1	Service 2	···
×	Room A −1	×	×	Printer 1 printing	Search location of print	···
×	Room A −1	×	Adm	Printer 10 printing	Search location of print	···
×	Room A −2	UserA	Adm	Printer 2 printing	Search location of print	···
×	Room A −3	UserB	×	Ramp Turn On	Computer Power On	···
⋮						

Fig. 4. Example of service composition using an ontology engine at COSEP

For a user, available services depend on his or her location, identifier, level, and etc. If with no level, a man is located at a room A-1 in Fig. 4, then he can use Printer 1. On the other hand, with an administration level a man can use "Printer 10." To explain a service composition, assume that a service consumer A is located at room A-1 and has an administration level. When the service consumer requests a printing service, COSEP identifies user's context information and discover an optimal service like "Printer 10 printing and search location of printer in Fig. 4." After that, a user A can use "Printer 10." Like this, the Ontology Engine only uses registered services and thus we need a Data Mining Engine for the sake of composing a new service at COSEP.

When a consumer requests a print service, a service composition should be processed as an event driven diagram of service provisioning protocol, as shown in Fig. 5. When a service consumer requests a query, COSEP receives context information from the Context Manager. The Service Discovery at COSEP discovers available resource types from the Service Databases. If the Service Discovery cannot find it in Service Databases, the Service Composer composes new complex services by the Ontology Engine and Data Mining Engine, and registers it to *the Ontology Databases*. Then the query will be evaluated and executed immediately.

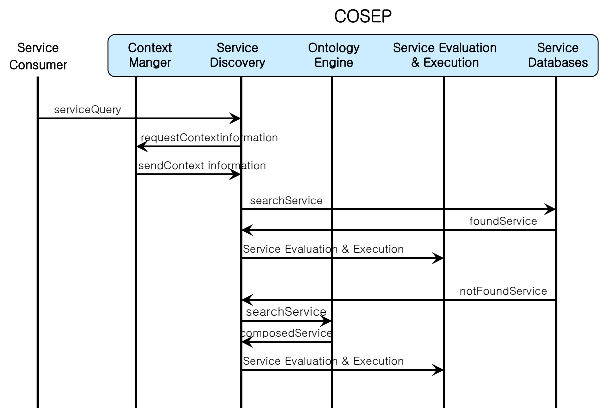


Fig. 5. An event diagram of service provisioning for a service request

4.2 The Ontology Engine with Data Mining

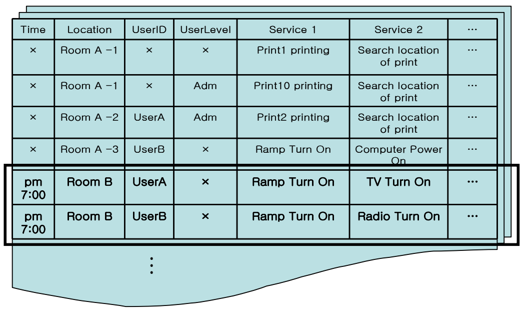
Data Mining refers to extracting or "mining" knowledge from large amounts of data [21, 23]. Data mining is also named "Knowledge mining from data." Data mining technique is used to describe patterns discovered through mining. For our ubiquitous computing environments, when a service consumer requests a query to COSEP, the query is stored into the Transaction Databases as service usage historical data. *The Data Mining Engine* discovers an association rule from the Transaction Databases and registers it at *the Ontology Engine* as a new complex service. When a service consumer comes back home, for example, he or she turns on lamps and watches TV everyday. These are stored into *the Transaction Databases* as historical data. Then, *the Data Mining Engine* periodically extracts some knowledge from *the Transaction Databases*. Consequently, two services, turning on lamps and then watching TV, will be composed as a new service for the service consumer and the new service is registered in *the Ontology Engine*. *The Ontology Engine* with data mining provides primitive services as well as complex services through service composition against the context information. Fig. 6 shows an algorithm of service composition using *Apriori* algorithm for data mining.

In Fig. 6, there are two input parameters: one is *the Transaction Databases (TDB)* that stores all the service queries; another is *a Minimum Support Count, MIN_SUPPORT_COUNT*. The support count is the number of transactions that contain the service queries. The number of transactions required to satisfy minimum support is referred to as the minimum support count. The output is a new composed service, named *ComposedServices*. At step 2, the algorithm begins to find service composition rules for each service query on *Transaction Databases*. At steps 3 to 4, the algorithm discovers new composed services using the apriori_gen(), *Apriori* algorithm, and then determines whether the composed service exist in *the Ontology Databases* or not. At step 5, if new, then it is added to *the Ontology Databases*. In this step, *the Ontology Engine* should be rebuilt by adding new composition rules. For each query in *the Transaction Databases*, the above processes will continue.

```
Procedure Service Composition
Input: Transactional Databases (TDB); MIN_SUPPORT_COUNT
Output: ComposedServices
1   initialize ComposedServices;
2   For each transaction $t \in TDB$
3     ComposedServices = apriori_gen(t, MIN_SUPPORT_COUNT);
                 // build a composed service using Apriori Algorithm
4     If ComposedServices is not found in OntologyDatabases then
                 // determine if they are existing composition services
5       add ComposedServices to OntologyDatabases;
6     End if
7   End for
```

Fig. **6**. Service composition using an *Aprior*i algorithm

For example, assume that service consumers A and B use different services at home in the evening. When a service consumer comes back home, a userA turns on lamps and watches TV everyday and a userB turns on lamps and listens to music. In these cases, the *Data Mining Engine* analyzes the service usage patterns for each user. For each user, new composed services are added to *the Ontology Engine* again, as shown in Fig. 7. Consequently, *the Ontology Engine* provides dynamic service composition for user queries in real-time, and *the Data Mining Engine* builds new complex services.



Time	Location	UserID	UserLevel	Service 1	Service 2	…
×	Room A −1	×	×	Print1 printing	Search location of print	…
×	Room A −1	×	Adm	Print10 printing	Search location of print	…
×	Room A −2	UserA	Adm	Print2 printing	Search location of print	…
×	Room A −3	UserB	×	Ramp Turn On	Computer Power On	…
pm 7:00	**Room B**	**UserA**	**×**	**Ramp Turn On**	**TV Turn On**	**…**
pm 7:00	**Room B**	**UserB**	**×**	**Ramp Turn On**	**Radio Turn On**	**…**

⋮

Fig. 7. An additional service composition through data mining

5. Experiments

In this section, we describe simulation environments, experiment the performance of ontology engine with data mining and analyze the performance of enhanced service composition through simulation results. In our experiments, we used Intel Pentium 4 2.8 GHz with 1 gigabytes RAM, 160 gigabytes HDD and Window XP. We have implemented a Service Composition using Eclipse editor in Java. We implement an *Apriori* algorithm that searches suitable association rules using support count and confidence.

5.1 Simulation Data

For our experiments, we used 30 primitive services that have 101 through 130 identifiers in *Service Databases*. Initially, *the Ontology Databases* stores the static primitive services. *The CreateOntology class* randomly selects the primitive services of from two to five and then composes them. In our simulation, there are 100 service compositions in *Ontology Databases*. Table 1 illustrates some possible service compositions for *Ontology Databases*. Transaction databases stores the service usage history information at a timestamp. For transaction databases, each transaction is built by the primitive services of two through five from *the PrimitiveService Databases*. For our transaction data, the primitive services 101 through 110 occurs very often, the primitive services 111 through 120 occurs medium, and the primitive services 121 through 130 occurs at rare intervals. For the above transactional data, we give different frequencies for primitive services, because it can increase the probability of service composition by data mining later. For our simulation, the number of transaction data is limited to 10,000 and its timestamp is classified by five groups. Table 2 illustrates some transaction data.

Table 1. Service composition in Ontology databases

Ontology ID	PrimitiveServices
1	106 113 127
2	117 119 129
3	108 112 114 115
4	113 120 128 129
5	101 129
6	101 121
7	116 121
8	129 130
9	103 105 113 117

Table **2**. Transaction data at timestamp 1

TransactionID	PrimitiveServices	Timestamps
1	107 109 117	1
2	101	1
3	105 109	1
4	107 110 116 120 125	1
5	106 107 120	1
6	101 106 115	1
7	102 105 120	1
8	109 117 118	1
9	101 105	1

5.2 Simulation Results

For our experiments, we measure the number of service composition by using *the Ontology Engine with data mining*. First, we measure the number of service compositions at single time. During the time, we examine the number of services composed by *the Ontology Engine* and *the Data Mining Engine*, respectively. Second, we measure the number of service compositions, depending on the number of transaction over time. After storing each transaction, *the Data Mining Engine* builds new service composition by analyzing the service history data.

(1) The number of services composed at single time

For our experiments, assume that we fix the number of transactions to 10,000. We only extract application services composed by *the Data Mining Engine* except the primitive services in *Ontology databases*. Fig. 8 illustrates the number of service composition generated by different engines such as *the Ontology Engine, the Data Mining Engine*, and *the Ontology Engine with Data Mining*. There are 100 service compositions by *the Ontology Engine*. This is the same as the number of predefined service compositions. The number of service composition generated by *the Data Mining Engine* is 210 and the total services composed by *the Ontology Engine with Data Mining* are 310. Notice that our approach (i.e. *the Ontology Engine with Data Mining*) enhances the service compositions about two times. Consequently, we believe that combining *the Ontology Engine* with *the Data Mining Engine* results in better service provision for users.

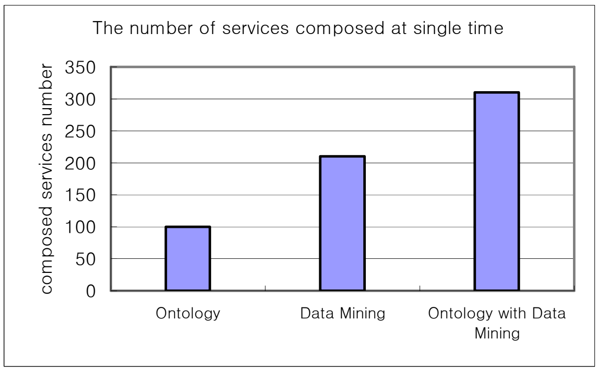


Fig. 8. The number of composed services at single time

(2) The number of service composition over time

In this simulation, we analyze transaction data over time. The number of transaction increase 2000 at each timestamp. Notice that service compositions using Data mining method increase every timestamp in Fig. 9. Like the above experiment (1), we only extract the service composition built by *the Data Mining Engine* except the primitive services in *the Ontology Databases*. The more the number of transaction exists, the more the service compositions generate. For our experiment, notice that our approach enhances the ratio of service compositions two times.

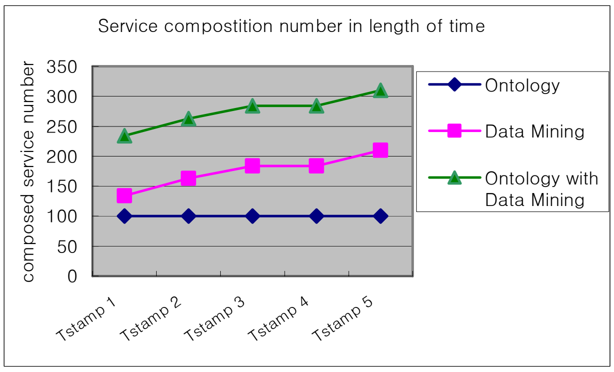


Fig. 9 The number of service composition over time

6. Conclusions

When users want to have services correctly in their surrounding circumstance in ubiquitous computing environment, it is very important to discover and compose primitive services according to the context information. However, previous works focused on only service discovery for user queries and lacked the consideration of context information. They also did not use the service history information on service composition. In this paper, we present a framework for a service provisioning middleware system that can discover and provide dynamic services against the context information. We also present a service composition method using the service history information and an Ontology Engine with data mining function. The Ontology Engine uses registered service compositions and provide the dynamic service composition for user queries. The Data Mining Engine also builds new composed services using the service history information. Thus, we combine the Ontology Engine with Data Mining Engine. For above two experiments, we make sure that our approach enhances the ratio of service compositions. In the future, we will integrate QoS-aware service composition with the current service provisioning framework.

Acknowledgments.

This work was supported by the research grant of the Chungbuk National University in 2006.

References

[1] S.raya K. Mostefaoui and Beat Hirsbrunner: Context Aware Service Provisioning. Proceedings of The IEEE/ACS International Conference on Pervasive Services (2004)71–80
[2] Dipanjan Chakraborty, Yelena Yesha and Anupam Joshi: A Distributed Service Composition Protocol for Pervasive Environments. Wireless Communication and Networking Conference, Vol. 4. IEEE (2004) 2575–2580
[3] SUN Microsystems.: Jini: Architectural Overview. Technical White Paper (1999)
[4] Microsoft Corporation: Understanding Universal Plug and Play: a White Paper. Microsoft (2000)
[5] Guttman, E., Perkins, C., Veizades, J. and Day M.: Service Location Protocol. Version 2 RFC2608 (1999)
[6] Salutation Consortium: Salutation Architecture Specification Version 2.1. Salutation (1999)
[7] Bluetooth Specification Part E: Service Discovery Protocol. http://www.bluetooth.com (1999)
[8] V. Sundramoorthy, Hans Scholten, Pierre Jansen and Piter Hartel.: Service Discovery at Home. Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing and the Fourth Pacific Rim Conference on Multimedia, Volume 3. (2003) 1929–1933
[9] Michiharu Takemoto, Tetsuya Oh-ishi, Tetsuya Iwata, Yoji Yamato and Yohei Tanaka.: A service-Composition and Service-Emergence Framework for Ubiquitous-Computing Environments. Proceedings of the 2004 International Symposium on Applications and the Internet Workshops (2004) 313–318
[10] Michiharu Takemoto, Hiiroshi Sunaga, Kenichiro Tanaka, Hiroaki Matsumura and Eiji Shinohara: The Ubiquitous Service-Oriented Network(USON): An Approach for a Ubiquitous World based on P2P Technology. Proceeding of the second International Conference on Peer-to-Peer Computing IEEE (2002)
[11] Qun Ni: Service Composition in Ontology enabled Service Oriented Architecture for Pervasive Computing. Workshop on Ubiquitous Computing and e-Research (2005)
[12] Keita Fujii and Tatsuya Suda: Dynamic Service Composition Using Semantic Information (2004)
[13] HAVi Consortium: HAVi Specification V1.0. (2000)
[14] Antonio Corradi, Rebecca Montanari and Daniela Tibaldi: Context-Based Access Control for Ubiquitous Service provisioning. Computer Software and Applications Conference, vol. 1 (2004) 444–451
[15] Choonhwa Lee and Sumi Helal: Context Attributes: An Approach to Enable Context-Awareness for Service Discovery. Symposium on Applications and the Internet (2003) 22–30
[16] Samir Chamri-Doudane and Nazim Agoulmine: Hierarchical Policy Based Management Architecture to Support the Deployment and the Discovery of Services in Ubiquitous Networks. The 29th Annual IEEE International Conference on Local Computer Networks (2004) 126–133
[17] S. Maffioletti, S. kouadri Mostefaoui and B. Hirsbrunner.: Automatic Resource and Service Management for Ubiquitous Computing Environments. Proceedings of the

Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (2004) 219–223
[18] Dipanjan Chakraborty, Anupam Joshi, YElena Yesha and Tim Finin: GSD: A novel group-based service Discovery Protocol for MANETS. In IEEE conference on Mobile and Wireless Communication Networks, Stockholm, Sweden (2002)
[19] Keita Fujii and Tatsuya Suda: Component Service Model with Semantics (CoSMoS): A New Component Model for Dynamic Service Composition. Proceedings of the 2004 International Symposium on Applications and the Onternet Workshops (SAINTW'04) (2004)
[20] Keita Fujii and Tatsuya Suda: Semantic-Based Dynamic Service Composition. IEEE Journal on selected areas in Communications, Vol. 23, no.12 (2005) 2361–2372
[21] Nam Shic Jang, Seong Wan Hong, Dai Ho Jang: Data Mining. Dachung Midia (1999) 19–47
[22] Jiawei Han and Micheline Kamber: Data Mining: Concepts and Techniques. Morgan Kaufmann (2001)
[23] Nikola Milanovic and Miroslaw Malek: Current Solutions for Web Service Composition. IEEE Internet Computing, Vol. 8, Issue 6. (2004) 51–59
[24] Sun Young Lee, Byong Cheol Shin, Jong Yun Lee, Jeong Suk Bae, Gyoung Cheol Shin: A Framework for Service Provisioning in Ubiquitous Computing Environment. Proceedings of the 32ed KISS Fall Conference, Vol. 32, No.2 (I). The Korea Information Science Society (2005) 604–606

Sun Young Lee received the B.S. and M.S. degrees in Electrical Engineering from Chungbuk National University in 2001 and 2005, respectively. She is now a candidate for the PhD in the department of Computer Education at Chungbuk National University. Her research interest includes databases, ubiquitous computing and bioinformatics.

Jong-Yun Lee received the BS and MS degrees in Computer Engineering from Chungbuk National University in 1985 and 1987, respectively and the PhD degree in Computer Science from Chungbuk National University, South Korea in 1999. He worked as a research/project leader in Software Research and Development Institute of Hyundai Electronics Industrial Company Ltd. and Hyundai Information Technologies Company Ltd. in South Korea from 1990 to 1996. Also he worked with Bit Computer Cooperation in 1989. He had worked for the department of Information and Communication Engineering at Samcheok National University as an assistant professor from March 1999 to February 2003. After that, he is an associate professor in the department of Computer Education of Chungbuk National University in South Korea. His current research interests include temporal databases, spatio-temporal databases, sensor data, u-learning, and especially query processing and optimization technologies in databases. He is a member of the IEEE, Korea Information Science Society, Multimedia Society, and served an international journal INFORMATION as an editorial board member in 2004 and Korea Information Processing Society as a journal editorial member since 2003.

Byung Il Lee received the B.S. deg ree in Computer Science from Hanbat Nati onal University in 1996 and the M.S. degre e in Computer Engineering from Chungbuk National University in 1998. He worked in Computer laboratory of Korea National Un iversity of Education from 2001 to 2004. H e is now a candidate for the PhD in the dep artment of Computer Engineering at Chung buk National University. His research interest includes data mini ng, bioinformatics and multi sequence alignment.