ICEB 2010 Proceedings

International Conference on Electronic Business (ICEB)

Winter 12-1-2010

Method for Web Service Composition Discovery Based on Association Rules

Fuzan Chen

Minqiang Li

Jisong Kou

Method for Web Service Composition Discovery Based on Association Rules

Chen Fuzan Li Minqiang Kou Jisong
School of Management, Tianjin University, Tianjin, China

Abstract

Web service plays an important role in implementing Service Oriented Architecture (SOA) for achieving dynamic business process. With the increased number of web services advertised in public repository, it is becoming vital to provide an efficient web service composition mechanism with respect to user's requirement. In this paper, a service composition approach based on association rules is proposed in the sense of knowledge discovery. This approach includes two steps: firstly, frequent web services will be enumerated in the dataset of history service composition transactions. Secondly, execution path, the basic unit of composite service, will be generated based on concepts extracted from SOA domain and association rules implied in frequent services. In addition, frequent services mining algorithm is put forward based on a structure of Adjacent-Lattice, and experiments are given to show the effectiveness of the mining algorithm.

Key Words: Service Oriented Architecture, Web Service, Service Composition, Association Rules

1. Introduction

Web services are considered as self-contained, self-describing, modular applications that can be published, located, and invoked across the Web [1]. Nowadays, an increasing amount of companies and organizations only implement their core business and outsource other application services over Internet. Thus, the ability to efficiently and effectively select and integrate inter-organizational and heterogeneous services on the Web at runtime is an important step towards the development of the Web service applications. In particular, if no single Web service can satisfy the functionality required by the user, there should be a possibility to combine existing services together in order to fulfill the request. In this case we speak of a composite service [2]. The process of developing a composite service in turn is called service composition, which can be either performed by composing elementary or composite services. This trend has triggered a considerable number of research efforts on the composition of Web services both in academia and in industry.

In service oriented application, value-added service is composed of some component services selected from the candidate services. How to discover services and composing them for users` requirements efficiently become the key in the application of Web service technology. For this problem, a service composition discovery approach based on association rules is proposed in the sense of knowledge discovery. Firstly, sets of frequent web services that can be used to constitute compose services will be discovered in dataset of history services composition information. Secondly, execution path, the basic unit of composite service, will be generated based on association rules implied in these frequent service sets.

The rest of this paper is organized as follows. Section 2 give an overview of the related work, and section 3 introduces the basic concept of services composition and association rule problem. Section 4 describes the mechanism of services composition based on association rules. The experimental results and analyses are presented in Section 5. Finally, the paper is concluded in Section 6.

2. Related Works

At present, after user bring a web service request, service register server checks whether a single registered web service could fulfill the user's request. But under some circumstances, the user's request can only be satisfied by a composition of several services. Therefore composition-based service discovery and some progress have been made in recent years. The large amount of services make composing of services a time consuming and impossible job. Hence, a variety of techniques to compose services some automated and semi-automated ways have recently been investigated. To that end, several methods for this purpose have been proposed. In particular, most researches conducted fall in the realm of workflow composition or AI planning.

For the former, one can argue that, in many ways, a composite service is similar to a workflow [3][4]. The workflow-based approach to Web service composition adopts composition schemas to define the abstract functionalities and interactions of component services, such as control flow, data flow, and transactional dependencies. In these approaches, the designer is required to explicitly identify the tasks that compose business processes and specify the relationships (e.g., execution sequence) among them. This mode of composition requires all task relationships to be established a priori. It also requires meticulous transformation of business rules and relationships into a particular process model. Industry extensively uses BPEL

(Business Process Execution Language), an extension of WSDL, for workflow-based web service composition [5]. However, the composition is carried out in the syntactic level, which is not able to express the semantic capabilities of services. The current achievements on flexible workflow, automatic process adoption and cross-enterprise integration provide the means for automated Web services composition as well. In addition, the dynamic workflow methods provide the means to bind the abstract nodes with the concrete resources or services automatically.

On the other hand, dynamic composition methods are required to generate the plan automatically. Among these techniques, AI planning-based and deductive theorem proving web service composition techniques have attracted considerable research interests. In these works, automated composition is described as a planning problem [6][7][8]: services that are available and published on the Web, the component services are used to construct the planning domain, composition requirements can be formalized as planning goals, and planning algorithms can be used to generate composed services, i.e., plans that compose the component services. This is achieved through the use of the Semantic Web. For example, the ontology language OWL-S [9] is specifically designed to connect with AI planning. Some other AI planning techniques are proposed for the automatic composition of Web services. For example, Rao et al. [10] introduced a method for automatic composition of semantic web services using liner logic (LL) theorem proving. The Colored Petri Nets (CPNs), that is an extension of Petri Nets, is used to model service composition and orchestration [11].

Furthermore, in order to address the challenges of scalability, flexibility and quality-of-service (QoS) management for distributed service composition, the correlations among resource services are taken into account during MGrid resource service composition, and a QoS description mode supporting resource service correlation is presented by F.Tao [12]. Data mining techniques are also used on web services log to improve quality and performance of web services composition. Service mining, a technology similar to web mining, has been proposed [13][14][15] in order to make use of the service artifacts in a registry-repository and to improve service composition via analysis of service usage patterns.

3. Problem Statement and Preliminaries

3.1 Composite Service

According to specific business process, the composite service is a complex service composed by relatively simple web services from a large set of candidates. It provides stronger and more integrated business functions than single service. Definition of composite service will be given in the following.

Definition 1 (Composite Service, CS): A composite service can be formally defined as CS = (SS, CF, DF), and hereinto:

1) SS is a web services set $\{S_1, S_2, \ldots, S_m\}$, and S_i is either atomic service or composite service. There exist logic control relations CF and data dependency relations DF on web services set SS.
2) CF is logic control relation set among web services. For any two services $S_i, S_j \in SS$, if there is any relation of the following four kinds of relation, namely, sequence, fork, iteration and parallel relation. It is said that there is logic control relation between S_i and S_j.
3) CF is data dependency relation set among web services. Given $S_i, S_j \in SS$, if the input of S_j is output of S_i, there is a dependency relation between S_i and S_j.

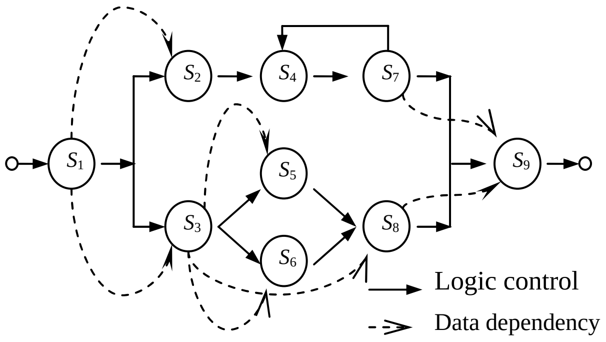

Figure 1 An example of composite service

Figure 1 is an example of composite service composed by component web services SS=$\{S_1, S_2, \ldots, S_9\}$. Thus, web service composition can be seen as a process to find a new service S, which consists of a set of component web services SS=$\{S_1, S_2, \ldots, S_9\}$. There is a logic control relation between S_2 and S_3, and data dependency relation between S_3 and S_8.

3.2 Association Rules (AR) Problem

Association rules in large databases of sales transactions was firstly proposed by Agrawal in 1993, which is one of the most important research topics in data mining [16]. Association rules problem can be stated as follows. Let I be the set of items and D a transaction database in which each transaction is a subset of I and is associated with a unique identifier called its TID. A set of items is called an itemset. We say that a transaction supports an itemset X if all the items in X are contained in the transaction. The support of an itemset X is the percentage of transactions in D which support X: $support(X) = \|\{t \in D | X \subseteq t\}\| / \|\{t \in D\}\|$. An association rule is an implication of the form $X \Rightarrow Y$, where $X, Y \subset I$ and $X \cap Y = \phi$. It means that if we find all of items of item set X, then we have a good chance of finding the items of Y.

Given an association rule, its support and confidence should be above certain thresholds. The confidence of association rule *R:*$X \Rightarrow Y$ describes the probability of finding Y on the condition of X: confidence($X \Rightarrow Y$) = support($X \cup Y$)/ support(X). The support of *R* is defined as: support($X \Rightarrow Y$) = support($X \cup Y$). So an association rule R:$X \Rightarrow Y$ holds in the transaction set D with confidence c if c% of transactions in D that contain X also contain Y, and has support s in the transaction set D if s% of transactions in D contain $X \cap Y$.

The problem of mining association rules in a database D is then traditionally defined as follows. Given the defined thresholds for the permissible minimum support threshold *min_sup* and confidence threshold *min_conf*, this procedure can be broken into two sub-problems:

1) Find all frequent itemsets X with support no less than the given *min_sup*.
2) For each frequent itemset X and $Y \subset X$, generate all strong rules $Y \Rightarrow X\text{-}Y$, with confidence no less than the given *min_conf*.

4. Services Composition Based on AR

When users give a service request, web services register server will select several registered services, compose them in some way and finally generate a composite service which could satisfy the user's request. This composition process usually complex and will lead to large overhead. Usually, specific user groups have some common request, for which service discovery algorithm will return the same composite service. Therefore, by recording the composite services returned by services register server will constitute services composition logs. Performing mining algorithms on such log files, some frequent composite services can be discovered, which correspond to the requirements of the specific user group. By doing this, the service discovery efficiency and quality can be greatly improved.

4.1 The Main Idea of Service Composition Based on AR

Association rule describes the probability of the associations among different objects. In the task of web service discovery, it is considered that web services are not independent on each other. In general, a frequent service set is the corresponding services of a composite service that are frequently executed together. It has been verified by many efficient executions in the past. It can be deduced that the frequent service sets would have better performance than other services. The implicit associations always happen among different web services and would reflect the most optimal composition of different web services. Inspired by the idea above, association rule can be applied for web service composition.

The main idea of our proposed approach is to utilize AR among different web services to identify the dependence between the given web service and other web services. The process can be divided into the following two steps:

1) Frequent service sets discovery: Enumerate all of the frequent web service sets that satisfy the minimum support threshold.
2) Composite services generation: Generate association rules from the frequent service sets that satisfy the minimum confidence threshold, and then configure reachable execution paths with selected association rules. This step involves pruning the association rules based on concepts extracted from SOA domain and confidence threshold.

4.2 Frequent Service sets Discovery

In order to enumerate all of the frequent web service compositions, service composition transaction datasets D should be constituted firstly, which consists of a certain number of composition transactions implemented with web services in the repository during the specified time interval. D is defined as $D = \{T_1, T_2, T_3, \dots, T_m\}$, T_i is a service composition transaction associated with a unique identifier. Each services composition transaction T contains a set of web services invoked in that transaction and $T = \{S_1, S_2, S_3, \dots, S_k\}$, where S_i is a web service invoked in that composition transaction. Therefore, we can get a web service set including all web services $SS=\{S_1, S_2, \dots, S_n\}$ invoked in all composition transactions. Consequently, an item is a web service, and an itemset is the set of services, named as serviceset, consisted in a certain composite service.

Let *I* be a service (item) set derived in transaction database *D*, $X,Y \in I$ and $X \neq Y$, if *Y* can be obtained from X by adding a single service, then *X* is said to be adjacent to *Y*. We call *X* is a parent of *Y*, and *Y* is a child of *X*. Thus, and serviceset may possibly have more than on parent and more than one child. For each serviceset *X* and each $x_k, \in X$, $X\text{-}\{x_k,\}$ is a parent of *X*.

Definition 2 (Adjacent-Lattice): Let $I = \{i_1, i_2, \ldots, i_m\}$ be the service set derived in transaction database D, the Adjacent-Lattice L on the ordered powerset P=(P(*I*), ⊆) of *I* be a DAG which each vertex is an serviceset $X \subseteq I$. L is constructed as follow: For each primary serviceset *X*, construct a graph with a vertex *v*(*X*), each vertex has a label corresponding to the value of its support, denoted as *S*(*X*). If and only if *X* is a parent of *Y*, a directed edge exists from *v*(*X*) to *v*(*Y*), denoted as *E*(*X*,*Y*).The vertex *v*(*X*) is said to be the head of the edge *E*(*X*,*Y*), and the vertex *v*(*Y*) is said to be the tail of the edge *E*(*X*,*Y*).

Thus, the search space of all servicesets can be represented by an Adjacent-Lattice, with the empty serviceset at the top and the set containing all services at the bottom. Apparently, the fact there is a directed path from vertex *v*(*X*) to vertex *v*(*Z*) in a adjacent-lattice L, implies $X \subset Z$. Especially, we call *X* is an ancestor of *Z*, and *Z* is a descendent of *X*.

Considering the database D illustrated in Table1, there are five different services, i.e.,*I*={A,B,C,D}, The corresponding Adjacent-Lattice L is illustrated in Figure 2. Each vertex has a label corresponding to its support value.

Table1 Transaction database D

Tid	servicesets
1	ABC
2	CD
3	ABC
4	ACD
5	ABCD
6	BCD
7	D
8	BC
9	ABCD
10	B

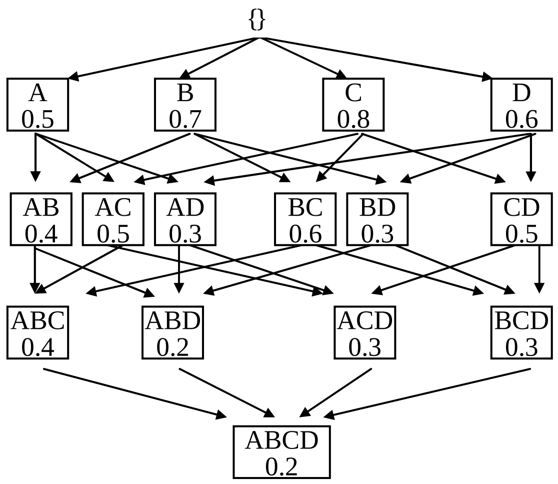

Figure2 Adjacent-Lattice L＝P(I)

Let *min_sup* be the minimum support theshod, to find all frequent servicesets, we need to solve the following search problem in the Adjacent-Lattice: for a given serviceset *X* (i.e. bottom element, top element or some medi-vertex), find all servicesets *Y* such that *v*(*Y*) is reachable from *v*(*X*) by a directed path in the lattice or sub-lattice, and satisfies the condition *S*(*Y*)≥ *min_sup*.

In practice, the number of vertices reachable from a given vertex may be quite large, though the number of vertices, which satisfy the minimum support condition, may be small. Using the lattice and sub-lattice structure can restrict the number of vertices checked. In order to restrict the number of vertices checked in the lattice, we sort the serviceset vertices by their support value in inverted sequence for each level.

We discuss an efficient strategy for enumerating the frequent servicesets in lattice, denoted Breadth-First search. The Breadth-First search approach starts with the bottom element of the sub-lattice, traverses the lattice level by level. Just the same as saying, we check each vertex at the next level only after the end of checking the certain level. This approach enumerates all frequent servicesets.

Algorithm 1 Breadth-First search strategy

```
Algorithm Breadth-First-search( $S$ )
Begin
  $FS = \{(v(B), S(B))\}$; $List$={v($B$)};
  While $List \neq \phi$ do
     Select $v(R)$ from $List$ ;
     For each unvisited child $v(T)$ of $v(X)$ do
          If $S(Y) \geq min\_sup$ do
             $List = List \cup v(Y)$ ;
             Add ( $v(Y)$ , $S(Y)$ ) to $FS$
             For each $i_j \in Y$ add $E(Y - \{i_j\}, Y)$
             to $FS$
      endfor;
   enddo;
   Delete $v(R)$ from $List$
End;
```

Each supper set of an infrequent serviceset is also infrequent. Thus, we can increase searching efficiency of the algorithm by pruning the vertices such that not satisfying the minimum support condition.

Performing the Breadth-First searching on the service composition transaction datasets D, all the frequent service sets can be enumerated. Web services in the same frequent set are frequently executed together, and have better performance than other infrequent services.

4.3 Composite Service Generation Based on selected Association Rules

A composite service consists of execution paths, which are the basic units for service composition.

Definition 3 (Execution Path, EP): Given a composite service CS = (SS, CF, DF), one execution path EP = (SS\`, CF\`, DF\`) of CS is still an composite service, and meets the following constraint conditions:

1) $SS' \subseteq SS \wedge CF' \subseteq \wedge DF' \subseteq DF$, namely EP is contained in the composite service CS;
2) There is the unique service sequence $s=<S_{i1}, S_{i2}, \ldots, S_{in}>$ corresponding to EP, hereinto $\cup S_{ij}=SS'$, S_{i1} is the first web service, S_{in} is the last one. S_{ik} is called to the direct predecessor of S_{ik+1}, and S_{ik+1} is the direct successor of S_{ik}. EP can be executed sequentially according to s.
3) $\forall S_i, S_j \in SS'$, it is impossible that S_i belongs to the different fork or parallel route from S_j;
4) $\exists S_{ij} \in <S_{i1}, S_{i2}, \ldots, S_{ik-1}>$, that is the direct predecessor of S_{ik};
5) Any direct successor of S_{ik} does not exist in $<S_{ik+1}, S_{ik+2}, \ldots, S_{in}>$.

Definition 4 (Reachable Execution Path, REP): Given an execution path EP = (SS, CF, DF) of a composite service CS, if EP can executed successfully, then we call EP is a reachable execution path of CS.

Definition 5 (Consistent Rule): Given an execution path EP = (SS, CF, DF) of a composite service CS, $e=<S_1, S_2, \ldots, S_n>$ is the service sequence corresponding to EP. $e'=<S_1, S_2, \ldots, S_k>$ is a sub service sequence of e, namely $Ant=\{S_1, S_2, \ldots, S_k\}$ is the set of services that has been selected by EP. Ant is called to the ancestor services set of EP, denoted by EP.Ant. Likewise $Con=SS\text{-}Ant=\{S_{k+1}, S_{k+2}, \ldots, S_n\}$ is called to the consequence services set of EP, denoted by EP.Con. Given an association rule $R: X \Rightarrow Y$, if $X \subseteq EP.Ant$ and $Y \supseteq EP.Con$, then we call that R is a consistent rule for EP.

It can be seen from the above definitions, that the execution paths, generated by composite service division, are the basic unit for web service selection. The key of this web service composition approach is how to generate reachable execution paths.The main idea of our proposed approach is to utilize the association rules among different web services to identify the dependence between the given web service and other web services. Namely, for an execution path EP of a composite service CS, let $SS=\{S_1, S_2, \ldots, S_n\}$ be the set of web services of EP, and ARS be the association rules set generated from frequent service sets. EP can be constituted by invoking these two procedures iteratively:

1) Given the last selected web service $S_i \in SS$ of EP, find the constituent rule $R: S_i \Rightarrow Y$ with the maximal confidence value in ARS that satisfying:

$$\begin{aligned} &Support(S_i \Rightarrow Y) \geq min\_sup \\ &Confidence(S_i \Rightarrow Y) \geq min\_conf \qquad (1) \end{aligned}$$

2) Add component web services in Y to SS, and generate services sequence of SS.

5. Experimental and Analytical Results

We run the simulation on PC: Intel P4 2*2.4G CPU, 2GB main memory and Windows 2000 Server. The synthetic service composition transactions are generated using a method provided by KDD Research Group in IBM, referring to Http://www.almaden.ibm.com/cs/quest/syndata#AssocSynData. D is the number of service composition transactions, T is the average transaction size, and S is the average size of potential maximal frequent service set. A data set is denoted by Tx.Sy.DmK. Our approach is a two stages process, and enumerates the frequent web service sets in the second stage:

1) Constructing the Adjacent-Lattice, decomposing the constructed lattice into sub-lattices if necessary.
2) Traversing the lattice and enumerating the frequent service sets.

Figure 3 shows the response time variation with average transaction size during lattice constructing. The data sets are Tx.S3.D100K and Tx.S3.D100K, the transaction size increases 1 every time. This shows that response time of lattice constructing and the average transaction size is an exponential relation. The computational effort and the scale of constructed lattice are more sensitive to the average transaction size, rather than to the number of transactions. We also find that the computational effort mushrooms when some transactions have very great length, even though the average transaction size is small.

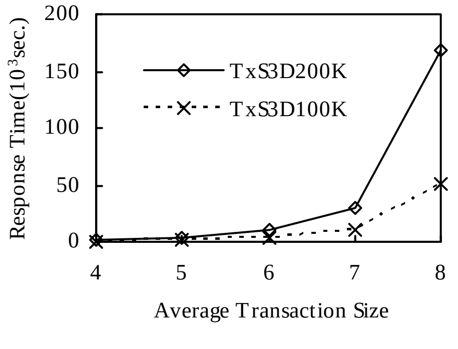

Figure 3 Response times with average transaction size during lattice constructing

Figure 4 shows the response time variation with the number of frequent service sets enumerated by applying Breadth-First search strategy to Adjacent-Lattice. The data sets are T5.S3.D100K and T5.S3.D200K. The response time of frequent service sets enumerating is more sensitive to the number of frequent service sets searched, rather than to the average transaction size and number of transactions in the data set.

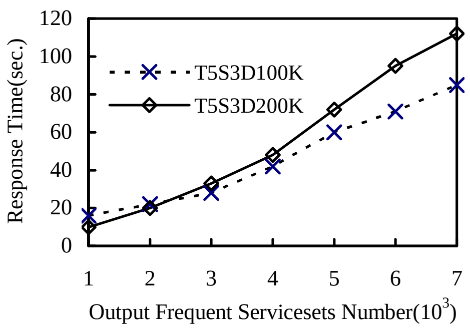

Figure 4 Response time variation with number of frequent servicesets searched

6. Conclusions

Web Service Composition problem, especially based on Quality of Service (QoS), has been extensively studied recently. The different approaches that have been followed so far in the research of QoS-driven service selection span from the use of QoS ontology, the proposal of ad-hoc methods in some general framework, the use of data mining and so on. Most approaches are complex and time consuming. Furthermore, current Web service technology can not support QoS or other non-functional aspects of a web service, so all the approaches above face the vital problem that how to acquire Web service QoS information. Compared to service QoS information, the service composition information and composite service execution information, which can be recorded on the service register server, are easier to acquire.

This paper presents a new service composition discovery approach based on service composition transaction mining, gives a complete solution of the new service composition discovery approach, and describes the related algorithm. Based on service-composition-data mining, the new services composition discovery approach presented in this paper would greatly improve the efficiency of the service compotation process, and achieve more robustness of the composite services. Furthermore, association rules would be discovered by analyzing all web service composition transactions related to that set of users. By combining user group and association rule mined from that group, a personalized web service recommendation would be presented.

However, the proposed approach has its own limitation. It would suffer from cold start problem, a typical problem in web applications, which means for a new web service, it is difficult to give recommendation since there have no history composition transactions on it.

Acknowledgements

The work was supported by the National Science Foundation of China (No. 70771074), the Self-Innovation Foundation of Tianjin University, and the Ph.D. Programs Foundation of Ministry of Education of China (No. 20090032110065).

References

[1] Web Services Architecture—W3C Working Group Note, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211, Feb. 2004

[2] Alonso, Gustavo, Fabio Casati, Harumi Kuno & Vijay Machiraju: Web Services. Concepts, Architectures and Applications. Springer-Verlag Berlin Heidelberg 2004

[3] F. Casati, S. Ilnicki, and L. Jin. Adaptive and dynamic service composition in EFlow. In Proceedings of 12th International Conference on Advanced Information Systems Engineering (CAiSE), Stockholm, Sweden, June 2000. Springer Verlag.

[4] Kunal Verma，Karthik Gomadam，The METEOR-S Approach for Configuring and Executing Dynamic Web Processes，Technical Report.2005，http://lsdis.cs.uga.edu/projects/meteor-s/techRep6-24-05.pdf

[5] M. t. Beek, A. Bucchiarone, and S. Gnesi. “Web service composition approaches: From industrial standards to formal methods,” 2nd Intl. Conf. on Internet and Web Applications and Services, 15-20, 2007

[6] J. Rao and X. Su, A survey of automated web service composition methods,” 1st Int. Work. on Semantic Web Services and Web Process Composition, SWSWPC-2004, LNCS, 2005(3387), 43–54

[7] P. Bertoli, M. Pistore, P. Traverso, Automated composition of Web services via planning in asynchronous domains, Artificial Intelligence, 2010, vol.174, 316–361

[8] E. Sirin, B. Parsiab, D. Wu, J. Hendler, D. Nau, HTN planning for web service composition using SHOP2, Journal of Web Semantics, 2004(4), 377–396

[9] D. Martin, et al., “OWL-S: Semantic markup for web services,” http://www.w3.org/Submission/OWL-S/, 2004.

[10] J.Rao, Kungas P, Matskin M, Composition of semantic Web services using linear logic theorem proving. Information System, 2006(31):340-360

[11] V. Gehlot, K. Edupuganti, Use of Colored Petri Nets to Model, Analyze, and Evaluate Service Composition and Orchestration, Proceedings of the 42nd Hawaii International Conference on System Sciences, 2009,1-8

[12] Fei Tao,Dongming Zhao, Hu Yefa, Zude Zhou, Correlation-aware resource service composition and optimal-selection in manufacturing grid, European Journal of Operational Research 201 (2010) 129–143

[13] G. Zheng, A. Bouguettaya, Service Mining on the Web, IEEE transactions on services computing, 2(1), 2009,65-78

[14] Wenge Rong, Kecheng Liu, Lin Liang, Personalized Web Service Ranking via User Group Combining Association Rule. ICWS 2009: 445-452

[15] Shuchuan Lo, Web service quality control based on text mining using support vector machine, Expert Systems with Applications, 34(1),2008, 603-610

[16] R. Agrawal, T. ImielinSki, A. Swami, Mining association rules between sets of items in large database. Proc. of the ACMSIG2 MOD International Conference on Management of Data, Washington, DC.1993, 2 : 207-216