Composite Process Oriented Service Discovery in Preserving Business and Timed Relation

Yu Dai, Lei Yang, Bin Zhang, and Kening Gao
College of Information Science and Technology
Northeastern University
P.R.China
NEU_DaiYu@126.com

ABSTRACT

This paper proposes an approach for solving the problem of composite process oriented service discovery with preserving business and timed relation. Key to our approach is the definition of requirement towards service. Such requirement not only constrains the functional semantics on an operation of service, but also constrains the timed and business relation among operations of service. Thus, services fulfilling such requirement can preserve the business and timed relation of the composite process. In order to automatically discover satisfied services, a multilevel matching relation model is defined which includes matching rules on the level of syntactic, functional semantics and behavioral semantics to test whether service is similar to the requirement. And based on such relation model, an algorithm for discovering services for the composite process is proposed. The experimentation shows that the proposed approach can preserve the timed and business relation effectively.

Categories and Subject Descriptors

D.2.12 [Software Engineering]: Interoperability–*Distributed Objects.*

General Terms

Algorithms, Experimentation.

Keywords

Composite process oriented service discovery, behavioral semantic, functional semantic

1. INTRODUCTION

Web services are loosely coupled software components, published, located and invoked across the Web. With the prosperity of web services, composing available services to form a new value-added one becomes a new business paradigm. Web service composition has been studied variously and a variety of frameworks are proposed [1]. These frameworks include AI planning based composition [2], workflow based composition [3], and interactive composition [4-6].

In this paper, we formulate a service composition problem which starts from a composite process specification and arises naturally from studies on interactive composition. A composite process is a process for a composite service with each task in the composite process as a placeholder for a concrete service. One of the essential technical elements in this framework is *service discovery* to find appropriate concrete service for each task in order to accomplish the composite process. Approaches in [4, 7] have reported composite process oriented service discovery problem while simplified such service discovery problem by assuming that composite process only have *stateless services*, where there exists no business relation between tasks and each task can be performed by different service. However, this assumption may not always hold. Composite processes with *stateful services* are often widely used in the domain area such as online shopping.

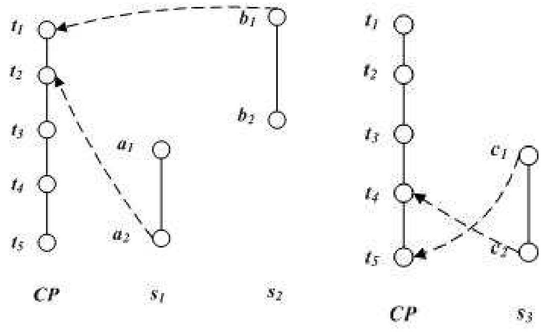


Fig 1 (a) Business Relation Disruption **Fig 1 (b) Timed Relation Disruption**

Fig.1. Problems of composite process oriented service discovery

Let us illustrate the composite process oriented service discovery problem with *stateful services* using the example in Fig 1, where there are three web services s_1, s_2 and s_3 as well as one composite process *CP*. Service s_1 and s_2 are provided by two companies for providing online air ticket booking. *CP* is a composite process for booking air ticket and hotel. In *CP*, t_1 is a task for querying if the air ticket is available, t_2 is a task for ordering desired ticket for users, t_5 is a task for receiving payment for the hotel booking and t_4 is a task for giving the booked room number. a_1 and b_1 are the operations which can fulfill the functional requirement of t_1 while a_2 and b_2 are the operations which can fulfill the functional requirement of t_2 as well as c_1 and c_2 are the operations which can fulfill the functional requirement of t_4 and t_5 respectively. Fig 1 (a) shows that task b_1 is assigned to t_1 and a_2 is assigned to t_2. In this situation, if s_1 has no air ticket of s_2, *CP* cannot run properly as the output of b_1 cannot feed to a_2; if the price of the air ticket booked by s_1 is above the price of the air ticket queried by s_2, the performance of *CP* may not be accepted by end users. The main reason for such phenomenon is that there exists no business relation requirement between tasks in the composite process *cp*. If it is required that t_1 and t_2 must be assigned by the same service, such phenomenon will not happen. Fig 1 (b) shows that task c_2 is assigned to t_4 and c_1 is assigned to t_5. In this situation, *CP* cannot run properly also as the timed relation of c_1 and c_2 in s_3 violates the timed relation of t_4 and t_5 in *CP*. The main reason for such phenomenon is that current service discovery approach overlooks the timed relation between operations in certain service. If it is required that operations which can assign to t_4 and t_5 must have the same timed relation as that of t_4 and t_5, such phenomenon will not happen.

The main contributions of this paper are three-fold. First, a novel approach is proposed to discovering services for composite process. Based on the requirement constraining the timed and business relation among tasks, services with operations satisfying such relation will be discovered. Second, business relation between tasks is added to composite process and the definition of requirement towards service is given. Third, a multilevel matching relation model including matching rules on the level of syntactic, functional semantics and business behavioral semantics to test whether service is similar to the requirement is defined. And based on such relation model, an algorithm for discovering services for the composite process is proposed. The experimentation shows that the proposed algorithm can assure the timed and business relation effectively.

2. RELATED WORKS

In this section, we review some key relevant developments in composite process oriented service discovery.

In [8], a service discovery algorithm is proposed, the key of which is a service template in order to solve the problem of heterogeneity among services and requirements. Based on such service template, operations of each service are described and the functional semantic requirement of each task in the composite process is also described. Thus, the process of service discovery is to find operation of certain service for each task in the composite process. Since that such discovery is lost of the requirements of timed relations and business relations between tasks in the composite process, the problem of business relation disruption and timed relation disruption will occur.

In [7], an algorithm for finding service for each task in the workflow is presented. Such algorithm is based on three dimensions: syntax, operational metrics and semantics. The operational metrics mentions the QoS requirements of the tasks in the workflow. However, as similar to [8], such approach is lost of the requirements of timed relations and business relations between tasks in the workflow, which will result in the disruption of business relation and timed relation.

[4] gives an automatic service composition approach with services exporting their behavior. This approach uses finite state automata to express business process and through matching the actions of the business process with actions of services, a business process can be implemented. Although such approach considers the service from the whole point of view of the service behavior which considers the timed relation between operations of certain service, this approach still takes the action in the business process as the discovering granularity which does not constrain the timed relation and business relation between actions of services. This approach may cause the disruption of business relation and timed relation also.

Compared with above works, the work done in this paper considers the timed and business relation among tasks which will constrain the relation between operations of discovered services. Thus, our work can solve the problem of disruption of business and time relation faced by current works.

3. APPROACH DESCRIPTION

This paper proposes an approach for solving the problem of composite process oriented service discovery with preserving business and timed relation. Based on the requirement constraining the timed and business relation among tasks, services with operations satisfying such relation will be discovered. In the following, we will give the definition of requirement towards service and service discovering algorithm.

3.1 Definition of Requirement towards Services

Currently, most of the process description languages such as BPEL only identify the control flow which constrains the timed relations between tasks. In order to support composite process oriented service discovery, it is needed to add business relation between tasks to the composite process. Business relation constrains that tasks having such relation will be provided by the same service. Through adding business relation to the composite process, in this paper, the definition of business relation added composite process is given as follows:

Definition 1. *Composite Process.* A composite process can be defined as a 4-vector: $CP=<Taskset, C, B, f>$, where:

- $Taskset=\{t_1, t_2,\dots, t_k\}$ is a set of task nodes in the composite process. For each $t\in Taskset$, $t=<ID, Input, Output>$, where *ID* is the identification of *t*, *Input* and *Output* are the input and output parameters and for each parameter *p*, it can be defined as $p=<ID, Meaning, DataType>$ where *Meaning* signifies the semantics of *p* and *DataType* is the data type of *p*.
- $C \subseteq Taskset \times Taskset \times condition$ is used to signify the timed relation. For each $e=(t_i, t_j, c_u)\in C$, it means that if condition c_u is true, after task t_i is finished, t_j will be invoked. We call t_i and t_j are respectively the source and target task of *e* which can also be signified as $t_i \prec_{c_u} t_j$.;
- $B \subseteq Taskset \times Taskset$ is used to signify the business relation. For each $g=(t_i, t_j)\in B$, it means that t_i and t_j have business relation and they must implemented by a same service which can also be signified as $t_i \Delta t_j$ or $t_j \Delta t_i$. Especially, for each task t_i, $(t_i, t_i)\in B$;
- $f{:}B \longrightarrow BRD$ is a mapping relation from *B* to *BRD*. $BRD=\{br_1, br_2,\dots,br_v\}$is a set of business descriptions

and for each br_i in *BRD*, it can be defined as *br=<ID, Fc>* where *ID* is the identification and *Fc* signifies the functional category. For each $g=(t_i, t_j)\in B$, $f(g)=br$ means that the business relation between t_i and t_j is for the function *br.Fc*.

Definition 2. *Path.* In composite process *CP*, there exists a set of tasks $t_1, t_2,\ldots, t_n$. If $t_i \prec_{c_1} t_2 \prec_{c_2} \ldots \prec_{c_{n-1}} t_n \prec_{c_n} t_j$, we call that in *CP*, t_i can achieve t_j which can be signified as $t_i \longrightarrow t_j$. $(t_i,t_2,\ldots,t_n,t_j)$ is called a path from t_i to t_j.

Definition 3. *Business Partition.* In composite process *CP*, there exist a set of sets $x_1, x_2,\ldots, x_u$. For each x_i, $x_i \subseteq CP.Taskset$ and $x_i \cap x_j = \phi\ (x_i \neq x_j)$ and $x_1 \cup x_2 \ldots \cup x_u = CP.Taskset$. For each $t_{ik}\in x_i$ and each $t_l\in CP$, if $t_{ik}\, \Delta\, t_l$ and $t_l\in x_i$, we call that $(x_1, x_2,\ldots, x_u)$ is a business partition of *CP*. And for each x_i in the business partition, we call it a gene.

A business relation added composite process is illustrated in Fig 2. In Fig 2, tasks t_1 and t_3 have business relations which mean that these two tasks must be bind with a same service. Task t_2 t_5 and t_6 have business relation which means that they should be implemented by a same service. Tasks t_4 and t_8 have business relations and they must be bind with a same service. In Fig 2, (t_1, t_2, t_3) is a path from t_1 to t_3. $(\{t_1, t_3\}, \{t_4, t_8\}, \{t_2, t_5, t_6\}, \{t_7\})$ is the business partition of the composite process and $x_1=\{t_1, t_3\}$, $x_2=\{t_4, t_8\}$, $x_3=\{t_2, t_5, t_6\}$, $x_4=\{t_7\}$ are the genes.

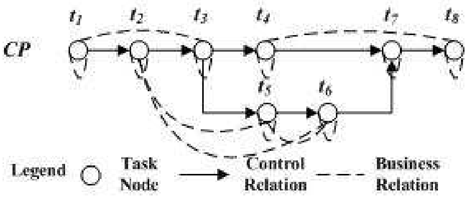


Fig 2 Example of Business Relation Added Composite Process

One of the important issues in the composite process oriented service discovery is to make the requirement towards service full of timed and business relation in order to solve the problem of disruption of timed and business relation illustrated in Fig 1. For each gene in the business partition, it is a set of tasks which have business relations and must be implemented by a same service. Gene constrains the business relation between operations of certain service. If taking the gene as the requirement, it will not disrupt the business relation. However, as the gene has no timed relation, it is needed to add the timed relation to gene, The timed relation between tasks in the gene must also be consistent with the timed relation between tasks in the composite process. In the following, we will give the definition of local composite process.

Definition 4. *Local Composite Process.* For business partition $(x_1, x_2,\ldots, x_u)$ on the composite process *CP* and certain gene x_i in the partition, if the following conditions are true, we call that $LCPx_i$ is the local composite process of x_i under *CP*.

- $LCPx_i.Taskset=x_i.Taskset$;
- $\forall\, t_i, t_j\in CP.Taskset$ and $t_i \prec_{c_u} t_j$, if $t_i, t_j\in LCPx_i.Taskset$, then in $LCPx_i$, $\exists\, t_i \prec_{c_u} t_j$; if $t_i \in LCPx_i.Taskset$ and $t_j \notin LCPx_i.Taskset$, then if $\exists\, t_k\in LCPx_i.Taskset$ and in *CP*, $t_j \prec_{c_u} t_k$, then in $LCPx_i$, $\exists\, t_i \prec_{c_u} t_k$; if $t_i\notin LCPx_i.Taskset$ and $t_j\in LCPx_i.Taskset$, then if $\exists\, t_k\in LCPx_i.Taskset$ and in *CP*, $t_k \prec_{c_u} t_i$, then in $LCPx_i$, $\exists\, t_k \prec_{c_u} t_j$.

The first condition in the above constrains that the local composite process has the same tasks as the gene. The second condition constrains that if there exists a path between two tasks in the local composite process, there also exists a path between these two tasks in the original composite process. It means that the timed relation between tasks in the local composite process must be consistent with the timed relation between tasks in the composite process.

Take Fig 2 as an example, for each gene in the business partition of the composite process, the local composite process of each gene is illustrated in Fig 3. For each local composite process, the timed relation between tasks is consistent with the timed relation between the tasks in the composite process.

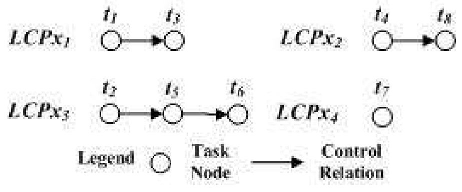


Fig 3 Local Composite Process

For that in the local composite process, all the tasks must be implemented by a same service and the operations of the service must have the consistent timed relation as the local composite process, the local composite process can be served as the requirement towards service.

Definition 5. *Requirement towards Service.* For business partition $(x_1, x_2,\ldots, x_u)$ on the composite process *CP*, for each gene x_i in the partition, under x_i the requirement towards service can be defined as *R=<Fc, LCP>*, where *Fc* is the function description $Fc=CP.f(x_i.t,\ x_i.t)$ and *LCP* is the local composite process of x_i under *CP*.

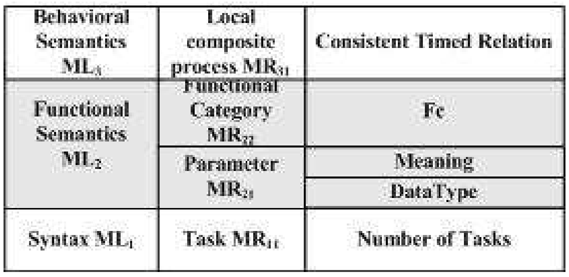
3.2 Discovering Services for Requirement

We define a service as a set of operations and such operations have timed relation with each other. In this paper, we assume that for each operation in certain service, it has business relation with all the other ones. Finding services for the composite process is to find a service for each requirement towards service of the composite process. The requirement towards service constrains the functional description, the operations in the service and the timed relation between operations in the service. Thus, the desired service must have the same business description, the operations and the timed relation as the requirement. In the following, we will give the definition of service, ontology and the multilevel matching relation model.

Definition 6. *Service.* A service can be defined as *s=<ID, Fc, ServiceModel>*, where *ID* is the identification of *s*; *Fc* is the functional category and *ServiceModel=<OPset, C>* is used to identify the operations and the timed relation between operations in the service (for each *op* in *OPset*, it has the same definition as task in *Definition 1* and *C* has the same meaning as *C* in *Definition 1*).

Definition 7. *Ontology*. An ontology Ω = {$concept_1$,…, $concept_n$} contains a set of classes. Each class $concept_j$ has an associated set of properties P_k={p_1,…, p_m}. An ontology relates more specific concepts to more general ones from which generic information can be inherited. Such links have been variously name "is a", "subset of", "member of", "sub concept of", "supper concept", etc.

Definition 8. *Multilevel Matching Relation Model.* The proposed multilevel matching relation model for matchmaking between service and the requirement contains rules organized into 3 levels (Fig 4). Each rule MR_{pq} at a level ML_p (p=1…3) compares a specific feature of requirements within ML_p. The first level ML_1 compares syntactic attributes such as the number of tasks (MR_{11}). The second level ML_2 compares functional semantic attributes. We define 3 groups of rules at this level. The first group compares the functional semantics of functional category of the requirement and that of the service. The second group compares the functional semantics of parameters of each task of the requirement and that of parameters of each operation of the service. The third level ML_3 compares the behavioral semantics of the requirement and service. We define one group of rules at this level which compares the timed relation among tasks of the requirement and service.



Behavioral Semantics ML3	Local composite process MR31	Consistent Timed Relation
Functional Semantics ML2	Functional Category MR22	Fc
	Parameter MR21	Meaning
		DataType
Syntax ML1	Task MR11	Number of Tasks

Fig 4 Multilevel Matching Relation Model

For service s and requirement r, the syntax and functional semantics and behavioral semantics rules is given in Table 1. Based on these rules, satisfied services can be discovered. Algorithm 1 gives the algorithm of discovering service for requirement towards service.

Table.1 Rules in Multilevel Matching Relation Model

MR	Rule Description
MR_{11}	$\lvert s.OPset \rvert \geq \lvert r.Taskset \rvert$
MR_{21}	$\forall$ $p\in$ *task.Output*, $\exists$ $p'\in$ *op.Output*, *p.Datatype* is the sub-type of *p'.Datatype* and Ω (*p.Meaning*)= Ω (*p'.Meaning*); $\forall$ $p'\in$ *op.Input*, $\exists$ $p\in$ *task.Input*, *p'.Datatype* is the sub-type of *p.Datatype* and Ω (*p'.Meaning*)= Ω (*p.Meaning*). (*task*$\in$ *r.Taskset*, *op*$\in$ *r.OP*)
MR_{22}	Ω (*s.Fc*)= Ω (*r.Fc*)
MR_{31}	$\forall$ $e\in$ *r.C*, in *s*, there exists a path which makes *e.target* $\longrightarrow$ *e.source.*

Algorithm1. Discovering Service for Requirement towards Service

```
DS Discovery(RS, Serviceset)// RS is the set of requirements generated based on Algorithm 1
1 begin
2   for each r in RS do
3     begin
4       DS[i].Fc=r.Fc;
5         for each s in Serviceset do
6           begin
7         if MR_11(s, r) and MR_21(s, r) and MR_22(s, r) and MR_31(s, r) then
8               put s into DS[i].S;
9           end
10        i++;
11    end
12 end
```

Through testing if the service has matching relation with the requirement, it can discover service for each requirement towards service of the composite process. Thus, for the problem illustrated in Fig 1, as t_1 and t_2 have business relation, they must be implemented by a same service. As in service s_1, the timed relation between b_1 and b_2 is consistent with that of t_1 and t_2 in the composite process. s_1 is a service that can implement t_1 and t_2. Let us illustrate a more sophisticated example. In Fig 5, as the timed relation between $s_1.op_4$ and $s_1.op_2$ is not consistent with the timed relation between t_2 and t_5, s_1 is not an appropriate service. In Fig 5, as there exist a path from $s_2.op_1$ to $s_2.op_3$, the timed relation between them is consistent with the timed relation between t_2 and t_5, s_2 will be an appropriate service. Also, service s_3 will be the appropriate service for the under x_3 requirement.

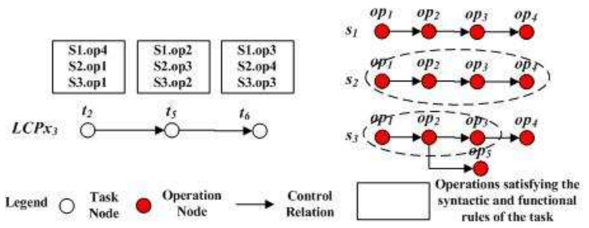


Fig 5 Example for Service Discovery

For composite process *CP*, the requirements generating from *CP* forms the set *R*={r_1, r_2,…, r_n}. There exists a set *S* of services, for each service s_i in *S*, there exists one and only one r_j in *R*, which can be implemented by s_i. Since that the service model of services may be heterogeneous from the local composite process of *r*, there may exist additional interactions between services in *S* which may cause deadlock among services. Thus, for services in *S*, a compatible detection of these services will be done in order to determine whether the service set is a candidate composite service. As lots of works have been done in this field [8], in this paper, for the limitation of this paper, we will not give algorithm for detecting the compatibility among discovered services.

4. EXPERIMENTATIONS

The algorithms are implemented in Java. All the experiments are carried out on Pentium IV 2.4GHz with 1GB RAM under

Windows XP. We will compare the performance of current composite process oriented service discovering algorithm [7] and the proposed algorithm.

Experimentation 1 Disruption of Business Relation

On the testing set where the number of service is 1600, randomly generating 10 requirements towards service, compare the performance of current composite process oriented service discovering algorithm and proposed service discovering algorithm in preserving the business relation. The experimentation is shown in Table 2 where if the discovered service satisfies functional semantic rules of MMR, it will not disrupt the business relation signifying as 1 and if such service does not satisfy the rule, it will disrupt the business relation signifying as 0.

Table2 Comparison of Disruption of Business Relation

Requirement towards Service	1	2	3	4	5	6	7	8	9	10
The Proposed Algorithm	1	1	1	1	1	1	1	1	1	1
Current Algorithm	0	0	1	0	1	0	1	1	0	0

Table 2 shows that current composite process oriented service discovering algorithm will be likely to disrupt the business relation while the proposed algorithm can preserve the business relation. This is because that in the proposed algorithm, the requirement towards service constrains that tasks in the requirement must be implemented by a same service, while in the current algorithm, tasks in the requirement can be implemented by operations from different services.

Experimentation 2 Disruption of Timed Relation

On the testing set where the number of service is 1600, randomly generating 10 requirements towards service, compare the performance of current composite process oriented service discovering algorithm and proposed service discovering algorithm in preserving the timed relation. The experimentation is shown in Table 3 where if the discovered service satisfies behavioral semantic rules of MMR, it will not disrupt the timed relation signifying as 1 and if such service does not satisfy the rule, it will disrupt the business relation signifying as 0.

Table3 Comparison of Disruption of Timed Relation

Requirement towards Service	1	2	3	4	5	6	7	8	9	10
The Proposed Algorithm	1	1	1	1	1	1	1	1	1	1
Current Algorithm	0	0	0	0	1	0	1	0	0	1

Table 3 shows that current composite process oriented service discovering algorithm will be likely to disrupt the timed relation while the proposed algorithm can preserve such relation. This is because that in the proposed algorithm, the requirement towards service constrains the timed relation between operations in the service while in the current algorithm, the requirement lacks of such constraint.

Experimentation 3 Discovering Efficiency

The experimentation is taken on 4 testing sets where the numbers of services are 200, 400, 800 and 1600 respectively. Randomly generate 20 requirements towards service, and take the average discovering time as the result. Compare runtime of the proposed algorithm and composite process oriented service discovering algorithm, the experimentation result is shown in Fig 6.

Fig 6 shows that the runtime of current algorithm is a little less than that of the proposed algorithm. This is because that in the proposed algorithm, it is needed to test whether the operations in the service have the consistent timed relation as tasks in the requirement towards service while in the current algorithm, this work is not done.

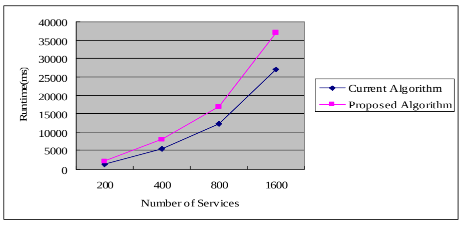


Fig 6 Comparison of Runtime

5. CONCLUSION

This paper proposes an approach for solving the problem of composite process oriented service discovery. In such approach, based on the requirement constraining timed and business relation among services, services with operations satisfying such relation will be discovered. The experimentation shows that the proposed algorithm can preserve the timed and business relation effectively.

In this paper, we assume that different genes in the business partition of the composite process will be implemented by different services. However, in reality, one service may implement multiple genes. In this situation, there is a work needed to be done in order to test if the service can satisfy the timed relation between these genes and in the future, we will works on such problem. This work is supported by the National Natural Science Foundation of China under Grant No.60773218.

REFERENCES

[1] R. Hull and J. Su. Tools for composite web services: A short overview. SIGMOD Record, 34(2), 2005: 86–95.

[2] S. Narayanan and S. McIlraith. Analysis and simulation of web services. Computer Networkds, 42(5), 2003:675–693.

[3] W. M. P. van der Aalst. On the automatic generation of work-flow processes based on product structures. Computer in Industry, 39(2), 1999: 97–111.

[4] C. Gerede, R. Hull, O. Ibarra, and J. Su. Automated composition of e-services: Look aheads. 2nd Int. Conf. on Service-Oriented Computing, 2004:252-262.

[5] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella. Automatic composition of transition-based

semantic web services with messaging. Int. Conf. on Very Large Data Bases, 2005: 613–624.

[6] J. Cardoso and A. Sheth. Semantic e-workflow composition. Journal of Intelligent Information Systems, 21(3), 2003:191-225.

[7] R. Aggarwal, K. Verma, J. Miller, and W. Milnor. Constraint driven web service composition in METEOR-S. Int. Conf. on Services Computing. Shanghai, China, 2004: 23-30.

[8] S. G. Deng, Z H. Wu, M. C. Zhou and J. Wu. Modeling Service Compatibility with Pi-calculus for Choreography. Int. Conf. on Conceptual Modeling. Arizona, USA, 2006: 26-39.