UDC

E-WsFrame：A Framework Support QoS Driven Web Services Composition

Chen Yan-ping, Li Zeng-zhi

Department of Computer Science and Technology, Xi'an Jiaotong University, 710049, Xi'an China
yanping@tom.com, lzz@mail.xjtu.edu.cn

Abstract.
Providing composed Web Services based on the QoS requirements of clients is still an urgent problem to be solved. In this paper, we try to solve this problem. Firstly, we enhanced the current WSDL to describe the QoS of services, and gave a way to choose the proper pre-exist services based on their QoS. Then, an evolved Web Service frame is proposed by adding a new role to the current Web Service frame to realize the QoS driven composition. Finally, we design and implement a prototype, which can flexible select and compose Web Services at run time.

Keywords:
Web Service Composition，Service Management，WSDL

1 Introduction

Web Service is a hotspot in the research of SOA as the best realization of SOA [1]. The primary goal of Web Service is to meet clients' requirements by integrating the pre-existing resources distributed in Internet, therefore, composing the existed service to provide an add-value complex service is a good way when the existing single service[1] cannot meet clients' requirements (functional and/or non-functional). As Web Services become more commercial, the service should not only satisfy the functional requirements of clients but also non-functional requirements. So, the main purpose of this paper is to dynamically provide composed Web Services according to clients' QoS requirements

There are several problems in Web Service frame needed to be studied such as service composition, data integration of services, security of services, etc. However, all these existing problems are aroused by the composition [2]. The task of composition is to combine and link existing Web services to create new Web services. Lots of researchers have paid their attentions to the service composition. SELF-SERV [3] is a platform that can provide composed service, but SELF-SERV emphasizes only on functional composition and ignores the QoS requirements of clients. A way to compose components based on QoS is proposed in [4], but it gives no details about how to describe the QoS of a service. In distributed environment, different service components may possess the same functions, and references [5-7] provide WSOL (Web Service Offering Language) that can change services at run time by dynamic switching in different service constrains. DAML-S [8] aims to define ontologies for service description that will allow software agents to reason about the properties of services.

[1] In this paper, a service indicates a Web Service

The above approaches solve some issues in composed Web Services from different views, but none of them can give a whole and realizable way to provide composed Web Services based on clients' QoS. To describe QoS requirements of clients, we proposed a new Web Service description language-EWSDL (Enhanced Web Service Description Language), and optimized concurrent Web Service frame with an evolved role of provider to meet non-functional requirements of clients to realize compositions.

The remainder of this paper is organized as follows. In section 2 we give an enhanced Web Service description language-EWSDL. In section 3, we propose an evolved Web Service frame to support Web Service composition based on EWSDL, we also give a prototype and make some experiments in section 4. The paper is concluded in section 5.

2 Web Service description model

2.1 QoS property of Web Services

There are many failed projects because of ignoring non-functional properties of software [9]. Definitions of non-functional properties of software are various and have no unified definition [10]. However, there are still some broadly accepted views about the non-functional properties. Generally speaking, the non-functional properties of software should include performance, reusability, maintainability, security, reliability, and availability. In Internet, we call the non-functional property as QoS property [13].

If a service provider only takes functional requirement of clients into account without considering non-functional requirements when providing the service, the provided service will not be accepted by clients at runtime. For example, a loanApprove service, which can provide a function of approving the loan, cannot guarantee the reliability and security then, obviously no client will use this service. How to assure QoS of services is a stringent problem for service providers. Now there are two different ways to solve this problem. One is syntax-based approach extending the current WSDL with more elements, and the other is to develop a new language based on semantic, such as XL, OWL-S. Both ways intend to add more information of the service when descript the service. But the language based on semantic is more complex. So, in this paper we extend WSDL to support more QoS descriptions.

WSDL2.0 describes three functional properties of Web Service: what to do, where to do, and how to do. However, WSDL cannot describe non-functional properties of services [11]. So, WSDL cannot be used to automatic service lookup and large-scale reuse and composition. Because of these defections, WSDL should be extended to include more information. In addition, only when including the non-functional properties it is integrity.

Essentially, how to describe and quantitative analyze non-functional properties of software is a complex problem and is still needed to solve, so how to describe the non-functional properties of Web Service has no existing way. Many researchers have paid their attentions to this problem. V.Tosic proposed a Web Service Offering Language[5-7](WSOL) to extend WSDL by adding new mark, such as price, time, etc. In essence, WSOL only provides some disperse, predefined, and limited property plate, so it lacks of flexibility Some non-functional properties are given in [12], such as availability, channels, charging style, settlement, payment obligations, etc. The author also indicates that the non-functional properties of Web Service are actually the QoS properties actually. A model for Web Service discovery with QoS is given in [13], it gives some definitions of QoS properties, but these properties are not from the view of managing composed Web Services. In this paper we explore an enhanced Web Service description language according to the need of managing the QoS properties of composed Web Service.

2.2 Description model of Web Service

Definition 1. Description model of Web Service. Let S be the description model of Web Service which can be expressed as *S={Func，QoS}*， where, *Func* denotes the functional properties of *S*, and *QoS* denotes the non-functional properties of *S*.

The functions of Web Service is described by the portTypes of WSDL, in order to describe the QoS properties, a tOperationInst element is added to tport element of WSDL. The diagram of tOperationInst is given in figure 1.

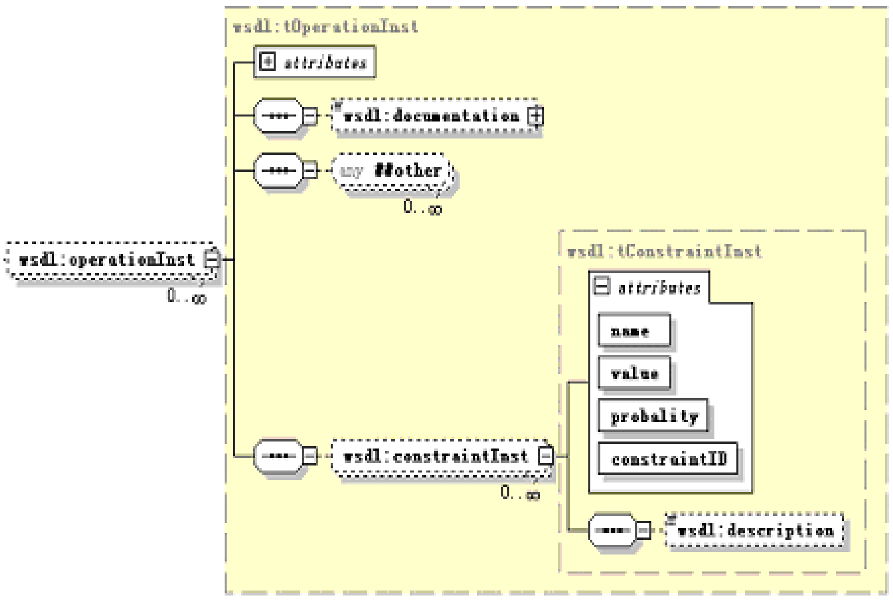


Figure 1. Graphic description of the grammar of tOperationInst in EWSDL

Considering that the QoS properties of Web Service should be independent to the domain of services and they also should be quantifiable, we use a vector in seven dimensions to describe the QoS of a Web Service (both element service[1] and composed service)

- **responseTime** represents the response time of a element service;
- **availability** represents the probability of the service can be used correctly;
- **concurrency** represents the maximum ability to support concurrent transactions;
- **expireTime** represents the expire time of a service, and the reliability of a service can be ensured before the expireTime,
- **price** represents the money the client should pay for this service,
- **fine** represents service provider(client) should compensate client(provider) for breaking the contract between them. Commonly, the **fine** has a linear relation with the **price**,
- **securityLevel** represents the security level of a service.

For a composed service CS=$\overset{m}{\underset{i=1}{Compose}}(Service_i)$, it also possesses the same parameters, but in a CS,

[1] Web Services that participate in the composition are all called element services. In this paper, there are no essences differences between the element service and the composed service except for the granularity.

these QoS parameters cannot be calculated by a simple mathematical function such as sum. For example, responseTime of CS is not the sum of responseTime of every element services when existing concurrent process in CS. Followings give a way to calculate the QoS parameters of CS.

- **ResponseTime**, considering that there exists concurrent service in composed Web Service, response time of composed service is not the sum of responseTime of all the element services, it should be the sum of critical route in execution process. CPA[15] is an algorithm to find the critical route.

$$\text{ResponseTime} = \text{CPA}\,(Service_1, Service_2, \ldots, Service_m),$$

- **Availability**$=\prod_{i=1}^{m} avalibility_i$,
- **Concurrency** = min (concurrency $_1$,concurrency $_2$,..., concurrency $_m$),
- **ExpireTime** = min (expireTime $_1$, expireTime $_2$,..., expireTime $_m$),
- **Price** = $\bigcup_{i=1}^{m} price_i$,
- **Fine** = *A*Price*, the client and the provider negotiate to get A,
- **SecurityLevel**=min(securityLevel $_1$,securityLevel $_2$,...,securityLevel $_m$).

These QoS properties are not independent, for example, a correlation between price and fine. This is why composition according to clients' QoS requirements is a difficult problem.

2.3 Composition selection algorithm

Definition 2 Service Class. Let A be a set of Web Services which have been registered into the Composer, and these services can be divided into several subsets according to the different functions, and each Web Service of A belongs to one subset at least, and the differences among services which belong to a same subset only are the QoS properties. Noticed, all the Service Classes form an overlay of A not a partition.

Definition 3 Candidate Relation. Let $Service_a, Service_b$ are two services , if

1) $\bigcup_{i=1}^{l} funtion(Service_a) \supseteq \bigcup_{i=1}^{k} funtion(Service_b)$, $l,k \in N, l \geq k$, or

2) $\bigcup_{i=1}^{k} funtion(Service_b) \supseteq \bigcup_{i=1}^{l} funtion(Service_a)$, $l,k \in N, k \geq l$,

There exist a Candidate Relation between Service$_a$ and Service$_b$, remembered as $Service_a \overset{\wedge}{\longleftrightarrow} Service_b$. Candidate Relation is an equivalence relationship.

For a service Composer, if exist $\text{Service}_a \overset{\wedge}{\longleftrightarrow} \text{Service}_b$, the Composer should order the proper service according to QoS requirements to clients. We define a new duality relation $\geq$ meaning "at least not worse than ", and deduced another two relations $\succ$ and $\approx$ based on $\geq$:

1) $Service_i^m \approx Service_i^n$ *iff* $Service_i^m \geq Service_i^n$ and $Service_i^n \geq Service_i^m$;

2) $Service_i^m \succ Service_i^n$ *iff* $Service_i^m \geq Service_i^n$ and not $Service_i^n \geq Service_i^m$.

It is a hard work for the service composer to provide a composition Web Service to clients according to their QoS requirements based on the following two reasons: first, there are no standard measurement of all these QoS properties, second, QoS properties are not independent, and one QoS property may favor/feed back others.

So, the multi-dimensional of the QoS properties cannot be merged into one dimension, we proposed a Composition Selection Algorithm, which relies on the following assumptions:

1) Clients' SLA is the QoS of the composed Web Services;
2) Composed services and element services use the same glossary;
3) The required QoS of clients can be pressed as a vector (ResponseTime$_c^*$, Availability$_c^*$, Concurrency$_c^*$, ExpireTime$_c^*$, Price$_c^*$, Fine$_c^*$, SecurityLevel$_c^*$);The requirements of clients usually fall into a range not a certain value, so we must define the relation between composed service and the required QoS (requirement) of clients, for example ResponseTime < ResponseTime$_c^*$, Availability > Availability$_c^*$, Concurrency < Concurrency$_c^*$, ExpireTime > ExpireTime$_c^*$, Price < Price$_c^*$, Fine < Fine$_c^*$, SecurityLevel > SecurityLevel$_c^*$;
4) The composition logic is predefined, and the aim is to simplify the composition and pay more attention to QoS.

Based on these assumptions, we use multi-object programming to model this problem and get the following programming matrix:

Table 1. Programming Matrix

	properties of composed Web services				
scheme	ResponseTime	Availability	Concurruncy	...	SecurityLevel
1	100ms	99%	1	...	3
2	70ms	97%	4	...	2
...					
m	80ms	97%	3	...	2

There are many ways to solve the multi-object programming [16,17,18], and in our prototype, we used the following method.

Step1, Construct programming matrix $A = (a_{ij})_{n\times m}$, and normalize A to $R = (r_{ij})_{n\times m}$ using proper ways;

Step2, Calculate $R = (r_{ij})_{n\times m}$ to get $\ddot{R} = (\dot{r}_{ij})_{n\times m}$, where $\dot{r}_{ij} = \frac{r_{ij}}{\sum_{i=1}^{n} r_{ij}}$, $i \quad N, j \in M$.

Step3, Get the information entropy E_j of every QoS, $E_j = -\frac{1}{ln\, n}\sum_{i=1}^{n} \dot{r}_{ij}\, ln\, \dot{r}_{ij}, j \in M$,

Step4, Get weight vector $\omega = (\omega_1, \omega_2, ..., \omega_m)$, where $\omega_j = \frac{1 - E_j}{\sum_{k=1}^{m}(l - E_k)}$,

Step5, Synthesis QoS of every scheme is defined as $z_i(\ \ ) = \sum_{j=1}^{m} r_{ij}\omega_j\omega, \quad i \in N$,

Step6, Sorting and selecting a scheme are according to $z_i(\omega)\,(i \in N)$.

3. Design of composition Web Services frame

3.1 Evolved Web Service frame

In order to realize the service composition based on EWSDL, we proposed an evolved Web Service Frame (E-WsFrame). Figure 2 gives the details of E-WsFrame.

There are three main roles in current Web Service framework: client, registry, and service provider. In E-WsFrame, we proposed a new role named after Composer. Composer takes charge of composing pre-exist services to meet clients' requirements (functional & non-functional). In figure 2, dashed lines represent interactions among roles in the concurrent Web Service framework, and solid lines represent the augment interactions in E-WsFrame.

Noticed that; in E-WsFrame, we set Composer as a part of provider, and some one may consider that Composer also can be looked as a part of registry. To simplify the problem, we consider Composer as a special provider. There is no need to extend the current registry because it cannot support EWSDL. How to evolve current registry to support EWSDL and how to lookup based on the EWSDL file are the next step of direction.

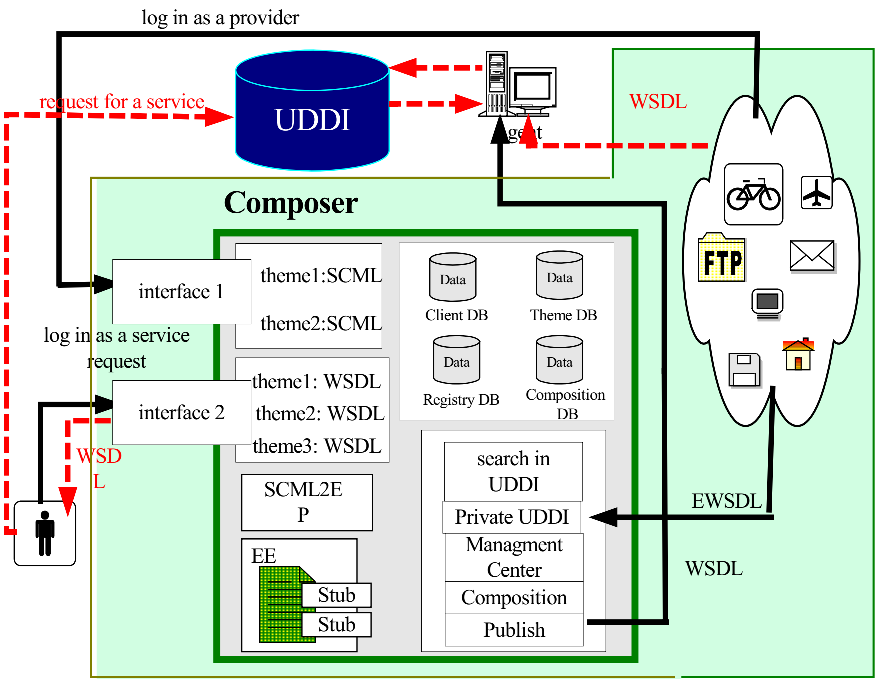


Figure 2. Structure of E-WsFrame

3.2 Service composition management language

Unlike other Web Service process description language, e.g. BPEL4WS, we defined a new language SCML (Service Composition Management Language). Using SCML, we can not only describe the process but also describe the QoS of services. Figure 3 gives the graphic description of the grammar of SCML.

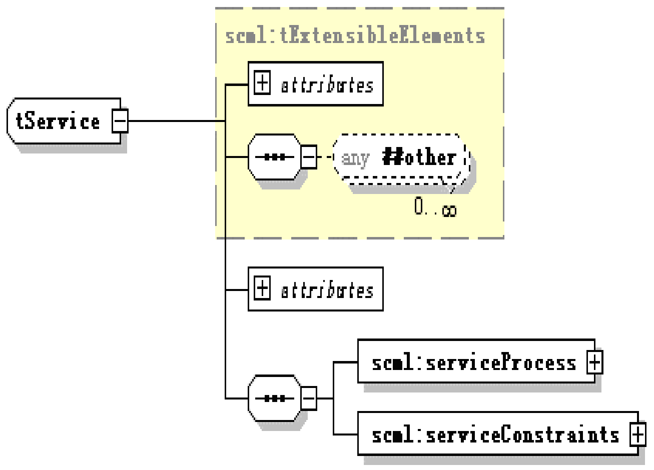


Figure 3. Graphic description of the grammar of SCML

In SCML, the root element tService is composed of two sub-elements. One is serviceProcess which describe the process and embody the functional properties of services, the other is serviceConstrains which describe constrains and embody the non-functional properties of services. Commonly used Web Service process description languages e.g. BPEL4WS [19], they only contain part information of serviceProcess of SCML. Figure 4 gives the graphic description of the grammar of element serviceConstraints in SCML.

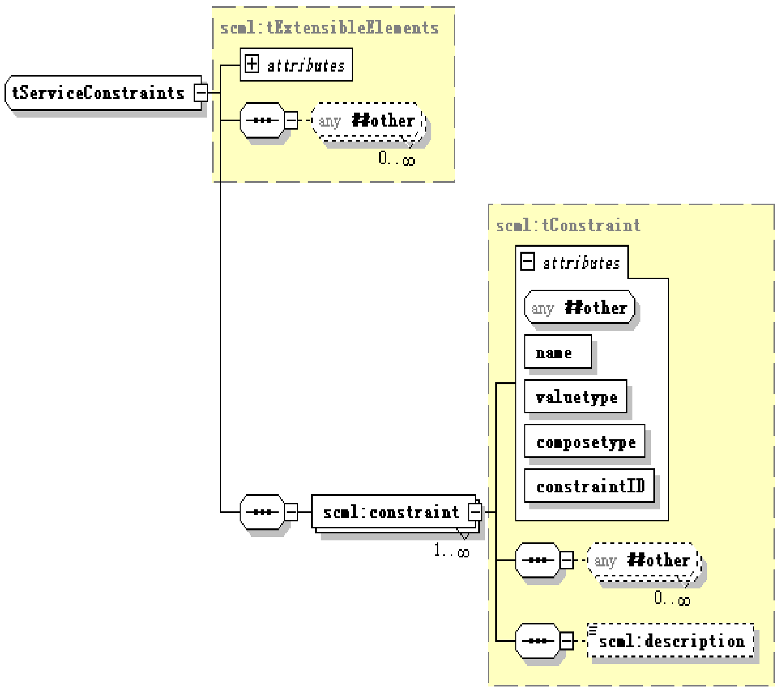


Figure 4. Graphic description of the grammar of serviceConstraints in SCML.

3.3 Implementation of E-WsFrame

3.3.1 Main steps before composition

Before providing the composed service, following works should be done:

- Specialist provides some plates of themes, which include composition logic;
- Programmers provide a SCML file of a composed service,
- Composer publish the SCML file on its homepage, and providers can provide its service according to SCML files,
- Service provider register their service description files (EWSDL) into the registry of Composer,
- SCML2EP converts a SCML into the executable process, and create a WSDL file of this composed Web Service then store this WSDL file into composition database of E-WsFrame;
- Once EE get the request from clients, EE execute the executable process that can get in step 5, and bind every element service at run time according to clients' QoS requirements.

3.3.2 Get composed services

Following give the steps to request and use a composed service:

- The client logs in Composer homepage, and finds the theme that he wants,
- The client and Composer should negotiate about the non-functional requirements,
- The management center of Composer chooses correct element services according to the contract, and submits the operation to stubs at run time.

4 Implement prototype of E-WsFrame

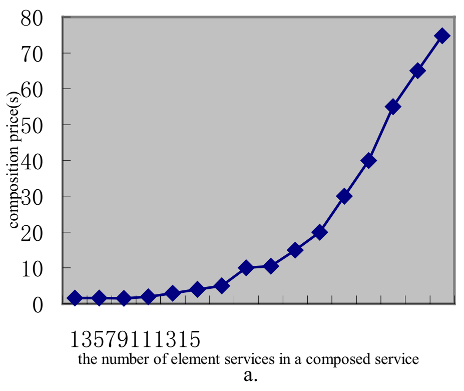
There are five main functional modules:

- ManagementCenter, it provides the functions such as register, lookup, and composition, etc.;
- DBs, it contains theme DB, Client DB, Registry DB, composition DB. Theme DB stores the information of the themes in current system. Client DB stores three kinds of interface: one is for the client who requests for composed Web Services, the second is for the element services providers, and the third is for E-WsFrame management. Registry DB stores the information of the service providers. Composition DB stores the schemes of themes.
- Login interface, there are two kinds of interface one is for clients who requests for composed services, the other is for service providers.
- SCML2EP, Converts the SCML files into executable processes, there are following main steps:
  1) Creates executable processes, and hands it to EE,
  2) Creates stubs,
  3) Fills the portType table in the portType table of theme DB,
  4) Fills the operation table in the operation table of theme DB,
  5) Fills the sequence table in the sequence table of composition DB, and
- EE, execution environment which can execute the process, and realize the dynamic composition.

4.2 Experiments and results

In E-WsFrame, client interface is developed by JSP+Win2000+Tomcat. ManagmentCenter is developed by Borland Delphi. DBs is developed by mysql, and an Active-BPEL engine is disposed on execution server.

We choose the travel problem as the scenario to realize the composed service. A travelService is composed by several element services such as accommodationService, sightseeingService, weatherService, and payService, etc. For two travelers even if they choose the same sightseeing, ManagementCenter can provide clients the different travelService according to their different QoS requirements such as price, accommodation requirement, etc., and fills the portType IDs into the EWSDL of element services in EE.



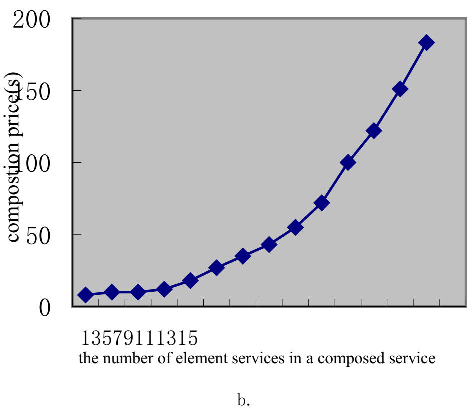


Figure 5. Composition price of Composer

In figure 5.a, each step in a composed service process has two element services of the Candidate Relation. While in figure 5.b, each step in a composed service process has five element services of the Candidate Relation. Examinations indicate that composition price will increase greatly with both the increasing numbers of candidate services in the composed Web Service and the number of element services in a composed service process. In order to deduce the cost, we should reduce the number of candidate services in composed service through ordering the candidate service by some algorithms and reduce the number of element services by increasing the granularity of services

5. Conclusion and future directions

The studies on Web Service management focus on two aspects, one is the self-management of the element Web Service; the other is the management of composed services [5-7]. The QoS properties are independent on above two aspects due to different targets. In this paper, we aim to manage the QoS of composed Web Services and propose a new service description language based on current WSDL, and then evolved the current Web Service architecture with a new role of Composer, finally give a prototype and do some examinations. Our experiments evaluate the correctness of the researches.

Short-term goals of our research contain three parts: firstly, we will extend the current UDDI to support EWSDL, which has been mentioned in section 3.1; secondly, we will add a monitor to E-WsFrame. The monitor can collect and analyze service data and make judgment when client or provider violates the agreement between them. Finally, we will order the candidate services by some algorithms to reduce composition price.

Acknowledgements

This research was sponsored by the National Science Foundation of China NSFC under the grant No. 90304006.

References

1. IBM dW 2004 special，http://www-128.ibm.com/developerworks/cn/.
2. YUE Kun, WANG Xiao-ling, ZHOU Ao-ying. Underlying Techniques for Web Services: A Survey. Journal of software ,2004，15(3):428-442.
3. Boualem Benatallah, Marlon Dumas, Quan Z. Sheng, Anne H.H. Ngu. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. Proceedings of the 18th International Conference on Data Engineering (ICDE.02).
4. LIAO Yuan, TANG Lei, LI Ming-Shu. A Method of QoS-Aware Service Component Compositon. Chinese Journal of computers, 2005,28(4):627-p634.
5. Esfandiari, B., Tosic, V. Requirements for Web Service Composition Management, in Proc. of the 11th HP-OVUA Workshop, Paris, France, June 21-23, 2004.
6. Tosic, V., Patel, K., Pagurek, B. WSOL - Web Service Offerings Language, in Proceedings of the Workshop on Web Services, e-Business, and the Semantic Web - WES Bussler, C.et al.(eds.), Toronto, Canada, May 2002.
7. Tosic,V, Ma, W, Pagurek, B, Esfandiari,B. Web Services Offerings Infrastructure (WSOI) - A Management Infrastructure for XML Web Services. In Proc. of NOMS (IEEE/IFIP Network Operations and Management Symposium) 2004, Seoul, South Korea, April 19-23, 2004, IEEE, 2004, pp. 817-830.
8. Chakraborty D, Joshi A, Yesha Y, Finin T. GSD: A novel group-based services discovery protocol for MANETS. In: Proc. of the 4th IEEE Conf. on Mobile and Wireless Communications Networks. 2002.
9. Finkelstein A, Dow ell J. A Comedy of Errors: the London Ambulance Service case Study. Proceedings of the 8th International Workshop on Software Specification and Design. 1996. 2-4.
10. YANG Fang-chun, LONG Xiang-ming. An Overview on Software Non-Functional properties Research. Journal of Beijing University of Posts and Telecommunications, 2004,27(3):1-11.
11. WSDL 2.0. http://www.w3.org/TR/2005/WD-wsdl20-primer-20050510/ .
12. JUSTIN O’SULLIVAN. What’s in a Service? Towards Accurate Description of Non-Functional Service Properties. Distributed and Parallel Databases,2002, 12:117 – 133
13. Shuping Ran. A Model for Web Services Discovery with QoS. ACM SIGecom Exchanges, 2003, 4(1):1-10.
14. McCall J.A.,Richards P.K.,Walters G.F. Factors in software quality. RADC:Technical report RADC-TR-77-363,1977.
15. Liangzhao Zeng, Boualem Benatallah, Marlon Dumas. Quality Driven Web Services Composition. Proceeding of WWW 2003, May 20-24, Hungary..
16. CHEN Ting, Programming Analysis. Science Press. Peking. 1987.
17. WANG Xiao-ping, CAO Li-ming. genetic algorithm- theory, application and realization. Xian Jiaotong University press. Xian, 2002.
18. Hwang C L, Yoon K. Multiple Attribute Decision Making and Applications. New York: Springer-Verlag, 1981.
19. OASIS Working Draft. BPEL: Business Process Execution Language 1.1.

Article received: 2006-01-06