Context-Sensitive QoS Model: A Rule-Based Approach to Web Service Composition

Tao Zhou
College of Computer Science, Zhejiang University,
Hangzhou, China
zt_zhoutao@zju.edu.cn

Xiaolin Zheng
(corresponding author)
College of Computer Science, Zhejiang University,
Hangzhou, China
xlzheng@zju.edu.cn

William Wei Song
Department of Computer Science, Durham University,
Durham, UK
w.w.song@durham.ac.uk

Xiaofeng Du
Department of Computer Science, Durham University,
Durham, UK
xiaofeng.du@durham.ac.uk

Deren Chen
College of Computer Science, Zhejiang University,
Hangzhou, China
drchen@zju.edu.cn

ABSTRACT
Generally, web services are provided with different QoS values, so they can be selected dynamically in service composition process. However, the conventional context free composition QoS model does not consider the changeability of QoS values and the context sensitive constraints during composition process. In this paper, we propose a rule based context sensitive QoS model to support the changeability of QoS values and the context sensitive constraints. By considering context in the QoS model, web service composition can be used widely and flexibly in the real world business.

Categories and Subject Descriptors
H.3.5 [**Information storage and retrieval**]: On-line Information Services –Web-based services; H.3.3 [**Information storage and retrieval**]: Information Search and Retrieval – Selection process.

General Terms: Algorithms, Economics, Measurement.

Keywords: Web service composition, Context model, Rule; QoS

1. INTRODUCTION
In the real world, web service selection is heavily affected by the environment. E.g., one service provider may announce that their web service is only compatible with some specified vendor's web services, so it excludes some other candidates; another service provider may claim that if you buy Service A from them, you will get Service B cheaper, so one service QoS values are dependent on other services. We call this situation as a context sensitive QoS based service composition. In the conventional context free QoS model, QoS value of the web services is constant during composition process. By contrast, in the context sensitive QoS model, a QoS value of a service is determined by this service together with other services related to this service. There are three main contributions in the paper. The first is to introduce a context sensitive QoS model and a context function *g*() to calculate the impact of context on each QoS value of a service. The second is to propose a rule model to represent context based on real world business perspective. The third is to apply the rule model in validation of service selections and changes of QoS values, which are not considered in the context free QoS model.

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

2. QOS MODELS
There are two layers in a web service composition model: abstract service (AS) layer describes the business process, and concrete service (CS) layer decides the concrete web services to use [1].

In traditional model, a business process P can be represented as a set of ASs: $P=\{AS_1, AS_2, ..., AS_n\}$. Each AS_i can have m CSs as its candidates: $AS_i=CS_{i1}|CS_{i2}| \ ... \ |CS_{im}$, $0<i\leq n$. Therefore, for each business process P, we have a set of service selections: $\rho=\{\rho_1, \rho_2, ..., \rho_w\}$. So *Service selection* is a set of concrete services that can achieve the goal of P. It's defined as follows $\rho_j=\{CS_{1,k1}, CS_{2,k2}, ..., CS_{n,kn}\} \mid \rho_j \in \rho, \ 0<j\leq w$. For each CS_{ij}, its quality metric Q_{ij} is a vector, $Q_{ij}=<c_{ij}, l_{ij}, a_{ij}, r_{ij}>$. And then an objective function $f_{k=1}^{w}(q(\rho_k))$ is introduced to represent user requirements. In traditional model, no matter which CS has been chosen previously, $Q_{ij}=<c_{ij}, l_{ij}, a_{ij}, r_{ij}>$ is constant, and all the candidates are available for AS_i.

In our context sensitive QoS model, not all the service selections are valid and $Q_{ij}=<c_{ij}, l_{ij}, a_{ij}, r_{ij}>$ is no more constant. Let ρ' be a set of service selections and R a set of constraints (from both service providers and users), then we can define a valid service selection as an element of a set ρ' whose elements satisfy the following condition: $\rho' = \{\mu \mid R, \mu \in \rho\}$. In order to cope with context sensitive situations, we introduce a context function *g*(). The function *g*() has two parameters, denoted below: $g(q_i, \rho_k)$ where $q_i=c_i|l_i|a_i|r_i$ $(0<i\leq n)$ is one of the QoS values for $CS_i \mid CS_i \in \rho_k, 0 < i \leq n$ and $\rho_k \mid \rho_k \in \rho', 0 < k < w'$ is the currently selected valid service selection. The return value of function *g*() indicate the impact of context on the QoS value of CS_{ij}. Below is an explanation for different return values: $g(q_i, \rho_k)>0$, the context positively influences q_i; $g(q_i, \rho_k)<0$, the context negatively influences q_i; $g(q_i, \rho_k)=0$, the context has no influence on q_i or there is no relevant context defined. We take the equation for the total cost C in the sequence pattern as an example. The equation for the cost is rewritten as $C=\sum_{i=1}^{n}(c_i + g(c_i, \rho_k))$, $0<k\leq w'$ (1).

This paper focus on business rules that adjust and restrict selecting suitable CS for AS, so we classify business rules into 3 types: **The qualitative constraints** which describe the validation of the selected candidates of CSs; **The quantitative rules** which describe the ways in which the CS QoS values change; and **the user satisfaction enhancement rules** which describe the extra

value added by the service providers to enhance user satisfaction. In general, each business rule can be represented as below: *H, LHS→RHS*, where *H* normally contains the attributes of the rule; *LHS* normally contains the rule fire conditions; *RHS* normally describes the actions to fire. Based on our requirements, we have extended *H* and *LHS* to contain more information about a rule. The extended *H* is denoted as a 4-tuple: $H=< n, p, V_g, V_c>$, where *n* is the name of the rule; *p* is a priority value of the rule; $V_g=\{v_{g1}, v_{g2}, ..., v_{gn}\}$: is a set of global variables; $V_c=\{v_{c1}, v_{c2}, ..., v_{cn}\}$: is a set of user defined variables. The extended *LHS* can be denoted as a 2-tuple: $LHS=< t, C>$, where *t* is a time period to specify when the rule is active; $C=\{c_1, c_2, ..., c_n\}$ is a set of conditions. The above three rule types can be formally defined as below:

Definition 3. A qualitative constraint is a rule that triggers a validation action to validate the validity of a service selection according to the constraints provided in *LHS*, denoted as *H, LHS→validate*(ρ_k)*=true/false*, where *validate()* is the validation action; $\rho_k \in \rho$ is a service selection; *true* is a possible return value of *validate*() action to indicate valid selection; *false* is a possible return value of *validate*() action to indicate invalid selection.

Definition 3.1 Validation phase: Assume we have a set of qualitative constraint $R=\{r_1,r_2,...,r_\tau\}$ and a set of service selections $\rho=\{\rho_1, \rho_2,..., \rho_w\}$, then after the validation phase all the service selections satisfy the following condition $\forall\ \rho_k\ \forall\ r_j\ [\rho_k \in \rho, r_j \in R | r_j.validate(s_i, \rho_k)=true]$, $0<k\leq w$, $0<j\leq \tau$, $0<i\leq n$.

Definition 4. **A quantitative rule** is a rule that triggers a series of actions to update the QoS values of the services in a valid service selection, denoted as *H, LHS→Update*, where $Update = \{update(\rho_k)_1, update(\rho_k)_2, ..., update(\rho_k)_m\}$, $m>0$: is a set of update actions to update the QoS values of the services in a service selection according to the conditions in *LHS*; $\rho_k \in \rho'$ is a valid service selection.

Definition 4.1 QoS value update phase: Once a quantitative rule is fired, its update actions generate a set of value $X=\{x_1, x_2, ..., x_n\}$ for each relevant QoS value of all the services in a service selection to indicate the impact of the context. The x_i can be considered as a return value of the function *g*(). Therefore, the Eq. (1) can be rewritten as $C = \sum_{i=1}^{n} (c_i + x_i)$.

Definition 5. A user satisfaction enhancement rule triggers a series of actions that add user satisfaction enhancement but do not directly update the QoS values of a selected service or validate a service selection, denoted as *H,* So *LHS→Action* where $Action=\{action(\rho_k)_1, action(\rho_k)_2, ..., action(\rho_k)_m\}$, $m>0$, which are a set of actions that add user satisfaction enhancement onto certain service selections. It satisfies the following condition: $Action \cap Update = \varnothing \wedge valicate(\rho_k) \notin Action; \rho_k \in \rho'$.

Definition 5.1 Satisfaction enhancement adding phase: By firing the *user satisfaction enhancement rules*, the relevant service selections will be added user satisfaction enhancement provided by the service providers. At the current stage of our work, the enhancement is represented to the user as messages.

3. EVALUATION

We emulated a web service composition environment by creating about 2000 web services, and generated 1000 test cases to evaluate the advantages of the context sensitive QoS model. The evaluation is performed through two steps:

QoS improvement evaluation: In this step, we evaluate whether the context sensitive QoS model can improve the overall QoS of a composite service. From the result we can see 58.4% positive effect, 39.8% no effect and 1.8% negative effect.

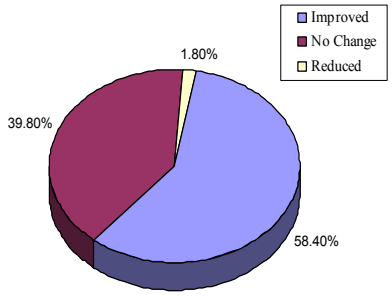

Fig.1. shows the QoS improvement evaluation result.

Execute-ability evaluation: In this step, we evaluate how many composite services generated under the two QoS models can be actually executed. From Fig.2 we can see that when the number of AS in a composite service getting bigger, more and more invalid composite services are generated from the context free QoS model.

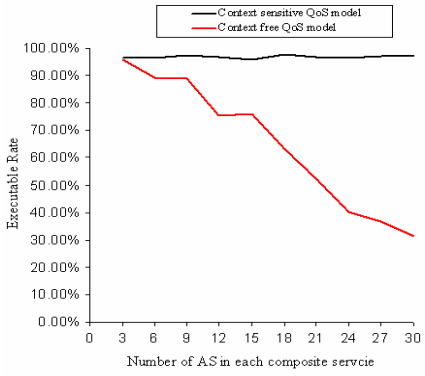

Fig.2. shows the execute-ability evaluation result.

4. CONCLUSION

In this paper, we proposed a context sensitive QoS model in contrast with the conventional context free QoS model. The model constraints in order to compute correct QoS value of a composite service. A context function *g*() is proposed to calculate the impact of context on each QoS value of a service. we also proposed a rule model which categorize and construct rules not only from the service composition perspective, but also from the business perspective so that it can correctly represent both user and service provider's requirements and constraints. We believe by combining the satisfactory model with the context sensitive QoS model, service composition can be performed more efficiently and effectively so that it can be applied widely in the real world business.

5. REFERENCES

[1] Canfora, G., Penta, Canfora, M. Di, Esposito, Penta, R. . Esposito, and Villani, M. Villani. A Lightweight Approach for QoS–Aware Service Composition. Proc. 2nd International Conference on Service Oriented Computing (ICSOC04), 2004.