Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 641361, 15 pages
doi:10.1155/2012/641361

Research Article

Service Selection of Ensuring Transactional Reliability and QoS for Web Service Composition

Guojun Zhang,[1] Liping Chen,[2] and Weitao Ha[2]

[1] *College of Communication Engineering and Center of Network Engineering Technology, Weinan Normal University, West Chaoyang Street, 714000 Weinan, Shaanxi, China*
[2] *College of Mathematics and Information Science and Center of Network Engineering Technology, Weinan Normal University, 714000 Weinan, China*

Correspondence should be addressed to Guojun Zhang, wnchlp@gmail.com

Received 11 April 2012; Accepted 26 June 2012

Academic Editor: Yuping Wang

Service-Oriented Architecture (SOA) provides a flexible framework of service composition. Using standard-based protocols, composite service can be constructed by integrating component services independently. As component services are developed by different organization and offer diverse transactional properties and QoS characteristics, it is a challenging problem how to select suitable component services which ensure reliable execution of composite Web service and construct the optimal composite Web service. In this paper, we propose a selection approach that combines transactional properties of ensuring reliability and QoS characteristics. In the selection approach, we build automaton model to implement transactional-aware service selection and use the model to guarantee reliable execution of composite Web service. We also define aggregation functions, and use a Multiple-Attribute Decision-Making approach for the utility function to achieve Qos-based optimal service selection. Finally, two scenarios of experiments are presented to demonstrate the validity of the selection approach.

1. Introduction

Service-Oriented Architecture (SOA) is becoming a major software framework for distributed applications such as e-business and enterprise systems. Using standard protocols, Web services developed by different organizations can be dynamically and flexibly composed. Web services composition can build value-added applications by aggregating several existing Web services together according to dynamic business requirements.

Service consumer submits business objective which the composition service should achieve, along with some constraints and preferences [1]. Based on those, YAWL [2] is used to be chosen to represent the workflows model and describe the logic of composition Web service. When each activity of a workflow is implemented by a component Web service,

we obtain a composition Web service. Component Web service of every activity fulfilling the user's goal is selected among a set of candidate services. In this paper we focus on this selection which could be dynamic and automatic, and we do not focus on the execution step, the recovery, or replanning problems.

Although the problem of web service selection and composition has received a lot of attention by many researchers in recent years, designing a composite Web service to ensure not only correct and reliable execution but also optimal QoS remains an important challenge [3]. Indeed, these two aspects of selection are always implemented separately. In [4], a solution combines the business process adequacy of workflow systems and the reliability of transactional processing. Similarly, in [5–9], only transactional properties are considered, and QoS-based selection is not involved. For these researches Web services composition based on transactional properties ensures a reliable execution; however, an optimal QoS composite Web service is not guaranteed. The approaches [10–14] implement conventional optimal QoS composition, but composing optimal QoS Web services does not guarantee a reliable execution of the resulting composite Web service. Therefore, transactional-based and QoS-based should be integrated.

Our research objective is to propose a reliable and efficient selection approach for automatic Web service composition, where transactional and QoS requirements are both integrated in the selection process. Transactional requirements should be considered firstly, because if the selection is done based on QoS firstly (transactional selection followed by a QoS), a local or global optimized QoS composition may not guarantee transactional execution. In other words, the overall consistency and successful termination of composition Web service are not ensured. For those reasons, the selection is done in two separate steps: transactional service selection starts firstly, and the QoS-aware service selection is embedded with the transaction-aware service selection.

The innovation of the paper mainly lies in a few aspects. Firstly, we present an ensuring transactional reliability and QoS service selection approach. The selection of the component Web services is done by matching the Web services properties with the user's desires. More precisely, the selection is realized depending on transactional and QoS user requirements. The former is established by means of a risk tolerance notion that is given in the paper. And it indicates if the results can be compensated or not. The latter is expressed as a weight over each QoS criterion. Secondly, we build automaton model to implement transactional-aware service selection, and using the model composite Web service can guarantee transactional execution. Moreover, our method is scalable because the user has only to define a global transaction requirement and does not have to define the possible termination states of all component Web service. Finally, nonfunctional QoS aspects (e.g. response time, availability, etc.) are also crucial for selecting the web services to take part in the composition. In the paper, we consider quantitative nonfunctional properties that can include generic QoS attributes like response time, availability, price, reputation, and so forth, as well as domain-specific QoS attributes, for example, bandwidth for multimedia Web services. We define aggregation functions and use a Multiple Attribute Decision-Making approach [15] for the utility function. The utility computation involves scaling the QoS attributes' values to allow a uniform measurement of the multidimensional service qualities independent of their units and ranges.

2. Web Service Transaction Descriptions

As composition web service is a cross-organizational collaborative system, unexpected behavior or failure implement of a component service might not only lead to its failure but

also may bring negative impact on all the participants of the composition. In order to ensure overall consistency, execution of either a component service or composition web service requires transactional properties. Section 2.1 describes web service transactional property. Section 2.2 defines composition web service transactional property.

2.1. Web Service Transactional Property

The main transactional properties of a Web service we are considering are retriable, compensatable, and pivot [16]. A service s is said to be retriable (r for short) if it is sure to be successfully completed after several finite activations. s is said to be compensatable (c for short) if it offers compensation policies to semantically undo its effects. Then, s is said to be pivot (p for short) if once it is successfully completed, its effects remain forever and cannot be semantically undone, and if it fails, it has no effect at all. A completed pivot Web service cannot be rolled back. Naturally, a service can combine properties, and the set of all possible combinations is {p, c, pr, cr}.

2.2. Composite Web Service Transactional Property

A composite Web service (CWS for short) is a conglomeration of existing Web services working in tandem to offer a new value-added service [17], which is often long-running, loosely coupled, and cross-organizational applications. It orchestrates a set of services, as a workflow-based composition, to achieve a common goal [18]. Transactional property of CWS depends on two sides, transactional property of every component service and workgroup patterns. Inspired by Mehrotra et al. [16], we have the following definitions.

Definition 2.1. Atomic property of CWS (a for short) is that if all component services are completed successfully, their effects remain forever and cannot be semantically undone, and if one component service cannot be completed successfully, previously successful component services have to be compensated. (In other words, if one component service fails, the execution result is compensated).

Definition 2.2. Compensatable property of CWS (c for short) is that all component services are compensatable.

Definition 2.3. Retriable property of CWS (r for short) is that all component services are retriable.

Definition 2.4. Transactional Composite Web Service (TCS) is CWS whose transactional property is in {a, ar, c, cr}.

In this paper, our object of transactional services selection makes composition service to be TCS. TCS can ensure composite service is completed successfully and the consistency of component services. TCS is composed of elementary services whose transactional property is in {p, c, pr, cr} or is composed of CWS whose transactional property in {a, ar, c, cr}.

3. Transactional Automaton Services Selection

Every activity of workgroup selects proper service that makes composition service not only become TCS but also satisfy use's requirement. The selection depends on two

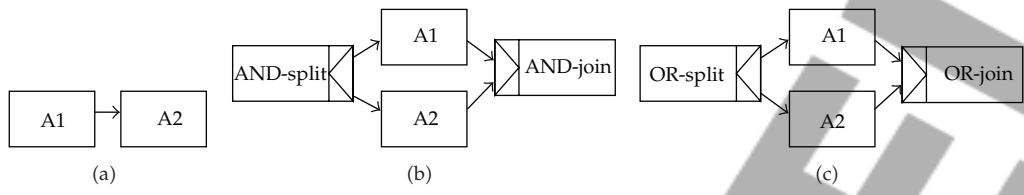


Figure 1: Workflow patterns. (a) Sequential pattern. (b) AND-split and AND-join patterns. (c) XOR-split and XOR-join patterns.

factors: workgroup pattern and use's transactional requirement. Use's transactional requirement is defined in terms of risk tolerance in Section 3.1, dependency between activities in the workgroup is workgroup pattern showing in Figure 1, that is, sequence, parallel split (AND-split), exclusive choice (XOR-split), synchronization (AND-join), and simple merge (XOR-join). When a service WS1 is assigned to activity A1 and a service WS2 is assigned to activity A2, the obtained composite Web service CWS1 is represented by SEQ(WS1,WS2), where symbol SEQ() represents a sequential execution: WS1 is executed before WS2. The obtained composite Web service CWS2 is represented by PAR(WS1, WS2), where symbol PAR() represents the AND-split and AND-join patterns. PAR(WS1, WS2) means that both services are executed in parallel. We do not consider the XOR-pattern (XOR-split and XOR-join) because in an XOR-pattern the resulting "Composite" WS contains only one Web service WSi, and the WS transactional property corresponds to the transactional property of WSi. How to select service to assign each activity in the different workgroup pattern is described in Section 3.2.

3.1. Definition of Risk Tolerance

Usually user expresses requirements and constraints of QoS easily, but it is difficult to express use's transactional criteria. In order to explain the transactional Web service selection process, it is necessary to establish how the user can express their transactional criteria. We define risk tolerance which expresses importance of the uncertainty of application completion and recovery. In terms of the transactional properties for CWS, we believe that properties a and ar are riskier than c and cr. Indeed, properties a and ar mean that once a service has been executed, it cannot be rolled back. Therefore, we define two levels of risk tolerance in a transactional system.

Risk Tolerance 0

The system guarantees that if the execution is successful, the obtained results can be compensated by the user. In this level the selecting process generates a compensatable workflow.

Risk Tolerance 1

The system does not guarantee the successful execution but if it is achieved the results cannot be compensated by the user. In this level the selecting process generates an atomic workflow.

3.2. Transactional Automaton Services Selection

We can use transactional automaton selecting transactional property of next service according to workgroup pattern and previous service transactional property. In order to get transactional automaton we will propose some rules below.

The parameters are described as follows.

A_i is an activity of workgroup.

$S_i = \{s_{i1}, s_{i2}, \ldots, s_{in}\}$, where S_i is set of candidate service for activity A_i.

$P = \{\mathrm{p}, \mathrm{a}, \mathrm{pr}, \mathrm{ar}, \mathrm{c}, \mathrm{cr}\}$ is set of transactional property.

$\mathrm{tp}(s)$: if s is a service, $\mathrm{tp}(s)$ expresses transactional property of service s, $\mathrm{tp}(s) \in P$. If s is a type of service, $\mathrm{tp}(s)$ expresses transactional property set of services s, $\mathrm{tp}(s) \subseteq P$.

CWS = (ES, TP, PA) expresses composite Web service, where ES is set of component service, and TP is transactional property set of component service, and PA is workgroup pattern.

3.2.1. Services Selection Rule in the Sequential Pattern

Rule 1. One has $\mathrm{CWS} = (\mathrm{ES}, \mathrm{TP}, \mathrm{PA}) \wedge (\mathrm{ES} = S_i \cup S_{i+1}) \wedge \mathrm{tp}(S_i) = \{\mathrm{p}, \mathrm{a}, \mathrm{pr}, \mathrm{ar}\} \wedge \mathrm{PA} = \mathrm{SEQ}(S_i, S_{i+1}) \rightarrow \mathrm{tp}(S_{i+1}) = \{\mathrm{pr}, \mathrm{ar}, \mathrm{cr}\} \wedge ((\mathrm{tp}(S_i) = \{\mathrm{pr}, \mathrm{ar}\} \rightarrow \mathrm{tp}(\mathrm{CWS}) \in \{\mathrm{ar}\}) \vee (\mathrm{tp}(S_i) = \{\mathrm{p}, \mathrm{a}\} \rightarrow \mathrm{tp}(\mathrm{CWS}) \in \{\mathrm{a}\}))$.

From Rule 1 we can find if previous service transactional property is p, a, pr, or ar in the sequential pattern, and to obtain selected TCS the candidate service transactional property of next activity is pr, ar, or cr. CWS transactional property is a or ar and is moreover ar if all its components are retriable.

Proof. $\mathrm{tp}(S_i) = \{\mathrm{p}, \mathrm{a}, \mathrm{pr}, \mathrm{ar}\}$ expresses effects of the previous service which cannot be semantically undone, and in the sequential pattern the next should ensure successful execution. Therefore the next transactional property must be retriable, pr, ar, or cr. □

Rule 2. One has $\mathrm{CWS} = (\mathrm{ES}, \mathrm{TP}, \mathrm{PA}) \wedge (\mathrm{ES} = S_i \cup S_{i+1}) \wedge \mathrm{tp}(S_i) = \{\mathrm{c}, \mathrm{cr}\} \wedge \mathrm{PA} = \mathrm{SEQ}(S_i, S_{i+1}) \rightarrow \mathrm{tp}(S_{i+1}) = P \wedge ((\mathrm{tp}(S_{i+1}) = \{\mathrm{c}, \mathrm{cr}\} \rightarrow \mathrm{tp}(\mathrm{CWS}) \in \{\mathrm{c}, \mathrm{cr}\}) \vee (\neg \mathrm{tp}(S_{i+1}) = \{\mathrm{c}, \mathrm{cr}\} \rightarrow \mathrm{tp}(\mathrm{CWS}) \in \{\mathrm{a}, \mathrm{ar}\}))$.

From Rule 2 we can find if previous service transactional property is c or cr in the sequential pattern, and to obtain selected TCS the candidate service transactional property of next activity is not required, as long as the next service is transactional service. When the WS assigned to the next activity is either c or cr CWS transactional property is c or cr. Moreover, when both component services are r, the resulting TCS is r.

Proof. Because the previous service transactional property is c or cr, if the next service failed the previous service is compensatable. Therefore whatever transactional property of next activity is, CWS is transactional. □

3.2.2. Services Selection Rule in the Parallel Pattern

Rule 3. One has $\mathrm{CWS} = (\mathrm{ES}, \mathrm{TP}, \mathrm{PA}) \wedge (\mathrm{ES} = S_i \cup S_{i+1}) \wedge \mathrm{PA} = \mathrm{PAR}(S_i, S_{i+1}) \wedge \mathrm{tp}(S_i) = \{\mathrm{p}, \mathrm{a}\} \rightarrow \mathrm{tp}(S_{i+1}) = \{\mathrm{cr}\} \wedge \mathrm{tp}(\mathrm{CWS}) = \mathrm{a}$.

Rule 3 is applied to parallel pattern. S_i and S_{i+1} are branches of parallel pattern. If assigned service transactional property of one activity is p or a, the other assigned parallel service should be cr.

Proof. In parallel pattern, when assigned service transactional property of one activity is p or a, if it is completed successfully, and its effect is not semantically undone, therefore, the other assigned parallel service is retriable (r), which can guarantee a successfully termination. If it failed, the other should be compensatable (c). So to ensure a successful termination and be compensatable simultaneously, transactional property of the other selected services should only be cr. From Definitions 2.1–2.3, TCS is a. □

Rule 4. One has $\mathrm{CWS} = (\mathrm{ES}, \mathrm{TP}, \mathrm{PA}) \wedge (\mathrm{ES} = S_i \cup S_{i+1}) \wedge \mathrm{PA} = \mathrm{PAR}(S_i, S_{i+1}) \wedge \mathrm{tp}(S_i) = \{\mathrm{pr}, \mathrm{ar}\} \rightarrow \mathrm{tp}(S_{i+1}) = \{\mathrm{pr}, \mathrm{ar}, \mathrm{cr}\} \wedge \mathrm{tp}(\mathrm{CWS}) = \mathrm{ar}$.

Rule 4 is applied to parallel pattern. S_i and S_{i+1} are branches of parallel pattern. If assigned service transactional property of one activity is pr or ar, the other selected parallel service should be pr, ar, or cr, and TCS is ar.

Proof. In parallel pattern, when assigned service transactional property of one activity is pr or ar, it can ensure to be completed successfully. Therefore the other assigned parallel service is pr, ar, or cr. From Definitions 2.1–2.3, TCS is ar. □

Rule 5. One has $\mathrm{CWS} = (\mathrm{ES}, \mathrm{TP}, \mathrm{PA}) \wedge (\mathrm{ES} = S_i \cup S_{i+1}) \wedge \mathrm{PA} = \mathrm{PAR}(S_i, S_{i+1}) \wedge \mathrm{tp}(S_i) = \{\mathrm{c}\} \rightarrow \mathrm{tp}(S_{i+1}) = \{\mathrm{c}, \mathrm{cr}\} \wedge \mathrm{tp}(\mathrm{CWS}) = \mathrm{c}$.

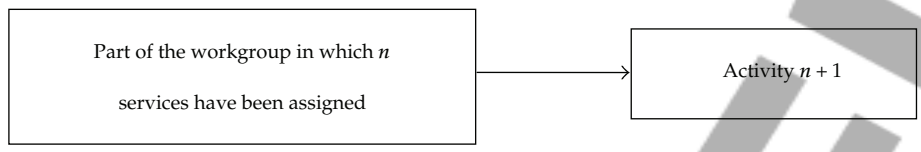
Rule 5 is applied to parallel pattern. S_i and S_{i+1} are branches of parallel pattern. If assigned service transactional property of one activity is c, the other selected parallel service should be c or cr, and TCS is c.

Proof. In parallel pattern, when assigned service transactional property of one activity is c, it offers compensation policies to semantically undo its effects, but it can fail. Therefore, the other assigned parallel service is c or cr. From Definition 2.2, TCS is c. □

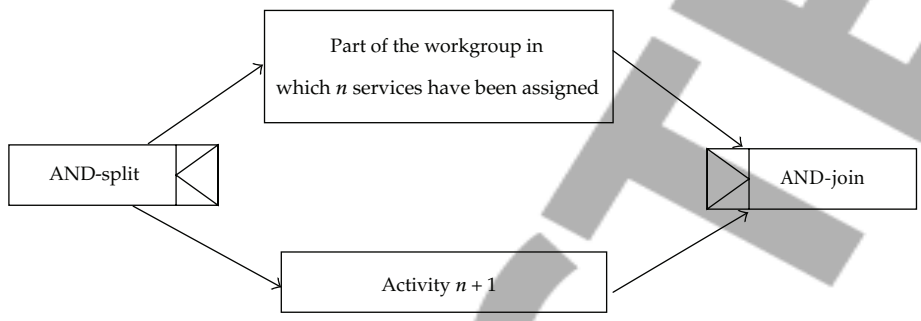
Rule 6. One has $\mathrm{CWS} = (\mathrm{ES}, \mathrm{TP}, \mathrm{PA}) \wedge (\mathrm{ES} = S_i \cup S_{i+1}) \wedge \mathrm{PA} = \mathrm{PAR}(S_i, S_{i+1}) \wedge \mathrm{tp}(S_i) = \{\mathrm{cr}\} \rightarrow (\mathrm{tp}(S_{i+1}) = \{\mathrm{p}, \mathrm{a}\} \rightarrow \mathrm{tp}(\mathrm{CWS}) = \mathrm{a}) \vee (\mathrm{tp}(S_{i+1}) = \{\mathrm{pr}, \mathrm{ar}\} \rightarrow \mathrm{tp}(\mathrm{CWS}) = \mathrm{ar}) \vee (\mathrm{tp}(S_{i+1}) = \{\mathrm{c}\} \rightarrow \mathrm{tp}(\mathrm{CWS}) = \mathrm{c}) \vee (\mathrm{tp}(S_{i+1}) = \{\mathrm{cr}\} \rightarrow \mathrm{tp}(\mathrm{CWS}) = \mathrm{cr})$.

Rule 6 is applied to parallel pattern. S_i and S_{i+1} are branches of parallel pattern. If assigned service transactional property of one activity is cr, the other selected parallel service should be in set of {p, a, pr, ar, c, cr}. When the other is p/a, pr/ar, c, or cr, corresponding to transactional property of CWS is a, ar, c, or cr.

Proof. In parallel pattern, when assigned service transactional property of one activity is cr, it offers compensation policies to semantically undo its effects, and it can ensure to be completed successfully. Therefore, the other selected parallel service is only transactional service. □



(a) $n+1$th service selecting in Web service composition of the sequential pattern



(b) $n+1$th service selecting in Web service composition of the parallel pattern

Figure 2: $n+1$th service selecting.

3.2.3. Transactional Automaton Model for Services Selection to Web Service Composition

Either elementary service or CWS can be assigned to activity of workgroup from the left to the right in the sequential patterns and from the top to the bottom in the parallel patterns. After n services assigned to $n(n \geq 1)$ activities, the different possible configuration of the activity $n+1$ is shown in Figures 2(a) and 2(b). As shown in Figure 2(a), from the previous workflow where services have been assigned, CWS is obtained which is transactional, and the next assigned service of activity $n+1$ is selected from candidate services according to the previous CWS transactional property and Rule 1 or Rule 2. To Figure 2(b), the CWS which is produced by part of workflow where services have been assigned is parallel to assigned service of activity $n+1$, and the $n+1$ assigned service is selected according to the CWS transactional property and one of Rule 3 to Rule 6.

To guide service selection that is driven by transactional property, we give transactional automaton model according to Rule 1 to Rule 6, and it represents all possible TCSs which could be obtained by the selection process. It is described by using Figure 2. I is initial state which is owned by service of first activity in the {p, a, pr, ar, c, cr}. {SEQ(,p), SEQ(,a), SEQ(,pr), SEQ(,ar), SEQ(,c), SEQ(,cr), PAR(p), PAR(a), PAR(pr), PAR(ar), PAR(c), PAR(cr)} represents components of different the transactional properties that are executed in sequential or in parallel. The final state is in {c, a, cr, ar} corresponding to transactional property of a TCS.

3.2.4. Example of Service Selection Driven by Transactional Automaton Model

The input workflow is shown in Figure 4, for simplicity, where every assigned service is an elementary service. If it is p for transactional property of assigned service WS1 of the first activity A1, then according to transactional automaton model, transactional state of workgroup will go from p/a branch of initial state I to the state a. Since the WS1 and

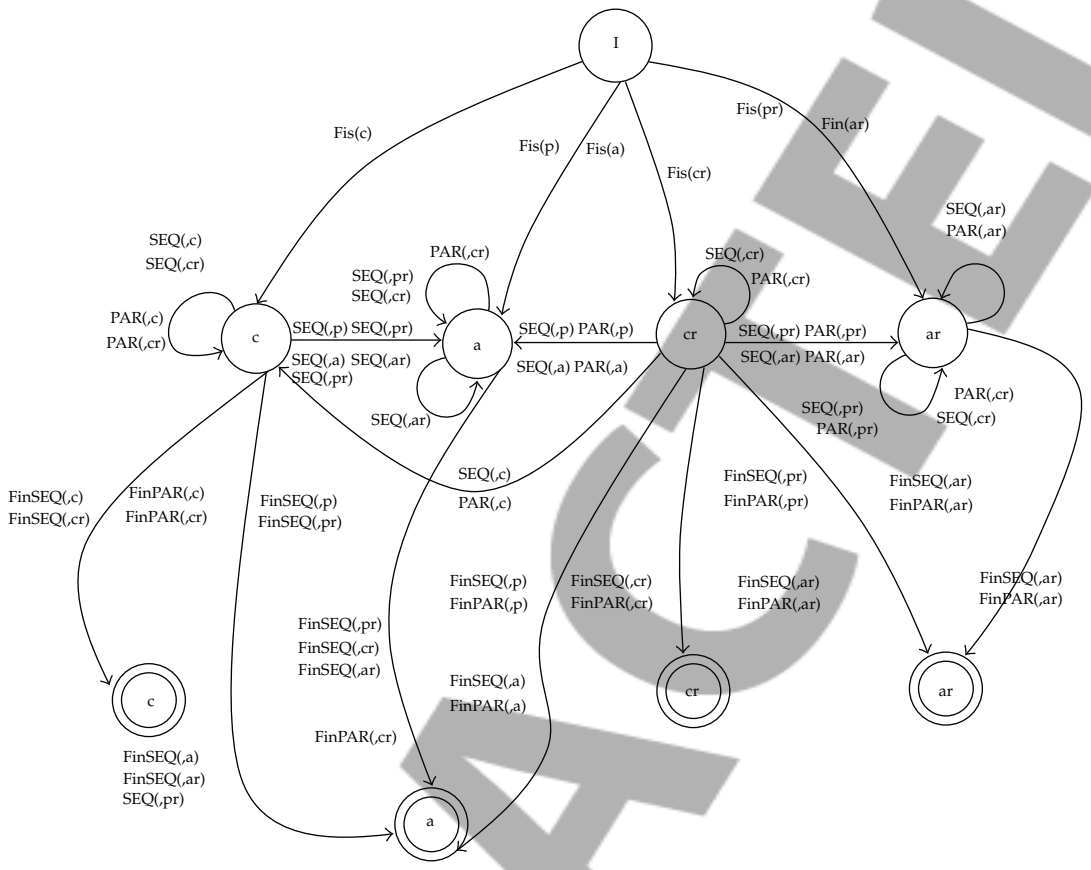


Figure 3: Transactional automaton model.

CWS2 compose in sequential pattern and CWS2 is composite service of WS2, WS3, and WS4, transactional property of the selected CWS2 must be ar or cr (according to SEQ(,ar) and SEQ(,cr) in the automaton model which can make transactional property of TCS be a). CWS2 is composed of CWS2′ and WS4, so CWS2′ should be ar or cr, and transactional property of the selected WS4 is pr or ar. CWS2′ is also composed of WS2 and WS3, so WS2 and WS3 are pr or cr. The TCS resulting from the composition of WS1 with CWS2 is a. Continuing in the same way, using the automaton we can deduce that the elementary Web services assigned to activities A5 to A8 could only be pr or cr. After selecting service assigned to activities A1 to A8, the ultimate TCS is formed and its transactional property is a. The TCS can ensure the overall consistency and successful termination Figure 3.

4. QoS-Based Web Service Selection

4.1. QoS-Based Web Service Model

In our approach, after transaction-based service selection many equivalent web services of transactional property are available to perform the same activity, their QoS properties such as price, reputation, and reliability become important in the next selection process. In order to

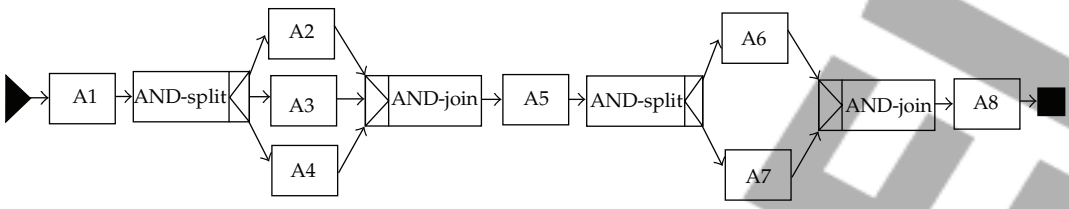


Figure 4: Example workgroup.

reason about QoS properties, a model is needed to take into account the fact that QoS involves multiple dimensions.

Assume a set WS of service classes. Each service class $S_{Ai} = \{S_{A_{i1}}, S_{A_{i2}}, \ldots, S_{A_{in}}\}$ can be assigned to activity A_i with suitable transactional properties in the workgroup after automation selection, but potentially differ in terms of nonfunctional properties. Some service providers might provide the same service in different quality levels, for example, at different response times and different prices. For the sake of simplicity, we model each variation of the service as a different service. In this paper, we assume that service brokers maintain and update information about existing service classes and candidate services of each class in their registries, making them accessible to service requesters.

We consider quantitative nonfunctional properties of web services, which can be used to describe the quality criteria of a web service. These can include generic QoS attributes like response time, availability, price, reputation, and so forth, as well as domain-specific QoS attributes, for example, bandwidth for multimedia web services, as long as these attributes can be quantified and represented by real numbers. We use the vector $Q_S = \{q_1(S), \ldots, q_r(S)\}$ to represent the QoS values of service ws, which are published by the service provider. The function $q_i(S)$ determines the published value of the ith attribute of the service ws.

QoS attributes may be positive or negative. The values of positive attributes need to be maximized (e.g., throughput and availability), whereas the values of negative attributes need to be minimized (e.g., price and response time). For simplicity, in this paper we consider only negative attributes because positive attributes can be easily transformed into negative by multiplying their values by −1.

4.2. QoS-Based Composite Web Service

The QoS attributes of CWS are decided by the QoS attributes of individual services and their composition relationships (which are workgroup patterns in the paper). There are different workgroup patterns that individual services can be composed to form a CWS. Having said that, the three workgroup patterns are; (a) Sequential pattern; (b) AND-split and AND-join patterns; (c) XOR-split and XOR-join patterns. In this paper, we only consider the Sequential pattern, which is the fundamental one. All the other models can be converted into sequential model. We can find how to do the conversions in many published research for example.

The QoS vector for a CWS $= \{S_{A_{1i}}, \ldots, S_{A_{nm}}\}$ is defined as $Q_{\text{CWS}} = \{q_1(\text{CWS}), \ldots, q_r(\text{CWS})\}$, where $q_i(\text{CWS})$ is the estimated end-to-end value of the ith QoS attribute and can be computed by aggregating the corresponding values of the component services. In our model, we consider three types of QoS aggregation functions: (1) summation, (2) multiplication, and (3) minimum relation.

(1) Summation Aggregation Function

Price and response time aggregation function is as follows:

$$q'(\text{CWS}) = \sum_{i=1}^{n} q(S_{Ail}), \tag{4.1}$$

where i is number of component services for a CWS that is composed of component services from each service class WS_i.

Reputation aggregation function is as follows:

$$q'(\text{CWS}) = \frac{1}{n} \sum_{i=1}^{n} q(S_{Ail}). \tag{4.2}$$

(2) Multiplication Aggregation Function

Availability and reliability aggregation function is as follows:

$$q'(\text{CWS}) = \prod_{i=1}^{n} q(S_{A_{il}}). \tag{4.3}$$

(3) Minimum Relation Aggregation Function

Throughput aggregation function is as follows:

$$q'(\text{CWS}) = \min_{i=1}^{n} q(S_{A_{il}}). \tag{4.4}$$

4.3. Utility Function Of CWS

In our approach, the QoS service selection is embedded within the transactional service selection. The set of potential Web services for each activity is restricted by the transactional requirement. Indeed, the selection of a Web service for an activity depends on the transactional property of Web services already assigned to the previous activities of the workflow. So each service class $S_{A_i} = \{S_{A_{i1}}, S_{A_{i2}}, \ldots, S_{A_{in}}\}$ is assigned to a same activity after transactional service selection.

In order to evaluate the multidimensional quality of a composite web service a utility function is used. The function maps the quality vector QoS into a single real value. The utility computation involves scaling the QoS attributes' values to allow a uniform measurement of the multidimensional service qualities independent of their units and ranges. The scaling process is then followed by a weighting process for representing user priorities and preferences. In the scaling process, each QoS attribute value is transformed into a value between 0 and 1, by comparing it with the minimum and maximum possible value according to the available QoS information about alternative services.

Utility function of CWS $F(\mathrm{CWS})$ is computed as follows:

$$F(\mathrm{CWS}) = \sum_{k=1}^{r} w_k \cdot \frac{\max q_k(\mathrm{CWS}) - q_k(\mathrm{CWS})}{\max q_k(\mathrm{CWS}) - \min q_k(\mathrm{CWS})}, \tag{4.5}$$

where r is dimension of quality vector and w_k is the weight of q_k to represent priorities of QoS attributes. $\max q_k(\mathrm{CWS})$ or $\min q_k(\mathrm{CWS})$ is the maximum and minimum aggregated values of the kth QoS attribute for a given composite service, and they are computed as follows:

$$\begin{aligned} \max q_k(\mathrm{CWS}) &= \mathrm{AF}\big(\max q_k(S_{A_1}), \max q_k(S_{A_2}), \ldots, \max q_k(S_{A_n})\big), \\ \min q_k(\mathrm{CWS}) &= \mathrm{AF}\big(\min q_k(S_{A_1}), \min q_k(S_{A_2}), \ldots, \min q_k(S_{A_n})\big), \end{aligned} \tag{4.6}$$

where AF is aggregation function of the kth QoS attribute, and $\max q_k(S_{A_i})$ or $\min q_k(S_{A_i})$ is the maximum or minimum value of kth QoS attribute of candidate services S_{A_i}assigned to activity A_i. They are computed as follows:

$$\begin{aligned} \max q_k(S_{A_i}) &= \max\big(q_k(S_{A_{i1}}), q_k(S_{A_{i2}}), \ldots, q_k(S_{A_{im}})\big), \\ \min q_k(S_{A_i}) &= \min\big(q_k(S_{A_{i1}}), q_k(S_{A_{i2}}), \ldots, q_k(S_{A_{im}})\big). \end{aligned} \tag{4.7}$$

We consider the QoS-based optimal service selection that maximizes the overall utility value $F(\mathrm{CWS})$.

5. Experimentation

In order to evaluate the behavior of our service selection approach, we write program whose input is a workflow composed of n activities and the output is a TCS corresponding to a list of elementary Web services or composite Web services assigned to each activity of the input workflow. Experiments were conducted by implementing the proposed service selection approach with the program on a PC Core i3 with 2 GB RAM, Windows 7, and Java 2 Enterprise Edition V1.5.0. The experiments involved composite services varying the number of activities and varying the number of Web services.

In the experiment we design the workgroup shown in Figure 5.

Different services can be generated randomly to implement the activities of workflow shown in Figure 5, so in the experiment each activity uniformly generates 15 Web services. Also the services are transactional whose transactional properties are in the set of {p, pr, c, cr, a, ar}. For each activity, we randomly generate from 1 to 10 services for each of the transactional properties. For each service, we randomly generate transactional property and a QoS vector, but there is the relation between the two. Particularly we assume that the execution price of service with c transactional property is more expensive than a p or a one, because the former provides additional functionality in order to guarantee that the result can be undone. Similarly, we believe that a pr, ar, or cr web service has execution duration higher than a nonretriable one, because the former provides additional operation in order to guarantee that it successfully finishes after a finite number of invocations. Table 1 shows the different set of values considered for each QoS criterion depending on transactional service property for experiment scenario.

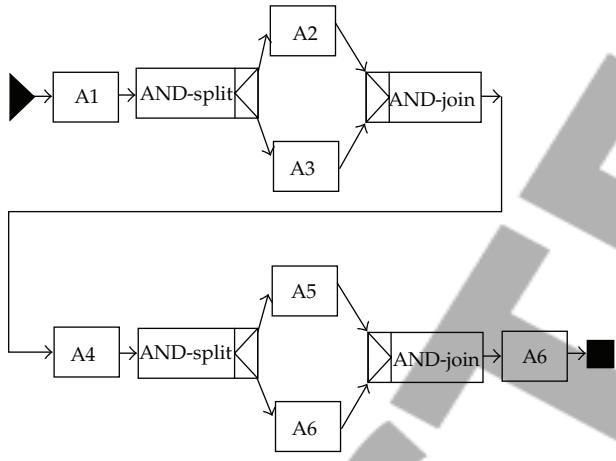


Figure 5: Experiment workgroup.

Table 1: Set of values for each QoS criterion.

Qos vector	Transactional property					
	p	pr	c	cr	a	ar
Execution price	0–60	0–60	60–100	60–100	0–60	0–60
Execution duration	10–60	60–100	10–60	60–100	10–60	60–100
Reputation	1–6	1–6	1–6	1–6	1–6	1–6
Reputation	0.00–0.10	0.00–0.10	0.00–0.10	0.00–0.10	0.00–0.10	0.00–0.10
Availability	0.00–0.10	0.00–0.10	0.00–0.10	0.00–0.10	0.00–0.10	0.00–0.10

We apply our proposed selection mechanism considering both levels of risk tolerance. For the experiment, we consider weights assigned by the user in such a way that price and duration constraints have always 60 percent of the total weight. With this condition, we execute the selection process for the weight distribution shown in Table 2. This experiment was executed 10 times.

In the experiment we observe relationship of utility value and price weight with different risk tolerance, which is shown in Figure 6. As depicted in Figure 6, the more important the price criteria to the user (which means having high price weight), the better a composition with risk tolerance 1 compared to a composition with risk tolerance 0. The appearance of Figure 6 shown coincides with the real application. When transactional constraint of user is risk tolerance 0, the TCS should ensure to be compensated and undone. That is more expensive than TCS of risk tolerance 1. As price weight is bigger utility value will be smaller. Therefore, if the execution price criterion is important to the user (i.e., price minimum), the better solutions are the ones with the lowest level of risk.

Figure 7 shows relationship of utility value and duration weight with different risk tolerance. As depicted in Figure 7, the more important the duration criteria to the user, the better a composition with risk tolerance 0 compared to a composition with risk tolerance 1.

Table 2: Weight distribution.

Qos property	Weight plan						
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Execution price	0	10	20	30	40	50	60
Execution duration	60	50	40	30	20	10	0
Reputation	10	10	10	10	10	10	10
Reputation	15	15	15	15	15	15	15
Availability	15	15	15	15	15	15	15

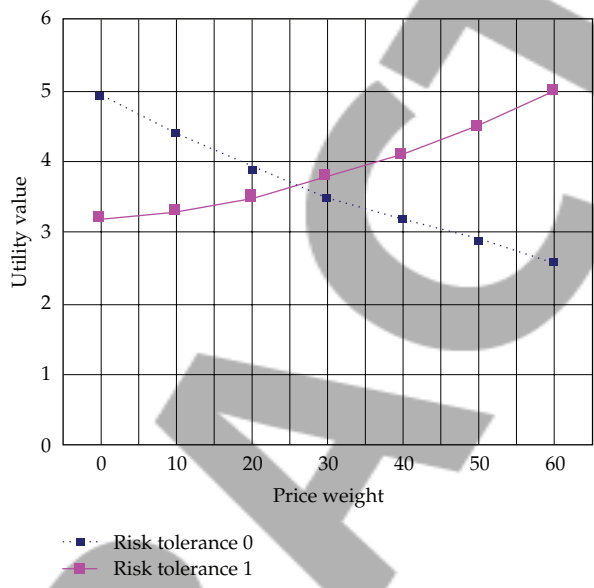


Figure 6: Experimental results for risk tolerance 0 and risk tolerance 1 by varying price weights.

6. Conclusion

In this paper, we have presented an ensuring transactional reliability and QoS service selection approach. The selection of the component Web services is done by matching the Web services properties with the user's desires. More precisely, the selection is realized depending on transactional and QoS user requirements. The former is established by means of a risk tolerance notion that indicates if the results can be compensated or not. The latter is expressed as a weight over each QoS criterion. We build automaton model to implement transactional-aware service selection, and with the model composition Web service can guarantee transactional execution. Moreover, our method is scalable because the user has only to define a global transaction requirement and does not have to define the possible termination states of all component Web service. Nonfunctional QoS aspects (e.g., response time, availability, etc.) are also crucial for selecting the web services to take part in the composition. In the paper, we consider quantitative nonfunctional properties that can include generic QoS attributes like response time, availability, price, reputation, and so forth, as well as domain-specific QoS attributes, for example, bandwidth for multimedia web services. We define aggregation functions, and use a Multiple Attribute Decision-Making approach for the

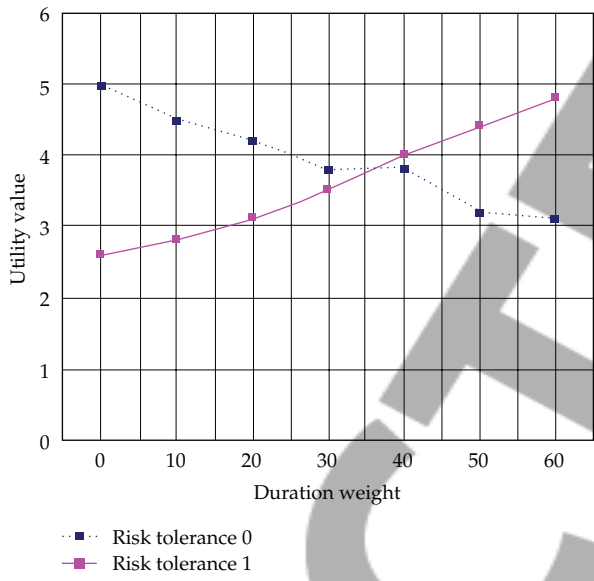


Figure 7: Experimental results for risk tolerance 0 and risk tolerance 1 by varying duration weights.

utility function, and in particular the Simple Additive Weighting (SAW) technique. The utility computation involves scaling the QoS attributes' values to allow a uniform measurement of the multidimensional service qualities independent of their units and ranges.

In the experimentation, in order to give a semantic meaning to the risk notion, we have considered two scenarios where the execution duration and execution price of a WS depend on additional operations required to guarantee their transactional properties. We used the risk tolerance notion for these scenarios. Under these conditions, the implementation shows that the QoS of TCS is in conformity with the user preferences. If the execution price criterion is important to the user, the better solutions are the ones with the lowest level of risk. If the execution duration criterion is more important to the user, then the riskier solutions are the best ones. The results also show that risk 0 is equivalent to risk 1 if compensatable services do not cost more than the others.

Acknowledgment

This work is partially supported by Shaanxi Education Department Foundation of China, (no. 12JK0745 and no. 12JK0746).

References

[1] Yet Another Workflow Language, 2009, http://www.yawl-system.com.

[2] Business Process Execution Language for Web Services, 2010, http://www.ibm.com/developerworks/library/ws-bpel.

[3] G. Alonso, D. Agrawal, and A. E. Abbadi, "Process synchronisation in workflow management systems," in *Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing (SPDS '97)*, New Orleans, La, USA, October 1996.

[4] P. Albert, L. Henocque, and M. Kleiner, "Configuration based workflow composition," in *Proceedings of IEEE International Conference on Web Services (ICWS '05)*, pp. 285–292, July 2005.

[5] N. Gioldasis and S. Christodoulakis, "Utml: Unified transaction modeling language," in *Proceedings of the 3rd International Conference on Web Information Systems Engineering*, pp. 115–126, IEEE Computer Society, 2002.

[6] D. Langworthy et al., Web services atomic transaction (ws-atomictransaction).

[7] B. Medjahed, B. Benatallah, A. Bouguettaya, A. H. H. Ngu, and A. K. Elmagarmid, "Business-to-business interactions: issues and enabling technologies," *The VLDB Journal*, vol. 12, no. 1, pp. 59–85, 2003.

[8] N. Kavantzas, D. Burdett, G. Ritzinger, and Y. Lafon, Web services choreography description language version 1. 0., October 2004, http://www.w3.org/TR/ws-cdl-10,.

[9] D. Langworthy et al., Web services business activity framework (ws-businessactivity).

[10] I. Bea and Microsoft, Business process execution language for web services (bpel4ws), 2003.

[11] S. Bhiri, O. Perrin, and C. Godart, "Ensuring required failure atomicity of composite web services," in *Proceedings of the 14th International Conference on World Wide Web (WWW '05)*, pp. 138–147, 2005.

[12] A. Elmagarmid, *Transaction Models For Advanced Database Applications*, Morgan-Kaufmann, 1992.

[13] D. Bunting, M. Chapman, O. Hurley et al., "Web services transaction management (ws-txm) version 1. 0," in *Arjuna, Fujitsu, IONA, Oracle, and Sun*, 2003.

[14] E. Gamma, R. Helm, R. Johnson, and J. Vlisside, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, Mass, USA, 1995.

[15] K. P. Yoon and C. L. Hwang, *Multiple Attribute Decision Making: An Introduction*, Quantitative Applications in the Social Sciences, Sage, 1995.

[16] S. Mehrotra, R. Rastogi, H. Korth, and A. Silberschatz, "A transaction model for multidatabase systems," in *Proceedings of the 12th International Conference on Distributed Computing Systems (ICDCS '92)*, pp. 56–63, June 1992.

[17] B. Medjahed, B. Benatallah, A. Bouguettaya, A. H. H. Ngu, and A. K. Elmagarmid, "Business-to-business interactions: issues and enabling technologies," *The VLDB Journal*, vol. 12, no. 1, pp. 59–85, 2003.

[18] P. Albert, L. Henocque, and M. Kleiner, "Configuration based workflow composition," in *Proceedings of IEEE International Conference on Web Services (ICWS '05)*, pp. 285–292, July 2005.