TQoS: Transactional QoS-driven Selection for Web Service Composition

Joyce El Haddad[1], Maude Manouvrier[1,3], Marta Rukoz[1,2,3]

[1] LAMSADE CNRS UMR 7024, Université Paris Dauphine, Place de Lattre de Tassigny, Paris, France

[2] Université Paris X-Nanterre, 200 Avenue de la République, 92001 Nanterre, France, On leave from Universidad Central Venezuela

[3] WISDOM, Federation of the three database research teams, LIP6 (Université Paris 6), LAMSADE and CEDRIC (CNAM)

Abstract

Composite Web services are often long-running, loosely coupled and cross-organizational applications. For such applications, advanced support is required to ensure quality reliable execution. This paper addresses the issue of selecting and composing Web services not only according to their functional requirements but also to their behavioral properties (e.g. transactions) and QoS characteristics. In our approach, Web services are selected in a way satisfying user preferences. These preferences are expressed as weights over QoS criteria and as risk level defining semantically the transactional requirements.

keywords: Web service composition; Transactional Web service; Quality of Service.

1. Introduction

Web services have been emerging as a promising technology for business integration [4]. The creation of value-added services by composing existing ones is gaining a significant momentum [3]. As composite Web services are often long-running and loosely coupled, resulting from aggregation of Web services offered by different organizations, different properties supports are required to ensure the overall consistency of data modified by its component service.

There are three different types of properties that must be considered when talking about services: (1) functional (i.e. capabilities), (2) behavioral (e.g. transactions) and (3) non-functional (i.e. QoS criteria). In this paper we discuss the last two types of properties. As explained in [12], a realistic Web service must meet both behavioral and non-functional requirements of its users. Therefore it is important that a composite Web service is augmented so that its characteristics can be determined and users are bound to services that best meet their behavioral as well as non-functional requirements.

More precisely, we investigate about constructing and selecting transactional composite Web service. The obtained composition maximizes the user satisfaction expressed in terms of weights over QoS criterion and satisfies the transactional requirements set by the user and by the structure of the composite service.

The paper is organized as follows. Section 2 presents the system architecture and Web services by the way of their behavioral and non-functional properties. Section 3, introduce our Transactional QoS (TQoS) Web service selection algorithm. Experimental results are shown in Section 4. In Section 5, we discuss some related work. Section 6 concludes and gives perspectives.

2. Preliminaries

In this section, we present our system architecture and some basic concepts as Web services and how to express their behavioral and non-functional properties.

2.1 System Architecture

The architecture of our system is presented in Figure 1. There are three distinct components namely, a *workflow*, a *Web services registry* and a *composition manager*.

1. *Workflow*: workflows are rule based management softwares that direct, coordinate and monitor execution of activities representing business processes. Several

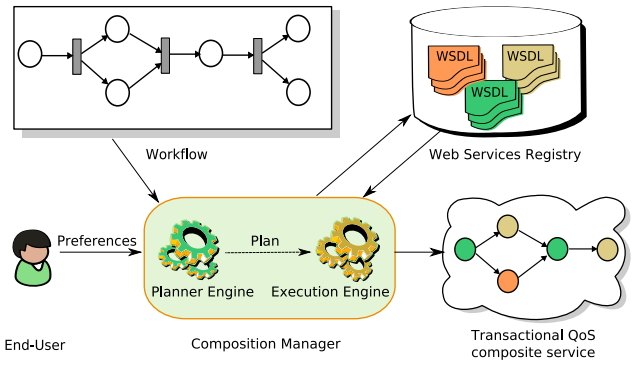

Figure 1. System architecture

workflow patterns [10] have been proposed to provide an uniform approach to describe workflow characteristics. In [2], several relevant patterns for the Web service composition have been identified. In our approach, we use the sequence, the parallel and sequential split (AND and XOR) and the parallel and sequential join (AND and XOR) patterns. Using these patterns, we define a workflow skeleton to represent the structure of an application in terms of activities and temporal dependencies between them. In our system, a workflow WF defines the execution order of a set of n activities performed by Web services: $WF = (a_i)_{i\in[1,n]}$ – for the sake of simplicity, we consider that one service executes only one activity. Thus, a composite Web service can be represented by a workflow.

2. *Web services registry*: The ability to register, discover, and manage services is an essential requirement for any Service Oriented Architecture (SOA) implementation. A service registry provides the means for registering and discovering Web services, and managing associated metadata and artifacts securely and reliably. A Web service description contains metadata that describes the service functional properties (i.e., capabilities), behavioral properties (e.g., transactions) and non-functional properties (e.g., QoS criterion).

3. *Composition manager*: The service composition manager is made up of a planner engine and an execution engine. When an instance of a composite service is initiated, the planner engine contacts the Web services registry to search for candidate component services, and, based on the candidate services retrieved, it generates an execution plan, i.e., an assignment of component services to the activities in the schema of the composite service. Based on the execution plan, the adaptive execution engine then orchestrates the component services to execute the instance of the composite service.

2.2 Web service description

Web services (WS) are autonomous software systems identified by URIs which can be accessed through messages encoded according to XML-based standards (e.g., SOAP, WSDL, and UDDI). Since Web services are intended to be discovered and used by other applications, they need to be described and understood in terms of functional capabilities as well as behavioral properties and non-functional properties. As said above, in this article, we only discuss the last two types of properties.

- **Behavioral properties.** Service discovery and composition techniques require both functional and behavioral aspects of a Web service to be accurately specified in its description. Otherwise, the located Web service may not actually provide the required functionality. The behavioral description is about how the functionality of a Web service can be achieved in terms of interaction with the other Web services. In a composition where several component Web services interact, unexpected behavior from a component Web service may, not only lead to its failure, but also may bring negative impact on all the participants to the composition. As for all cross-organizational collaborative systems, the execution of composite Web services requires Transactional Properties (TP) so that the overall consistency is ensured. Within behavioral properties, we distinguish between transactional and non-transactional behavior. A transactional Web service is a Web service that emphasizes transactional behavior for its characterization and correct usage. However, Web services may provide dissimilar transactional behavior. With the main transactional properties [6], *pivot* (p) and *compensatable* (c), we have the following definitions:

  Definition 1 ***Pivot.*** *A service is said to be pivot if its behavior supports atomic transactions. In other word, a service is pivot if once it successfully completes, its effects remains forever and cannot be semantically undone. On one hand, there is no guarantee that this type of service can be executed successfully and if it fails it has no effect at all. On the other hand, a completed pivot service cannot be rolled back.*

  Definition 2 ***Compensatable.*** *A service is said to be compensatable if its behavior supports compensatable transactions. In other words, a service is compensatable if it is able to offer compensation policies to semantically undo the original activity.*

  Next, among behavioral properties of a Web service other than transaction, let us consider the behavioral property called *retriable* (r). We have the following definition:

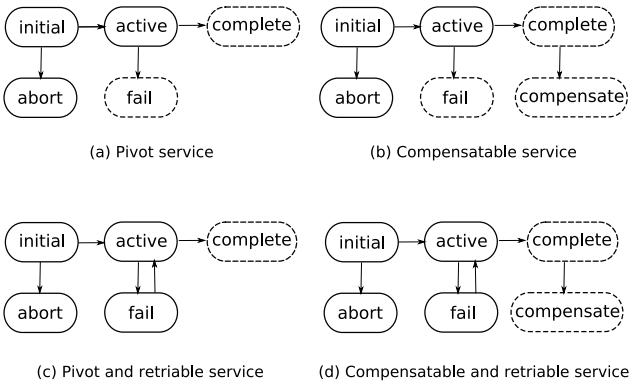

Figure 2. Service state diagrams according to their transactional properties

Definition 3 ***Retriable.*** *A service is retriable if it guarantees a successfully termination after a finite number of invocations. In other words, a service with this property is able to offer forward recovery.*

Naturally, a service can combine behavioral properties and the set of all possible combinations is $\{p, c, pr, cr\}$.

In order to model the internal behavior of a service, we adopt the states/transitions model. A service has a minimal set of states (initial, active, abort, fail and complete). Figure 2.(a) shows the internal state diagram of a pivot service. When a service is instantiated, the state of the instance is initial. Then this instance can be either aborted or activated. Once it is active, the instance can normally continues its execution. In this case, it can achieve its objective and successfully completes or it can fail. The requested transactional properties can be expressed by extending the service states and transitions. For instance, for a compensatable service, a new state compensate is introduced – see in Figure 2.(b). Figures 2.(c) and 2.(d) illustrate the states diagram of services combining transactional and retriable behavior properties.

- **Non-functional properties.** When several functionally and transactionally equivalent Web services are available to perform the same activity, their QoS properties such as price, availability, reliability and reputation become important in the selection process. In order to reason about QoS properties in Web services, a model is needed which captures the descriptions of these properties from a user perspective. Such model must take into account the fact that QoS involves multiple dimensions. In this paper, we consider the following five generic quality criteria for a Web service:

  1. Execution price ($q_{ep}(s)$): which is the fee that a requester has to pay for invoking of the Web service s.
  2. Execution duration ($q_{ed}(s)$): that measures the expected delay time between the moment when a Web service s is invoked and when the results are received.
  3. Reputation ($q_r(s)$): which is a measure of trustworthiness of service s, generally this measure is defined as the average ranking given to the service by end users.
  4. Successful execution rate ($q_{sr}(s)$): that is the probability that service s responds correctly to the user request.
  5. Availability ($q_a(s)$): which is the probability that service s is accessible.

The following section presents how to compose Web services having the properties defined above.

2.3 Composite Web service specification

A Composite Web Service (CWS) is a conglomeration of existing Web services interacting together to offer a new value-added service. It coordinates a set of Web services as a cohesive unit of work to achieve common goals. Currently, lots of process modeling languages including BPEL have been proposed to capture the logic of a composite Web service, and some of them are still evolving.

Rather than choosing a particular modeling language, we adopt workflows model to describe the composition. In a Web services environment, a workflow represents a composite Web service, and an activity of a workflow is implemented by a component Web service. These terms may be used interchangeably in the following discussion. Next, we describe composite Web services in terms of transactional (i.e., behavioral) and quality (i.e., non-functional) models.

2.3.1 Composite transactional model

It is time to analyze the behavioral property of a composite Web service with the presence of component Web services within behavioral properties in $\{p, c, pr, cr\}$. We are interested in properly assigning component Web services in order to have a Transactional Composite Web Service (TCWS). A TCWS is a composite Web service that emphasizes transactional behavioral properties for composition and synchronization of component Web services. It takes advantage of component service behavioral properties to specify mechanisms for failure handling and recovery.

A TCWS defines component services orchestration by specifying dependencies between them. Theses dependencies are defined by the workflow patterns that specify how

services are coupled and how the behavior of certain service(s) influence the behavior of other service(s). The behavioral properties of a TCWS highly depends on the properties of individual services and on the structure of the workflow. The composition manager exploits a combination of Web services by treating the execution of them as an unit of work.

In order to give a precise definition of a composite Web service with transactional behavioral property, we need to define retriable, atomic and compensatable workflows:

Definition 4 ***Retriable workflow.*** *An workflow is retriable when all its activities (component Web services) are retriable. In the following, r is used to represent a retriable workflow.*

Definition 5 ***Atomic workflow.*** *A workflow is atomic if it can be treated as an unit of work. In other words, if all the activities (component Web services) of the workflow completes successfully then their effect remain forever and cannot be semantically undone. On the other hand, if one activity does not complete successfully then all previously successful activities have to be compensated. In the following, $\vec{a}$ is used to indicate that the transactional property of a workflow is atomic while $\tilde{a}$ is non-atomic.*

Definition 6 ***Compensatable workflow***. *A workflow is compensatable if all its activities can be compensated. In the following, c is used to represent a compensatable workflow.*

Now, we can specify a transactional composite Web service as:

Definition 7 ***Transactional Composite Web Service.*** *A TCWS is a worklfow that can be atomic or that can be compensated. In other words, the transactional behavioral property of a TCWS is in $\{\vec{a}, \vec{a}r, c, cr\}$.*

Inspired by [4], we focus on the description of a workflow as an execution from the start point to the end point. Thus, we derive below the different behavioral properties for a TCWS with the sequential and concurrent patterns, as shown in Table 1.

- **Sequential execution:** $(t_1; t_2)$ where $t_1, t_2 \in \{p, c, pr, cr\}$ represents a sequential invocation of two activities a_1 followed by a_2 with respectively transactional properties t_1 and t_2. The derived transactional property of this sequential execution is shown in Table 1. There are 12 sequential invocations leading to a transactional property : $(p; pr)$, $(p; cr)$, $(c; p)$, $(c; c)$, $(c; pr)$, $(c; cr)$, $(pr; pr)$, $(pr; cr)$, $(cr; p)$, $(cr; c)$, $(cr; pr)$ and $(cr; cr)$ – see lines 3 to 8, 11 to 16 of Table 1.

  For example, $(p; pr)$ is atomic because if both activities complete successfully then the result cannot be semantically undone and if the first activity (p) does not complete successfully, then the second one is not executed. $(c; p)$ is also atomic: if the first activity and the second one complete successfully, then the result cannot be semantically undone and if the last activity (p) does not completes successfully, the first one can be compensated. On the other hand, $(p; c)$ is not atomic: if the last activity (c) does not complete successfully, the first one (p) cannot be undone. $(pr; pr)$, $(pr; cr)$ and $(cr; pr)$ are atomic retriable because all the component web services are retriable. $(c; c)$, $(c; cr)$ and $(cr; c)$ are compensatable because all the component web services are compensatable, and $(cr; cr)$ is compensatable retriable because all the component web services are compensatable retriable.

- **Concurrent execution:** $(t_1//t_2)$ where $t_1, t_2 \in \{p, c, pr, cr\}$ represents the invocation of two activities a_1 and a_2 simultaneously with respectively transactional properties t_1 and t_2. The derived transactional property of this concurrent execution is shown in Table 1. A concurrent execution is defined such that the involved activities can be executed independently. There are 9 concurrent invocations leading to a transactional property: $(p//cr)$, $(c//c)$, $(c//cr)$, $(pr//pr)$, $(pr//cr)$, $(cr//p)$, $(cr//c)$, $(cr//pr)$ and $(cr//cr)$ – see 4, 6, 8, 11 to 16 of Table 1.

  Indeed, $(p//cr)$ and symmetrically $(cr//p)$ are atomic because, if both activities complete successfully, then the result cannot be semantically undone and if the activity p does not complete successfully, then the other activity (cr) can be compensated. $(pr//pr)$, $(pr//cr)$, $(cr//pr)$ and $(cr//cr)$ are atomic retriable because all the component web services are retriable. $(c//c)$, $(c//cr)$, $(cr//c)$ are compensatable because all the component web services are compensatable, and $(cr//cr)$ is compensatable retriable because all the component web services are compensatable retriable.

Above we have presented the composition between two component WS. Next, we present the composition between a TCWS and a component WS before presenting the composition of two TCWS.

Table 2 presents the transactional behavioral property of sequential and concurrent execution of an atomic TCWS (retriable or not) with a component Web service. For compensatable TCWS (property in $\{c, cr\}$) the composition result is the same as lines 5 to 8 and 13 to 16 of Table 1.

For example, the parallel or sequential composition between an atomic TCWS $(\vec{a})$ and a compensatable retriable Web service (cr) is atomic because if both components complete successfully then the result cannot be semantically undone. Moreover, for $(\vec{a}//cr)$, if the atomic TCWS $(\vec{a})$ does not complete successfully, then the compensatable retriable component (cr) can be compensated. This is not the case with $(\vec{a}//c)$: if the second component (c) does not

	t_1	t_2	$t_1;t_2$	$t_1//t_2$
(1)	p	p	$\tilde{a}$	$\tilde{a}$
(2)	p	c	$\tilde{a}$	$\tilde{a}$
(3)	p	pr	$\vec{a}$	$\tilde{a}$
(4)	p	cr	$\vec{a}$	$\vec{a}$
(5)	c	p	$\vec{a}$	$\tilde{a}$
(6)	c	c	c	c
(7)	c	pr	$\vec{a}$	$\tilde{a}$
(8)	c	cr	c	c
(9)	pr	p	$\tilde{a}$	$\tilde{a}$
(10)	pr	c	$\tilde{a}$	$\tilde{a}$
(11)	pr	pr	$\vec{a}r$	$\vec{a}r$
(12)	pr	cr	$\vec{a}r$	$\vec{a}r$
(13)	cr	p	$\vec{a}$	$\vec{a}$
(14)	cr	c	c	c
(15)	cr	pr	$\vec{a}r$	$\tilde{a}r$
(16)	cr	cr	cr	cr

Table 1. Transaction property of sequential and concurrent execution of component WS

complete successfully, the first one ($\vec{a}$) cannot be compensated.

Table 3 presents the transactional behavioral property of sequential and concurrent execution of two TCWS (retriable or not).

2.3.2 Composite quality model

In the composite Web service specification previously presented, Web services will be grouped together according to their functionality and behavioral properties. To differentiate the members of a set, quality properties must be considered. In this section we analyze the QoS of a composite Web service.

	WF	t	$WF;t$	$WF//t$
(1)	$\vec{a}$	p	$\tilde{a}$	$\tilde{a}$
(2)	$\vec{a}$	pr	$\vec{a}$	$\tilde{a}$
(3)	$\vec{a}$	c	$\tilde{a}$	$\tilde{a}$
(4)	$\vec{a}$	cr	$\vec{a}$	$\vec{a}$
(5)	$\vec{a}r$	p	$\tilde{a}$	$\tilde{a}$
(6)	$\vec{a}r$	pr	$\vec{a}r$	$\vec{a}r$
(7)	$\vec{a}r$	c	$\tilde{a}$	$\tilde{a}$
(8)	$\vec{a}r$	cr	$\vec{a}r$	$\vec{a}r$

Table 2. Transaction property of sequential and concurrent execution of an atomic or atomic retriable TCWS with a component Web service

	WF_1	WF_2	$WF_1;WF_2$	$WF_1//WF_2$
(1)	$\vec{a}$	$\vec{a}$	$\tilde{a}$	$\tilde{a}$
(2)	$\vec{a}$	$\vec{a}r$	$\vec{a}$	$\tilde{a}$
(3)	$\vec{a}r$	$\vec{a}$	$\tilde{a}$	$\tilde{a}$
(4)	$\vec{a}r$	$\vec{a}r$	$\vec{a}r$	$\vec{a}r$
(5)	c	$\vec{a}$	$\vec{a}$	$\tilde{a}$
(6)	c	$\vec{a}r$	$\vec{a}$	$\tilde{a}$
(7)	cr	$\vec{a}$	$\vec{a}$	$\tilde{a}$
(8)	cr	$\vec{a}r$	$\vec{a}r$	$\vec{a}r$

Table 3. Transaction property of sequential and concurrent execution of two TCWS

Criteria	**Aggregation function**
Price	$q_{ep}(CWS) = \sum_{i=1}^{n} q_{ep}(s_i)$
Duration	$q_{ed}(CWS) = \sum_{i=1}^{n} q_{ed}(s_i)$
Reputation	$q_r(CWS) = \frac{1}{n} \sum_{i=1}^{n} q_r(s_i)$
Success rate	$q_{sr}(CWS) = \prod_{i=1}^{n} q_{sr}(s_i)$
Availability	$q_a(CWS) = \prod_{i=1}^{n} q_a(s_i)$

Table 4. Aggregation functions

A composite Web service has the same quality properties as a Web service, i.e. execution price, execution duration, reputation and successful execution rate. When a user wants to execute a composite Web service, it indicates, between other things, the quality of the wished result. This one is expressed as weight in each of the quality criterion. In this paper we use a local optimization selection algorithm as follows: according to the transactional requirements for each activity, a set of transactional Web services is selected (see Section 2.3.1), then a QoS-driven service selection, as defined in [13], is executed. For the Web service selection in each activity, the system uses the classical Multiple Criteria Decision Making (MCDM) approach. This selection is based on the weight assigned by the user to each quality criterion. A simple additive weighting technique is used to assign a quality score to each Web service as follows: $Score(s_i) = \sum w_j q_{ij}$, where $w_j \in [0, 1]$ is the weight assigned by the user to the quality criterion j such that $\sum w_j = 1$ and q_{ij} is the value of criterion j for the service s_i. The Web service with maximal score is selected to execute the activity a_i. If there are several services with maximal score, one of them is selected randomly.

In order to evaluate the QoS of a composite Web service, it is necessary to look into workflow composition constructs such as AND-split and XOR-split. A split construct produces several execution paths. Each one of them is composed of one or more activities. Similarly a join construct

transforms several execution paths in one path. The QoS evaluation of a CWS is realized by using the aggregation functions defined in Table 4 in such a way that activities in all execution paths between AND-split and AND-join constructs are considered and activities in only one execution path between XOR-split and XOR-join constructs is considered. In [3] the most frequently used path is considered in the aggregation functions for the XOR-split. In this work we select the path that has the Web service with the maximal score in its first activity, according to the previous selection algorithm. If there are several paths with the maximal score in its first activity one of them is selected randomly.

3 TQoS-driven WS selection

3.1 Definition of risk

We assume that the execution price of a $compensatable$ service is more expensive than a $pivot$ service. Indeed, the former provides additional operation in order to guarantee that the result can be undone. Similarly, we believe that a $retriable$ service has an execution duration higher than a $non\text{-}retriable$ service. Indeed, the former provides additional operation in order to guarantee that it successfully finishes after a finite number of invocations.

Under theses hypotheses and in order to explain the services selection process, it is necessary to establish how the user can express their transactional criteria. Although, expressing its transactional criteria is significant for the user, the risk or possibility that a transaction will be unsuccessful have a more significant effect on its decision. The importance of the uncertainty of transaction completion and recovery is semantically expressed under a factor (criteria) called the *risk*. Under this notion, the set $\{\vec{a}, \vec{a}r, c, cr\}$ of the behavioral properties of composite Web service can be divided into two subsets $\{\vec{a}, \vec{a}r\}$ and $\{c, cr\}$ each one of them representing a level of risk. For instance, in terms of the transactional properties, we believe that properties $\vec{a}$ and $\vec{a}r$ are riskier than c and cr but cost less than properties c and cr. Indeed, properties $\vec{a}$ and $\vec{a}r$ mean that once a service has been executed it can not be rolled back. By the other way, to have a composite Web service with property c or cr we have to select only component Web services with properties c or cr (see Section 2.3.1). In both cases, as we mentioned above, this will have either a higher execution price or a higher execution duration. Therefore, we define two notions of risk of execution in a transactional system like:

- **Risk 0***: the system guarantees that if the execution is successful, the obtained results can be compensated by the user.*
- **Risk 1***: the system does not guarantee the successful execution but if it achieves the results cannot be compensated by the user.*

In the following section, we present the service selection algorithm used by the composition manager for service composition with risk and QoS preferences.

3.2 Service selection algorithm

In this paper, we use a local optimization selection algorithm. For each activity a_i, we select a Web service taking only into account the properties of the Web services able to execute a_i. Moreover, when a_{i-1} exists, we also take into account the transactional properties of the selected Web service for the previous activity of a_i.

In the local optimization approach, the service selection process is done for each activity individually. Although service selection is locally optimized, the global quality of the compose Web service may be suboptimal. In this article, we would like to enforce constraints over the composite Web service execution such as: "*the risk level of the composite service execution should be* 0" or "*the risk level of the composite service execution should be at most* 1".

To find the optimal assignment, we assign a service to each workflow activity based on an iterative process. Depending on the $risk$ level (i.e., transactional requirement) and the QoS of the services available for each activity, different scenarios can occur:

- $Risk = 0$: select for each activity the best QoS available service with transactional property in $\{c, cr\}$.
- $Risk = 1$: to select the optimal service, the algorithm must compute different combinations. That is the assignment must take place by considering the current activity, its predecessor and the workflow pattern.

According to Definition 7 and Table 1 presented above, the transactional property for a TCWS is in $\{\vec{a}, \vec{a}r, c, cr\}$. Figure 3 represents the automaton modeling the TQoS Web service selection algorithm. It contains five states. State I is the initial one. The final states $c, cr, \vec{a}$ and $\vec{a}r$ correspond to the transactional properties of a TCWS. When one of the final states is reached, a TCWS is obtained. The alphabet of the language accepted by the automaton is $\{\ 'p'\ ,\ 'c'\ ,\ 'cr'\ ,\ 'pr'\ ,\ ';p'\ ,\ '//p'\ ,\ ';pr'\ ,\ '//pr'\ ,\ ';c'\ ,\ '//c'\ ,\ ';cr'\ ,\ '//cr'\ ,\ ';\vec{a}'\ ,\ ';\vec{a}r'\ ,\ '//\vec{a}r'\ \}$ representing the transactional properties of components (activities or TCWS) executed in sequence (;) or in parallel ($//$). For example, using a transition $'p'$ from the initial state, the following state is $\vec{a}$. Indeed, any sequential execution beginning by a pivot service can be atomic, but will never be able to be compensated. The next state could be again $\vec{a}$ using a transition $'//cr','; cr','; pr'$ or $';\vec{a}r'$, meaning that an atomic TCWS

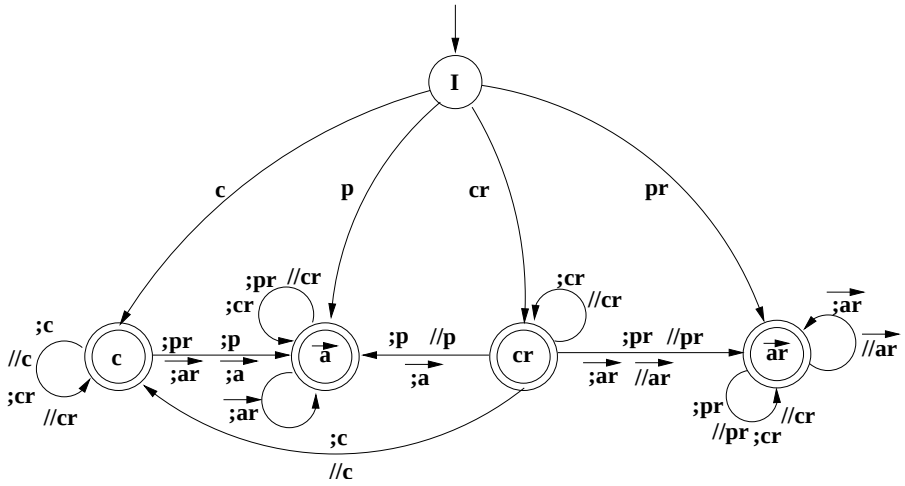

Figure 3. Automaton modeling the TQoS Web service selection algorithm

can be composed with a retriable component. When one of the final states $\vec{a}$ or $\vec{a}r$ is reached then we obtain a TCWS with risk=1. When one of the final states c or cr is reached then we obtain a TCWS with risk=0.

4. Experimentation

In order to evaluate the behavior of the proposed service selection approach, experiments were conducted by using the TQoS algorithm proposed. We use the workflow WF presented in Figure 4 as a composite Web service. A randomly generation of different services that can implement the activities of WF was realized as follows:

For each activity we uniformly generate 15 web services implementing it. For each generated service its transactional property was randomly pick up in the set $\{p, c, pr, cr\}$ and each one of its QoS criterion was randomly generated. Table 5 shows the different set of values considered for each QoS criterion depending on transactional properties.

For the generated environment we apply our selection algorithm for the two level of risk. In our experiments, the sum of price and duration weights represents 60% of the weights assigned by the user. With this condition we execute the selection process for the weight distribution showed in table 6. This experiment was realized 10 times. Figure 5a shows the obtained score results for both levels of risk and for different weights over the execution price criteria. Figure 5b shows the obtained score results for both levels of risk and for different weights over the execution duration criteria. As depicted is Figure 5, the more the price criteria is important to the user, the better is a composition with risk 1 compared to a composition with risk 0. By the other way, the more the duration criteria is important to the user, the better is a composition with risk 0 compared to a composition with risk 1.

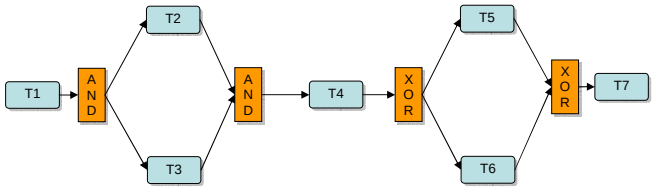

Figure 4. Input workflow WF

The following section presents the approaches related to our.

5. Related work

There are several contributions addressing the composition of Web services.

By one hand, a number of QoS-aware Web services selection mechanisms have been developed. These mechanisms focus on performance improvement in order to facilitate Web service composition in an open and dynamic environment [7, 2, 13]. Menasce [7] studies the QoS of component Web services in terms of cost and execution time. The author employs probability techniques to measure the cost and execution time of component Web services by considering different execution scenarios such as parallel, sequential, fastest-predecessor-triggered and synchronization. This study helps in selecting appropriate component Web services for Web service composition. Jaeger *et al.* [2] propose a mechanism for composite Web services with pattern-based QoS aggregation. The QoS aggregation is used to verify that a set of services satisfies the QoS requirement for the selected composite Web services. In the same year, Zeng *et al.* [13] propose two QoS-driven ser-

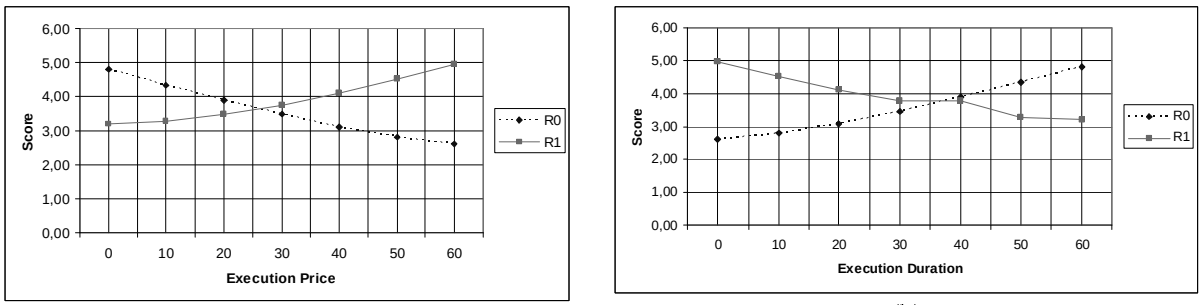

Figure 5. Experimental results for risk 0 and risk 1 varying the price weight and the duration weight

TP	$Price$	$Duration$	$Reputation$	$Availability$	$Successfullrate$
cr	$0.20 - 0.30$	$0.20 - 0.30$	$0 - 5$	$0.00 - 1.00$	$0.00 - 1.00$
c	$0.20 - 0.30$	$0.01 - 0.10$	$0 - 5$	$0.00 - 1.00$	$0.00 - 1.00$
pr	$0.00 - 0.10$	$0.20 - 0.30$	$0 - 5$	$0.00 - 1.00$	$0.00 - 1.00$
p	$0.00 - 0.10$	$0.01 - 0.10$	$0 - 5$	$0.00 - 1.00$	$0.00 - 1.00$

Table 5. Set values considered for each QoS criterion

$Combination$	$Price$	$Duration$	$Reputation$	$Availability$	$Sucessfullrate$
1	0	60	10	15	15
2	10	50	10	15	15
3	20	40	10	15	15
4	30	30	10	15	15
5	40	20	10	15	15
6	50	10	10	15	15
7	60	0	10	15	15

Table 6. Weight distribution

vice selection approaches: local optimization and global planning. In the former, the selection of the Web service that will execute a given task of a composite service is done without taking into account the other tasks involved in the composition. In this way, service selection is locally optimized, but the global quality of the execution may be suboptimal. The latter considers Web service selection as a global optimization problem and linear programming is used to find the solution representing the service composition. Recently, Kokash [9] modifies the Zeng *et al.* method in order to consider the probability of component web service failures, their response time and the execution cost along with the structure of composite graph. In [14], the authors propose a solution for Web service selection taking into account one or many global constraints set by users. In [11], the authors use an approach allowing service selection with best results with slow selection time, by using dynamic programming or good enough results with fast selection time. However, none of these approaches takes into account the transactional behavior of the composite Web service.

By the other hand, several transactional composition mechanism have been proposed to ensure the overall consistency of data modified by complex service resulting from aggregation of Web services offered by different organizations[1, 8]. In [1], Bhiri *et al.* present an approach specifying relaxed atomicity requirements for composite Web services based on Acceptable Termination States (ATS) model and transactional rules to validate a given composite service with respect to defined Transactional Requirements (TR). In [8], Montagut et Molva propose a selection mechanism enabling the automatic design of transactional composite Web service by using the ATS model. The drawback of these approaches is the definition of all the ATS by the user, which is not simple. Moreover, the mechanism does not take into account any QoS criterion in the selection process. In [4], the transactional behavior of composite Web services in presence of transactional component Web service are studied but without taking into account the QoS. Moreover, the authors do no specify the behavior of retriable Web services in case of compensation or abortion of a part of the workflow. Liu *et al.* [5] propose several transactional composition operators for Web services and evaluate the QoS of the composite Web service, considering the abortion cases. They only analyze the transactional effects on QoS, without ensuring the optimal QoS requirement. In our approach, we propose a model for the selection of transactional Web services with the best QoS. Thus, our approach not only fulfills the global transactional requirement but also guarantees locally the best QoS component Web service.

6. Conclusion

In this article, we have presented TQoS, a Web service selection approach supporting transactional and quality driven Web service compositions. In this approach the transactional properties of a composite Web service are established based on the transactional properties of its component Web services. The selection is realized depending on transactional and QoS user requirements. The former are established using a risk notion that indicates if the user can or cannot compensate the result. The latter are expressed as weight over each QoS criterion.

Our contribution is double. Firstly, we have expressed the Web service composition selection taking into account the user preferences in terms of transactional and QoS criteria. Secondly, we have defined formally a Transactional Composite Web Service (TCWS). We have derived the transactional behavioral property of the composition of two component Web services. Then, we have established the transactional behavioral property of the composition of a TCWS with a component Web service before generalizing to the composition of two TCWS. Our experimentation has been done using one input workflow composed of patterns with non composite WS. For this instance, we have supposed an environnement with a specific set of values for the QoS criteria and a specific weight for the execution price and the duration. Our approach has been validated by the experimental results.

More experiment should be done with more complex workflows and different environements and variations in weights for the quality criteria. Currently, we are studying different approaches to replan the selection of Web services in order to take into account dynamic changes that may occur, e.g., a component Web service becomes unavailable or the QoS of one of the component services changes significantly.

Acknowledgement

We would like to thank Guillermo Ramirez, from the Universidad Central de Venezuela, for the experimentations.

References

[1] S. Bhiri, O. Perrin, and C. Godart. Ensuring required failure atomicity of composite web services. *Proceedings of the 14th international conference on World Wide Web (WWW'2005)*, pages 138–147, 2005.

[2] M. C. Jaeger, G.Roec-Goldmann, and G. Muehl. QoS Aggregation for Web Service Composition using Workflow Patterns. *Eighth IEEE International Enterprise Distributed Object Computing Conference(EDOC'04), IEEE Computer Society*, 00:149–159, 2004.

[3] M. C. Jaeger, G. Muehl, and S. Golze. Qos-aware composition of web services: An evaluation of selection algorithms. *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, Lecture Notes in Computer Science 3760*, pages 646–661, October 2005.

[4] L. Li, C. Liu, and J. Wang. Deriving Transactional Properties of Composite Web Services. *IEEE International Conference on Web Services (ICWS'2007)*, pages 631–638, July 2007.

[5] A. Liu, L. Huang, and Q. Li. Qos-aware web services composition using transactional composition operator. *7th International Conference Advances in Web-Age Information Management (WAIM'2006), Lecture Notes in Computer Science 4016*, pages 217–228, June 2006.

[6] S. Mehrotra, R. Rastogi, H. Korth, and A. Silberschatz. A transaction model for multidatabase systems. *Proceedings of the International Conference on Distributed Computing Systems*, pages 56–63, June 1992.

[7] D. Menasce. Composing Web Services: A QoS view. *IEEE Internet Computing*, 6(8):88–90, December 2004.

[8] F. Montagut and R. Molva. Augmenting web services composition with transactional requirements. *IEEE International Conference on Web Services (ICWS'2006)*, pages 91–98, September 2006.

[9] V. D. N. Kokash. Evaluating quality of web services: A risk-driven approach. *Business Information Systems (BIS'2007), Lecture Notes in Computer Science 4439*, pages 180–194, 2007.

[10] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros. Advanced Workflows Patterns. *7th International Conference on Cooperative Information Systems (CoopIS 2000), Lecture Notes in Computer Science 1901*, pages 18–29, 2000.

[11] B. Wu, CH.Chi, and S.Xu. Service selection model based on qos reference vector. *2007 IEEE Congress on Services (Services 2007), IEEE Computer Society*, pages 270–277, 2007.

[12] U. S. Zafar and A. A. Shah. Extended web services framework to meet non-functional requirements. In *Workshop proceedings of the 6th international conference on Web engineering (ICWE '06)*, page 21, 2006.

[13] L. Zeng, A. N. B. Benatallah, M. Dumas, J. Kalagnanam, and H. Chang. QoS-Aware Middleware for Web services Composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, May 2004.

[14] W. Zhang, Y. Yang, S. Tang, and L. Fang. Qos-driven service selection optimization model and algorithms for composite web services. *31st Annual International Computer Software and Applications Conference (COMPSAC'2007), IEEE Computer Society*, 2:425–431, 2007.