Evaluating Quality of Web Services: A Risk-driven Approach

Natallia Kokash and Vincenzo D'Andrea

DIT - University of Trento, Via Sommarive, 14, 38050 Trento, Italy
{kokash, dandrea}@dit.unitn.it

Abstract. Composing existing web services to obtain new functionalities is important for e-business applications. Deficiencies of aggregated web services can be compensated involving a redundant number of them for critical tasks. Key steps lie in Quality of Service (QoS) evaluation and selection of web services with appropriate quality characteristics in order to avoid frequent and severe faults of a composite web service. This paper, first, surveys the existing approaches for QoS-driven web service selection. Then, it proposes a novel approach for evaluating quality of redundant service compositions through analysis of risk related to the use of external web services. Finally, we describe an improved selection algorithm that takes into account success rate, response time and execution cost of involved web services.

Keywords. Web services, Quality of Service, Selection, Composition, Risk Analysis

1 Introduction

Web services are software applications with public interfaces described in XML. According to the established standards, web service interfaces are defined in Web Service Description Language (WSDL). Published in Universal Description, Discovery and Integration (UDDI) directory web services can be discovered and invoked by other software systems. These systems interact with web services using XML-based messages conveyed by Simple Object Access Protocol (SOAP). Web services are considered a promising technology for Business-to-Business (B2B) integration. A set of services from different providers can be composed together to provide new functionalities. One of the most notable efforts in the area of web service composition is the Business Process Execution Language for Web Services (BPEL4WS). BPEL4WS is a language for describing service-based business processes and specifying interaction protocols for involved services.

Web service composition is a complex process involving analysis of process requirements, semantics and behavior of existing services, service testing, adaptation, contracting and management. Despite all the efforts problems both on technical and semantical levels may appear. Modifications of the involved services and their unexpected faults may affect a client application. Erroneous services

can be replaced with analogues to allow for correct behavior of client applications in such situations. Since much work is required to safely introduce a new component in a system, alternative services must be known in advance. In *redundant* service compositions a set of services are not normally used but cater for *fault-tolerance*, i.e. the ability of a system to behave in a well-defined manner once faults occur.

Definition 1. *A web service composition c is said to be* redundant *iff for all executions E of c in which no faults occur, the set S of all services of c contains services that are not invoked in E.*

Due to unsteadiness of business environments service-based systems require run-time monitoring. Statistics about user experience with web services may be used to select well-behaved services. Quality of Service (QoS) is a set of parameters such as service execution cost, performance, reliability, robustness and the like. In this paper, we will refer to quality of composite web services as to Quality of Composition (QoC).

Analysis of QoS of web services is of paramount importance. Multiple proposals aiming at QoS evaluation and selection of better services have appeared because of multi-dimensionality and volatility of QoS parameters. They will be surveyed in the next section. In this paper, we present a novel web service selection algorithm. In contrast to existing work, it does not rely on a simple additive weighting technique for involving QoS parameters such as success rate, response time and execution cost into an objective function. A generalized strategy for QoC evaluation inspired from risk analysis is proposed. We apply our strategy to evaluate quality of redundant compositions, assuming failures of atomic services and regarding composition structure.

The paper is structured as follows. Section 2 discusses related work. In section 3, a notation for modelling redundant service compositions is explained. Section 4 discusses risk management and its application to analysis of QoC. Sections 5 studies failures of composed web services and evaluates impact of these failures on the service composition. In Section 6, an example is given that helps better understand how QoC is calculated. Our service selection algorithm is presented in Section 7. Section 8 presents experimental results. Conclusions and future work are sketched in the last section.

2 Related Work

Description of quality characteristics of web services can be found in [1]. Among them are *throughput* (the number of requests served in a given time period), *capacity* (a limit of concurrent requests for guaranteed performance), *response time* (the time taken by a service to process its sequence of activities), *execution cost* (the amount of money for a single service execution), *availability* (the probability that a service is available), *reliability* (stability of a service functionality, i.e., ability of a service to perform its functions under stated conditions), and so on.

There are several proposals aiming at measuring and specifying QoS for web services. Tosic et al. [2] developed a Web Services Offering Language (WSOL) that allows a service provider to specify five QoS-related constructs: constraints (functional constraints, QoS and access rights), statements, constraint groups, constraint group templates and service offerings. Maximilen and Singh [3] propose an agent-based framework and ontology for QoS measurement. In this approach service providers publish their services to registries and agencies, and service consumers use their agents in order to discover the desired service. The metrics concept is absent in this ontology. A QoS ontology proposed in [4] fills this gap. It consists of three layers: the QoS Profile Layer, used for matchmaking; the QoS Property Definition Layer, used to present the property's domain and range constraints; and the Metrics Layer that provides measurement details. Multiple QoS profiles can be attached to one service profile in this approach.

An interesting task is how to choose web services to be used by a new (composite) web service in order to have a guarantee that required quality level of the composition is reached. Cardoso et al. [5] introduced several models for QoS measurement in workflows. In particular, the authors evaluate expected response time, execution cost and reliability of a workflow applying sequential, parallel, conditional, loop and fault-tolerant system reduction rules. For example, the expected execution cost of a composition including two parallel services s_1 and s_2 is $q_{cost}(s_1) + q_{cost}(s_2)$ while its response time equals $max(q_{time}(s_1), q_{time}(s_2))$. Lakhal et al. [6] extends this work by reviewing the estimation of reliability and response time of fault-tolerant compositions.

Service compositions that embed low-quality services inherit their drawbacks as well. This poses a challenge for software developers that build new systems on the basis of available components. One can compensate deficiency of such systems if many web services with compatible functionality exist. Elaborating this idea, a good number of QoS-driven service selection algorithms have appeared. One of the first works in this direction is done by Zeng et al. [7]. They consider web service selection as a global optimization problem. Linear programming is applied to find the solution that represents the service composition optimizing the target function. The target function is defined as a linear combination of five parameters: availability, successful execution rate, response time, execution cost and reputation. In [8] the service selection is considered as a mixed integer linear program where both local and global constraints are specified. The model by Yu and Lin [9] comes to the complex multi-choice multi-dimension 0-1 knapsack problem. In this approach, the practice of offering different quality levels by services is taken into consideration. Gao et al. [10] apply integer programming to dynamic web service selection in the presence of inter service dependencies and conflicts. Wang et al. [11] consider the measurement of non-numerical qualities. Accuracy, security and exception handling are taken into account. As in the previous work, QoS-driven web service selection is based on assessment of a linear combination of scaled QoS parameters. Yang et al. [12] turn QoS factors to a form following the ascent property. Along with the five QoS attributes used in [7] a service matching degree is analyzed. Matching degree defines a compliance

between composed services and, in principle, is a functional parameter. The Multiple Criteria Decision Making (MCDM) technique is used to give an overall evaluation for a composite web service.

The above solutions depend strongly on user weights assigned to each parameter. There is no clear mechanism allowing a user to set up these weights in order to obtain the desired result. Several approaches try to avoid a user involvement in the selection procedure. For example, in [13] service selection is formulated as Multiple Attribute Decision Making (MADM) problem. Four modes for determining relative weights for QoS attributes are proposed: subjective, single, objective and subjective-objective. Claro et al. [14] follows the quality model proposed in [7] with several improvements. The first extension concerns the concept of reputation that is ranked based on the user's knowledge of the service domain. Secondly, a multi-objective problem is considered as opposed to Zeng's aggregation functions. It is resolved using multi-objective genetic algorithm called NSGA-II, without giving any weight to any quality criterion. Canfora et al. [15] extend works by Cardoso [5] and Zeng [7] in a similar way. A genetic algorithm for QoS-aware workflow re-planning is proposed. An interesting approach is taken by Martin-Diaz et al. [16] who propose a constraint programming solution for procurement of web services with temporal-aware demands and offers. Bonatti and Festa [17] formalize three kinds of service selection problems to optimize the quality of the overall mapping between multiple requests and multiple offers with respect to the preferences associated to services and invocations. In particular, they prove that the problem of cost minimization is NP-hard by reduction from the Uncapacitated Facility Location Problem. Exact and approximated algorithms to solve the formulated problems are proposed. Several works experiment with expressing user QoS preferences in a fuzzy way. Mou et al. [18] set up a QoS requirement model with support of fuzzy metrics for expressing user requirements on target service QoS. In [19] the service selection problem is formalized as a Fuzzy Constraint Satisfaction Problem. Deep-first branch-and-bound method is applied to search for an appropriate web service composition.

Assuming that the failure of any individual web service causes the failure of the composite service, the overall reliability of composite service is the product of the reliability of constituent web services. Therefore, one unreliable web service can decrease the overall reliability to a very low level. The upper bound of overall reliability is often determined by the weakest constituent web services. Jaeger and Ladner [20] consider identification of such weak points. For each weak point they identify alternative candidates that meet the functional requirements. Three possible replacement patterns are analyzed: Additional Alternative Candidate, AND-1/n and XOR-XOR. Dealing with the analogous problem, Stein et al. [21] apply an algorithm for provisioning workflows that achieve higher success probability in uncertain environments through varying the number of providers provisioned for each task.

Diversity of quality metrics, their value ranges and measurements makes it difficult to provide a single QoS-driven selection algorithm for web services.

The existing approaches cover a wide range of methods for multi-objective optimization, but fail to provide a valid formalization of the problem that would sufficiently reflect the real world conditions. A consistent analysis of pros and cons of the listed algorithms is out of the scope of this work. Among the negative characteristics of state-of-the-art algorithms that will be addressed are:

- *Choice of an objective function.* Dependency between different QoS factors is not considered by the existing solutions. Suppose that two weather forecasting services are available: the first one provides reliable forecasts but, as a consequence, it is expensive and rather slow, whereas the second one is cheap and fast but generates forecasts at random. Algorithms based on a simple additive technique are likely to select the second web service despite the fact that the service response time and execution cost are not important if the service is unreliable. Another grave drawback of the methods comparing a weighted sum of relative scores for each quality factor is that bigger compositions are likely to have higher total score. On the other hand, constraint satisfaction algorithms consider QoS separately and do not reason about overall quality of a service.
- *Absence of redundancy.* Despite the efforts aimed at insuring web service reliability (e.g., contracts), service composition failures are almost inevitable. Nevertheless, they can be gently treated without leading to the composition breakdown. As failure-tolerance can be reached through composition redundancy, an important characteristic of a service is the number and quality of services compatible with it in a particular environment. This implies that services must be selected with respect to their context within the composite service and its structure. So, the problem must not be reduced to the selection of a simple execution path where only one web service is assigned to each task.

3 Modelling Redundant Web Service Compositions

Composite web services can be defined using a set of workflow patterns [22]:

- *Sequence.* Several services are executed in a sequence.
- *Loop.* The execution of a service is repeated several times.
- *AND split followed by AND join.* Several services are invoked in parallel and all services must be executed successfully.
- *AND split followed by m-out-of-n join.* Several services (n) are invoked simultaneously, but only $m \leq n$ of them must be executed successfully.
- *XOR split followed by XOR join.* Only one service is invoked from a set of available services. The synchronizing operation considers only the invoked service.
- *OR split followed by OR join.* Several services (n) from all available (k) are invoked and all of the invoked services are required to be finished successfully for synchronization.

Table 1. A notation for representing composite web services.

Graphical	Syntactical	Description
s_i	s_i	A web service.
		The start and the end states.
$\rightarrow$	$(s_1; s_2; ...; s_k)$	Sequential operator.
$\|_n^m$	$(s_1 \| s_2 \| ... \| s_k)_n^m$	Parallel operator. Indices m and n are used to represent AND split followed by m-out-of-n join (bottom index $n = k$ and upper index $m = n$ can be omitted).
$+_n^m$	$(s_1 + s_2 + ... + s_k)_n^m$	Choice operator. Indices m and n are used to represent OR split followed by m-out-of-n join (bottom index $n = k$ and upper index $m = 1$ can be omitted).

– *OR split followed by m-out-of-n join.* Several services (n) from all available (k) are invoked and $m \leq n$ services must be executed successfully.

The above workflow patterns form a set of functional and structural requirements which cover most service-based flow languages (e.g., BPEL4WS). We suppose that sequential composition prescribes an order for the execution of services. The situation when the execution of a set of services can be performed in an arbitrary order is also possible in practice. It can be modelled as XOR split followed by XOR join of all alternative sequences with a prescribed order. Loop can be seen as a special case of sequential composition. For specifying composite web services with redundancy we will use a notation drawn in Table 1.

Several services are composed in an application that can be available as a new web service (see Fig. 1). The provider of this service is in a difficult situation as (s)he must guarantee a certain level of QoS to end users, and in the same time, quality of the provided service depends on agreements established among the partners and quality of the involved services. For example, one of the possible problems is a limited capacity of the atomic services. A composite service will be forced to pay penalties to its clients because it cannot satisfy all requests in a required time. To avoid such bottlenecks, the maximum capacity of the composition must be controlled. A set of run-time changes in the composition model should be taken into account:

– *Service capacity correction.* It reflects changes in the monitored service performance related to the increase/decrease of service load by external clients, problems in a communication network or middleware, etc.
– *Service deletion.* Some web service is not available for the invocation.
– *Service addition.* A new service is introduced into a composition.
– *State deletion.* All services that can be invoked from this state are deleted.
– *Sub-composition deletion.* Any service can be deleted if there is no path between the start state and the end state that includes this service. Iteratively repeating state and service deletion we can delete a sub-composition.
– *Sub-composition addition.* The reverse operation to the sub-composition deletion arises if a new sub-composition is involved in the model.

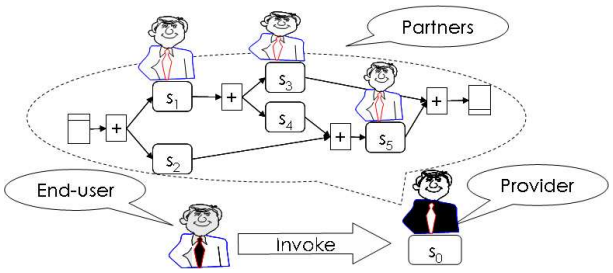


Fig. 1. An example of a service-based workflow

Distributions of the service capacity and the expected number of concurrent requests must be compared to guarantee a stable execution of the composite web service. Generally, we may speak about risk that quality of a composite web service will be affected because of problems with involved services. If this risk is significant we must try to mitigate it, for example, negotiating quality of service with partners or adopting other services.

In the next section we discuss risk management and its application to QoS-driven web service selection.

4 Risk Management

The purpose of risk management is to reduce or neutralize potential risks and offer opportunities for some positive improvements. Risk management operates with notions of *threats* (danger sources), *probabilities* of threats (likelihoods that given threats will be triggered) and their *impacts* on the organization (monetary loss, breach of reputation, etc.). Risk r can be expressed mathematically as a probability p of a threat e multiplied by a respective magnitude q of its impact: $r(e) = p(e)q(e)$. Three main steps of risk management include *identification,* which is needed for surfacing risks before they become problems, *analysis,* when identified risk data is converted into decision-making information, and *control,* which consists of monitoring the status of risk and actions taken to reduce them. Appropriate risk metrics must be developed to enable the evaluation of the risk status and mitigation plans.

Risk management covers all steps of software development and business process modelling lifecycle. Several risk management frameworks [23][24] have been created to keep software projects within established time and budget constraints. Other related work concerns risk analysis for business processes [25][26]. A *business process* is a structured set of activities which are designed to produce a specified output for a particular customer or market. Business processes are subject to errors in each of their components: to enable the successful completion of a business process it is important to manage risks associated with each process activity and with the overall process. Defining a business process, a company can rely on *outsourcing*, i.e., a formal agreement with a third party to

perform a service for an organization. In this case, the risk management process requires evaluation of proposals, identification of best providers and sources, evaluation of supplier financial viability, country risk assessment, etc. along with costs of failure or non-delivery of outsourced functional modules [27].

Further we will consider risks specific for execution of service-based business processes to that extent as they may affect composition design. Such risks can be divided into three basic categories:

- *Inter-organizational risks* are caused by providers of web services used in a service composition. Into this category we can put risks related to such events as disposal of a service by the provider, changes in interface and behavioral logics of a service, contract violation, obtrusion of a new contract with worse conditions, intentional disclosure of private user information, etc.
- *Technical risks* are related to technical aspects of distributed systems such as network or service failures.
- *Management risks* may be caused by the use of automatic management systems. For example, requests from some unprivileged clients may be ignored or delayed because of the limited capacity of a web service, etc.

Risk analysis uses two basic types of techniques, namely *quantitative* and *qualitative*. Qualitative analysis involves the extensive use of mathematics and reports based on probabilities of threats and their estimated costs. Qualitative analysis is a verbal report based on system knowledge, experience, and judgment. Statistical information about service behavior is essential for risk analysis. The assessment has to be ongoing and evaluations of the probability of events happening revised as the system is used and evolves. Without the benefit of a quantitative assessment risk analysis is subjective and has to be based largely on common sense and experience. For example, services provided by a large well-known company can be considered less risky than services of a small unknown company.

In ideal risk management a prioritization process is followed: the higher risks are handled first, and the lower risks are handled later. Each external web service can be seen as a black box with a certain QoS. Several actions are possible to manage risks caused by use of external web services:

- Communicate with the service provider in order to establish an agreement that can help to mitigate the risk.
- Mitigate the impact of the risk by identifying a triggering event and developing a contingency plan.
- Try to avoid risks by changing the design of the application. In particular, functionality of unreliable services can be (1) implemented from scratch, (2) taken from open source projects, (3) provided by software components that are deployed locally.
- Accept the risk and take no further actions, thus, accepting the consequences.
- Study the risk further to acquire more information and better determine the characteristics of the risk to enable decision making. For example, conditions when failures of external services are more likely can be discovered.

As standard protocols simplify involvement of new web services, we can try to reduce risks through web service selection. However, if too many services are included in the composition, its cost increases. A composition that maximizes the overall profit must be selected. As risks define expected loss in some period of time, the problem can be formalized as selection of a composition c_0, such that

$$Q_{profit}(c_0) = Q_{income}(c_0) - R(c_0) = \max_{c \in C}(Q_{income}(c) - R(c)),$$

where $Q_{income}(c)$ is an income expected by the provider, and

$$R(c) = \sum_{e_j \in E(c)} r(e_j) = \sum_{e_j \in E(c)} p(e_j) Q_{loss}(e_j)$$

is an estimated risk of the composition c internally used by the composite service. Here, C denotes a set of all available compositions, $E(c)$ is a set of independent risk-related events identified for a composition c, $p(e_j)$ is a probability that an event e_j will occur and $Q_{loss}(e_j)$ is an estimated loss function of this event. In a ProRisk Management Framework [28] other strategies for risk assessment are provided, given that some events are not totally independent of others. For example, assuming that there exist one dominating threat, the following fuzzy model for risk assessment is valid[1]:

$$R(c) = \max_{e_j \in E(c)} r(e_j) = \max_{e_j \in E(c)} p(e_j) q(e_j).$$

5 Failure Risk

Failure risk is a characteristic considering probability that some fault will occur and the resulting impact of this fault on the composite service. For an atomic service s_i it equals

$$r(s_i) = p(\overline{s}_i) q(s_i),$$

where $p(\overline{s}_i)$ is a failure probability of service s_i, and $q(s_i)$ is a *loss function*. Service s_1 is better than service s_2 if it has a smaller failure risk $r(s_1) < r(s_2)$.

Let $c = (s_1; s_2; ...; s_k)$ be a sequential composition. If a service s_i fails the results of services $\{s_1, s_2, ..., s_{i-1}\}$ will be lost as well whereas their response time and execution cost increase the total expenses to satisfy a user request. These expenses are included in a loss function of a service s_i failure. Hence, a failure risk of a service s_i in a sequential composition equals:

$$r(s_1; ...; s_i) = p(s_1; ...; s_{i-1}; \overline{s}_i) q(s_1; ...; s_i) = \prod_{j=1}^{i-1} p(s_j) p(\overline{s}_i) \sum_{j=1}^{i} q(s_j).$$

Let us consider an example. Suppose that a user needs to translate a text from Belarusian to Turkish provided that five translation web services are available:

[1] The original formula includes also weighting coefficients to scale the impact of the threat into an appropriate utility measure.

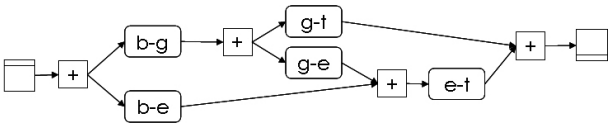


Fig. 2. A redundant service composition with three possible execution paths.

b-e translates from Belarusian to English, *b-g* from Belarusian to German, *g-t* from German to Turkish, *e-t* from English to Turkish, and *g-e* from German to English (see Fig. 2). There are three configurations that can fulfill the user goal, i.e., the text can be initially translated (1) from Belarussian to English and then from English to Turkish, (2) from Belarussian to German and then from German to Turkish, (3) from Belarussian to German, then from German to English and finally from English to Turkish. Suppose we have chosen the first composition. If the service *e-t* fails, the task will not be completed and the translation done by the service *b-e* will be lost. Instead, if the second composition is chosen, in case of a *g-t* failure, the task still can be completed successfully by switching to the third composition without rollback.

In our example, $r(e\text{-}t) = p(b\text{-}e)p(\overline{e\text{-}t})(q(b\text{-}e)+q(e\text{-}t))$, where $p(e\text{-}t)$ is a failure probability of the service *e-t*. Loss function $q(.)$ can refer either to execution cost of the services *b-e* and *e-t* or to their expected response time. In the latter case, the estimation of time loss will be obtained. It may be important for tasks with deadlines, i.e., with limits on the latest time for a task to be accomplished.

In complex service oriented systems calculation of a loss function may involve analysis of transactional aspects of the process. The loss function of the AND split followed by AND join pattern with service sequences in each branch $c = (s_1; ...; s_n)|(t_1; ...; t_m)$ depends on the service coordination mechanism. We may distinguish *centralized* and *decentralized* compositions. In the first case, parallel branches can be interrupted immediately after the fault detection, therefore loss functions of a service s_i failure are:

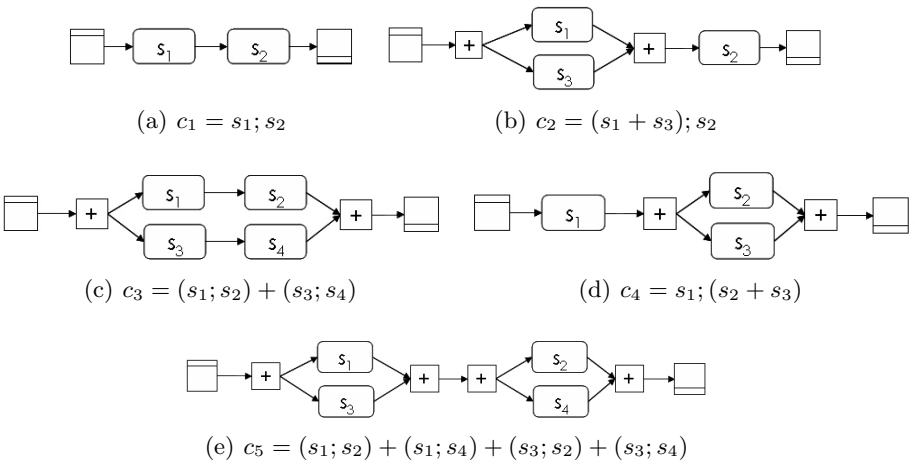
$$q_{time}(c|\overline{s}_i) = \sum_{j=1}^{i} q_{time}(s_j), \quad q_{cost}(c|\overline{s}_i) = \sum_{j=1}^{i} q_{cost}(s_j) + \sum_{j=1}^{k} q_{cost}(t_j),$$

where $k\,(1 \leq k \leq m)$ is the number of services executed before the s_i failure has been detected. In the second case, additional expenses can be involved since additional time is required to forward an error message to a place where it can be correctly processed.

6 Failure Risk of Redundant Service Compositions

In this section, we provide an example of failure risk management for redundant service compositions.

Redundant compositions include one or more XOR/OR split followed by XOR/OR/m-out-of-n join patterns that define alternative ways to accomplish



(a) $c_1 = s_1; s_2$ (b) $c_2 = (s_1 + s_3); s_2$

(c) $c_3 = (s_1; s_2) + (s_3; s_4)$ (d) $c_4 = s_1; (s_2 + s_3)$

(e) $c_5 = (s_1; s_2) + (s_1; s_4) + (s_3; s_2) + (s_3; s_4)$

Fig. 3. Composite web services with different structure.

some tasks. In our scenario, end users invoke a composite web service, which invokes a set of other services to fulfill user requests. If the user task is not satisfied, the provider of the composite service pays a compensation. Similarly, if an atomic service fails, the provider of the composite service receives a compensation from the provider of the failed service. A Service Level Agreement (SLA) with the end user can be established in such a way that a service composition will satisfy the constraints on response time and execution cost provided that no faults occur. Unexpected failures of component services lead to resource loss and may cause a violation of negotiated parameters. If there are stand-by resources (the maximum budget for a task is not reached, there is time before a task execution deadline), a user task can be completed by other web services. Therefore, it is reasonable to create a contingency plan in order to improve fault resistance of a composite web service. For redundant compositions a *contingency plan* is a set of triples $\langle a_k, t_j, c_i \rangle$, expressing the fact that a sub-composition c_i is started from a state t_j after an event a_k. Here, t_j is one of the XOR/OR split states, and a_k refers to the actions discussed in Section 4.

Let $q_{cost}(s_i)$ be an execution cost and $q_{pnlt}(s_i)$ be a penalty of an atomic service s_i. We will denote a difference between the service execution cost and the penalty paid if the service s_i fails by $q_{df}(s_i) = q_{cost}(s_i) - q_{pnlt}(s_i)$. Let also $q_{pnlt}(c)$ be a penalty paid by the provider of the composite web service in case of its failure. Figure 3 shows five web service compositions with different internal structure. Their failure risk is shown in Table 2. It is assumed that service failures are independent, i.e., $p(\overline{s}_i|\overline{s}_j) = p(\overline{s}_i)$, $1 \leq i, j \leq n, i \neq j$. Failure risk defines an expected amount of money the provider will lose exploiting external services with certain QoS, provided different levels of redundancy: in the first case, only one service is assigned to each task; in the last case, two services are assigned

Table 2. Failure risk for compositions in Fig. 3. The risk values are given for the following parameters: $p(s_i) = p(\overline{s}_i) = 0.5$, $q_{cost}(s_i) = q_{pnlt}(s_i) = 1$, $q_{pnlt}(c) = 2$.

c_i	**Failure risk calculation formula**	$R(c_i)$
c_1	$p(\overline{s}_1)\big(q_{df}(s_1) + q_{pnlt}(c)\big) + p(s_1)p(\overline{s}_2)\big(q_{cost}(s_1) + q_{df}(s_2) + q_{pnlt}(c)\big)$.	1.75
c_2	$p(\overline{s}_1)\big(q_{df}(s_1)+p(\overline{s}_2)(q_{df}(s_2)+q_{pnlt}(c))+p(s_2)p(\overline{s}_3)(q_{cost}(s_2)+q_{df}(s_3)+ q_{pnlt}(c)\big) + p(s_1)p(\overline{s}_3)\big(q_{cost}(s_1) + q_{df}(s_3) + q_{pnlt}(c)\big)$.	1.625
c_3	$p(\overline{s}_1)\big(q_{df}(s_1)+p(\overline{s}_3)(q_{df}(s_3)+q_{pnlt}(c))+p(s_3)p(\overline{s}_4)(q_{cost}(s_3)+q_{df}(s_4)+ q_{pnlt}(c))\big) + p(s_1)p(\overline{s}_2)\big(q_{cost}(s_1) + q_{df}(s_2) + p(\overline{s}_3)(q_{df}(s_3) + q_{pnlt}(c)) + p(s_3)p(\overline{s}_4)(q_{cost}(s_3) + q_{df}(s_4) + q_{pnlt}(c))\big)$.	1.5625
c_4	$p(\overline{s}_1)\big(q_{df}(s_1)+q_{pnlt}(c)\big)+p(s_1)p(\overline{s}_2)\big(q_{df}(s_2)+p(\overline{s}_3)(q_{cost}(s_1)+q_{df}(s_3)+ q_{pnlt}(c))\big)$.	1.375
c_5	$p(\overline{s}_1)\big(q_{df}(s_1)+p(\overline{s}_3)(q_{df}(s_3)+q_{pnlt}(c))+p(s_3)p(\overline{s}_4)(q_{cost}(s_3)+q_{df}(s_4)+ q_{pnlt}(c))\big) + p(s_1)p(\overline{s}_2)\big(q_{df}(s_2) + p(\overline{s}_4)(q_{cost}(s_1) + q_{df}(s_4) + q_{pnlt}(c))\big)$.	1.25

to each task. An alternative combination is used only if the previous one fails to complete the process.

Failure risk is a compound measure considering probability of constituent service failures, their response time and/or execution cost along with the structure of a composition graph. Intuitively, compositions with many OR branches are more reliable. However, which configuration will be selected depends on the balance between the above parameters. For example, if only two web services can accomplish some task and one of them failed, it might be better for the composite service to stop the execution instead of trying a second service if it is too expensive.

7 Web Service Selection Algorithm

In this section we present a modification of the method by Zeng et al. [7] for web service selection based on the above ideas.

Let us consider a composite service that invokes a set of external web services. *Success rate* of a service s can be defined statistically as $p(s) = N_{suc}(s)/N_{total}(s)$, where N_{suc} is the number of successful service responses and N_{total} is the total number of observed invocations. A service invocation is considered *successful* if the user goal is satisfied or we can proceed along with the execution, i.e., (1) the service is available, (2) the successful response message is received within an established timeout, (3) no errors are detected automatically in the output, (4) service effects are satisfied, (5) preconditions of a subsequent service are met. If necessary, we can distinguish the above situations and consider several metrics for service successful invocation. Success rate defines the *probability of success* $p(s)$ for future service invocations. Along with the probability of success we can consider the *probability of failure* $p(\overline{s}) = 1 - p(s)$.

The main idea of the service selection algorithm is to search for a simple path $(s_1; ...; s_k)$ between the start and the end states in the composition graph that

maximizes the following target function:

$$f(c) = p(c)(q^{max} - q(c)) = p(s_1; ...; s_k)(q^{max} - q(s_1; ...; s_k)) = \\ = \prod_{i=1}^{k} p(s_i)(q^{max} - \sum_{i=1}^{k} q(s_i)),$$

where q^{max} defines the resource limit, taken from an SLA or chosen big enough to guarantee the positive value of $f(c)$. This formula can be inferred from the more general one (see Section 4) assuming that the provider pays for external web services only if all of them are executed successfully, and no penalty is paid to a user if the composite service fails. Thus, to maximize the profit, the percentage of the successful invocations must be maximized and cost of each invocation must be minimized. Here $q(.)$ can refer to response time, execution cost, or a function including both of them. For instance, we can use a linear combination of execution cost and response time: $q = w_1 q_{time} + w_2 q_{cost} \,|\, w_1 + w_2 = 1, 0 \leq w_1, w_2 \leq 1$. Service availability is not considered explicitly as in [7], however, according to our definition, this aspect is characterized by the service success rate.

8 Experimental evaluation

In order to analyze our approach empirically, we compared the proposed service selection algorithm with the linear programming approach by Zeng et al. [7]. We developed a simulation of a web service composition engine and generated a large number of random service compositions. For the data presented in this paper, we used 100 compositions of 10 atomic web services. Such a relatively small number of services included in one composition is chosen to follow the realistic scenarios. For each atomic web service its execution cost, response time and success rate are defined randomly with uniform distribution, from 0 to $maxCost = 1000\$$ for execution cost, from 0 to $timeout = 1000ms$ for response time, and from 0.5 to 1 for success rate (values greater than 0.5 are generated to avoid services with very low success rate). We compared the performance of our method with the performance of the linear programming approach by recording the expected response time, execution cost and success rate of the compositions chosen by these two methods. We also simulated invocation of the chosen compositions and compared their real success rates. We assigned weights $w_i = \frac{1}{3}$ for each of the three parameters for the linear programming approach, and $w_i = 0.5$ for response time and execution cost in our approach. The solutions proposed by our modified algorithm had better response time and execution cost than the solutions found by the linear programming approach, in 96% and 89% of tests, correspondingly. In the same time, expected success rates of these solutions were always better. More details about the presented results are available in our technical report [29].

9 Conclusion

We have proposed a risk-driven methodology for QoS evaluation. This approach may help to simplify web service selection by considering cost equivalent of various QoS factors. We have demonstrated how risk analysis can be used to measure impact of atomic service failures on a service composition. Our experiments prove that the difference between expected income and expenses better characterizes the problem of QoS-driven web service selection from provider's perspective than a linear combination of scores for various QoS factors.

An obvious drawback of our metric is that different redundant compositions require risk recalculation, which makes the approach computationally less efficient than methods relying on the QoS evaluation of well defined patterns [20]. In our previous work [30] a polynomial-time greedy heuristic selecting a less risky sub-composition in each XOR split state was proposed.

Service oriented systems are open to various risks. Different techniques might be needed for their identification, analysis and control. In our future work we are going to systematize and elaborate the above ideas.

References

1. Ran, S.: A model for web services discovery with QoS. ACM SIGecom Exchanges **4**(1) (2003) 1–10
2. Tosic, V., Pagurek, B., Patel, K.: WSOL - a language for the formal specification of classes of service for web services. In: Proceedings of the ICWS, CSREA Press (2003) 375–381
3. Maximilien, M., Singh, M.: A framework and ontology for dynamic web services selection. IEEE Internet Computing **8**(5) (2004) 84–93
4. Zhou, C., Chia, L., Lee, B.: DAML-QoS ontology for web services. In: Proceedings of the ICWS, IEEE Computer Society (2004) 472 – 479
5. Cardoso, J., Sheth, A., Miller, J., Arnold, J., Kochut, K.: Quality of service for workflows and web service processes. Journal of Web Semantics **1**(3) (2004) 281–308
6. Lakhal, N.B., Kobayashi, T., Yokota, H.: A failure-aware model for estimating the efficiency of web service compositions. In: Proceedings of the IEEE Pacific Rim International Symposium on Dependable Computing, IEEE Computer Society (2005) 114–121
7. Zeng, L., Benatallah, B., Ngu, A.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for web services composition. IEEE Transactions on Software Engineering **30**(5) (2004) 311–327
8. Ardagna, D., Pernici, B.: Global and local QoS constraints guarantee in web service selection. In: Proceedings of the ICWS, IEEE Computer Society (2005) 805–806
9. Yu, T., Lin, K.: Service selection algorithms for composing complex services with multiple QoS constraints. In: Proceedings of the ICSOC. Volume 3826 of LNCS., Springer (2005) 130 – 143
10. Gao, A., Yang, D., Tang, S., Zhang, M.: QoS-driven web service composition with inter service conflicts. In: Frontiers of WWW Research and Development - APWeb: Asia-Pacific Web Conference, Springer Berlin Heidelberg (2006) 121 – 132

11. Wang, X., Vitvar, T., Kerrigan, M., Toma, I.: A QoS-aware selection model for semantic web services. In: Proceedings of the ICSOC. Volume 4294 of LNCS., Springer (2006) 390–401
12. Yang, L., Dai, Y., Zhang, B., Gao, Y.: Dynamic selection of composite web services based on a genetic algorithm optimized new structured neural network. In: Proceedings of the International Conference on Cyberworlds, IEEE Computer Society (2005) 515 – 522
13. Hu, J., Guo, C., Wang, H., Zou, P.: Quality driven web services selection. In: Proceedings of the ICEBE, IEEE Computer Society (2005)
14. Claro, D., Albers, P., Hao, J.K.: Selecting web services for optimal composition. In: Proceedings of the International Workshop on Semantic and Dynamic Web Processes. (2005) 32–45
15. Canfora, G., Penta, M.D., Esposito, R., Villani, M.L.: QoS-aware replanning of composite web services. In: Proceedings of the ICWS, IEEE CS Press (2005)
16. Martin-Diaz, O., Ruize-Cortes, A., Duran, A., Muller, C.: An approach to temporal-aware procurement of web services. In: Proceedings of the ICSOC, Springer (2005) 170–184
17. Bonatti, P., Festa, P.: On optimal service selection. In: Proceedings of the International WWW Conference, ACM Press (2005) 530–538
18. Mou, Y., Cao, J., Zhang, S., Zhang, J.: Interactive web service choice-making based on extended QoS model. In: Proceedings of the CIT, IEEE Computer Society (2005) 1130–1134
19. Lin, M., Xie, J., Guo, H., Wang, H.: Solving QoS-driven web service dynamic composition as fuzzy constraint satisfaction. In: Proceedings of the Int. Conference on e-Technology, e-Commerce and e-Service, IEEE Computer Society (2005) 9–14
20. Jaeger, M., Ladner, H.: Improving the QoS of WS compositions based on redundant services. In: Proceedings of the NWeSP, IEEE Computer Society (2005)
21. Stein, S., Gennings, N.R., Payne, T.R.: Flexible provisioning of service workflows. In: Proceedings of the ECAI, IOS Press (2006) 295–299
22. van der Aalst, W.M., ter Hofstede, A.H., Kiepuszewski, B., Barros, A.: Workflow patterns. Distributed and Parallel Databases **14**(3) (2003) 5–51
23. Verdon, D., McGraw, G.: Risk analysis in software design. IEEE Security and Privacy (2004) 33–37
24. Freimut, B., Hartkopf, S., et al.: An industrial case study of implementing software risk management. In: Proceedings of the ESEC/FSE, ACM Press (2001) 277–287
25. Muehlen, M., Ho, D.T.Y.: Integrating risks in business process models. In: Proceedings of Australasian Conference on Information Systems (ACIS). (2005)
26. Neiger, D., Churilov, L., Muehlen, M., Rosemann, M.: Integrating risks in business process models with value focused process engineering. In: Proceedings of the ECIS. (2006)
27. O'Keeffe, P., Vanlandingham, S.: Managing the risks of outsourcing: a survey of current practicies and their effectiveness. White paper, http://www.protiviti.com/downloads/PRO/pro-us/product_sheets/business_risk/Protiviti_ORM_WhitePaper.pdf (2004)
28. Roy, G.: A risk management framework for software engineering practice. In: Proceedings of the ASWEC, IEEE Computer Society (2004) 60 – 67
29. Kokash, N., D'Andrea, V.: Evaluating quality of web services: A risk-driven approach. Technical Report DIT-06-099, DIT-University of Trento, Italy (2006)
30. Kokash, N.: A service selection model to improve composition reliability. In: International Workshop on AI for Service Composition, University of Trento (2006) 9–14