Evaluating Quality of Web Services: A Risk-driven Approach

Natallia Kokash and Vincenzo D’Andrea

December 2006

Technical Report # DIT-06-099

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 good quality in order to avoid frequent and severe faults of a composite service. This paper, first, surveys the existing approaches for QoS-driven web service selection. Then, it describes an improved selection algorithm that takes into account success rate, response time and execution cost of involved web services. Finally, we propose a novel approach for evaluating quality of redundant service compositions through analysis of risk related to the use of external web services.

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 complicated process involving careful 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 behavioral levels may appear. For example, 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 the business environment service-based systems require constant run-time monitoring. Statistics about user experience with web services then 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. We also 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 works, 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. A new service selection algorithm is presented in Section 3. In section 4, a notation for modelling redundant service compositions is explained. Section 5 discusses risk management and its application for analysis of QoC. Sections 6 studies failures of composed web services and evaluates impact of these failures on the service composition. In Section 7, an example is given that helps better understand how QoC is calculated. Section 8 presents experimental results. Finally, conclusions and future work are sketched in the last section.

2 Related Work

Numerous works devoted to quality of web services were published in the last years. They cover various research questions, for example:

- How to specify a variety of QoS factors?
- How to define run-time QoS information taking into account their volatility and complexity?
- How to match user requirements with existing services in terms of QoS?
- How to specify user preferences about web services?
- How to perform ranking of similar services with respect to user preferences?
- How to predict QoS factors under certain environmental conditions implying dependencies among QoS parameters and relations to contextual factors?

2.1 Representation of QoS information and user preferences

There are several proposals aiming at measuring and specifying QoS for web services. Tosic et al. [1] 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 [2] 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 [3] 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. Tian et al. [4] extend the matchmaking mechanisms with the concept of service broker. QoS parameters are classified into two main categories: network and server-client parameters. An extension of service publication and discovery mechanisms with QoS features is proposed in [5]. This approach exploits the finite automata theory in multidimensional spaces. The proposed model performs well only with static QoS parameters. One more ontology and vocabulary adequate for arbitrary web services is proposed in in [6]. Bleul and Weiss [7] support service packaging and include a Unit-Transformation-Ontology to define functional relations between metrics. Four ontologies presented in [8]: requirements, measurement, traceability and quality management, aim to minimize ambiguities in QoS evaluations.

An approach to specify user requirements with help of a QoS ontology and a requirement matching tool are presented in [9]. Vu et al. [10] propose a model for the users to describe their QoS selection criteria taking into account environmental conditions. These conditions are specified by providers in their service descriptions via description logics and Horn rules. The discovery, ranking and selection of the matched services are customizable via the use of appropriate domain ontologies that represent both user preferences and provider specifications. The basic steps of the discovery process are defined as algebraic operators of a query execution model. This enables plug-in of different algorithms into the discovery framework. Kerrigan [11] identifies two types of preferences that are useful for service selection: *filtering preferences*, which filter the list of services found during discovery, and *ordering preferences*, which sort the list services found during discovery. Liu et al. [12] provide a model to measure QoS factors, including such of them as compensation, penalty policies and transaction, through an active users feedback and monitoring.

Service Level Agreement (SLA) defines the agreed level of performance for a particular service between a service provider and a service user. SLA parameters can be measured with different metrics, including composite ones like maximum response time or average availability. In [13] implementation of a rule engine for SLA evaluation is presented. Ranganathan and Dan [14] present a framework for allocating/de-allocating of physical resources and provisioning/de-provisioning of service instances in Grid environment to meet SLA constraints.

2.2 QoS analysis and service selection

Another 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. [15] introduced several useful 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. Lakhal et al. [16] extends this work by reviewing the estimation of reliability and response time of fault-tolerant web service compositions. These results are consistent with reliability models valid for general component-based software systems [17].

Norton [18] indicates several inconsistencies in QoS measures by Cardoso and Sheth used in the METEOR-S approach [19]. He also proposes a new algorithm to calculate the service fitness metric, normalized from 0 to 1. Balke et al. [20] examine different techniques towards advanced personalization of web service selection. Sharma et al. [21] experiment with dynamic request prioritization schemes for web services. Sensoy [22] proposes an experience-based approach for service provider selection, in which consumers record their experiences with service providers rather than the overall, subjective ratings. Menasce et al. [23] provide a methodology for planning service capacity. Knowledge about the number of potential users, frequency of service invocations and their time distributions are essential for the analysis. An accurate capacity planning can be quite problematic because of parameter uncertainty and limited budget.

Service compositions that embed low-quality services inherit all their drawbacks. This poses a big challenge for the software developers building new systems on the basis of available components. One can compensate composition deficiency 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. [24]. They consider the service selection task 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. If the global characteristics, like total amount of money to accomplish a task, are not restricted, an optimal solution can be found by a modified Dijkstra's algorithm searching on the graph of available compositions [25]. In [26] 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 [27] 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. [28] apply integer programming to dynamic web service selection in the presence of inter service dependencies and conflicts. Wang et al. [29] consider the measurement of non-numerical qualities. For example, reputation of a service may be evaluated as *low*, *medium* or *high*. Accuracy, security and exception handling are taken into account. However, as in the previous works, QoS-driven web service selection is based on assessment of a linear combination of scaled QoS parameters. Yang et al. [30] turn QoS factors to a form following the ascent property. Along with the five QoS attributes used in [24], 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 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 [31] 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. [32] follows the quality model proposed in [24] 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. [33] extend works by Cardoso and Zeng in a similar way. A genetic algorithm for QoS-aware workflow re-planning is proposed. Hacigumus [34] formalizes the problem of cost-driven web service composition as a Weighted Set-Covering Problem. Similar to the previous work, Cao et al. [35] examine only execution cost. They propose a genetic algorithm to search for the web service composition with optimal cost, arguing that other methods are too expensive. Interesting approach is given in the work by Martin-Diaz et al. [36] where a constraint programming solution for procurement of web services with temporal-aware demands and offers is proposed. Bonatti and Festa [37] 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. For example, Mou et al. [38] set up a QoS requirement model with support of fuzzy metrics for expressing user requirements on target service QoS. In [39] 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. Gao et al.[40] study quality of service compositions from the provider perspective. The objective of this work is to optimize the aggregate bandwidth utilization within an operator's network. The task is formalized as a problem of mapping a given arbitrary service graph to a graph of physical network.

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 could 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 [41] 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*. In the first template the original service is replaced by an alternative service. In the second one, a structure containing the alternative candidates in a parallel AND-split with 1-out-of-n-join pattern is involved instead of the original service. These arrangements reduce the response time, improve probability for the successful execution of the task, and raise execution cost by the sum of all additionally executed services. In *XOR-XOR* structure only one of the available alternative candidates is invoked. Response time is reduced if the selected candidate executes quicker. Execution cost is raised by the mean value of the individual costs. Reputation of services in all patterns is reduced if additional candidates have lower reputation. Dealing with the analogous problem, Stein et al. [42] apply an algorithm for provisioning workflows that achieve higher success probability in uncertain environments through varying the number of providers provisioned for each task.

A consistent analysis of pros and cons of the listed algorithms is out of the scope of this work. However, we must point out that this would be useful since algorithms in the original works are compared mostly only with primitive or random strategies. Among the negative characteristics of state-of-the-art algorithms that will be addressed in this paper 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 redun-

dancy, 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 Web Service Selection Algorithm

Description of QoS factors for web services can be found in [43] or in the works that propose QoS ontologies, e.g. [3][7]. 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.

Success rate of a service s is 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* $p(\overline{s}) = 1 - p(s)$.

For sequential composition we may distinguish *additive metrics*, such as response time and execution cost, *multiplicative metrics,* such as availability, and *concave metrics* such as throughput or capacity. QoS aggregation functions for different patterns can be found in [15]. For example, execution cost of two parallel services $c = (s_2|s_3)$ is $q_{cost}(c) = q_{cost}(s_1) + q_{cost}(s_2)$, their response time $q_{time}(c) = max(q_{time}(s_1), q_{time}(s_2))$, and probabilities of success and failure $p(c) = p(s_1)p(s_2)$ and $p(\overline{c}) = p(\overline{s}_1) + p(\overline{s}_2) - p(\overline{s}_1)p(\overline{s}_2)$, correspondingly.

Our service selection algorithm is a modification of the method proposed in [24]. This algorithm searches 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 $f(c)$). Here, $q = \{q_{time}, q_{cost}\}$ may refer either to response time or to execution cost. In this case we have a *dual criteria optimization* problem. The basic approach is to put the less important parameter into an objective function provided that the most important criterion meets some constraints. Although it is hard to predict which of the identified dimensions is more important (response time or execution cost), we suppose that for a provider of composite web services focus should be put on the response time. Usually, the internal structure of services is hidden from the end-user, so (s)he expects to pay a fixed price for a single service execution (provided the same quality level). At the same time, service delays can be indemnified by penalties. On conditions that response time constraint is satisfied, a provider can optimize its own expenses. If the user preferences are given, the above function can incorporate several scaled QoS parameters, e.g.: $q(s) = f(q_{time}, q_{cost})$. Assuming that f is a linear combination, we have $q = w_1 q_{time} + w_2 q_{cost} \,|\, w_1 + w_2 = 1,\ 0 \leq w_1, w_2 \leq 1$. We do not consider explicitly service availability, however, according to our definition, this aspect is characterized by a service success rate.

4 Modelling Redundant Web Service Compositions

Composite web services can be defined using a set of workflow patterns [44]:

- *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.
- *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.

Workflow patterns form a set of functional and structural requirements which are applied to most flow languages for web service composition [45]. 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,

Table 1. A notation for representing composite web services.

Graphical	**Syntactical**	**Description**
$\boxed{s_i}$	s_i	A web service.
▭ ▭		The start and the end states.
$\rightarrow$	$(s_1; s_2; ...; s_k)$	Sequential operator.
$\boxed{\mid^m_n}$	$(s_1 \mid s_2 \mid ... \mid s_k)^m_n$	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).
$\boxed{+^m_n}$	$(s_1 + s_2 + ... + s_k)^m_n$	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).

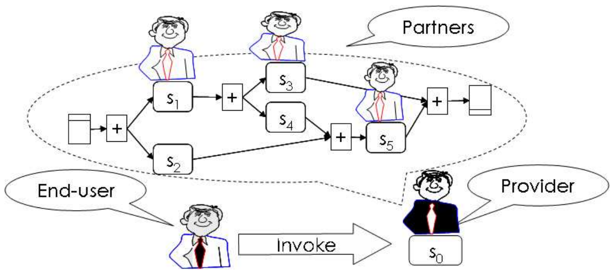


Fig. 1. An example of a service-based workflow

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 one 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 in 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 a 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 sub-composition deletion arises if a sub-composition is involved in the model.

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

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

5 Risk Management

The purpose of risk management [46] is to reduce or neutralize potential risks and offer opportunities for some positive improvement. Risk is defined as a potential impact that may arise from some present process or from some future event. By this definition, risk can be expressed mathematically as the probability of occurrence of loss/gain multiplied by its respective magnitude. Risk is commonly associated with negative outcomes. Thee main steps of risk management are:

– *Identification.* Risk identification is needed for surfacing risks before they become problems.
– *Analysis.* Risk analysis is a process of converting identified risk data into decision-making information.
– *Control.* Risk control consists of monitoring the status of risk and actions taken to ameliorate 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. We will consider only risks specific for execution of service-based business processes to that extent as they may affect the design of service composition and selection of services to be involved.

5.1 Risk Identification

Risks that can affect web service compositions at run-time can be divided into several categories:

– *Inter-organizational risks.* This category includes risks 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.
– *Management risks.* Problems related to use of automatic management systems may occur. For example, requests from some unprivileged clients may be ignored or delayed because of the limited capacity of a web service, etc.
– *Technical risks.* This group includes risks related to technical aspects of distributed systems such as network or service failures.

5.2 Risk Analysis

Risk analysis uses two basic types of techniques: quantitative and qualitative. Qualitative analysis involves the extensive use of mathematics and reports based on probabilities of events 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. It allows qualitative assessment of the identified risks. Each external web service is seen as a black box encapsulating unknown realization with a certain QoS. Without the benefit of a quantitative assessment of the event probability 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. The assessment has to be ongoing and evaluations of the probability of events happening revised as the system is used and evaluates, and the risk becomes clearer.

5.3 Risk Control

In ideal risk management, a prioritization process is followed whereby the risks with the greatest loss and the greatest probability of occurring are handled first, and risks with lower probability of occurrence and lower loss are handled later. Several actions are possible to manage the risk 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 risk by careful selection of constituent services. 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 $R(c)$ is an estimated risk of the composition c internally used by the composite service. In order to enable the effective risk assessment it is necessary to consider the dependency between risk events. It may be necessary to elicit a number of probability values or probability distributions and estimate conditional probabilities amongst selected events. For a set $E(c)$ of substantially independent risk-related events identified for a composition c

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

Here, C denotes a set of all available compositions, $p(e_j)$ is a probability that an event e_j will occur and $q_{loss}(e_j)$ is an estimated loss function in this case. The risk of mutually exclusive events $E(c)$ can be defined as

$$R(c) = \prod_{e_j \in E(c)} r(e_j) = \prod_{e_j \in E(c)} p(e_j) q_{loss}(e_j).$$

Further, assuming that "if anything can go wrong the event with the largest risk value will be the one", the following fussy model for risk estimation can be proposed:

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

Roy [47] in addition to the above strategies uses the weighting coefficient w_j to scale the risk effect of the event e_j occurring into an appropriate utility measure.

Let us consider a composite service that sequentially invokes a set of external services. Suppose that the provider pays for these services only if all of them are executed successfully, and no penalty is paid to a user if the composite service fails. In order to maximize the profit, the percentage of the successful invocations must be maximized and cost of each invocation must be minimized. From here we get an objective function used in Section 3:

$$f(c) = \prod_{i=1}^{k} p(s_i)(q_{income}(c) - \sum_{i=1}^{k} q(s_i)),$$

where $q_{income}(c)$ denotes the execution cost paid by the end-user of the composite service. Naturally, this cost is higher than provider expenses to use other services.

6 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 $\{s_1, s_2, ..., s_i\}$ will be lost as well whereas their response time

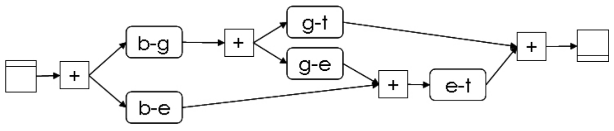


Fig. 2. A redundant service composition with three possible execution paths.

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:

$$q(s_1; ...; s_i) = \sum_{j=1}^{i} q(s_j).$$

A failure risk of a service s_i in a sequential composition is

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

where

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

is the probability of the composition failure while service s_i is being executed.

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: *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. 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 (see Fig. 2). 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 failure of the service *g-t*, 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* and $q(.)$ refers either to execution cost or response time of the services *b-e* and *e-t*. Time losses may be important for tasks with *deadlines*. A deadline defines the latest time for a task to be accomplished. *Soft deadlines* can be violated with penalties. Tasks with *hard deadlines* should be accomplished within a deadline or rejected immediately with a fixed penalty.

In complex service oriented systems calculation of a loss function may involve analysis of transactional aspects of the process as rollback of a whole transaction can be caused by a service failure. 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:

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

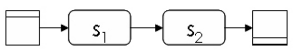
where k $(1 \leq k \leq m)$ is the number of services executed before the s_i failure has been detected. In the second case, additional expenses can be involved as time is required to forward an error.

7 Failure Risk of Redundant Service Compositions

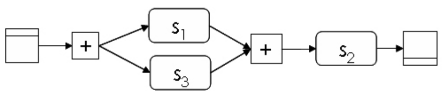
In this section, we provide an example of failure risk management for redundant service compositions. Redundant compositions include one or more XOR/OR split followed by XOR/OR/m-out-of-n join patterns that define alternative ways to accomplish some task.

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 this service. An 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 the normal conditions, i.e., that no faults occur. The unexpected failures of component services lead to resource loss and may cause the violation of negotiated parameters. If there are reserve resources (the maximum budget for a task is not reached and there is time left before 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 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 5.1.

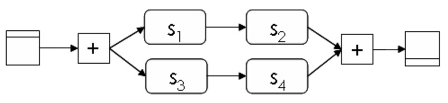
Let $q_{cost}(s_i)$ be the execution cost and $q_{pnlt}(s_i)$ the penalty for an atomic service s_i. By $q_{diff}(s_i) = q_{cost}(s_i) - q_{pnlt}(s_i)$ we will denote a difference between service execution cost and penalty paid if a service execution fails. Let also $q_{pnlt}(c)$ be a penalty paid by the provider of the composite service in case of its failure. Assuming that service failures are independent, i.e., $p(\overline{s}_i|\overline{s}_j) = p(\overline{s}_i)$, $1 \leq i, j \leq n, i \neq j$, the failure risk for the composite services drawn in Fig. 3 is shown in Table 2. It defines an expected amount of money the provider will loose, 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 to each task. An alternative combination is used only if the previous one fails to complete the process.



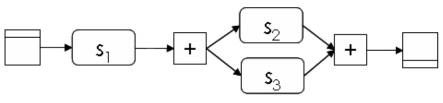
(a) $c_1 = s_1; s_2$



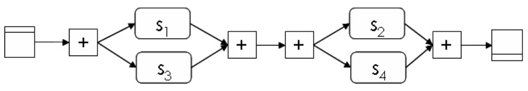
(b) $c_2 = (s_1 + s_3); s_2$



(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.

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.

8 Experimental evaluation

In order to analyze our approach empirically, we compared a service selection algorithm described in Section 3 with the linear programming approach proposed in [24]. 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

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

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. The corresponding experimental results are shown in Fig. 4. In this figure, arrows show the difference between total response time, execution cost and success rates (expected and obtained in simulations) of the service compositions selected by two algorithms.

9 Conclusion

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.

We have proposed a risk-driven methodology for QoS evaluation. Risk is the probability of occurrence of some (negative) event multiplied by its respective magnitude. 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 QoC metric for web service selection is that different compositions require risk recalculation, which makes the approach computationally less efficient than methods relying on the QoS evaluation of well de-

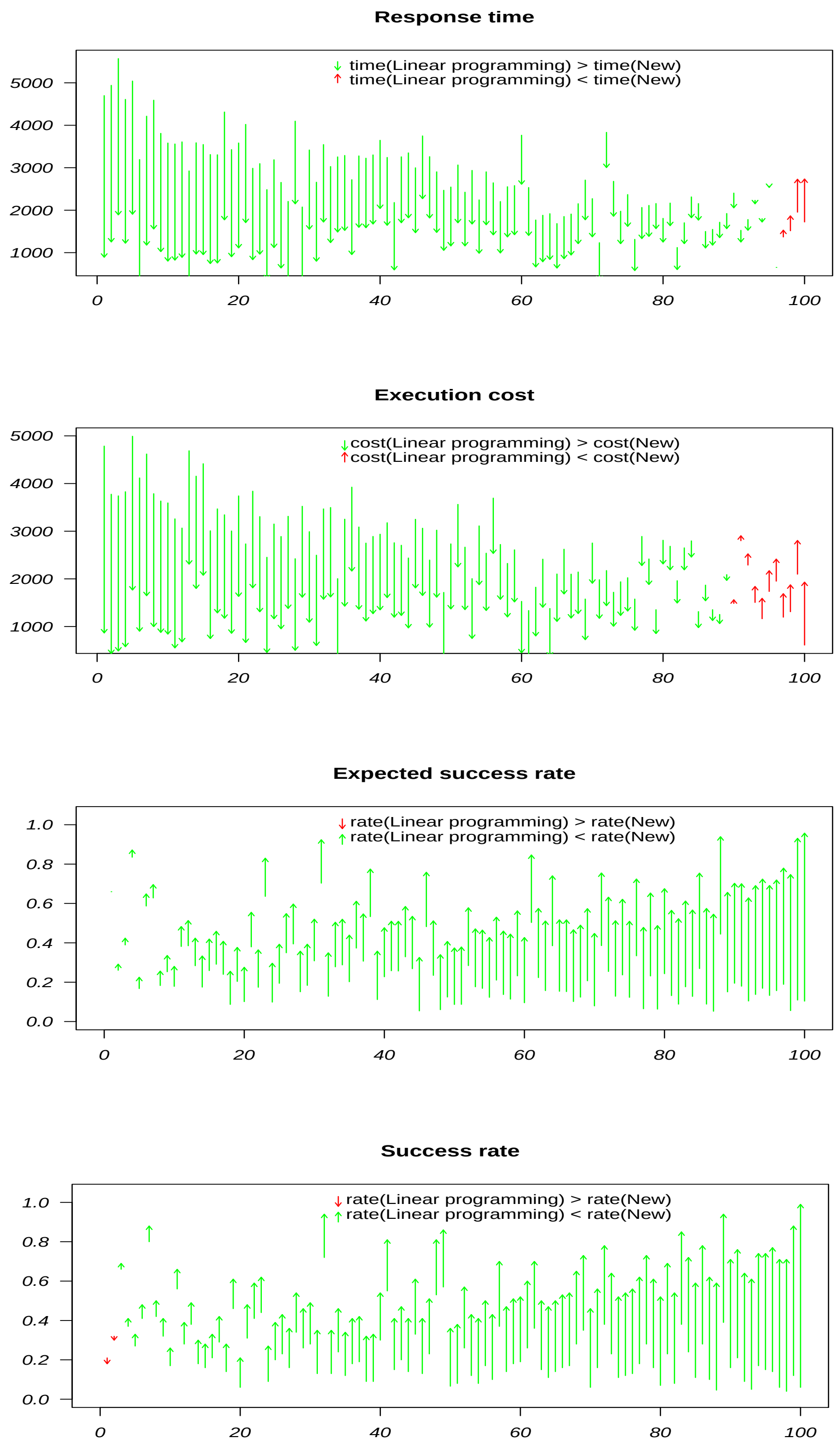


Fig. 4. Comparison of QoS of web service compositions selected by two methods

fined patterns [41]. In our previous work [48] a polynomial-time greedy heuristic selecting a less risky sub-composition in each XOR split state is 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.

10 Acknowledgments

We thank Artsiom Yautsiukhin for useful comments on a preliminary version of this paper.

References

1. Tosic, V., B.Pagurek, Patel, K.: WSOL a language for the formal specification of classes of service for web services. In: Proceedings of ICWS, CSREA Press (2003) 375–381
2. Maximilien, E.M., Singh, M.P.: A framework and ontology for dynamic web services selection. IEEE Internet Computing **8**(5) (2004) 84–93
3. Zhou, C., Chia, L., Lee, B.: DAML-QoS ontology for web services. In: International Conference on Web Services (ICWS), IEEE Computer Society (2004) 472 – 479
4. Tian, M., Gramm, A., Naumowicz, T., Ritter, H., Schiller, J.: A concept for QoS integration in web services. In: International Conference on Web Information Systems Engineering Workshops (WISEW), IEEE Computer Society (2003) 149–155
5. Bianchini, D., Antonellis, V.D., Melchiori, M.: QoS in ontology-based service classification and discovery. In: International Workshop on Web Semantics. (2004)
6. Papaioannou, I., Tsesmetzis, D., Roussaki, I., Miltiades, E.: QoS ontology language for web-services. In: Proceedings of the International Conference on Advanced Information Networking and Applications (AINA), IEEE Computer Society (2006)
7. Bleul, S., Weiss, T.: An ontology for quality-driven web service discovery. In: Workshop on Engineering Service Compositions. (2005) 35–42
8. Kim, H., Sengupta, A., Evermann, J.: MOQ: Web services ontologies for QoS and general quality evaluations. In: European Conference on Information Systems (ECIS). (2005)
9. Dobson, G., Lock, R., Sommerville, I.: Quality of service requirements specification using an ontology. In: SOCCER Workshop, Requirements Engineering. (2005)
10. Vu, L.H., Porto, F., Hauswirth, M., Gerlach, S., Tajmouati, O., Aberer, K.: QoS-enabled semantic web service discovery: a user's perspective approach. Technical Report LSIR-REPORT-2006-012, Distributed Information Systems Laboratory (2006)
11. Kerrigan, M.: Web service selection mechanisms in the web service execution environment (WSMX). In: Proceedings of the ACM Symposium on Applied Computing (SAC), ACM Press (2006) 1664 – 1668
12. Liu, Y., Ngu, A.H., Zeng, L.: QoS computation and policing in dynamic web service selection. In: Proceedings of 13th International World Wide Web (WWW) Conference, ACM Press (2004) 66–73

13. Bostrom, G., Giambiagi, P., Olsson, T.: Quality of service evaluation in virtual organizations using SLAs. In: 1st International Workshop on Interoperability Solutions to Trust, Security, Policies and QoS for Enhanced Enterprise Systems (IS-TSPQ). (2006)
14. Ranganathan, K., Dan, A.: Proactive management of service instance pools for meeting service level agreements. In: Proceedings of the International Conference on Service-Oriented Computing (ICSOC). Volume 3826 of LNCS., Springer (2005) 296–309
15. 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
16. Lakhal, N.B., Kobayashi, T., Yokota, H.: A failure-aware model for estimating the efficiency of web service compositions. In: Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing, IEEE Computer Society (2005) 114–121
17. M.Ross, S.: Introduction to Probability Models. Harcourt Asia PTE LTD (1997)
18. Norton, B.: A sound mathematical basis for quality of service profiles in web service discovery. Technical Report Technical Report CS-04-11, University of Sheffield (2005)
19. Cardoso, J., Sheth, A.: Semantic e-workflow composition. Journal of Intelligent Information Systems **21** (2003) 191–225
20. Balke, W.T., Wagner, M.: Towards personalized selection of web services. In: Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing. (2005) 114–121
21. Sharma, A., Adarkar, H., Sengupta, S.: Managing QoS through prioritization in web services. In: Proceedings of the International Conference on Web Information Systems Engineering Workshops, IEEE (2004) 140–148
22. Sensoy, M.: A framework for context-aware service selection. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2006)
23. Menasce, D., Almeida, V.: Capacity Planning for Web services. Prentice Hall, Upper Saddle River, NJ (2002)
24. 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
25. Gu, X., Chang, R.: OoS-assured service composition in managed service overlay networks. In: Proceeding of The IEEE 23rd International Conference on Distributed Computing Systems (ICDCS), IEEE Computer Society (2003)
26. Ardagna, D., Pernici, B.: Global and local qos constraints guarantee in web service selection. In: IEEE International Conference on Web Services, IEEE Computer Society (2005) 805–806
27. Yu, T., Lin, K.: Service selection algorithms for composing complex services with multiple QoS constraints. In: Proceedings of the International Conference on Service-Oriented Computing (ICSOC). Volume 3826 of LNCS., Springer (2005) 130 – 143
28. 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: 8th Asia-Pacific Web Conference, Springer Berlin Heidelberg (2006) 121 – 132
29. Wang, X., Vitvar, T., Kerrigan, M., Toma, I.: A QoS-aware selection model for semantic web services. In: Proceedings of the International Conference on Service-Oriented Computing (ICSOC). Volume 4294 of LNCS., Springer (2006) 390–401

30. 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
31. Hu, J., Guo, C., Wang, H., Zou, P.: Quality driven web services selection. In: Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE), IEEE Computer Society (2005)
32. Claro, D.B., Albers, P., Hao, J.K.: Selecting web services for optimal composition. In: Proceedings of the ICWS Second International Workshop on Semantic and Dynamic Web Processes. (2005) 32–45
33. Canfora, G., Penta, M.D., Esposito, R., Villani, M.L.: QoS-aware replanning of composite web services. In: Proceedings of the International Conference on Web Services, IEEE CS Press (2005)
34. Hacigumus, H.: Cost-effective service composition. In: WESC, in conjunction with ICSOC. (2005) 93–100
35. Cao, L., Li, M., Cao, J.: Cost-driven web service selection using genetic algorithm. In: Workshop on Internet and Network Economics. LNCS, Springer (2005) 906–915
36. Martin-Diaz, O., Ruize-Cortes, A., Duran, A., Muller, C.: An approach to temporal-aware procurement of web services. In: International Conference on Service-Oriented Computing, Springer (2005) 170–184
37. Bonatti, P.A., Festa, P.: On optimal service selection. In: Proceedings of the 14th international conference on World Wide Web, ACM Press (2005) 530–538
38. Y. Mou, J. Cao, S.Z., Zhang, J.: Interactive web service choice-making based on extended QoS model. In: Proceedings of the International Conference on Information Technology (CIT), IEEE Computer Society (2005) 1130–1134
39. Lin, M., Xie, J., Guo, H., Wang, H.: Solving QoS-driven web service dynamic composition as fuzzy constraint satisfaction. In: IEEE International Conference on e-Technology, e-Commerce and e-Service. (2005) 9–14
40. Gao, X., Jain, R., Ramzan, Z., Kozat, U.: Resource optimization for web service composition. In: Proceedings of IEEE Conference on Service Computing (SCC), IEEE Computer Society (2005)
41. Jaeger, M.C., Ladner, H.: Improving the QoS of WS compositions based on redundant services. In: Proceedings of the International Conference on Next Generation Web Services Practices (NWeSP), IEEE Computer Society (2005)
42. Stein, S., Gennings, N.R., Payne, T.R.: Flexible provisioning of service workflows. In: Proceedings of the 17th European Conference on Artificial Intelligence, IOS Press (2006) 295–299
43. Ran, S.: A model for web services discovery with QoS. ACM SIGecom Exchanges **4**(1) (2003) 1–10
44. van der Aalst, W.M., ter Hofstede, A.H., Kiepuszewski, B., Barros, A.: Workflow patterns. Distributed and Parallel Databases **14**(3) (2003) 5–51
45. van der Aalst, W.M.: Dont go with the flow: Web services composition standards exposed. Issue of IEEE Intelligent Systems **18**(1) (2003) 72–76
46. Charette, R.N.: Software Engineering Risk Analysis and Management. New York: McGraw-Hill (1989)
47. Roy, G.G.: A risk management framework for software engineering practice. In: Proceedings of the Australian Software Engineering Conference (ASWEC), IEEE Computer Society (2004) 60 – 67
48. Kokash, N.: A service selection model to improve composition reliability. In: International Workshop on AI for Service Composition, in conjunction with ECAI, University of Trento (2006) 9–14