QoS-Aware Service Selection Considering Potential Service Failures

Bernd Heinrich and Lars Lewerenz

Universität Regensburg, Germany
{Bernd.Heinrich, Lars.Lewerenz}@wiwi.uni-regensburg.de

Abstract. Nowadays, service compositions are increasingly used to execute business processes. During the execution of a service composition, a service failure leads to a necessary re-planning. Due to such runtime events, the ex-post realized Quality of Service (QoS) values and thus the realized utility of an executed service composition may be significantly lower than the ex-ante computed one. The presented paper examines how the consideration of the effects of potential service failures can be modeled for an ex-ante QoS-aware service selection using expected utilities. Furthermore, we analytically evaluate our approach and demonstrate its applicability by an example. By doing so, we show that considering the effects of potential service failures leads to substantial better decisions about the QoS-aware service selection.

Keywords: QoS-aware service selection, IT services, service failures

1 Introduction

Service oriented computing (SOC) was and still is one of the major research topics as well as a main driver for the software industry (cf. [1]) in the last years. The characteristics of SOC, loose coupling, dynamic binding, open standards, simplicity and security [2], create the possibility of flexible ad-hoc collaboration between two or more business partners [3]. Besides the use of a single service, multiple services can be composed to support the execution of business processes. As the services market enhances (e.g., [4], [5]), more and more services are offered by different providers which offer an equal or quite similar functionality [3] (e.g. webservices.seekda.com and programmableweb.com offer in the meantime over 30,000 web services with partly equal or similar functionality). Given such functional-equivalent services, non-functional criteria like execution price or availability of services (cf. [6]) become very relevant for selecting services.

The possibility of composing services brings greater flexibility for realizing a process. But gained flexibility is not without a cost. The price that has to be paid is in particular a greater complexity [7]. According to Yu and Lin [8] there are three main factors which are responsible for the greater complexity: (1) The large number of atomic services that may be available; (2) The different possibilities of integrating atomic services into a composed service; (3) Various performance requirements of an atomic
as well as composed service. Scholars and practitioners (e.g., software companies like IBM) put great effort particularly on the third point. Especially from an economic point of view it is very interesting to know how single services (so called service candidates) can be selected and compiled to a service composition without violating given time or price requirements to name but a few. Service compositions meeting such requirements are called feasible solutions. Given a utility function of a user, one optimal service composition or more out of the set of feasible solutions can be determined by maximizing the utility value. To deal with this optimization problem suitable selection approaches are needed (cf. [6], [9–12]).

All of these approaches select ex-ante the optimal QoS-aware service composition, i.e. before executing the services and without considering the following aspects:

1. In case, an invoked service is not available during process runtime, a re-planning of the selected service composition is necessary ([6], [13], [14]). Due to such runtime events, the ex-post realized end-to-end QoS values and thus the realized utility of a service composition may (significantly) differ from the ex-ante computed ones [15]. This effect occurs, for instance, when a service fails and has to be replaced by another service having worse QoS values (e.g. in terms of execution price). However, existing approaches do not assess and take into account these effects of potential service failures in their ex-ante determination of the optimal QoS-aware service composition.

2. As discussed in 1. a service composition needs to be re-planned in case an invoked service is not available (cf. [14], [16]). Thereby, current approaches for selecting ex-ante the optimal service composition neglect to which extent such a re-planning effects the feasibility of different service compositions regarding the end-to-end QoS requirements (e.g. an upper limit regarding the end-to-end costs) of the process.

3. In case of a service failure it will take a certain time till the failure is noticed and compensated (comparable to time-to-repair [17–19]). This time interval is left unconsidered by current approaches, although it has a direct influence on the end-to-end response time and thus on the utility of the affected service composition.

4. A re-planning of a service composition may cause a switch on an alternative service composition during the runtime [13], [20]. Thereby, losses could occur, in case services that have already been executed are not used in the alternative service composition again. As these losses directly influence the end-to-end QoS values and thus the utility of the service composition, they have to be considered within an optimization.

As a consequence of the aspects 1.-4. the ex-ante optimal QoS-aware service composition could significantly differ from the ex-post optimal one after the process execution, a feature that has to be considered within the selection problem. These reflections (cf. aspect 1.-4.) may be especially interesting for business processes with valuable output that are executed very often. An inferior selection made here can lead to a high loss of resources, such as time and money, during the process execution. Therefore, the research questions of this paper are as follows:
How to design an ex-ante optimization approach for a QoS-aware service selection that can cope with the effects of potential service failures? Can this approach lead to a better decision about the optimal QoS-aware service selection?

In order to contribute to these questions, we structure the paper as follows: In the next section, the prior research concerning the QoS-aware service selection is discussed. Then, we introduce a running example that is used on the one hand to show how a QoS-aware service selection is done by current approaches and on the other hand for the evaluation of our approach later on. Afterwards, we present this approach to address the aspects 1.-4. The penultimate section is not only dedicated to an analytical evaluation of our approach. In addition, we demonstrate by an example its strength and benefit compared to existing approaches. Finally, the limitations, conclusion and an outlook on future research will be given.

2 Literature Review

Several literature streams have already covered approaches for the QoS-aware service selection as well as re-planning approaches in case of a service failure. To ensure an overview over the existing literature we conducted a literature review according to Webster and Watson [21]. In a first step appropriate papers for our research were ascertained. Therefore, we used the TOP 30 journals of the ranking of the Association of Information Systems (including several IEEE and ACM journals) as well as the ICIS and ECIS conference papers as the basis of our review. The journals were searched for suitable papers with the terms: service selection, service composition, composite service, QoS-aware service, end-to-end QoS service, service re-planning, service re-binding, QoS-aware re-binding. In the second step we reviewed the citations of the identified papers in order to determine further papers. Finally, we used Google Scholar to find papers citing the key papers identified in the previous steps. Thereby, we obtained 426 papers. First we read the titles and abstracts. We considered an article relevant, if the main contribution of the article described an approach for a QoS-aware service composition indeed. After this review 72 papers were included. To further contain relevant papers, we read the articles in detail and selected only those of them, which were concerned with the topics of availability, service failure and the possible effects resulting from service failure (cf. aspect 1.-4.). Finally, due to the length restriction of the paper at hand, we selected at least one representative article for each identified selection or re-planning approach.

An overview over these approaches can be seen in Table 1 (see appendix). We briefly discuss in the following the works dealing with an optimal QoS-aware service selection. Afterwards four selected papers offering approaches for a re-planning are presented. For the determination of the optimal QoS-aware service composition [6], [22] proposed a global optimization approach by applying the method of integer programming. They maximize a given utility function under adherence of specific QoS requirements. Focusing on the same objective, Ardagna and Pernici [10] propose the method of mixed integer programming. Wan et al. [23] applied a branch and bound algorithm as well as a divide-and-conquer algorithm that separates the service compo-
sition in smaller segments which are then being optimized. Yu et al. [9] offer two approaches to address the selection problem. The first approach (BBPL) is for a multiple choice multiple dimension knapsack problem (MCKP) and is based upon a branch and bound algorithm. In an earlier work Yu et al. [7] also applied the method of dynamic programming to solve the MCKP. The second approach (MCSP) is based upon a graph constrained optimum path model which finds the optimal path in a service candidate graph according to a utility function. As a heuristic, a frequently applied approach are genetic algorithms (cf. [11], [15], [24], [25]). Thereby, a fitness function of a population (service composition of randomly selected service candidates) is maximized through the construction of several follow-up generations that can be created through the methods of mutation, crossover or selection. This procedure is repeated, until a defined termination condition is fulfilled. In contrast to the use of genetic algorithms as a heuristic, Berbner et al. [3] applied the method of mixed integer programming and improve the gained solution with the help of two meta heuristics called H2_SWAP and H3_SIM_ANNEAL (based upon simulated annealing). Yu et al. [9] provide a quite similar procedure in their heuristic approach WS_HEU by improving a feasible solution in two further steps. Yang et al. [26] use a genetic algorithm to determine the input parameters for the ant colony algorithm and use the latter algorithm then to optimize the service composition.

Besides this, several approaches have been developed concerning a re-planning during the runtime of a service composition. Lin et al. [13], [27] try to repair the service composition by exchanging the service candidate that has failed. They iteratively expand the number of service candidates that are exchanged, starting from the faulty one, till a feasible solution is found or the service composition needs to be terminated if a) no feasible solution is available or b) the re-planning region is too big (e.g. defined as the maximum percentage of services to be repaired; Lin et al.[13]). Berbner et al. [14] optimize the unexecuted part of the service composition after every single service invocation. This procedure ensures that the service composition stays feasible, valid and optimal during its execution. Contrary to this approach, Canfora et al. [15] monitor the realized QoS values and decide based on a local and global threshold whether to re-plan the current service composition, rather than re-optimizing after every service invocation as [14].

To sum up: In all of the above discussed selection approaches a utility function is optimized subject to given end-to-end QoS requirements. Thereby, the availability of a service candidate is considered by a single QoS attribute (cf. Table 1 in the appendix) which is used in combination with the other QoS attributes to determine the utility of a service candidate resp. the entire service composition. However, the effects (cf. aspects 1.-4.) resulting from a potential failure of a service candidate (i.e. the effects in case a service candidate is actually not available) are left unconsidered. In this context, the availability of a service could be determined in several ways. One possibility are service providers which often offer performance reports about their offered services (cf. [28]). In addition, service intermediaries like programmableweb.com offer monitoring tools [29] that allow the user to monitor the availability of any provided service. Besides these possibilities, many service management software tools (e.g. IBM’s WebSphere Integration Developer) offer the possibility to track and monitor
QoS values like the availability probability. In this case, not only external services, but also intra-company services can be monitored resp. their availability probability can be determined.

Furthermore, current re-planning approaches leave the time interval till a failure of a service is noticed and compensated unconsidered (so called time-to-repair). The same holds for losses that can occur e.g. when, after a re-planning, services that have already been executed are not used in the new service composition anymore. As losses and the time-to-repair have a direct influence on the end-to-end QoS values of the service composition and thus on their utility, they need to be considered within a re-planning as well. In the following, we introduce a running example to illustrate these effects and our approach.

3 Running Example (cf. also [9])

In the running example the following service classes $S_1$ to $S_6$ (see Figure 1. in the appendix) each with different service candidates (e.g. $s_{11}$ and $s_{12}$ for class $S_1$) are given. In total, there are four possible paths ($S1$-$S2$-$S3$-$S4$; $S1$-$S2$-$S6$; $S5$-$S2$-$S3$-$S4$; $S5$-$S2$-$S6$; each starting with the source node $V_s$ and ending with the sink node $V_d$) how the service composition and thus the process can be executed. The corresponding service candidates of each service class $S_1$ to $S_6$ with their QoS values are given in Table 2 (see appendix). All in all there are 54 possible service compositions that can be defined (cf. [9]). In contrast to these possible service compositions the number of the feasible service compositions depends on the end-to-end QoS requirements which are given by: end-to-end response time $T \leq 600$, end-to-end costs $C \leq 250$ and end-to-end availability $A \geq 0.85$. Moreover, to evaluate which of the feasible service composition is the optimal one [9] use a utility function $U$, which is defined as follows:

$$U(y) = \sum_{\alpha=1}^{x} w_{\alpha} \left( \frac{q_{\alpha} - \mu_{\alpha}}{\sigma_{\alpha}} \right) + \sum_{\beta=1}^{y} w_{\beta} \left( 1 - \frac{q_{\beta} - \mu_{\beta}}{\sigma_{\beta}} \right)$$

(1)

Considering the utility function $U$, there are $x$ QoS attributes (with $\alpha=1...x$) that will be maximized (e.g. the availability) and $y$ QoS attributes (with $\beta=1...y$) which will be minimized (e.g. the response time). $q_{ij}$ is a QoS vector for each service candidate $s_{ij}$; $\mu$ and $\sigma$ are the mean and standard deviation for each QoS attribute, considering the QoS values of all service candidates $s_{ij}$ in service classes $S_i$. The user can set up preferences ($\omega_{\alpha}$, $\omega_{\beta}$) for each QoS attribute, where (0<$\omega_{\alpha}$, $\omega_{\beta}$<1) and $\sum_{\alpha=1}^{x} w_{\alpha} + \sum_{\beta=1}^{y} w_{\beta}$ holds. In our example, the response time got the highest preference with a value of 0.5, followed by the costs with 0.4 and the availability 0.1.

Given that selection problem, we search for the optimal QoS-aware service composition provided by any of the existing (analytical) approaches (cf. Table 1 in the appendix). Without loss of generality, we selected for that task the MCSP approach proposed by [9]. A reason why this approach was chosen is that it can easily be implemented, since [9] offer a pseudo code for MCSP in their paper. Using this approach for ex-ante optimization, nine feasible service compositions were determined.
Out of these feasible solutions, the optimal service composition that was found is $s_{11}$-$s_{22}$-$s_{32}$-$s_{42}$ with a response time of 560, costs of 225, availability of 0.895.

As discussed above, the effects resulting from a potential failure of a service candidate (cf. aspects 1-4.) are left unconsidered so far. More precisely: Let us suppose two invoked service candidates $s_{11}$ and $s_{22}$ for the service classes $S_1$ and $S_2$ that possess the same probability of availability. In case service candidate $s_{11}$ fails, a substitute service candidate $s_{32}$ is available but with really worse QoS values. Opposed to that, if service candidate $s_{22}$ fails, a substitute service candidate $s_{32}$ is also available that has nearly the same QoS values as service candidate $s_{22}$. In other words, although both service candidates $s_{11}$ and $s_{32}$ are evaluated equally regarding the probability of availability, significantly different QoS values will be realized in case of their particular failure. The reason is the characteristic of the QoS attribute availability, as it got the ability to change the realization of the other end-to-end QoS values and thus the utility of a service composition as a consequence of a service failure. However, if the availability of a service candidate is only treated as a QoS attribute and thus the availability of a service composition is only optimized subject to a given end-to-end QoS requirement, the effects caused by a potential service failure are neglected.

Furthermore, the violation of the end-to-end QoS requirements may be another effect of a service failure. More precisely: Given two feasible service compositions $s_{11}$-$s_{22}$-$s_{32}$-$s_{41}$ and $s_{12}$-$s_{22}$-$s_{32}$-$s_{42}$. Furthermore, we suppose that the service composition $s_{11}$-$s_{22}$-$s_{31}$-$s_{42}$ nearly exceeds the end-to-end QoS requirement response time, but is still feasible as said before. Now during the runtime of the service composition $s_{11}$-$s_{22}$-$s_{31}$-$s_{41}$ the service candidate $s_{41}$ fails and should be replaced by the service candidate $s_{42}$. However, this is not feasible anymore, as realizing the QoS values of service candidate $s_{42}$ after the execution of $s_{11}$-$s_{22}$-$s_{31}$ would violate the end-to-end QoS requirements. The outcome would be a premature termination of the invoked service composition. Therefore, it can be reasonable to initially select the service composition $s_{12}$-$s_{22}$-$s_{32}$-$s_{42}$ even if the utility is smaller than the one of the service composition $s_{11}$-$s_{22}$-$s_{31}$-$s_{41}$ as illustrated in the example. Otherwise this may lead to a loss of resources (e.g. time and money). Thus a special treatment of the QoS availability is again important within an optimization approach to avoid the waste of these resources.

4 An Approach Considering the Effects of Potential Service Failures

The idea of our approach is to consider the effects of potential service failures within the ex-ante selection of the optimal QoS-aware service composition, i.e. before executing this composition. On the one hand, the effects can be determined by calculating an expected utility for service compositions using the probability of availability for each of their included service candidates. On the other hand, it is also analyzed in case of a potential service failure whether alternative service compositions violate the given end-to-end requirements and are thus not feasible to continue the interrupted service composition. As a result, the so calculated end-to-end QoS values as well as the expected utility contain the effects of potential service failure (cf. aspect 1-4.).
For the setup of the approach, we use the following notation (according to [6, 9]):

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_i$</td>
<td>Service class $i$ that includes all service candidates $x_j$ that implements the action $i$ (with $i = 1$ to $I$) of the considered service composition $SC$</td>
</tr>
<tr>
<td>$s_j$</td>
<td>Service candidate $s_j$ (with $j = 1$ to $J$, [9]) for the service class $i$ with $x_j = 1$ if the service candidate $s_j$ is selected for class $S_i$ and $x_j = 0$ otherwise</td>
</tr>
<tr>
<td>$q_{ij}$</td>
<td>QoS vector for each service candidate $s_j$ with $q_{ij} = [q_{ij}^1, \ldots, q_{ij}^N]$ including the single value for each QoS attribute $n$ with $n = 1$ to $N$ (excluding the QoS attribute ‘availability’)</td>
</tr>
<tr>
<td>$Q$</td>
<td>Global (end-to-end) QoS requirements vector $Q = [Q^1, \ldots, Q^N]$ for a service composition including the single requirements $Q^i$ for each QoS attribute $n$ with $n = 1$ to $N$</td>
</tr>
<tr>
<td>$p_i$</td>
<td>Probability $p_i$ of failure of a certain service candidate $s_j$, representing the QoS attribute ‘availability’</td>
</tr>
<tr>
<td>$U$</td>
<td>Utility function for a risk neutral decision maker to calculate the utility $U(x_j)$ for a single service candidate $x_j$ based on its QoS vector $q_{ij}$</td>
</tr>
<tr>
<td>$E^R(U(x_j))$</td>
<td>Expected utility for a single service candidate $x_j$ based on its QoS vector $q_{ij}$ as well as considering the effects of a potential failure of service candidate $s_j$, here, the indexation $R$ symbolizes the re-planning necessary after a potential failure of the service candidate $s_j$</td>
</tr>
<tr>
<td>$\Phi_n$</td>
<td>An aggregation function $\Phi_n$ for each QoS attribute $n$ in order to aggregate the QoS values $q_{ij}^n$ of each service candidate $x_j$, included by the considered service composition</td>
</tr>
</tbody>
</table>

Table 3. Notation

Given that notation, our optimization problem is specified as follows (according to [9]):

$$\arg\max_{SC} \sum_{s_j \in S_i} E^R(U(s_j)) * x_j \forall s_i \in SC$$

Subject to:

$$\Phi_n(q_{ij}^n * x_j, \forall i, j) \leq Q^n \forall n = 1, \ldots, N$$

$$\sum_{s_j \in S_i} x_j \geq 1 \text{ with } x_j \in [0,1], \forall s_i \in SC$$

(2)

For each service classes $S_i$ of the considered service composition $SC$, the optimization problem must select at least one service candidate $s_j$ (where $x_j$ is set to 1 if the service candidate $s_j$ is selected for class $S_i$ and 0 otherwise) so that the expected utility $E^R(U(s_j))$ for all selected service candidates $s_j$ (with $s_j \in S_i$) is maximized subject to the given end-to-end QoS requirements $Q$. This optimization has to be done for all service composition $SC$ where the argument of the maximum selects the service composition for which the utility attains its maximum value. Based on the optimization problem in (2) the challenge for our approach is to determine the expected utility $E^R(U(s_j))$ that considers the effects resulting from the potential failure of the service candidate $s_j$. Considering potential failures means that in a first step a service candidate $s_j$ is performed only with a probability of $(1-p_j)$ where $p_j$ represents the probability of failure. Considering potential failures is easy to compute based on the probability that the service candidate $s_j$ is available (cf. [6, 7, 9, 15]). Thus, the utility $U(s_j)$ – in case the service candidate $s_j$ is available – is weighted with the factor $(1-p_j)$ and taken into account when determining the expected utility $E^R(U(s_j))$ (cf. term (3)). In a second step, it is necessary to reflect the options that may be available in case of a re-planning, i.e. if a service candidate $s_j$ fails with a probability $p_j$. The following options are conceivable:
1. Select the next best service candidate $s_{ij}'$ from service class $S_i$ according to its expected utility and which is feasible subject to the end-to-end QoS requirements.

2. Select the next best alternative service composition according to its expected utility avoiding the service class $S_i$ and which is feasible subject to the end-to-end QoS requirements. The selection of an alternative service composition may be reasonable, for instance, if all other feasible service candidate $s_{ij}$ of the same service class $S_i$ as the faulty service candidate $s_{ij}$ have worse QoS values.

3. Termination of the process execution, if no alternative and feasible service composition exists that allows continuing the interrupted service composition execution subject to the end-to-end QoS requirements.

The utility for each option i. to iii. needs to be calculated in order to evaluate which of the options i. till iii. creates the highest expected utility. For a better understanding, this is illustrated with the help of Figure 4 referring to the running example above.

For instance, focusing on the service candidate $s_{11}$ (red ellipse) we calculate the corresponding expected utility for each option i. till iii. Option i. is illustrated with the blue line, meaning that in the service class $S_1$ the next best service candidate $s_{12}$ is taken into account regarding its expected utility value (in our example there is only one alternative service candidate). Note that due to the substitution of $s_{11}$ through $s_{12}$ the optimal service candidates for the upcoming service classes have also changed (cf. initial service composition $s_{11}$-$s_{21}$-$s_{31}$-$s_{41}$ vs. re-planned service composition $s_{12}$-$s_{21}$-$s_{32}$-$s_{61}$). Option ii. is illustrated with the orange line, meaning that the next best service composition $s_{12}$-$s_{22}$-$s_{62}$ avoiding the service class $S_1$ is considered.

![Diagram](image)

**Fig. 2.** Re-planning options i. to ii. in case of a potential failure of service candidate $s_{11}$

ad 1. Considering option i., the corresponding expected utility for each service candidate within the same service class $S_i$ of the faulty service candidate $s_{ij}$ needs to be calculated. The service candidate $s_{ij}'$ which creates the highest expected utility among all other service candidates is selected. In doing so it is necessary to consider that due to the substitution of service candidate $s_{ij}$ with another service candidate of service class $S_i$, the optimal service candidates of upcoming service classes within the corresponding service composition can change (cf. example). In particular this is caused by a different load of the requirements by service candidate $s_{ij}$ compared to other service candidates of service class $S_i$. Moreover, time delays till the failure is noticed and compensated (time-to-repair) needs to be considered within option i). For instance, the expected value of the time interval until a failure of a service candidate
is noticed can be defined as $t_{ij}/2$ (with $t_{ij}$ representing the response time of a service candidate $s_{ij}$). In this case, the time interval has to be added to the end-to-end response time of the re-planned service composition.

ad 2. Here, the corresponding expected utility of an alternative feasible service composition needs to be calculated. Thereby, the service candidates of the current service composition that have already been executed before the service candidate $s_{ij}$ fails need to be considered. Losses will occur if these services that have already been executed are not considered in the alternative service composition. The calculation of losses is handled as follows: First, the QoS values of the service candidates that have already been executed are aggregated and considered within the end-to-end QoS values. Then, the QoS values of the already executed service candidates are changed as follows: The values of the response time and the costs are set to zero. This procedure prevents a double calculation of the QoS values if the service candidates are to be used in the alternative service composition again. If the already executed service candidates are not part of the alternative service composition, their response time and the costs are already considered within the end-to-end QoS values, thus they constitute losses.

Example: Given that after a re-planning of the service composition $s_{11}$-$s_{21}$-$s_{31}$-$s_{42}$ (failure of service candidate $s_{42}$), the service composition $s_{11}$-$s_{21}$-$s_{61}$ (option ii.) is selected as the next best one. Then, the QoS values of the service candidate $s_{ij}$ create losses, as it is not selected for the alternative service composition again. In addition, the time delay till a failure of a service candidate $s_{ij}$ is noticed and compensated needs also to be considered within option ii. as well (see option i.).

ad 3. Here, the corresponding (penal-)cost for a general termination of the service composition needs to be appointed (e.g. data loss as a result of a process termination caused by a service failure [30]). Therefore, the (penal-) costs will be added to the end-to-end costs of the interrupted service composition and the corresponding (dis)utility is calculated.

The expected utility of each option i. to iii. can be determined under the consideration of already calculated feasible service compositions which avoids multiple calculations. After the calculation of the options i. to iii., the option which creates the highest expected utility, which we note as $E^R(\ldots)$, will be selected.

More precisely, in case of the service candidate $s_{ij}$ this expected utility $E^R(\ldots)$ must be multiplied with the probability of failure $p_{ij}$ whereas the utility $U(s_{ij})^2$ (in case the service candidate $s_{ij}$ does not fail) must be multiplied with $(1-p_{ij})$. Hence, the expected utility $E^R_{ij}(\ldots)$ considering a potential failure of the service candidate $s_{ij}$ is at first given by:

$$E^R_{ij}(\ldots)= U_{ij}(\ldots) (1-p_{ij}) + E^R(\ldots) p_{ij}$$ (3)

1 Supposing a uniform distribution with a time interval of 0 to $t_{ij}$ till the failure of the service candidate is noticed, the expected value of the time interval is given by $t_{ij}/2$.

2 The utility $U(s_{ij})$ constitutes not an expected utility, as it is known under certainty which utility value will be realized if the service candidate $s_{ij}$ does not fail.
Moreover, in case the option i. is chosen, meaning that the previous service candidate \( s_j \) will be substituted with the service candidate \( s_{ji} \): then, \( U^{R*}(\ldots) \) denotes the utility for service candidate \( s_j \) that will be realized in case service candidate \( s_j \) fails but service candidate \( s_{ji} \) not. Thus, this utility has to be multiplied with \( p_{ij} \) and \((1-p_{ij})\). However, the potential failure of service candidate \( s_{ji} \) needs to be considered in a next step as well. Hence, a reanalysis of the named options i. to iii. has to be done, but now with the difference that the potential failure of the service candidate \( s_{ji} \) will be considered. Again, the option i. to iii. that creates the highest expected utility, which we note as \( E^{R*}(\ldots) \), will be selected. Hence, the expected utility considering a potential failure of the service candidate \( s_{ji} \) is at first given by:

\[
E_{ij}^R(\ldots) = U_{ij}(\ldots) \cdot (1 - p_{ij}) + U^{R*}(\ldots) \cdot p_{ij} \cdot (1 - p_{ij}) + E^{R*}(\ldots) \cdot p_{ij} \cdot p_{ij}^R
\]

The term (4) is the iterative extension of term (3) by the consideration of a potential failure of service candidate \( s_{ji} \). These extension can be done till option iii. is triggered, meaning the process execution is terminated. For option ii., the calculations for the term (3) can be iteratively extended in the same way as it was done for the calculations in option i. shown in term (4). Obviously, these extensions for option i. and ii. terminate, as both the number of alternatives of next best service candidates resp. and the number of feasible service compositions subject to the end-to-end QoS requirements are limited. Based on the expected utility \( E_{ij}^R(\ldots) \) in term (4) each service candidate and thus each sequential service composition can be evaluated to select the optimal QoS-aware service composition considering the effects of potential service failure. In detail:

1. The effects of potential service failure can be considered within the ex-ante optimization by calculating the expected utility (cf. terms 3, 4 resp. options i. till iii.). In doing so, our approach is able to determine how a service composition will perform in case of a potential service failure, even before the real execution.
2. Furthermore, the effects of a potential re-planning on the end-to-end QoS values of an alternative service composition can be calculated and therefore its feasibility can be determined (cf. options i. and ii.). In that sense, the waste of resources like time and money can be reduced or prevented.
3. Moreover, the temporal delays till a failure of a service candidate is noticed and compensated in case of a potential re-planning are now considered within the ex-ante optimization.
4. Finally, losses that can occur due to a potential re-planning are considered within the ex-ante optimization. In doing so, a waste of resources like time and money can be prevented.

5 Evaluation of the Novel Approach

As defined in the introduction, the purpose of this paper is to examine if an approach for the QoS-aware service selection considering the effects of potential service fail-
ures can lead to a better decision. Therefore, we analytically examine at first due to the length restrictions of the paper – the easiest case of a service selection problem. However, if it is possible for this simple case to show that it is better to consider potential service failures, then it is obvious that this is particularly reasonable in case more re-planning alternatives exist. Secondly, we examine the applicability of the approach using the running example and compare our results with the one of existing approaches.

5.1 Analytical Evaluation of the Novel Approach

For our analytical evaluation we consider the easiest case of a selection problem:

Without any loss of generality, we consider a service class $S_i$ with two different service candidates $s_{ij}$ and $s_{ij}'$, each characterized by two QoS attributes, specifically, the response time of both service candidates, represented by $t_{ij}$ and $t_{ij}'$; and their probabilities of failure, represented by $p_{ij}$ and $p_{ij}'$. Given that both service candidates meet the QoS requirements (i.e. they are feasible; otherwise the selection problem would be extremely simple) and any utility function of an existing approach would prefer service candidate $s_{ij}$ against service candidate $s_{ij}'$ essentially because it holds $t_{ij} < t_{ij}'$. Therefore, the service candidate $s_{ij}$ is selected as first choice of the service class $S_i$. In case a service candidate fails, the time interval until a failure of a service candidate is noticed and compensated is supposed as $t_{ij}/2$ resp. $t_{ij}'/2$.

Given that service selection problem, the expected value $E_{ij}(\ldots)$ focusing on the response time $t_{ij}$ of the service candidate $s_{ij}$ that fails with a probability of $p_{ij}$ can be calculated as follows.

$$E_{ij}(\ldots)=t_{ij}(1-p_{ij})+(t_{ij}+t_{ij}'/2)*p_{ij}(1-p_{ij})+T*p_{ij}*p_{ij}'$$

The expected value $E_{ij}(\ldots)$ has three terms of the sum, whereas the case that the service candidate $s_{ij}$ will not fail is described by the first term of the sum. The second term of the sum describes the case that the service candidate $s_{ij}$ will fail and a re-planning on the service candidate $s_{ij}'$ with a certain delay $t_{ij}/2$ is necessary. The third term of the sum gives the time period $T$ (“penal time”), which describes the time interval till the process can be restarted after a failure of both service candidates.

The expected value $E_{ij}(\ldots)$ contains not only the QoS attribute response time $t_{ij}$ of the service candidate $s_{ij}$. As the service candidate $s_{ij}$ could fail with a probability $p_{ij}$, the QoS attribute response time $t_{ij}$ of the alternative service candidate $s_{ij}'$ as well as the time delay $t_{ij}/2$ till the failure of the service candidate $s_{ij}$ is noticed and compensated through the invocation of service candidate $s_{ij}'$ also needs to be considered.

Similar to the service candidate $s_{ij}$, the expected value $E_{ij}(\ldots)$ for the service candidate $s_{ij}'$ can be defined as follows:
The service candidate \( s_{ij} \) is selected as first choice of service class \( S_i \), as \( t_{ij} < t_{ij'} \) holds. But, when we include the effects of potential service failures however, the expected values \( E^R_{ij}(...) \) resp. \( E^R_{ij'}(...) \) are crucial. Hence, the condition \( E^R_{ij}(...) < E^R_{ij'}(...) \) (which results to \( U^R_{ij} > U^R_{ij'} \) as the response time has to be minimized) needs to be analyzed to decide whether the service candidate \( s_{ij} \) is still selected as first choice of service class \( S_i \). Specifically, we have to prove a contradiction to \( E^R_{ij}(...) < E^R_{ij'}(...) \) (i.e. \( E^R_{ij}(...) \geq E^R_{ij'}(...) \)) although it holds \( t_{ij} < t_{ij'} \). In the following we show this contradiction (here \( t_{ij} < t_{ij'} \) is mathematically substituted by \( t_{ij} = t_{ij} + \Delta \), i.e. \( \Delta \) represents the difference between \( t_{ij} \) and \( t_{ij'} \)):

\[
E^R_{ij}(...) > E^R_{ij'}(...) \iff t_{ij} \cdot (1 - p_{ij}) + \left(t_{ij} + \frac{t_{ij}'}{2}\right) \cdot p_{ij} \cdot (1 - p_{ij}) + T^* p_{ij} \cdot p_{ij'} \\
> t_{ij} \cdot (1 - p_{ij}) + \left(t_{ij} + \frac{t_{ij}'}{2}\right) \cdot p_{ij} \cdot (1 - p_{ij}) + T^* p_{ij} \cdot p_{ij'} \\
\Rightarrow \frac{1}{2} t_{ij} \cdot p_{ij} + \Delta p_{ij} - \Delta^{\#} p_{ij} > \Delta + \frac{1}{2} t_{ij} \cdot p_{ij} - \frac{\Delta^{\#}}{2} p_{ij'} \\
\Rightarrow \Delta \left(2p_{ij} - p_{ij} \cdot p_{ij'} + p_{ij'} - 2\right) > t_{ij} \cdot p_{ij'} - t_{ij} \cdot p_{ij} \\
\Rightarrow \Delta < \frac{t_{ij}(p_{ij'} - p_{ij})}{2p_{ij'} - p_{ij} + p_{ij'} - 2} \text{with } (2p_{ij} - p_{ij} \cdot p_{ij'} + p_{ij'} - 2) < 0
\]

The term (7) solved for the difference \( \Delta \) show that there are cases, where the selection of service candidate \( s_{ij'} \) instead of service candidate \( s_{ij} \) is beneficial. The condition applies if the difference \( \Delta \) is smaller than the quotient on the right considering the response time \( t_{ij} \) as well as the failure probabilities \( p_{ij} \) and \( p_{ij'} \). Thereby, the numerator shows the response time \( t_{ij} \) weighted with the difference of the probabilities \( p_{ij} \) and \( p_{ij'} \). This means, the more the failure probabilities of the two service candidates are far apart from each other, ceteris paribus the greater the value of the numerator and the value of the whole quotient will be. The service candidate \( s_{ij'} \) will be beneficial, as the value of the quotient rises above the difference \( \Delta \).

5.2 Demonstration of the Applicability of the Novel Approach

The goal of this second evaluation step is to examine the applicability of the approach. We intentionally use the running example presented by [9] in order to address transparency and reproducibility. In this example the optimal service composition that was determined is \( S_{11,S_{32}} \cdot S_{32} \cdot S_{42} \) with a response time of 560, costs of 225, availability of 0.895. Remember, this service composition is the result of any existing analytical selection approach (not only the one proposed by [9]).
In contrast, applying our approach, potential service failures are taken into account when solving the optimization problem before the actual process execution. Here, we consider the utility function defined in terms of potential service failures before the actual process execution. Here, we consider the utility function defined in terms of potential service failures before the actual process execution. Furthermore to realize the approach, for each feasible service composition, the paths that were terminated a) due to a violation of the requirements or b) due to the fact that no alternative service composition exists anymore, as well as the corresponding path probabilities were stored. This was done in order to get an insight of the robustness of different service compositions. For a termination of the service composition we set the (penal-)costs to 1,000 which prevent a premature termination of the considered service composition as long as at least one feasible service composition exists. After determining the effects of potential service failures, the results show that now the optimal service composition is $s_{11}$-$s_{21}$-$s_{61}$. As Table 4 (see appendix) demonstrates, the service composition $s_{11}$-$s_{21}$-$s_{32}$-$s_{42}$ which is supposed to be optimal by existing approaches is only at the fifth position when considering the effects of potential service failures. Specifically the service compositions $s_{11}$-$s_{21}$-$s_{61}$, $s_{11}$-$s_{22}$-$s_{61}$, $s_{11}$-$s_{23}$-$s_{61}$ and $s_{11}$-$s_{23}$-$s_{42}$ have a higher expected utility than the service composition $s_{11}$-$s_{21}$-$s_{32}$-$s_{42}$. One of the reasons why the service composition $s_{11}$-$s_{21}$-$s_{32}$-$s_{42}$ is worse compared to the other service compositions can be found in its robustness. Here, the service composition $s_{11}$-$s_{21}$-$s_{32}$-$s_{42}$ has with a probability of a premature termination of 3.47% (due to service failures) a much lower robustness compared to the service composition $s_{11}$-$s_{21}$-$s_{61}$ with a probability of a premature termination of just 0.89%. The result of these terminations is a huge waste of resources. Here, our approach can help to save resources by considering ex-ante the effects of potential service failures.

The next section contains the conclusion, discusses important limitations of our approach and determines possible starting points for future research.

5.3 Conclusion, Limitations and Future Research

In this paper, we propose an approach for the QoS-aware service selection that considers the effects of potential service failures before starting the process execution. The results provide some evidence for the research questions presented in the introduction. Precisely, an approach considering the effects of potential service failures can lead to a methodically well-founded decision making about the optimal QoS-aware service selection regarding the expected utility. The reason is the consideration of the effects of a re-planning, the consideration of losses as well as the consideration of the time interval till the service failure is noticed and compensated already within the ex-ante optimization.

Due to the dynamic nature of the Internet, such an approach is especially relevant since it is possible that values of the QoS attributes will change during the execution of a process (e.g. see also the availability statistics at [28], [29], [31]). Moreover, in many scenarios, an ex-ante planned service candidate is no longer available. Hence, neglecting the effects of service failures can lead to a loss of resources (e.g. money, time) during the runtime of a process. Here, our approach is thought to contribute to
these challenges. The evaluation was done on the one hand by mathematical methods showing that our approach can lead to better results. On the other hand, we use an existing example provided in the literature. With the latter we demonstrate that considering service failures in an ex-ante QoS-aware service selection leads to a better utility value, as the results of existing approaches. To compute this example as well as other cases, the approach has been prototypically realized. Summarizing, we evaluated the approach with respect to its applicability and the practical utility provided.

Some limitations have to be discussed which are the starting points for future research: In the paper, an evaluation and demonstration of the strength and benefit of our approach is provided. Nevertheless, future work is needed and intended supporting the further assessment and justification in different real-use situations. Moreover, the expected utility is a valid decision criterion if the process and thus the service composition are executed many times (“law of large numbers”). This has to be taken into account, when applying the approach. A further goal for research is how existing heuristics (e.g. [3]) can be combined with our ideas to consider expected utilities, losses etc. In the example above but also in larger cases with many service classes and service candidates the runtime of the optimization using our approach is low. Still, in very large cases heuristics may be useful. However, the goal of this paper is not to provide a runtime optimized approach or a heuristic. It is rather about the question, how the effects resulting from potential failures of services can be considered in a well-founded manner. The approach presented here forms an appropriate fundament for this as well as for the aforementioned enhancements and thus serves as a suitable basis for further research.

Acknowledgements

The research was funded by the Austrian Science Fund (FWF): P 23546-G11.

References

30. AWS Team: Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US East Region, http://aws.amazon.com/de/message/65648/
# Appendix

## Table 1. Relevant selection and re-planning approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Authors</th>
<th>Considered QoS-Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Selection approaches for an optimal QoS-aware service composition</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Analytical Approaches</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integer Programming</td>
<td>[6]</td>
<td>price, duration, reputation, availability</td>
</tr>
<tr>
<td>Mixed Integer Programming</td>
<td>[22]</td>
<td>response time, reliability, availability, price</td>
</tr>
<tr>
<td>BBLP/MCSP</td>
<td>[9]</td>
<td>response time, price, availability</td>
</tr>
<tr>
<td>Branch and Bound</td>
<td>[23]</td>
<td>price, duration, reliability, availability</td>
</tr>
<tr>
<td>Genetic algorithm</td>
<td>[11, 15]</td>
<td>duration, costs, availability, reliability</td>
</tr>
<tr>
<td><strong>Heuristics</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ant Colony Algorithm + Genetic Algorithm</td>
<td>[26]</td>
<td>time, cost, reliability, availability, reputation</td>
</tr>
<tr>
<td>Dynamic Programming</td>
<td>[7]</td>
<td>response time, costs, availability, reliability</td>
</tr>
<tr>
<td><strong>Re-planning approaches</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Analytical approaches</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service exchange</td>
<td>[13]</td>
<td>duration, costs, reliability, availability</td>
</tr>
<tr>
<td>Integer Programming</td>
<td>[6]</td>
<td>price, duration, reputation, availability</td>
</tr>
<tr>
<td><strong>Heuristics</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic algorithm</td>
<td>[15]</td>
<td>duration, costs, availability, reliability</td>
</tr>
</tbody>
</table>

---

**Fig. 1.** Service classes and service candidate in the example ([9])
Table 2. Service candidates and their values of the three considered QoS attributes

<table>
<thead>
<tr>
<th>Service class</th>
<th>Service candidate</th>
<th>Response time</th>
<th>Cost</th>
<th>Probability of availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>$s_{11}$</td>
<td>100</td>
<td>50</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>$s_{12}$</td>
<td>180</td>
<td>60</td>
<td>0.92</td>
</tr>
<tr>
<td>$S_2$</td>
<td>$s_{21}$</td>
<td>200</td>
<td>50</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>$s_{22}$</td>
<td>160</td>
<td>100</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>$s_{23}$</td>
<td>180</td>
<td>80</td>
<td>0.97</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$s_{31}$</td>
<td>150</td>
<td>100</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>$s_{32}$</td>
<td>120</td>
<td>85</td>
<td>0.99</td>
</tr>
<tr>
<td>$S_4$</td>
<td>$s_{41}$</td>
<td>130</td>
<td>60</td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>$s_{42}$</td>
<td>140</td>
<td>40</td>
<td>0.97</td>
</tr>
<tr>
<td>$S_5$</td>
<td>$s_{51}$</td>
<td>200</td>
<td>150</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>$s_{52}$</td>
<td>170</td>
<td>100</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>$s_{53}$</td>
<td>180</td>
<td>130</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Table 4. Comparison of the results

<table>
<thead>
<tr>
<th>Service composition</th>
<th>Results based on existing approaches</th>
<th>Results based on the novel approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Response time</td>
<td>Costs</td>
</tr>
<tr>
<td>$s_{11}$-$s_{21}$-$s_{32}$-$s_{42}$</td>
<td>560</td>
<td>225</td>
</tr>
<tr>
<td>$s_{11}$-$s_{21}$-$s_{32}$-$s_{41}$</td>
<td>550</td>
<td>245</td>
</tr>
<tr>
<td>$s_{11}$-$s_{31}$-$s_{61}$</td>
<td>470</td>
<td>200</td>
</tr>
<tr>
<td>$s_{11}$-$s_{33}$-$s_{61}$</td>
<td>450</td>
<td>230</td>
</tr>
<tr>
<td>$s_{11}$-$s_{22}$-$s_{61}$</td>
<td>430</td>
<td>250</td>
</tr>
<tr>
<td>$s_{11}$-$s_{21}$-$s_{62}$</td>
<td>480</td>
<td>230</td>
</tr>
</tbody>
</table>