QoS-aware service selection considering potential service failures

Bernd Heinrich[1] and Lars Lewerenz[1]

[1] 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

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

11th International Conference on Wirtschaftsinformatik,
27th February – 01st March 2013, Leipzig, Germany

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:

❶ 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.

❷ As discussed in ❶ a service composition needs to 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.

❸ 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.

❹ 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 ❶-❹ 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 ❶-❹) 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 ❶-❹. 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.

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 ❶-❹). 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. 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.

Table 1. Relevant selection and re-planning approaches

	Approach	Authors	Considered QoS-Attributes
Selection approaches for an optimal QoS-aware service composition			
Analytical Approaches	Integer Programming	[6]	price, duration, reputation, availability
		[22]	response time, reliability, availability, price
	Mixed Integer Programming	[10]	price, reputation, execution time, availability, data quality
	BBLP/ MCSP	[9]	response time, price, availability
	Branch and Bound	[23]	price, duration, reliability, availability
Heuristics	Genetic algorithm	[11, 15]	duration, costs, availability, reliability
		[24]	response time, price, reliability, availability
		[25]	availability, reputation, cost, time
	H1_Relax_IP; H2_SWAP;H3_SIM ANNEAL	[3]	response time, availability
	WS_HEU/ MCSP-K	[9]	response time, price, availability
	Ant Colony Algorithm + Genetic Algorithm	[26]	time, cost, reliability, availability, reputation
	Dynamic Programming	[7]	response time, costs, availability, reliability
Re-planning approaches			
Analytical Approaches	Service exchange	[13]	duration, costs, reliability, availability
	Integer Programming	[6]	price, duration, reputation, availability
Heuristics	H1_Relax_IP	[14]	response time, availability
	Genetic algorithm	[15]	duration, costs, availability, reliability

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 composition 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) 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 ❶-❹) 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.

Running example (cf. also [9])

In the running example the following service classes S_1 to S_6 each with different service candidates (e.g. s_{11} and s_{12} for class S_1) are given.

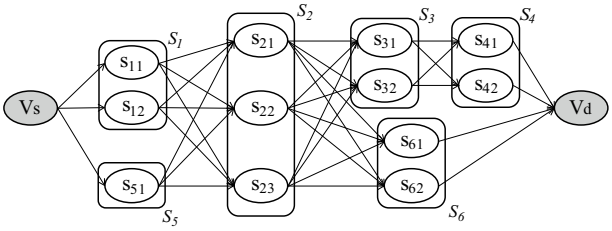

Figure 1. Service classes and service candidate in the example ([9])

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 *S1* to *S6* with their QoS values are given in Table 2.

Table 2. Service candidates and their values of the three considered QoS attributes

Service class	Service candidate	Response time	Cost	Probability of availability
S_1	s_{11}	100	50	0.95
	s_{12}	180	60	0.92
S_2	s_{21}	200	50	0.98
	s_{22}	160	100	0.95
	s_{23}	180	80	0.97
S_3	s_{31}	150	100	0.94
	s_{32}	120	85	0.99
S_4	s_{41}	130	60	0.93
	s_{42}	140	40	0.97
S_5	s_{51}	200	150	0.96
S_6	s_{61}	170	100	0.97
	s_{62}	180	130	0.99

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:

$$U(s_{ij}) = \sum_{\alpha=1}^{x} w_\alpha * \left(\frac{q_{ij}^{\alpha} - \mu^{\alpha}}{\sigma^{\alpha}}\right) + \sum_{\beta=1}^{y} w_\beta * \left(1 - \frac{q_{ij}^{\beta} - \mu^{\beta}}{\sigma^{\beta}}\right) \quad (1)$$

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}. μ and σ 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 (w_α, w_β) for each QoS attribute, where ($0<w_\alpha, w_\beta<1$) and $\sum_{\alpha=1}^{x} w_\alpha + \sum_{\beta=1}^{y} w_\beta = 1$ 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). 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_{21}-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 ❶-❹) are left unconsidered so far. More precisely: Let us suppose two invoked service candidates s_{11} and s_{21} 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_{12} is available but with really worse QoS values. Opposed to that, if service candidate s_{21} fails, a substitute service candidate s_{22} is also available that has nearly the same QoS values as service candidate s_{21}. In other words, although both service candidates s_{11} and s_{12} 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_{21}-s_{31}-s_{41} and s_{12}-s_{22}-s_{32}-s_{42}. Furthermore, we suppose that the service composition s_{11}-s_{21}-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_{21}-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_{21}-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_{21}-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.

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 ❶-❹).

For the setup of the approach, we use the following notation (according to [6, 9]).

Table 3. Notation

S_i	Service class S_i that includes all services candidates s_{ij} that implements the action i (with i=1 to I) of the considered service composition SC
s_{ij}	Service candidate s_{ij} (with j=1 to J_i, $\forall i$) for the service class S_i with x_{ij} = 1 if the service candidate s_{ij} is selected for class S_i and x_{ij} = 0 otherwise
q_{ij}	QoS vector for each service candidate s_{ij}, q_{ij} =$[q_{ij}^1, \dots, q_{ij}^N]$ including the single value for each QoS attribute n with n = 1 to N (excluding the QoS attribute 'availability')
Q	Global (end-to-end) QoS requirements vector Q=$[Q^1, \dots, Q^N]$ for a service composition including the single requirements Q^n for each QoS attribute n with n = 1 to N
p_{ij}	Probability p_{ij} of failure of a certain service candidate s_{ij} (representing the QoS attribute 'availability')
U	Utility function for a risk neutral decision maker to calculate the utility $U(s_{ij})$ for a single service candidate s_{ij} based on its QoS vector q_{ij}
$E^R(U(s_{ij}))$	Expected utility for a single service candidate s_{ij} based on its QoS vector q_{ij} as well as considering the effects of a potential failure of service candidate s_{ij} (here, the indexation R symbolize the re-planning necessary after a potential failure of the service candidate s_{ij})
Φ_n	An aggregation function Φ_n for each QoS attribute n in order to aggregate the QoS values q_{ij}^n of each service candidate s_{ij} included by the considered service composition

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

$$\arg\max_{SC} \sum_{s_{ij} \in S_i} E^R(U(s_{ij})) \cdot x_{ij} \quad \forall S_i \in SC$$

$$Subject\ to: \Phi_n(q_{ij}^n \cdot x_{ij}, \forall i, j) \leq Q^n \qquad \forall n = 1, .., N$$

$$\sum_{s_{ij} \in S_i} x_{ij} \geq 1 \ with\ x_{ij} \in \{0,1\};\ \forall S_i \in SC \tag{2}$$

For each service classes S_i of the considered service composition SC, the optimization problem must select at least one service candidate s_{ij} (where x_{ij} is set to 1 if the service candidate s_{ij} is selected for class S_i and 0 otherwise) so that the expected utility $E^R(U(s_{ij}))$ for all selected service candidates s_{ij} (with $s_{ij} \in S_i$) 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 de-

termine the expected utility $E^R(U(s_{ij}))$ that considers the effects resulting from the potential failure of the service candidate s_{ij}. Considering potential failures means that in a first step a service candidate s_{ij} is performed only with a probability of (1-p_{ij}) where p_{ij} represents the probability of failure (this probability is easy to compute based on the probability that the service candidate s_{ij} is available (cf. [6, 7, 9, 15])). Thus, the utility $U(s_{ij})$ – in case the service candidate s_{ij} is available – is weighted with the factor (1-p_{ij}) and taken into account when determining the expected utility $E^R(U(s_{ij}))$ (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_{ij} fails with a probability p_{ij}. The following options are conceivable:

i. 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.
ii. 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.
iii. 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_{61}). Option ii. is illustrated with the orange line, meaning that the next best service composition s_{51}-s_{23}-s_{62} avoiding the service class S_1 is considered.

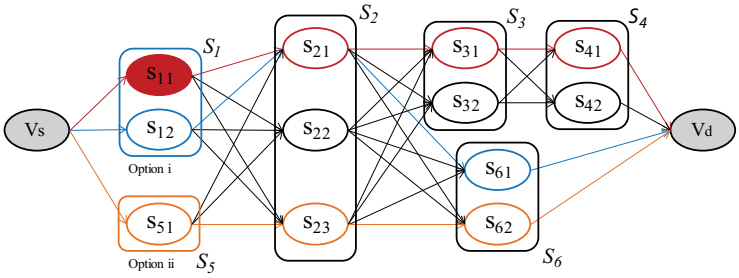

Figure 2. Re-planning options i. to ii. in case of a potential failure of service candidate s_{11}

ad i. 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 $t_{ij}/2$[1] (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 ii. 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_{31} 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 notices and compensated needs also to be considered within option ii. as well (see option i.).

ad iii. 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 high-

[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 $\frac{t_{ij}}{2}$.

est 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 $E_{ij}^{R}(\ldots)$ considering a potential failure of the service candidate s_{ij} is at first given by:

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

Moreover, in case the option i. is chosen, meaning that the previous service candidate s_{ij} will be substituted with the service candidate $s_{ij'}$: then, $U^{R*}(\ldots)$ denotes the utility for service candidate $s_{ij'}$ that will be realized in case service candidate s_{ij} fails but service candidate $s_{ij'}$ not. Thus, this utility has to be multiplied with p_{ij} and $(1 - p_{ij'})$. However, the potential failure of service candidate $s_{ij'}$ 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 $s_{ij'}$ will be considered. Again, the option i. to iii. that creates the highest expected utility, which we note as $E^{RR*}(\ldots)$, will be selected. Hence, the expected utility considering a potential failure of the service candidate $s_{ij'}$ is at first given by:

$$E_{ij}^{R}(\ldots) = U_{ij}(\ldots) * (1 - p_{ij}) + U^{R*}(\ldots) * p_{ij} * (1 - p_{ij'}) + E^{RR*}(\ldots) * p_{ij} * p_{ij'} \quad (4)$$

The term (4) is the iterative extension of term (3) by the consideration of a potential failure of service candidate $s_{ij'}$. 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:

❶ 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.

❷ 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.

❸ 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-

[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.

ante optimization.

❹ 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.

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 failures 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.

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 $E_{ij}^{R}(\dots)$ 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.

$$E_{ij}^{R}(\dots) = t_{ij} * (1 - p_{ij}) + \left(t_{ij'} + \frac{t_{ij}}{2}\right) * p_{ij} * (1 - p_{ij'}) + T * p_{ij} * p_{ij'} \quad (5)$$

The expected value $E_{ij}^{R}(\dots)$ 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 $E_{ij}^{R}(\dots)$ 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 $E^R_{ij'}(\dots)$ for the service candidate $s_{ij'}$ can be defined as follows:

$$E^R_{ij'}(\dots) = t_{ij'} * (1 - p_{ij'}) + \left(t_{ij} + \frac{t_{ij'}}{2}\right) * p_{ij'} * (1 - p_{ij}) + T * p_{ij} * p_{ij'} \quad (6)$$

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 $E^R_{ij}(\dots)$ resp. $E^R_{ij'}(\dots)$ are crucial. Hence, the condition $E^R_{ij}(\dots) < E^R_{ij'}(\dots)$ (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 $E^R_{ij}(\dots) < E^R_{ij'}(\dots)$ (i.e. $E^R_{ij}(\dots) \geq E^R_{ij'}(\dots)$) 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. Δ represents the difference between $t_{ij'}$ and t_{ij}):

$$\begin{aligned} E^R_{ij}(\dots) > E^R_{ij'}(\dots) &\Leftrightarrow t_{ij} * (1 - p_{ij}) + \left(t_{ij'} + \frac{t_{ij}}{2}\right) * p_{ij} * (1 - p_{ij'}) + T * p_{ij} * p_{ij'} \\ &> t_{ij'} * (1 - p_{ij'}) + \left(t_{ij} + \frac{t_{ij'}}{2}\right) * p_{ij'} * (1 - p_{ij}) + T * p_{ij} * p_{ij'} \\ &\Leftrightarrow \frac{1}{2} t_{ij} * p_{ij} + \Delta p_{ij} - \frac{\Delta}{2} p_{ij} * p_{ij'} > \Delta + \frac{1}{2} t_{ij} * p'_{ij} - \frac{\Delta}{2} * p_{ij'} \\ &\Leftrightarrow \Delta(2p_{ij} - p_{ij} * p_{ij'} + p_{ij'} - 2) > t_{ij} * p_{ij'} - t_{ij} * p_{ij} \\ \Leftrightarrow \Delta < \frac{t_{ij(}p_{ij'} - p_{ij})}{2p_{ij} - p_{ij} * p_{ij'} + p_{ij'} - 2} &\; with\; (2p_{ij} - p_{ij} * p_{ij'} + p_{ij'} - 2) < 0 \end{aligned} \quad (7)$$

The term (7) solved for the difference Δ 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 Δ 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 Δ.

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_{21}-s_{32}-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 3 and 4 in order to be able to calculate the effects of potential failures of a service candidate. Therefore, for every service candidate the utility (cf. option i. till iii.) considering a potential re-planning was calculated. 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 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_{21}-s_{62} 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.

Table 4. Comparison of the results

Service composition	**Results based on existing approaches**			**Results based on the novel approach**			
	Response time	Costs	Order	Expected response time	Expected costs	Expected utility value	Order
s_{11}-s_{21}-s_{32}-s_{42}	560	225	1	561,29	258,80	-5,5084	5
s_{11}-s_{21}-s_{32}-s_{41}	550	245	2	551,99	317,96	-6,1520	6
s_{11}-s_{21}-s_{61}	470	200	3	480,40	210,74	-5,0936	1
s_{11}-s_{23}-s_{61}	450	230	4	459,96	264,52	-5,4808	3
s_{11}-s_{22}-s_{61}	430	250	5	444,51	278,15	-5,3763	2
s_{11}-s_{21}-s_{62}	480	230	6	487,47	234,44	-5,4985	4

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

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

1. TechTarget and Forrest Research: State of SOA 2010: Executive Summary, http://cdn.ttgtmedia.com/searchSOA/downloads/TTAG-State-of-SOA-2010-execSummary-working-523%5B1%5D.pdf
2. Melzer, I.: Service-orientierte Architekturen mit Web Services. Konzepte - Standards - Praxis Spektrum Akademischer Verlag, Heidelberg, Neckar (2010)
3. Berbner, R., Spahn, M., Repp, N., Heckmann, O., Steinmetz, R.: Heuristics for QoS-aware Web Service Composition. In: International Conference on Web Services, 2006. ICWS '06, pp. 72–82 (2006)
4. Legner, C.: Is There a Market for Web Services? In: Service-Oriented Computing-ICSOC 2007 Workshops, pp. 29–42. Springer Berlin Heidelberg (2007)
5. Nüttgens, M., Dirik, I.: Business Models of Service Oriented Information Systems - A Strategic Approach towards the Commercialization of Web Services. Wirtsch. Inform. 50, 31–38 (2008)
6. Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for Web services composition. IEEE Transactions on Software Engineering 30, 311–327 (2004)
7. Yu, T., Lin, K.-J.: Service selection algorithms for Web services with end-to-end QoS constraints. ISeB 3, 103–126 (2005)
8. Yu, T., Lin, K.-J.: Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints. In: Proceedings of the Third International Conference on Service-Oriented Computing, pp. 130-143. Springer-Verlag, Berlin, Heidelberg (2005)
9. Yu, T., Zhang, Y., Lin, K.-J.: Efficient algorithms for Web services selection with end-to-end QoS constraints. ACM Trans. Web 1 (2007)
10. Ardagna, D., Pernici, B.: Adaptive Service Composition in Flexible Processes. IEEE Transactions on Software Engineering 33, 369–384 (2007)
11. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for QoS-aware service composition based on genetic algorithms. In: Proceedings of the 2005 Conference on Genetic and Evolutionary computation, pp. 1069-1075. ACM, New York, NY, USA (2005)
12. Yan, G., Jun, N., Bin, Z., Lei, Y., Qiang, G.: Optimal Web Services Selection Using Dynamic Programming. In: Proceedings of the 11th IEEE Symposium on Computers and Communications, 2006. ISCC '06., pp. 365–370 (2006)
13. Lin, K.-J., Zhang, J., Zhai, Y., Xu, B.: The design and implementation of service process reconfiguration with end-to-end QoS constraints in SOA. Service Oriented Computing and Applications 4, 157–168 (2010)
14. Berbner, R., Spahn, M., Repp, N., Heckmann, O., Steinmetz, R.: Dynamic Replanning of Web Service Workflows. In: Digital EcoSystems and Technologies Conference, 2007. DEST '07. Inaugural IEEE-IES, pp. 211–216 (2007)
15. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: A framework for QoS-aware binding and re-binding of composite web services. Journal of Systems and Software 81, 1754–1769 (2008)
16. Shen, Y.-h., Yang, X.-h.: A self-optimizing QoS-aware service composition approach in a context sensitive environment. J. Zhejiang Univ. - Sci. C 12, 221–238 (2011)

17. Maximilien, E., Singh, M.P.: A Framework and Ontology for Dynamic Web Services Selection. IEEE Internet Computing 8, 84–93 (2004)
18. Mani, A. and Nagarajan, A.: Understanding quality of service for Web services, http://www.ibm.com/developerworks/webservices/library/ws-quality/index.html
19. Hwang, S.-Y., Wang, H., Tang, J., Srivastava, J.: A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. Information Sciences 177, 5484–5503 (2007)
20. Yu, T., Lin, K.J.: Adaptive algorithms for finding replacement services in autonomic distributed business processes. In: 2005. ISADS 2005. Proceedings Autonomous Decentralized Systems, pp. 427–434 (2005)
21. Jane Webster and Richard T. Watson: Analyzing the past to prepare for the future: Writing a literature review. MIS Q 26 (2002)
22. Huang, A.F., Lan, C.-W., Yang, S.J.: An optimal QoS-based Web service selection scheme. Information Sciences 179, 3309–3322 (2009)
23. Wan, C., Ullrich, C., Chen, L., Huang, R., Luo, J., Shi, Z.: On Solving QoS-Aware Service Selection Problem with Service Composition. In: 2008 Seventh International Conference on Grid and Cooperative Computing, pp. 467–474. IEEE (2008)
24. Maolin, T., Ai, L.: A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. Proceedings of the 2010 World Congress on Computational Intelligence (2010)
25. Jaeger, M.C., Muehl, G.: QoS-based Selection of Services: The Implementation of a Genetic Algorithm. 2007 ITG GI Conference Communication in Distributed Systems (KiVS), 1–12 (2007)
26. Yang, Z., Shang, C., Liu, Q., Zhao, C.: A Dynamic Web Services Composition Algorithm Based on the Combination of Ant Colony Algorithm and Genetic Algorithm. Journal of Computation Information Systems 6, 2617–2622 (2010)
27. Lin, K.-J., Zhang, J., Zhai, Y.: An Efficient Approach for Service Process Reconfiguration in SOA with End-to-End QoS Constraints. In: IEEE Conference on Commerce and Enterprise Computing, 2009, pp. 146–153. IEEE (2009)
28. serviceobjects.com: Performance Report. DOTS Adress Validation, http://www.serviceobjects.com/AlertSite/March2006.pdf
29. programmableweb.com: API Pulse, http://monitor.programmableweb.com
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/
31. webservice.seekda.com: Web Service Details: XigniteFinancials. response time, http://webservices.seekda.com/providers/xignite.com/XigniteFinancial