A Probabilistic Approach to Service Selection with Conditional Contracts and Usage Patterns

Adrian Klein[1,2,3], Fuyuki Ishikawa[4], and Bernhard Bauer[1]

[1] University of Augsburg, Germany
bauer@informatik.uni-augsburg.de
[2] Technical University Munich, Germany
[3] Ludwig-Maximilians-University Munich, Germany
adrian.klein@campus.lmu.de
[4] National Institute of Informatics, Tokyo, Japan
f-ishikawa@nii.ac.jp

Abstract. Service selection is a central challenge in the context of a Service Oriented Architecture. Once functionally sufficient services have been selected, a further selection based on non-functional properties (NFPs) becomes essential in meeting the user's requirements and preferences. However, current descriptions of NFPs and approaches to NFP-aware selection lack the ability to handle the variability of NFPs, that stems from the complex nature of real-world business scenarios. Therefore, we propose a probabilistic approach to service selection as follows: First, to address the inherent variability in the actual values of NFPs at runtime, we treat them as probability distributions. Then, on top of that, we tackle the variability needed in describing NFPs, by providing conditional contracts. Finally, from usage patterns, we compute user-specific expectations for such NFPs. Further, we depict a typical scenario, which serves both as a motivation for our approach, and as a basis for its evaluation.

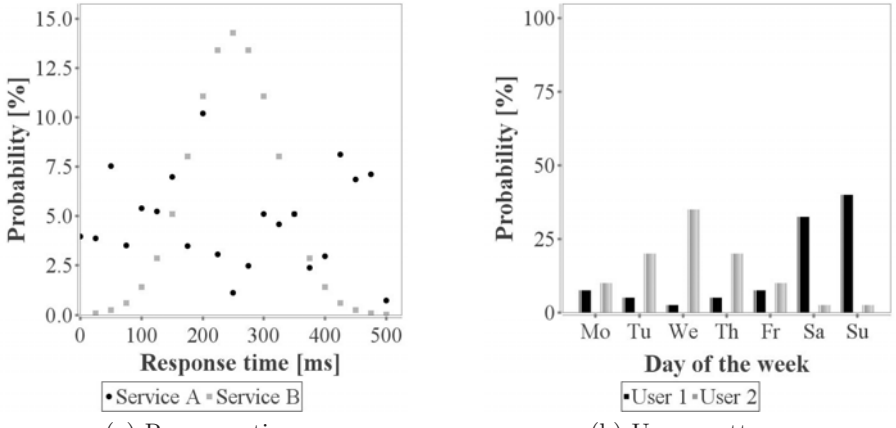
1 Introduction

A Service Oriented Architecture (SOA) lays the ground for loose coupling of interoperable services [1]. In a SOA, there are service providers, that offer services under certain conditions, and service users, that need services that fulfill certain criteria. Service contracts provide the basis necessary for the interaction between both of them, by describing the functional and non-functional properties (NFPs) [2] of a service. Service selection deals with finding the service that best matches the user's criteria and, as such, is a central challenge in the context of any SOA. The selection process is usually twofold: First, consider only the services matching the functional criteria. Then, to find the best one, rank those services according to which extent they fulfill the non-functional criteria.

While functional matching is a necessary part of any service selection, it has already been studied intensively [3]. On the other hand, matching based on NFPs such as price, response time, availability, or reliability has been drawing more

L. Baresi, C.-H. Chi, and J. Suzuki (Eds.): ICSOC-ServiceWave 2009, LNCS 5900, pp. 253–268, 2009.

and more attention, but it is far from being as well-understood. Sure, there is a consensus on the need for Service Level Agreements (SLAs), and several standards, like WSML [4] or WSLA [5], exist to define SLAs. Still, we think that probabilistic aspects are not covered in the necessary detail at the moment. Therefore, we want to focus on the aspect of variability found both in NFPs' values and their contractual descriptions.



(a) Response time (b) Usage patterns

Fig. 1. Examples

The values of a lot of NFPs exhibit an inherent variability. For instance, response time is not a constant value at all. Yet, in typical contracts, NFPs are represented by their maximum value only, whereas probability distributions would be much more accurate. As you can see in Fig. 1(a), the distributions of two services regarding response time can be quite different, though they share the same maximum value. This means that different users might prefer one over the other, which we think should be reflected in the selection. We therefore propose using probability distributions throughout the whole selection process.

The description itself also often exhibits variability. Commonly, in a contract, there is only one description for each NFP, specifying its value(s). A provider can, of course, offer multiple contracts, but, since customization is not possible, cannot really tailor a single contract for a specific kind of user. For example, if we take a look at mobile phone providers: They succeeded in catering to specific kinds of users, because what is charged per minute depends on several conditions, enabling each user to find a contract that best matches his needs. We think that, in a SOA, this possibility for providers to differentiate themselves is becoming increasingly important, as more and more functionally equivalent services are made available. We therefore propose the concept of conditional contracts. Providers can specify several descriptions of each NFP in a contract, and conditions according to which one of them will be chosen. Additionally, to

help users finding the service that best matches their needs, the selection process should be adapted accordingly by taking usage patterns into account: A usage pattern tells us how and when a service is expected to be used by a user, and, as such, can be represented by a probability distribution. For instance, consider a contract for a service s that, compared to the average, is very expensive during the week, yet very cheap during the weekend: Given two users u_1 and u_2 and their corresponding usage patterns from Fig. 1(b), that tell us exactly which days of the week they usually use s, it is clear that u_1 should use s, whereas an average service would be a better choice for u_2.

In a nutshell, we propose a probabilistic approach to leverage the variability of NFPs: we use conditional contracts and usage patterns, while treating NFPs as probability distributions throughout the whole selection process. This gives providers an opportunity to differentiate themselves and users a very easy way to find the best match. Our evaluation shows that not only does our approach make novel kinds of scenarios possible, it also allows for better selection in existing scenarios.

The structure of this paper is as follows: Section 2 gives an overview over related work. Section 3 introduces a realistic scenario. Section 4 describes our approach. Section 5 evaluates our approach against our scenario, and finally Section 6 concludes the paper.

2 Related Work

In this section we survey work that is related to ours. We show both the impact and the differences in relation to our work.

Regarding NFPs, while, in general, there is a lot of research out there, probabilistic approaches are less common. Probability distributions for NFPs have mainly been used in computing the NFPs of service compositions with different kinds of techniques: Dynamic programming and the greedy method [6], as well as Monte-Carlo simulations [7] have been applied for this purpose. This gives us confidence that using probability distributions for NFPs is feasible, because even calculating those for composed services works reasonably well. Besides, to the best of our knowledge, there is no work that uses probabilistic NFPs directly for service selection, as we do in our approach.

While we have not seen conditional contracts, as such, there exists an approach [8] that formalizes the obligations of a service provider as logical rules, which are specified with the Web Service Modeling Language (WSML) [4]. This allows to use reasoning on top of those rules to compute the actual NFPs for a specific service request. In a way, this is quite similar to what you can do with conditional contracts, yet we found two important differences: First, there exists no designated way to use such logical rules in conjunction with probabilistic computations. Second, given the possible complexity of such rules in WSML, it might not be feasible to derive direct conditions that imply certain values for NFPs. As we will see, both are strictly necessary for our approach. Hence, we deem conditional contracts a better choice for our purposes.

Concerning usage patterns, there seems to be little research besides the introduction of the notion to refer to patterns in how users usually compose services [9]. On the contrary, we only refer to usage patterns in how a single service is used, e.g. at which day of the week, or at which time, a user usually calls a service

In conclusion, our contribution does not only lie in defining conditional contracts and usage patterns, but in combining them together with probability distributions into a probabilistic approach, which leverages the variability of NFPs both in their values and descriptions.

3 Scenario

Now, we start by depicting a realistic scenario to illustrate the real-world problems that we are trying to solve. First, we describe the setting, the service in question and the assumed infrastructure in a general overview. Then, we give detailed descriptions of the involved providers and users. Finally, we pinpoint the challenges that arise from this scenario.

3.1 Overview

Our overall setting is the stock market. The service we envision provides mobile news about companies listed in the stock market. A stock market, in general, is only open on certain days of the week during specific time slots. For our means, we assume a stock market that is open from Monday to Friday during 9 and 17 o'clock. Consequently, the demand for the service is generally highest when the stock market is open, and lowest on the weekend.

Through their mobile clients, users can request news when and how often they want. Payment is specified per service request. The service discovery happens through brokers, that are commonly found in a SOA. These brokers have access to all service contracts available from different providers. Furthermore, brokers not only provide service discovery, but can also compute the contract that best matches a user's requirements and preferences. However, as consulting the broker incurs a notable fee, the users' clients usually only update their contracts once in a while, e.g. once a month.

3.2 Providers

Providers all offer functionally equivalent services. Therefore, they differentiate themselves only through the NFPs defined in their contract(s). Though, to make things comparable, all providers have agreed on a common pricing schema. First, to distinguish them based on their quality, providers are classified into different service classes $A, B, \ldots$ according to their maximum response times. For those classes, base prices are fixed. Starting from the best service class, A, for each class the maximum response times increases, while the base price decreases.

Then, to cater to specific kinds of users or to differentiate themselves, providers can introduce as many different service options as needed. These options modify the base price, given by the service class, depending on the time of usage, which can be classified into: Stock market open (Mo-Fr 9-17), during the week (Mo-Fr else), and during the weekend (Sa-Su). While this discretization of the time of usage is standardized, the providers are free to choose the according prices.

Therefore, the final price for the user is calculated by multiplying the base price, determined by the service class, with a constant factor, determined by the service option.

3.3 Users

As for the users, of course, each of them has different needs and uses the service differently. Nevertheless, most can either be categorized as business users, that deal with the stock market for a living, or as casual users, that engage in the stock market as a hobby.

The most essential difference is the time when they use the service. Business users mostly use the service when the stock market is open, sometimes during the rest of the week, and only rarely on the weekend. Casual users, on the other hand, use the service rarely when the stock market is open, also sometimes during the rest of the week, but mostly on the weekend.

Additionally, they have different needs, resulting in different requirements and preferences that mainly relate to the following two NFPs: Response time and price. Regarding response time, both value the response time in terms of the throughput, which means the number of news updates they can receive in a certain time interval. Also, both have an optimal throughput that allows them to make the best use of the service, and a minimum throughput that is the limit of what is actually usable or tolerable for them in terms of productivity or patience. Naturally, business users have much higher requirements for the throughput than casual users. Regarding the price, both, similarly, have optimal and maximum values, and, as expected, casual users are more price sensitive than business users. Throughput and price each contribute a part to the overall utility of the service for the users: Business users value throughput the most, while casual users are more concerned about price.

3.4 Challenges

So far, we have described the scenario, but not yet analyzed what actually is challenging about it. On a top level, we identify two areas that pose new challenges: The definition and the selection of contracts.

Definition. The definition of contracts that meet the needs of the providers is the first challenge, because, with normal contracts, it is not possible to realize the service options mentioned: Representing an option as a separate contract

would only allow to realize a base option that offers a constant price. On the other hand, any meaningful option would have to specify multiple alternatives at least for the price, depending on the time of the usage of the service. Otherwise, there would be no way for the service providers to cater to business or casual users by tailoring their contracts.

Selection. Selecting the best contract that accurately matches the users' needs poses the second challenge. While choosing the right service class according to its maximum response time is not that hard, choosing the best contract of providers from the same class is. The same maximum response time tells us nothing about the actual distribution of the response time's values: What is the average value, or how probable is an interval of values that is of special interest to the user? So selecting the best contract by choosing the right service class and service option is not trivial, because not only the needs, but also the usage patterns of users might differ, which might have a high impact on selection. Hence, accurately taking the long term prospect of a contract for a user into account is not easy, but especially important, as the contract can only be changed so often, because of the incurred fee when invoking the broker.

4 Approach

In this section we introduce our approach. While we want to illustrate how it can solve the challenges posed by the scenario, we first present it in its generality here, and then adapt it specifically to our scenario later on in the evaluation. First, we introduce our notions of conditional contracts, usage patterns and probability distributions, before going into the details of our probabilistic approach.

4.1 Conditional Contracts

First, we need contracts that allow us to express the services of our scenario. As already mentioned, we could model each service option with a separate contract, but we also would have to model the conditional pricing for each option. Therefore, we propose conditional contracts.

Definition 1. *A conditional contract ($cc \in \mathcal{CC}$) consists of a conditional statement. A conditional statement ($cs \in \mathcal{CS}$) can either be a statement ($s \in \mathcal{S}$) or a tuple of a condition ($c \in \mathcal{C}$) and two conditional statements, of which the first corresponds to the condition being true and the second to it being false.*

$$\begin{aligned} &< cc > ::= < cs > \\ &< cs > ::= < s > \mid (< c >, < cs >, < cs >) \end{aligned} \tag{1}$$

As syntax we propose something similar to typical programming languages, so a sample conditional contract could look like this:

```
contract {
        if time.weekDay = Monday
                if 8 <= time.hour <= 9
                        price = 4
                else
                        price = 2
        else
                price = 1
}
```

The semantics are also similar to what one would expect in a typical programming language, so by evaluating the conditions, one can easily deduce which statement actually holds when a service is called.

4.2 Usage Patterns

Then, to select the best service option for a user, we need to know how he uses the service. Thus, we introduce usage patterns.

Definition 2. *A usage pattern function up takes a condition $c \in \mathcal{C}$ as a parameter and returns a probability between 0 and 1 for the likeliness that c is true.*

$$up : \mathcal{C} \rightarrow [0, 1] \tag{2}$$

For a given user, we compute[1] his usage pattern function, so we can evaluate the contracts for him.

4.3 Probability Distributions

Finally, to find the best provider not only on a service class level, but also within a service class, we treat all our NFPs as probability distributions. This way, we can later differentiate even between providers within the same service class.

Given an utility function, instead of applying it only to maximum values, we can apply it first to the values of the probability distributions themselves, before aggregating everything into a single utility value. To aggregate these utility values properly, we need the probability for each combination of values for the NFPs involved. Therefore, we introduce a NFP function.

Definition 3. *A NFP function nfp defined for several NFPs $P_1, \ldots, P_n \in \mathcal{NFP}$ takes possible values for those NFPs as parameters and returns a probability between 0 and 1 for the likeliness of this combination of values.*

$$nfp : P_1 \times \cdots \times P_n \rightarrow [0, 1] \tag{3}$$

[1] There are many ways how to compute or approximate such a usage pattern function, e.g. one could compute it from the history of the user's previous requests.

4.4 Probabilistic Selection

Now, that we have introduced our notions of conditional contracts, usage patterns and probability distributions, we can introduce our probabilistic approach in its entirety. In the following we layout the steps of our approach one by one.

Setup. Our approach finds the best contract cc_u for a given user u from a given list of contracts $\mathcal{CC}_a \subseteq \mathcal{CC}$. For that, we only consider a limited number of NFPs, $P_a := \{P_1, \ldots, P_n\} \subseteq \mathcal{NFP}$, for which we also define the following notation:

$$\overrightarrow{P_a} := P_1 \times \cdots \times P_n \tag{4}$$

To compute the utility of a contract cc, we need the user's utility function $util_u$ that implies certain values for our NFPs P_a:

$$util_u : \overrightarrow{P_a} \rightarrow \mathbb{R} \tag{5}$$

Compute Conditions. As a first computation step, we compute all "full" conditions that directly determine if statements in the given contracts hold or not. For this, we need the following auxiliary function ac:

$$ac(cc, c_{ctx}) := \begin{cases} \{c_{ctx}\} & \text{for } cc = s \\ ac(cs_1, c_{ctx} \wedge \ c) \\ \cup\, ac(cs_2, c_{ctx} \wedge \neg\, c) & \text{for } cc = (c, cs_1, cs_2) \end{cases} \tag{6}$$

Then, we compute the relevant conditions $\mathcal{RC}_a$ for all given contracts, as follows:

$$\mathcal{RC}_a := \bigcup_{cc \in \mathcal{CC}_a} ac(cc, true) \tag{7}$$

Compute Usage Patterns. The next step is to compute[2] the user's usage pattern. While the usage pattern function may only be partial, it is important that it is defined for all relevant conditions:

$$\forall c \in \mathcal{RC}_a \,.\, up_u(c) \text{ is defined} \tag{8}$$

Compute NFPs. In the next step, we compute the user-specific probability distributions of all NFPs for each contract cc. In order to do this, we determine the relevant statement that corresponds to a given condition c with an auxiliary function rs:

$$rs(cc, c_x, c_{ctx}) := \begin{cases} \{s\} & \text{for } cc = s \wedge c_x = c_{ctx} \\ \{\} & \text{for } cc = s \wedge c_x \neq c_{ctx} \\ rs(cs_1, c_x, c_{ctx} \wedge \ c) \\ \cup\, rs(cs_2, c_x, c_{ctx} \wedge \neg\, c) & \text{for } cc = (c, cs_1, cs_2) \end{cases} \tag{9}$$

[2] There are several ways to compute this, but we do not focus on this in our approach.

Furthermore, for all statements s of cc, we compute[3] the NFP function $nfp_{cc,s}$ that returns the likeliness of a combination of values under the assumption that statement s holds:

$$nfp_{cc,s} : \overrightarrow{P_a} \rightarrow [0,1] \tag{10}$$

Given all this, we can now compute the probability distribution for NFPs specified in cc, as follows:

$$nfp_{cc,u}(\vec{p}) = \sum_{\substack{c \in ac(cc,true) \\ s \in rs(cc,c,true)}} up_u(c) \cdot nfp_{cc,s}(\vec{p}) \tag{11}$$

Compute Utility. As a last step, we compute the utility of the contract cc for the user u, using the utility function $util_u$. As already explained, we first apply the function to the values of the probability distributions itself, before aggregating everything into one utility value uv to make full use of the probability distributions:

$$uv_u(cc) := \sum_{\vec{p} \in \overrightarrow{P_a}} nfp_{cc,u}(\vec{p}) \cdot util_u(\vec{p}) \tag{12}$$

Thus, we can compute the utility value for any given contract, and then select the contract cc_u with the highest utility value for our user u.

5 Evaluation

In this section, we evaluate our approach against the scenario introduced before. The goal is to show two points:

1. Applying our approach to existing scenarios improves selection.
2. Using our approach allows for selection in novel scenarios.

Response time is a suitable NFP to show (1), because its variability is already inherent, independent of any scenario. So we take our scenario without its pricing aspects and show that applying our approach improves selection. Because our results do not depend on introducing novel pricing schemes, they can be generally applied to existing scenarios.

Price, on the other hand, is a NFP for which we can clearly show (2), directly following our scenario. Nevertheless, because of the prominence of pricing schemes, we think our results can be easily transfered to other scenarios as well, given some domain specific adaptation.

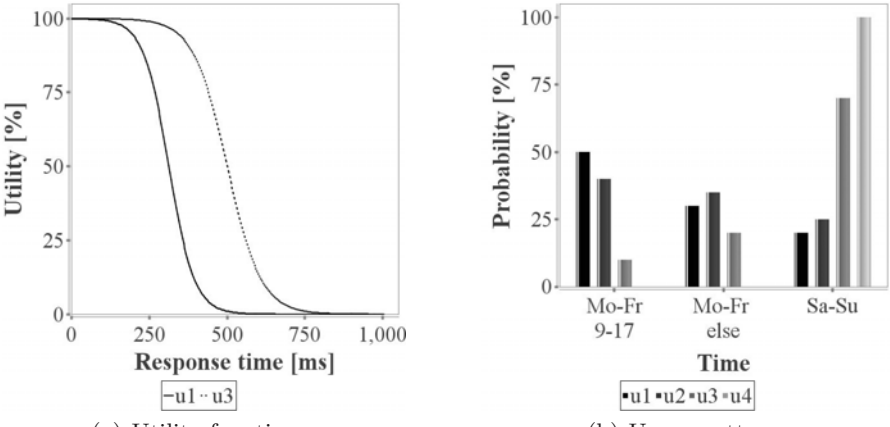
Hence, we conduct our evaluation in two parts, focusing on response time in the first part and on price in the latter to show (1) and (2), respectively.

[3] The complexity of the computation mainly depends on what kind of statements are allowed. For example, computation of $nfp_{cc,s}$ should be easy when directly assigning constant values or probability distributions to specific NFPs.

5.1 Response Time

In order to evaluate our approach for the NFP response time, we first define concrete providers and users. Then, we compute the utility of the providers for each user.

Users. We have four users: u_1, u_2, u_3 and u_4. Out of them, u_1 and u_2 are business users, and u_3 and u_4 are casual users. Therefore, u_1 and u_2 have sharper requirements regarding response time, meaning a higher optimal and minimum throughput.



(a) Utility functions (b) Usage patterns

Fig. 2. Users

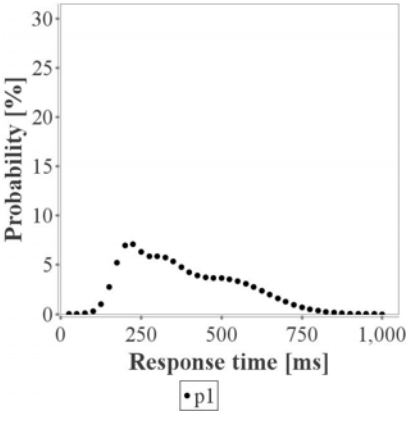
In Fig. 2(a) we see the corresponding utility functions for them[4], that are of the following form[5], similar to those described in [10]:

$$f(x) := \frac{1}{1+e^{a(x+c)}} \tag{13}$$

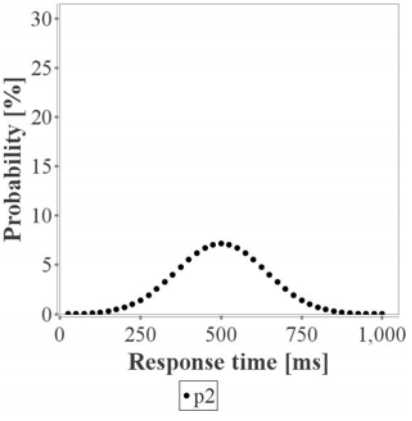
Additionally, our users have usage patterns, as seen in Fig. 2(b), that are typical for business and casual users, respectively. Two aspects are of special interest: First, while the usage pattern of u_1 directly corresponds to the load of p_1, u_2's usage pattern is still similar, but slightly different. Second, while both u_3 and u_4 represent casual users, only u_4 uses the service solely on the weekend.

Providers. We have two providers, p_1 and p_2, with the same maximum response time. This implies, they are in the same service class and therefore share the same base price. Thus, it makes sense to compare them just in terms of their response time. The probability distributions of their (aggregated) response times are shown in Fig. 3.

[4] The utility functions of $u1$ and $u2$ are identical. The same is valid for $u3$ and $u4$.
[5] Both a and c can be computed from the optimal and minimum throughput.

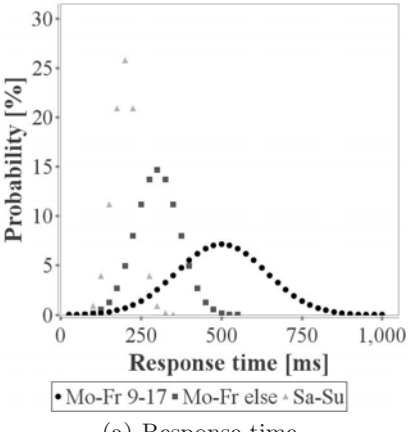


(a) Provider p_1

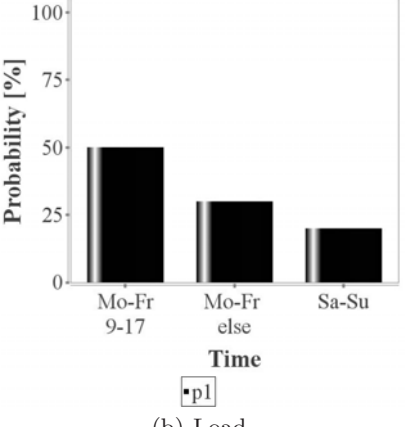


(b) Provider p_2

Fig. 3. Aggregated response time



(a) Response time



(b) Load

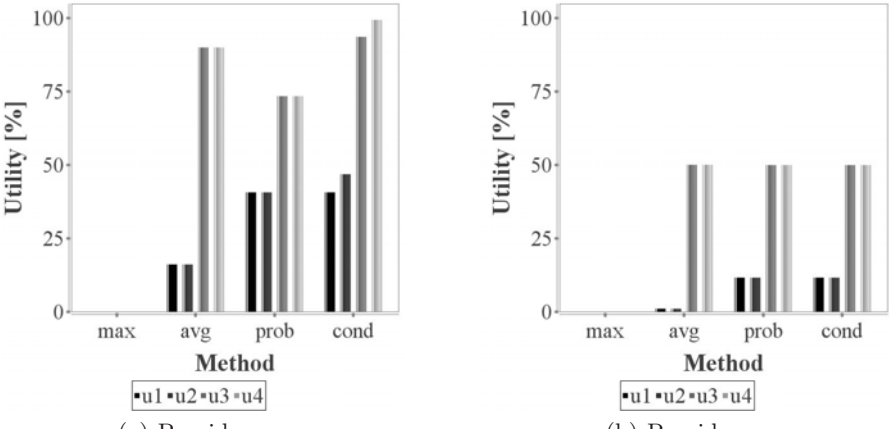
Fig. 4. Provider p_1

As you can see, for p_2 the probability distribution follows a simple normal distribution, while for p_1 it seems to follow no obvious pattern. That is, because, contrary to p_2, the response time of p_1 is dependent on the load, which varies throughout the week, as shown in Fig. 4(b). So actually, the distribution of p_1 is made up of all the normal distributions from Fig. 4(a), that, when aggregated, result in the distribution from Fig. 3(a).

Selection. Now, we compute the utility of each provider for each user, using four different methods: *max*, *avg*, *prob*, and *cond*. This yields the results seen in

Fig. 5. First, as a baseline we compute the utility using the maximum response time of the providers (*max*). This yields a utility of 0% for both providers, as their maximum response time corresponds to a throughput that is lower than the users' minimum throughput. We proceed by computing the utility using the average response time (*avg*), which is already better, because we get utilities greater than 0% for the providers.

Next, we compute the utility using the probability distribution of the (aggregated) response time (*prob*), and compare *prob* to *avg*. For p_1, we see that there is quite a notable difference for all users. Interestingly, while for u_1 and u_2 the utility increases with *prob*, for u_3 and u_4 it decreases. This shows using *avg* can serve neither as a lower nor as an upper bound for the utility computed using the inherent probability distributions.



(a) Provider p_1 (b) Provider p_2

Fig. 5. Utility values

Finally, we compute the utility using our full approach with conditional contracts and usage patterns (*cond*), and look at the difference between *prob* and *cond*. This means that instead of just having one probability distribution for the aggregated response time in each provider's contract, we give three conditional definitions with different distributions, one for each time of usage. In the case of p_1, those conditional definitions correspond to the probability distributions given in Fig. 4(a). For p_2, on the other hand, there is just a single distribution from Fig. 3(b), so introducing conditions in p_2's contract does not change anything. This explains why the utility of p_2 does not change for a single user, as, without conditions, the usage patterns have no impact on the utility computation. So for p_1, we can see differences for all users, except for u_1, which is because of the correspondence of the load of p_1 and the usage pattern of u_1: The different

normal distributions of p_1's response time are aggregated according to p_1's load, and each distribution corresponds to conditions in the contract; the same conditions the usage patterns correspond to. Hence, the bigger the difference between usage pattern and load, the bigger the difference in utility. For u_1 this means no difference, for u_2 a slight difference, and for u_3 and u_4 big differences, as their usage patterns are quite different from p_1's load. Overall, the comparison between *max* and *cond* shows quite remarkable differences in utility, especially regarding p_1: For u_3 and u_4 the utility goes up from 0% to over 40%, and for u_3 and u_4 it even goes up from 0% to over 90%.

5.2 Price

In order to evaluate our approach for the NFP price, we first refine the concrete providers and users, we already introduced. Then, we again compute the utility of the providers for each user, now considering price and response time.

Users and Providers. In order to compute the utility of a provider for our users, we first define the additional utility functions you can see in Fig. 6.[6]

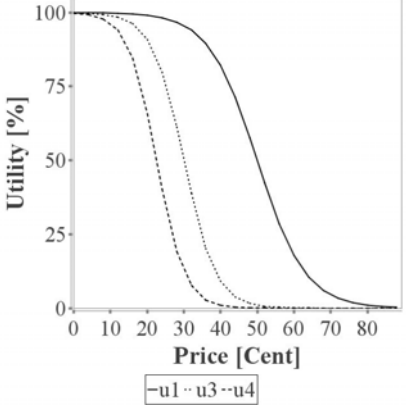
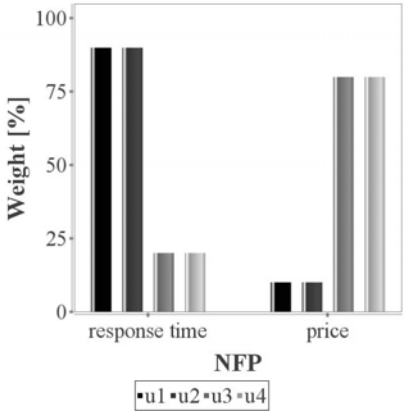


Fig. 6. Utility functions

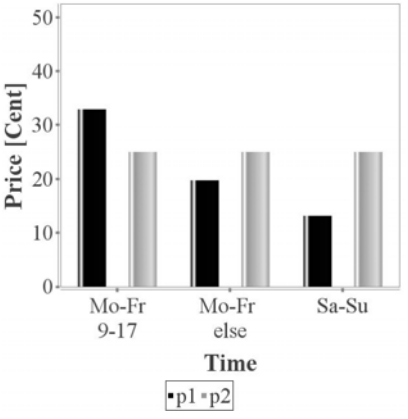
Then, we use a typical weighted sum, as for instance used in [8] and [11], to combine the utilities for response time and price. Figure 7(a) shows the corresponding weights for each user, which reflect that u_1 and u_2 are business, and u_3 and u_4 are casual users. As you can see in Fig. 7(b), similar to response time, the price for p_1 also depends on the load, while p_2's price is constant.[7]

[6] Both u_1 and u_2 have the same utility function.

[7] The aggregated average price (according to the load) for p_1 is equal to p_2's price.

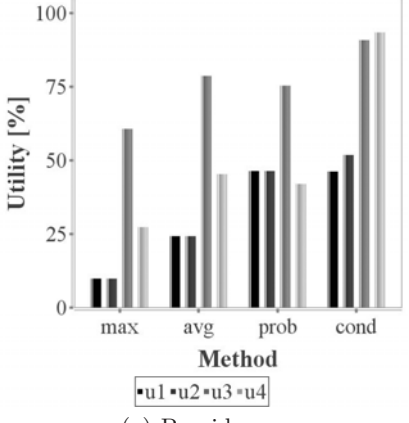


(a) Weights

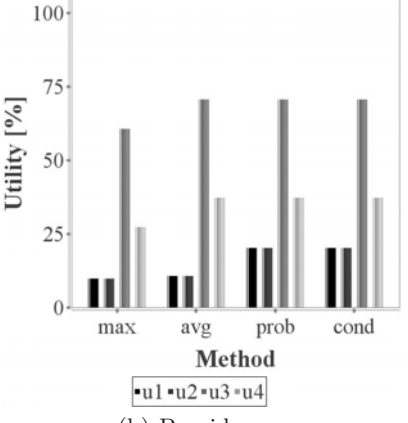


(b) Price

Fig. 7. Distributions



(a) Provider p_1



(b) Provider p_2

Fig. 8. Utility values

Selection. Again, we compute the utility of each provider for each user. The results from Fig. 8 are interesting in many respects, but we will pick just some observations: Comparing p_1's utility for u_4 to the previous evaluation, we see that, while the *max* utility is not 0% anymore, it is still less than 30%.

Also there is a sharp drop-off in *avg* and *prob* utility, that is not visible in the *cond* utility, which still is over 90%. The utilities for p_2, on the other hand, are not that much different, except that you can clearly see the different price expectations of u_3 and u_4 reflected in the increase of utility for u_3 and decrease for u_4, who is more price sensitive.

6 Conclusion

In this paper, we have proposed a probabilistic approach that leverages the variability of NFPs by using conditional contracts and usage patterns, together with probability distributions. We also have described a realistic scenario to illustrate the settings in which such variabilities occur. Furthermore, we have evaluated our approach against this scenario, with concrete examples, focusing on the two NFPs response time and price. The results of our evaluation show that our approach improves selection for existing scenarios and makes selection for novel scenarios possible.

Applying our approach in practice could yield some insights into how different ways of computing usage patterns or utility functions would fare, or what the impact of introducing conditional contracts in existing systems would be.

Last but not least, our work could also be extended in several ways. First, if anyone is going to use conditional contracts, the question is how to best formalize them, which could e.g. be done using WSLA [5]. Second, while we introduced conditional contracts, our approach could also, more generally, be applied to conditional descriptions of NFPs that are not necessarily contained in a contract. This also leaves the questions how to get such descriptions, and from where and from whom to get them. Finally, we have only looked into selection so far, but applying our approach to adaptation could be very interesting, because conditional descriptions of NFPs, and usage patterns could probably change at runtime.

References

1. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing. Communications of the ACM 46, 25–28 (2003)
2. O'Sullivan, J., Edmond, D., Ter Hofstede, A.: What's in a Service? Distributed and Parallel Databases 12(2-3), 117–133 (2002)
3. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.P.: Semantic Matching of Web Services Capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
4. de Bruijn, J., Lausen, H., Krummenacher, R., Polleres, A., Predoiu, L., Kifer, M., Fensel, D., Toma, I., Steinmetz, N., Kerrigan, M.: The Web Service Modeling Language WSML. Technical report, WSML, WSML Final Draft D16.1v0.3 (2007), `http://www.wsmo.org/TR/d16/d16.1/v0.3/`
5. Ludwig, H., Keller, A., Dan, A., King, R., Franck, R.: Web Service Level Agreement (WSLA) Language Specification, Version 1.0, IBM Corporation (2003), `http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf`
6. Hwang, S., Want, H., Tang, J., Srivastava, J.: A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. Information Sciences: an International Journal 177(23), 5484–5503 (2007)
7. Rosario, S., Benveniste, A., Haar, S., Jard, C.: Probabilistic QoS and Soft Contracts for Transaction-Based Web Services Orchestrations. IEEE Transactions on Services Computing 1(4), 187–200 (2008)

8. Toma, I., Roman, D., Fensel, D., Sapkota, B., Gomez, J.M.: A Multi-criteria Service Ranking Approach Based on Non-Functional Properties Rules Evaluation. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 435–441. Springer, Heidelberg (2007)
9. Balke, W., Wagner, M.: Towards Personalized Selection of Web Services. In: WWW 2003 (May 2003)
10. Menasce, D.A., Dubey, V.: Utility-based QoS Brokering in Service Oriented Architectures. In: ICWS 2007, July 2007, pp. 422–430 (2007)
11. Haddad, J.E., Manouvrier, M., Ramirez, G., Rukoz, M.: QoS-Driven Selection of Web Services for Transactional Composition. In: ICWS 2008, September 2008, pp. 653–660 (2008)