Service Selection for Happy Users: Making User Intuitive Quality Abstractions

Dionysis Athanasopoulos, Apostolos Zarras, Panos Vassiliadis

Service Selection for Happy Users: Making User-Intuitive Quality Abstractions

Dionysis Athanasopoulos
Dep. of Computer Science - Univ. of Ioannina Greece
dathanas@cs.uoi.gr

Apostolos V. Zarras
Dep. of Computer Science - Univ. of Ioannina Greece
zarras@cs.uoi.gr

Panos Vassiliadis
Dep. of Computer Science - Univ. of Ioannina Greece
pvassil@cs.uoi.gr

ABSTRACT

The state of the art service search engines allow the users to pick the services they need, based on the quality properties, offered by these services. To this end, the users should interact with the search engines based on the quality models that are imposed by the engines. This is a significant restriction towards making the service-oriented paradigm attractive to the general public. In this paper, we propose an approach that allows a user to specify his perception of quality in terms of a simple, user-defined quality model. The proposed approach automatically maps the user-defined quality model to the search engine's quality model. This mapping forms the basis for ordering, grouping and, in general manipulating, the results of the user's service discovery requests.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures - Domain-Specific Architectures

Keywords

Service Selection, Quality of Service

1. INTRODUCTION

The SWaN story: 30/6/2012 - It's a rainy Friday afternoon in the headquarters of Shiny-Warm-and-Nice (SWaN), a travel agency specializing in exotic travel. Baldus, the manager and Dilbertus, the developer from the IT department are discussing in Baldus' office:

B: I have spent all week trying to configure a work-flow for receiving customers at the airport and helping them move to their destination easily, but I simply can't pick the right services.

D: Pick the right services?

B: Yes, the idea is that we develop a registration process at our Web site, where the customers book a reception method, along with their travel and hotel arrangements. So, we need to cooperate with limo & car rental companies that offer Web services that are cheap, fast, reliable and come along with high reputation.

D: Well, you need to search a registry for services based on the functionality that you need. Then, you have to select the services that you are going to use based on your quality requirements. I think you should pick up services with a response time of less than 1 sec, more than 90% reliable and ...

B: Stop! Stop! Why 90% of ... whatever you said ... and not 95%? I am having a headache with your numbers that I forget anyway and I simply don't want to have to decide the right ones! It simply has to be cheap, fast, reliable and reputable! Can you have a nice list printed on my desk by the end of the day?

D: ... or maybe have a search engine that lets you do the work on your own (God forbid)...

The challenge: The moral of this short story is that *the discovery of the right services for a particular business process should be as easy as possible.* Achieving this goal is a key issue towards making the service-oriented paradigm really attractive to the general public.

Where we stand: Currently, the state of the art in service discovery does not allow us to fully achieve this goal. Service discovery is a process that consists of two main phases. The first phase, which we call **service scoping**, is driven by the users' functional requirements. Service scoping results in a possibly large set of services that satisfy to some extent the given requirements. This set of services is the starting point of the second phase, which we call **service selection**. The goal of service selection is to pick up the services that are going to be used. The state of the art service search engines facilitate this task based on quality descriptions that characterize the services returned from the scoping phase. Given these quality descriptions the user can select the services that meet his quality requirements.

Taking a real-world example, Figure 1 shows the scoping and the selection phases in the case of the WSCE engine [1]. In the scoping phase, the user's requirements are expressed as a set of keywords (Figure 1(a)). Next, WSCE returns to the user a set of services, characterized by corresponding quality descriptions (Figure 1(b)). The quality descriptions of the services are based on a quality model that reflects the technical perception of quality that is assumed by the WSCE engine. In the case of WSCE, the technical

SIGSOFT'12/FSE-20, November 11–16, 2012, Cary, North Carolina, USA.

quality model consists of 10 quality properties among which we have response time (RT), availability (AV), throughput (TH), reliability (RL), etc.

(a) Scoping phase.

Name	Response Time (ms)	Throughput (hits/sec)	Reliability (%)	Best Practices (%)	Documentation (%)	Class
DictionaryService	45	27.2	97.4	91	58	1
MyService	71.75	14.6	85.5	80	86	1
aba	117	23.4	88	87	59	1
AlexaWebSearch	70	5.4	79.3	75	91	1
ErrorMailer	105.2	18.2	92.2	84	91	1
getJoke	224	24.6	80	87	88	1
States_x0020__x0026__x0020_Provinces	99.2	13.7	76.3	83	89	1
XigniteRetirement	108.2	16.8	90.7	77	94	1
DOTSEmailValidate	125.2	16.4	89.2	84	93	1
XigniteNews	110.3	13.9	87.5	77	93	1

(b) Selection phase.

Figure 1: A typical service discovery scenario based on WSCE [1].

Regarding service scoping, we have very promising user-intuitive approaches that scope the search space and allow the users to express their functional requirements in terms of semantic description languages (see [7] for an interesting survey), or even in natural language [13].

For service selection, we also have very promising approaches. The core concept in these approaches is the quality model that they assume. Quality models come in 3 flavors. The simplest and most popular model is a *flat model* [11, 14, 2, 8, 10, 5, 3, 12], where the quality properties are listed as a flat vector of values characterizing each service. Second, we have *quality models with property structuring* [6] (e.g., generic properties, business properties). Third, we have *semantically rich* quality models, specified in terms of ontologies [4].

The main shortcoming in the aforementioned approaches is that they rely on the fundamental assumption that the users should interact with service search engines via the quality models that are imposed by the engines. From our point of view, *this is a significant restriction*, since service-oriented computing (SOC) does not only target developers with strong technical knowledge. On the contrary, SOC is supposed to promote software reuse to everyday users who may have very diverse backgrounds. From this perspective, we believe that *the quality-driven service selection should rely on a combination of:*

- *Technical quality models* that reflect the quality properties of available services.
- *Simple, user-defined quality models* that reflect the perception of quality of ordinary users.

What we propose: Inline with the aforementioned view, in this paper we propose a service selection approach that allows a user to specify his perception of quality in terms of a *simple, user-defined quality model* that describes quality properties in layman's terms. We call the resulting user-defined property definitions, **quality abstractions** to signify that these definitions overlay the underlying unique, reference technical definition of the service search engine. Next, the proposed approach automatically finds a mapping between the user-defined quality model and the search engine's *technical model*, which reflects the quality properties of available services in concrete technical terms. The derived mapping forms the basis for translating, ordering, grouping and, in general manipulating, the results of the users discovery requests.

2. APPROACH

Fundamentals: We start by formally defining the involved concepts. First, we provide a general definition of the technical quality model that we assume for the service search engine.

Definition 1. A technical quality model is a set of quality properties $\mathbf{Q}^T = \{Q_1^T, Q_2^T, \ldots, Q_N^T\}$. Each quality property Q_i^T is associated with (a) a name, and, (b) a corresponding ordered domain of values, $dom(Q_i^T)$.

In the rest of our deliberations, all domains include the special value `UNKOWN`. Domains can be either discrete or dense. We refer to the space of values constructed by the Cartesian product of the domains of the quality properties as $dom(\mathbf{Q}^T) = dom(Q_1^T) \times \ldots \times dom(Q_N^T)$.

To alleviate the burden of dealing with the search engine's complexities, we provide two simplifications to the users: (a) they can deal only with a subset of quality properties that interest them, and, (b) they can define their own vocabulary of values for these quality properties (which we call quality abstractions) and form a user-defined, *abstract quality model*. The crux of our approach is to automate the mapping between the user's quality abstractions and the engine's technical quality model.

Definition 2. An abstract quality model $\mathbf{Q}^A$, is defined as a set of quality abstractions $\mathbf{Q}^A = \{q_1^A, q_2^A, \ldots, q_n^A\}$. Each quality abstraction q_i^A is associated with (a) a name, and, (b) a corresponding ordered, discrete and finite domain of values, $dom(q_i^A)$.

Typically, user-defined domains are of small cardinality. As with the technical quality model, we assume that `UNKNOWN` is part of every domain and that $dom(\mathbf{Q}^A) = dom(q_1^A) \times \ldots \times dom(q_n^A)$.

Taking the example of the WSCE engine, assume that the user provides an abstract quality model $\mathbf{Q}^A = \{RT^A, AV^A\}$ which is a subset of the WSCE technical quality model. Following, the user defines the domain for response-time abstraction $dom(RT^A) = \{slow, bearable, fast\}$ and the domain for the availability abstraction $dom(AV^A) = \{low, medium, high\}$.

Of course, we need to have quality abstractions that are mappable to the underlying technical quality model. To

facilitate our task, we assume that technical and abstract models share the same property names. At the same time, we need to map every value of the abstract model's domain to a range of values of the technical model's domain. This way, we partition the domain of a quality property in pairwise discrete ranges to which abstract values are mapped. Again, the desideratum is to produce this mapping automatically.

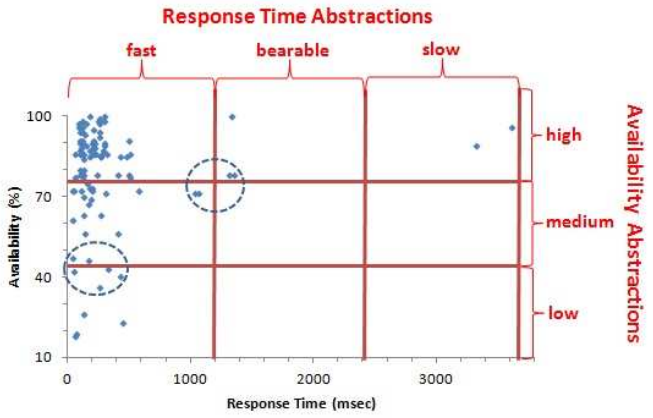
Definition 3. A *well-defined* abstract quality model $\mathbf{Q}^A = \{q_1^A, q_2^A, \ldots, q_n^A\}$, defined over a technical model $\mathbf{Q}^T = \{Q_1^T, Q_2^T, \ldots, Q_N^T\}$, $n \leq N$,guarantees that $\forall\ q_i^A \in \mathbf{Q}^A$, $\exists\ Q_j^T \in \mathbf{Q}^T$, called $\mu_q(q_i^A)$, such that:

- $q_i^A.name = Q_j^T.name$, and,
- $\forall\ v \in dom(q_i^A)$, $\exists\ v_{low}, v_{high} \in dom(Q_j^T)$, $v_{low} \leq v_{high}$ and a mapping $\mu_d(v)=[v_{low}, v_{high}]$
- for any two $v, v' \in dom(q_i^A)$, $v \neq v'$, $\mu_d(v) \cap \mu_d(v')=\varnothing$

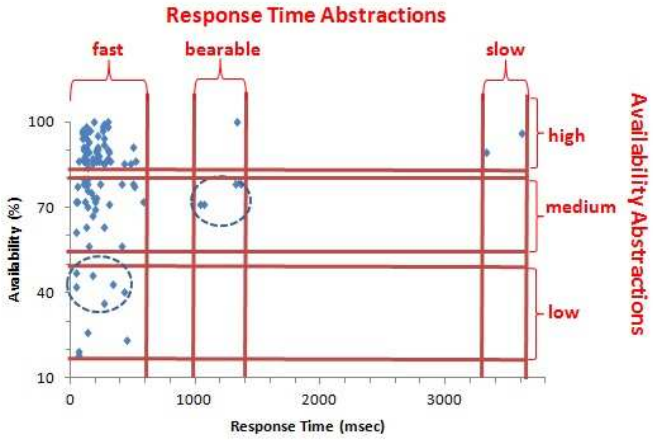
In the rest of our deliberations, we deal only with well-defined abstract quality models. Assume now a discovery request (or simply, query) d posed over the service search engine $\mathbf{E}$. In our example, such a query could be in the form $d := (RT^A = fast) \wedge ((AV^A = high) \vee (AV^A = medium))$ [1]. Each service s_i belonging to the result of d, $res(d|\mathbf{E})$ is characterized by a vector $v_{s_i}^T$ of quality values in the technical domain of the engine $dom(\mathbf{Q}^T)$ (e.g., $[RT^T = 103ms, AV^T = 91\%]$). The service s_i is further associated with a respective vector $v_{s_i}^A$ in the user abstractions $dom(\mathbf{Q}^A)$; this vector is calculated based on $v_{s_i}^T$ and the mapping between $\mathbf{Q}^A$ and $\mathbf{Q}^T$ (e.g., $[RT^A = fast, AV^A = high]$). This way, query results can be translated, ordered, grouped and, in general manipulated, via the user vocabulary. The only open issue left, is of course, the construction of the mappings among abstract and technical domains.

Mapping construction: Let q_i^A denote a quality abstraction in $\mathbf{Q}^A$ and Q_j^T denote its respective quality property in $\mathbf{Q}^T$, i.e., $Q_j^T = \mu_q(q_i^A)$. A naive method for mapping $dom(q_i^A)$ to $dom(Q_j^T)$ comprises the following steps. First, we calculate the overall range $R_{Q_j^T}^{in} = [v_{min}^j, v_{max}^j]$ of the Q_j^T values that characterize the services of $res(d|\mathbf{E})$ such that, $v_{min}^j = \min(v_{s_j}^T[Q_j^T], \forall\ s_j \in res(d|\mathbf{E}))$ and $v_{max}^j = \max(v_{s_j}^T[Q_j^T], \forall\ s_j \in res(d|\mathbf{E}))$. Then, we divide $R_{Q_j^T}^{in}$ in a set $R_{Q_j^T}^{out} = \{r_1^j, r_2^j, \ldots, r_{|dom(q_i^A)|}^j\}$ that comprises $|dom(q_i^A)|$ equally sized sub-ranges. Finally, we map each value $v_k^i \in dom(q_i^A)$ of the quality abstraction q_i^A to a corresponding range $r_k^j \in R_{Q_j^T}^{out}$, i.e., $\mu_d(v_k^i) = r_k^j$.

In our example, the scatter chart in Figure 2(a) gives the results of this naive method in the case of 100 services that we retrieved based on the WSCE engine. The horizontal axis gives the values of response time for the services, with respect to the WSCE technical quality model. The vertical axis gives the availability values, with respect to the WSCE technical quality model. The mapping between the domain of the user-defined response time abstraction and the WSCE response time values is given on top of the scatter chart. Moreover, the mapping from the domain of the user-defined availability abstraction to the WSCE availability values is given on the left of the scatter chart. Apparently, this naive method is quite problematic. In several cases (grouped with the dashed circles in Figure 2(a)), we observe that similar values in the domain of the WSCE technical quality model are mapped to different values in the domain of the user-defined abstract quality model.



(a) Naive approach.



(b) K-means approach.

Figure 2: Mappings construction.

To overcome such problems we turned to more advanced data mining techniques towards deriving mappings between abstract and technical domains. In general, we consider that finding the right technique(s) and adapting them to our problem is an open issue. However, we currently investigate the use of partitional clustering and specifically K-means [9]. Typically, the K-means clustering technique accepts as input a set of elements and a required number of clusters k. Then, it divides the given set of elements in k clusters, based on a similarity metric that is defined for the elements. Each cluster contains elements that are more similar to each other than the elements of the other clusters.

We employ K-means to map $dom(q_i^A)$ to $dom(Q_j^T)$ as follows. We give as input to K-means the Q_j^T values $C_{Q_j^T}^{in} =$

[1]We intentionally avoid delving into the formal definition of a query. For all practical purposes, in this paper it suffices to treat it as a predicate over any available properties of the services of $\mathbf{E}$.

$\{v_1^j, v_2^j, \ldots, v_{|res(d|\mathbf{E})|}^j\}$ that characterize the services of $res(d|\mathbf{E})$. Then, we use K-means to divide $C_{Q_j^T}^{in}$ in a set of clusters $C_{Q_j^T}^{out} = \{c_1^j, c_2^j, \ldots, c_{|dom(q_i^A)|}^j\}$ that contain similar values. The required number of clusters is $|dom(q_i^A)|$. Finally, we map each value $v_k^i \in dom(q_i^A)$ of the quality abstraction q_i^A to a range $r_k^j = [\min(c_k^j), \max(c_k^j)]$ that is defined based on the contents of a corresponding cluster c_k, i.e., $\mu_d(v_k^i) = r_k^j$.

The scatter chart in Figure 2(b) gives the results of the K-means mapping method in the case of the 100 services that we retrieved based on the WSCE engine. The time for producing the results by executing the method on a typical Intel 2 Core, 2.20GHz, 3GB, was $\simeq$ 800ms. We observe that the issues that we identified in the mappings produced by the simple method (grouped with the dashed circles in Figure 2(a)) are now resolved in the mappings produced by the K-means method.

3. STATUS & EMERGING RESULTS

At this stage, we have implemented a prototype of the proposed idea as part of an extensible service search engine that is developed in the context of the CHOReOS project[2]. As a first step towards the assessment of the proposed idea, we performed an informal review based on a small working group that comprised partners (both service users and service providers) from the CHOReOS project. The goal of the review was to present our prototype and collect feedback from the participants. The participants found the overall idea of letting the users specify their own abstract quality models and manipulating search results based on these models interesting. Nevertheless, we also received certain reasonable criticisms and interesting suggestions. More specifically, the participants found the flat technical quality model that we assume quite simple and suggested to extend the proposed approach to account for more complex models (e.g., structured [6], semantically rich [4]). The participants also suggested to provide different alternative means for mapping abstract quality models to technical quality models and letting them choose the one(s) that better suit their preferences, concerning the produced mappings and the time required for producing the mappings. Another interesting point that was raised was to provide the ability to refine abstract quality models based on the experience gained by the usage of the search engine.

4. CONCLUSION

Starting from the overall vision of making service discovery as simple as possible, we proposed an approach that allows a user to select services, based on his own perception of quality which is reflected by a *simple, user-defined quality model*. Possible research directions include exploring the possibility of letting the users more free to specify their abstract quality models as well as verifying the usability of our approach via a large-scale user study.

5. ACKNOWLEDGEMENTS

This work received funding from the European Community's FP7/2007-2013 under grant agreement number 257178 (project CHOReOS).

[2]http://www.choreos.eu/bin/view/Main/

6. REFERENCES

[1] E. Al-Masri and Q. H. Mahmooud. Investigating Web Services on the World Wide Web. In *17th International Conference on World Wide Web (WWW)*, pages 795–804, 2008.

[2] C. Ding, P. Sambamoorthy, and Y. Tan. QoS Browsing for Web Service Selection. In *Proceedings of the International Conference on Service-Oriented Computing (ICSOC)*, pages 285–300. 2009.

[3] R. Karim, C. Ding, and C.-H. Chi. An Enhanced PROMETHEE Model for QoS-Based Web Service Selection. In *Proceedings of the IEEE International Conference on Services Computing (SCC)*, pages 536–543, 2011.

[4] K. Kritikos and D. Plexousakis. Requirements for QoS-Based Web Service Description and Discovery. *IEEE Transactions on Services Computing*, 2(4):320–337, 2009.

[5] M. Li, J. Zhao, L. Wang, S. Cai, and B. Xie. CoWS: An Internet-Enriched and Quality-Aware Web Services Search Engine. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pages 419–427, 2011.

[6] Y. Liu, A. H. Ngu, and L. Zeng. QoS Computation and Policing in Dynamic Web Service Selection. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 66–73, 2004.

[7] M. Rambold, H. Kasinger, F. Lautenbacher, and B. Bauer. Towards autonomic service discovery: A survey and comparison. In *IEEE International Conference on Services Computing, (SCC)*, 2009.

[8] A. Srivastava and P. G. Sorenson. Service Selection Based on Customer Rating of Quality of Service Attributes. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pages 1–8, 2010.

[9] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.

[10] Y. Xia, P. Chen, L. Bao, M. Wang, and J. Yang. A QoS-Aware Web Service Selection Algorithm Based on Clustering. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pages 428–435, 2011.

[11] Z. Xu, P. Martin, W. Powley, and F. Zulkernine. Reputation-Enhanced QoS-based Web Services Discovery. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pages 249 –256, 2007.

[12] S. S. Yau and Y. Yin. QoS-Based Service Ranking and Selection for Service-Based Systems. In *Proceedings of the IEEE International Conference on Services Computing (SCC)*, pages 56–63, 2011.

[13] K. Zachos and N. Maiden. Inventing Requirements from Software: An Empirical Investigation with Web Services. In *Proceedings of the 16th IEEE International Conference on Requirements Engineering (RE)*, pages 145–154, 2008.

[14] G. Zou, Y. Xiang, Y. Gan, D. Wang, and Z. Liu. An Agent-based Web Service Selection and Ranking Framework with QoS. In *Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, pages 37–42, 2009.