Selecting Web Services Based on Past User Experiences

Philipp Leitner, Anton Michlmayr, Florian Rosenberg, Schahram Dustdar
Distributed Systems Group
Vienna University of Technology
Argentinierstrasse 8/184-1, 1040 Vienna, Austria
lastname@infosys.tuwien.ac.at

Abstract

Since the Internet of Services (IoS) is becoming reality, there is an inherent need for novel service selection mechanisms, which work in spite of large numbers of alternative services and take the user-centric nature of services in the IoS into account. One way to do this is to incorporate feedback from previous service users. However, practical issues such as trust aspects, interaction contexts or synonymous feedbacks have to be taken into account. In this paper we discuss a service selection mechanism which makes use of structured and unstructured feedback to capture the Quality of Experience that services have provided in the past. We have implemented our approach within the SOA runtime VRESCo*, where we use freeform tags for unstructured and numerical ratings for structured user feedback. We discuss the general process of feedback-based service selection, and explain how the problems described above can be tackled. We conclude the paper with an illustrative case study and discussion of the presented ideas.*

1. Introduction

The term "Internet of Services" [1] (or IoS for short) describes the unification of Web 2.0, Service-Oriented Architectures [2] (SOA), Ubiquitous and Pervasive Computing, Semantic Web Services and Mobile Computing that we are currently experiencing. Soriano et al. define the IoS as "the notion of a global, user-centric SOA enabling a real Web of Services made up of a global mesh of interoperable user-centric services" [3]. In such a global network of human-provided and human-consumed services there is an evident need for mechanisms which allow (human) users to select services from a vast number of alternatives. Besides actual functionality, the main factor influencing the service selection process is the Quality of Service (QoS) of the alternatives. However, in addition to the software parts, services in the IoS often also contain a physical part, which may be as or even more important to the user than the technical service itself. For example, considering a hotel booking service (which we will use as a running example in the remainder of this paper), we can safely assume that for the majority of users the technical booking service is of much less relevance for the overall satisfaction with the business transaction than the quality of the hotel itself. Consequently, typical fine-grained QoS metrics such as service response time are not enough to capture the full user experience in the IoS. Instead, the user needs quality metrics which describe the quality of the business transactions in an end-to-end fashion. We refer to this less formalized end-to-end quality metric as Quality of Experience (QoE [4]).

The problem of QoE is that it is inherently hard to quantify and measure automatically because of the physical part mentioned above. However, the Web 2.0 trend of folksonomies [5] provides a reasonable way to capture QoE by incorporating human factors, and evaluate the quality of services based on feedback from earlier transactions, e.g., using numerical ratings. Additionally, folksonomies often make use of the concept of emergent semantics [6] (i.e., determining the semantics of resources through large amounts of independent, publicly available metadata, such as tags) to functionally describe resources.

The main contribution of this paper is the combination of these two concepts into a service selection framework for the IoS, which captures human-perceived QoE using both freeform tags (unstructured feedback similar to Web 2.0 tags) and numerical ratings (structured feedback, i.e., ratings) to narrow down and rank available service alternatives. Unlike current systems, unstructured feedback is used to describe the satisfaction of users with a service instead of its semantics. We explain how this leads to practical challenges such as tag merging, preventing

spamming of service providers or including the context of invocations, and show how these issues can be tackled. Furthermore, we explain a system implementation based on the VRESCo [7] SOA runtime which realizes these ideas.

The remainder of this paper is structured as follows. Section 2 gives an overview over the most important related work. In Section 3 we introduce our approach to feedback-based service selection. In Section 4 we explain how the concepts presented in this paper are integrated into the VRESCo SOA runtime. Section 5 describes the implementation of our prototype tool, which is discussed in Section 6. Finally, some concluding remarks and an outlook on our future work are provided in Section 7.

2. Related Work

Web service discovery based on QoS and service semantics has been a hot topic in research for some time [8], [9]. Generally, these approaches have a similar scope to our work in that they aim at allowing users to find not just any service to fulfill a task, but the best one. However, as argued in Section 1, we strongly feel that QoS alone is too narrow to fully represent the experience of service users in the IoS. The idea that technical QoS parameters such as availability and response time are too limited to capture the experience provided to end users has originally been introduced by van Moorsel in [4]. This work also concluded that "the relationship between system quality, user experience (...) is hard to make concrete", however, no ideas are presented on how QoE can be estimated in advance, which is essentially the scope of this paper. Much research has already been conducted in the area of collaborative tagging systems, e.g., in [10]. These works do not apply the concepts of tagging to the area of services engineering, but focus on general Web 2.0 systems such as Del.icio.us[1]. However, these earlier works provided many helpful insights which helped shape the contributions in this paper. For instance, [10] was first to identify that tagging is often used not only to classify information, but also to rate it, a notion that we also use throughout our work.

Combining Web 2.0 and SOA has been explored in [11]. This work introduces the concepts of service communities, and discusses a prototype middleware to support Web 2.0 aspects such as tagging. Unfortunately, little technical detail about this prototype is openly available. Generally, our work bears some similarities to the idea of discovering service semantics through tagging [12]. A conceptually similar approach is implemented in SOALive [13]. SOALive is a system for describing and discovering situational enterprise services based tag clouds, which is similar to the work we present. The main difference of our work and these approaches is that we use tagging for an entirely different purpose – in these works tagging is used to describe what a service does, and not how well it performs. In our work, the "what" aspect is handled using the VRESCo metadata model, while feedback is used purely to evaluate "how well" a service performs.

The idea of using past transactions for suggestion is the foundation of recommender systems [14]. In a recommender system people provide recommendations as input, which the system aggregates and transforms to support the selection of others. At the core, our system is a recommender system for IoS services, therefore, our work faces similar problems. One of these problems is "free-riding" users, i.e., users which consume feedback from others while not contributing themselves. Another one is the "vote early and often" phenomenon [14], i.e., service owners spamming the system by providing massive amount of negative feedback for competitors and/or positive feedback for their services. In our approach, we use the concepts of trust to deal with the latter issue. The former problem is not addressed directly, however, we argue that free-riding becomes more and more irrelevant with increasing scale, as demonstrated daily by many Web 2.0 platforms which work perfectly on the notion of voluntary contributions (e.g., Wikipedia[2]). An early recommender system for Web service selection has been presented in [15]. However, this system is only based on numerical ratings and seems therefore limited. A more advanced Web service recommender system has been presented in [16]. The IC-Service system presented there actively monitors Web service usage (using a remote client which has to be incorporated into client applications) and suggests Web services based on how other applications use services in similar contexts. The main difference between this work and ours is that they focus purely on implicit feedback (i.e., information that their client application collects), while we use a more explicit approach where users actively contribute feedback through tags and ratings. A system based on explicit feedback is discussed in [17], where explicit structured feedback from clients is used to evaluate if service providers violate their Service Level Agreements (SLAs), and an incentive system is presented which motivates clients to report truthfully on their perceived service quality.

In the database community the Skyline operator [18] is a well-known concept to find the most interesting alternatives in a large set of data points. This is related to our work, however, the Skyline approach relies on a well-structured set of dimensions to rank data (i.e., in order to use the Skyline approach all services need to be

1. http://delicious.com/

2. http://www.wikipedia.org/

ranked according to similar features). This is in line with traditional QoS models, however, we argue that the IoS is inherently less structured. In such a world the Skyline approach is, unlike our work, not applicable.

3. Solution Outline

Feedback-based service selection considers the selection of services from a large number of alternatives based on existing feedback from past user transactions. In our approach we use two different types of feedback. Firstly, we make use of unstructured feedback given as any number of freeform strings (tags). This type of feedback is used to characterize the QoE provided by a service in a very open form. Secondly, we use numerical ratings between 1 and 5 as structured feedback to numerically rank services for selection.

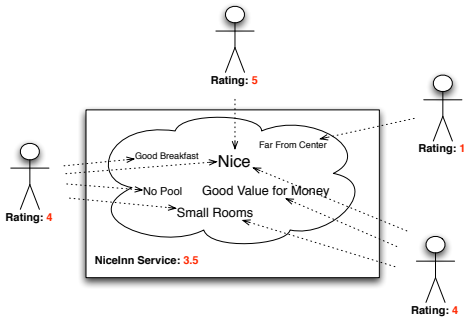


Figure 1. Illustrative Example

Figure 1 depicts the type of feedback used in this paper for an imaginary hotel booking service (NiceInn). There is feedback from four different customers for the service. The average numerical rating is *3.5* (on a scale from 1 to 5, with 5 being the best). However, much more interesting is the unstructured feedback associated with the service (visualized as tag cloud). From these tags more detailed information about the service can be learned, such as that three of four customers seemed rather satisfied, but the rooms were considered too small, and there is no pool. Clearly, these tags transport more fine-grained quality information. Some of this additional information may be unimportant for some future clients, but essential for others (e.g., if a client searches for a hotel for a business trip she may not care about there being a pool in the hotel, however, for a client who wants to go on summer vacation the pool might be a must-have). Note that in our model these tags are used to capture quality rather than semantics or functionality, even though some tags (like `No Pool` in the example) could also be considered functional descriptions.

The high-level process of feedback-based service selection is depicted in Figure 2. Generally, the process is a positive feedback loop, i.e., successful executions of the service selection process feed back into the system, which in turn (in tendency) improve the overall performance of the system. The selection process has five stages, which are executed sequentially.

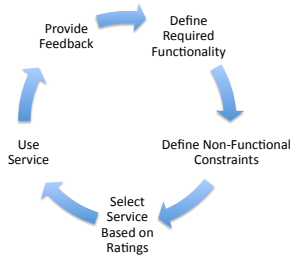


Figure 2. Service Selection Cycle

Firstly, the needed functionality is defined (e.g., being able to book a hotel). This is outside the scope of this paper. In our solution we mainly use the means provided by the VRESCo system, however, other approaches such as Semantic Web Services would be suitable as well. Secondly, in order to reduce the number of alternatives presented, non-functional constraints are defined. This is done by defining which tags should and should not be associated with the service to find (e.g., services should have the tags `Nice` and `Good Breakfast` associated, but not `No Pool`). Logically, this step is used to pre-filter the number of alternative services using hard constraints on the QoE of the service. Thirdly, a concrete service is picked from the remaining alternatives. This step is supported using both the structured and unstructured feedback: services are ordered based on average ratings given by users, and tag clouds can be used to quickly get an impression of what the general opinion of past users about the service was, besides the hard constraints formulated in the second step. When a concrete service is selected, it can finally be consumed. Lastly, the user should herself contribute back to the system and provide both tags and numerical rating to characterize the perceived quality of the service. In the following section we will explain how this cycle is implemented and supported in the VRESCo system, and how the challenges evident in such systems can be tackled.

4. Feedback Metadata and Extensions

We have integrated our feedback-based service selection model into the VRESCo (Vienna Runtime Environment for Service-Oriented Computing) SOA runtime environment. Details about VRESCo can be found elsewhere [19], and will be omitted here due to space restrictions. In the following we will present how the metadata necessary for feedback-based service discovery have been

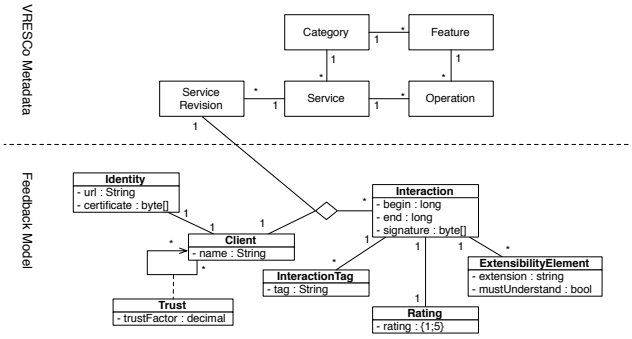


Figure 3. Service Feedback Metadata Model

implemented (grounded mostly on work presented in [20]), and which extensions have been provided in order to deal with real-world issues.

4.1. Basic Metadata Model

In [20], we have presented the general metadata model for the VRESCo infrastructure. This paper introduced the notion of *Feature*s and *Categories*, which are used to represent and group activities in a domain, e.g., booking a hotel. In VRESCo, features are semantically well-defined activities in the domain, with clearly defined inputs and outputs. In this paper we use *Feature*s to capture the functional aspects of service discovery, i.e., the first step in the cycle in Figure 2 is implemented by selecting a concrete *Feature* from the VRESCo metadata model.

Figure 3 shows an outtake of the original VRESCo metadata model ("VRESCo Metadata"), and the new data items that we have added in order to capture structured and unstructured user feedback ("Feedback Model"). Note that we have simplified the model for clarity in this figure. At the core, the idea is to capture *Interaction*s of *Service Client*s with concrete *Service Revision*s. For every *Interaction* we store exactly one *Rating* and any number of *Interaction Tags*. In our model we use the *Rating* as structured feedback with defined semantics (i.e., the rating is always a value between 1 and 5, 5 again being the best), while the *Interaction Tags* serve as means for the client to express additional feedback on the *Interaction* in a less structured way. Any additional information can be expressed using *Extensibility Elements*, which may contain arbitrary information, usually encoded in XML. Extensibility elements are (unlike rating and interaction tags) not standardized, i.e., usually not every extensibility element is interpretable for every client. The boolean "mustUnderstand" flag is used to indicate how a client should proceed when she receives an interaction with an extensibility element that she cannot interpret. If the flag is set to "true" the client ignores the interaction as a whole, if it is set to "false" the client ignores the extensibility element. Finally, begin and end date of the interaction are stored (as UNIX timestamp) to allow clients to e.g., only consider the more recent feedback.

4.2. Extensions

Using the model described above we are able to represent the basic structured and unstructured feedback necessary to implement a service selection process such as the one described in Section 3. However, there are a number of challenges associated with this basic model:

- Firstly, tagging systems such as the one described above face the problem of redundancy in tags, i.e., many different tags with very similar or identical semantics (e.g., `Nice`, `Good`, `Very Well`), which is further amplified by varying spellings and typing errors. In collaborative tagging systems the idea of tag merging based on tag similarity has been proposed to overcome this issue [21]. We provide two different merging strategies (based on co-occurence and cosine similarity) to define similarity of tags.
- Secondly, tagging systems are known to be susceptible to spamming of service providers [6]. We use a trust model to represent the reliability of a feedback source, in order to mitigate the influence of spam. On this account it is also necessary to provide means to verify the integrity of feedback (i.e., to verify that the feedback has not been forged).
- Finally, the above model ignores the context [22] in which a service has been used when it has been rated. This may be problematic for services which are often used in varying contexts, and exhibit a different QoE depending on it (e.g., the QoE provided by a hotel service may be totally different depending on whether the service is used in the context of a business or leisure trip). Therefore, we allow to annotate feedback with context information, so that future users may choose to incorporate only feedback which has been collected in equivalent circumstances of usage.

In the following we will discuss our solutions to these problems in more detail.

4.2.1. Tag Merging. To define similarity of freeform tags a number of possibilities exist [23]. Most of these similarity measures focus on how often tags have been used to describe the same service (co-occurence) or on the similarity of their usage patterns (cosine similarity). In VRESCo, we allow system administrators to select from three different tag merging strategies. The most basic one is the $none$ strategy (no merging at all). *Co-occurence merging* considers all tags $t_1,\ t_2$ to be similar which have a

normalized co-occurce ($nfreq(t_1, t_2)$) greater than a user-defined threshold. The co-occurence of two tags t_1 and t_2 is defined as the number of times t_1 and t_2 have been associated with the same service. We use the normalized co-occurence (co-occurence divided by the total number of times t_1 and t_2 have been used), because it is in $[0; 1]$, with $nfreq(t_1, t_2) = 0$ meaning that the tags are never used together, and $nfreq(t_1, t_2) = 1$ meaning that the tags are only used in conjunction. Finally, *cosine similarity merging* considers tags to be similar if they have a cosine similarity $cossim(t_1, t_2)$ greater than a given threshold. We use the definition of cosine similarity presented in [21]:

$$cossim(t_1, t_2) := \frac{v_1 \cdot v_2}{\|v_1\| \cdot \|v_2\|} \quad (1)$$

v_i are vectors, with $v_{t,t'} := freq(t, t')$ (i.e., every vector v_i consists of the non-normalized distances to all tags). $v_{t,t}$ (the distance of a tag to itself) is defined as 0. $\|v_1\|$ denotes the magnitude of v_1.

The actual process of tag merging is implemented using the following algorithm: for each tag we check if we should merge it with any other tag in the database (i.e., if the similarity of those tags is higher than a given threshold), and if both tags have been used more than once (we explicitly exclude tags which have been used only once, since we have too little information to merge these tags). If this is the case we update all occurrences of the tag which has been used less frequently with the more popular one. After merging a tag, we recalculate its similarity to every other tag. Consider for example the case of co-occurence merging with a threshold of 0.2. In this case, the two tags `Good` and `Very Nice` are merged if and only if they are used together (i.e., describing the same service) at least 20% of the time.

4.2.2. Trust Relationships. Another problem is spamming of service providers. One approach to handle this problem is to consider only feedback from trusted other users. However, this would arguably hamper the usefulness of the system, since too little feedback would be available to most services. One compromise to prevent spamming on the one hand and still incorporate a reasonable amount of feedback is to propagate trust relationships in a "friend of a friend" type of way.

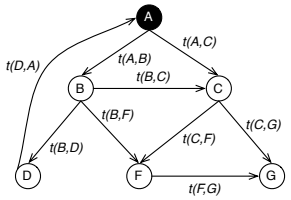


Figure 4. Example Trust Graph

Trust relationships between users in VRESCo are weighted using a "trust factor", a decimal value in $[0; 1]$, where 0 represents "no trust" and 1 "full trust". Inherently, the trust relationships between *Service Client*s form a weighted, directed graph, which we refer to as trust graph. Figure 4 sketches a small example trust graph, in which trust is propagated using a simple algorithm: the trust between two clients is the weighted length of the unweighted shortest path (i.e., the path with the smallest number of edges, ignoring edge weights) between these two clients in the trust graph; if two or more equally long shortest paths exist the one yielding a lower trust is used. Client *A* trusts *B* with a trust factor of $t(A, B)$, *C* with a trust factor of $t(A, C)$ and *F* with either $t(A, B) \cdot t(B, F)$ or $t(A, C) \cdot t(C, F)$, depending on which is less. Note that clients can explicitly assign a trust factor of 0 to essentially block a user. Clients choose a threshold of trust factor up to which they consider a client still trustworthy. Greater trust thresholds prevent spamming with higher certainty, however, this also leads to fewer interactions being used for selection, decreasing the value of the feedback-based service selection in general.

Trust relationships as described above inherently rely on the system being able to guarantee that feedback has not been forged (i.e., that a given feedback is indeed from the user that it claims to be from, and that the feedback has not been changed). We use digital signatures and OpenID[3] to let users explicitly verify the feedback that they use for their selection.

4.2.3. Context Information. The feedback metadata model as depicted in Figure 3 allows for *Extensibility Element*s, which incorporate more information about *Interaction*s than the basic model of tags and rating. One concrete usage of *Extensibility Element*s in our system is to capture the context of *Interaction*s. In our current prototype we re-use the context model presented in [24] for the VieCAR framework. In this model *Activities* can be modeled, which represent the context in which action has been carried out, e.g., booking a leisure trip. This context information may be used to filter out *Interaction*s which are less relevant, e.g., if a client is searching for a hotel service for a leisure trip interactions which have been conducted in a leisure context are more likely relevant to her than interactions in a business context.

For our prototype we have predefined a number of different contexts for the travel case study as described in Section 3. Currently, users can only select a context from these provider-defined activities. Clearly, these predefined contexts are scenario-dependent. As part of our future work we therefore plan to improve on this facet by allowing

3. http://openid.net/

users to define their own contexts and how they relate to existing ones, e.g., by refining existing contexts.

5. Prototype

We have implemented an end-to-end prototype system as an ASP.NET application on top of the VRESCo SOA runtime environment. The system is implemented in the C# programming language, using .NET 3.5 and the Windows Communication Foundation[4] (WCF) framework.

Our application supports the service selection cycle as described in Figure 2, i.e., the usage of the tool follows the five central steps "Defined Functionality", "Define Constraints", "Select Service", "Use Service" and "Provide Feedback". Required functionality can be selected using the VRESCo construct of *Feature*s. Non-functional constraints are defined by selecting tags describing qualities that the service should or should not exhibit, while the actual selection of a service is based on the rating given by previous users. Finally, the system allows to invoke a service and provide feedback. Note that obviously the system supports only the invocation of the "technical" part of a service, e.g., in the hotel scenario our application supports the invocation of the hotel booking service, but the feedback provided by the user should be based not only on her experience with the booking service, but with the hotel stay in general. Therefore, the last step "Provide Feedback" may be executed significantly after the actual technical invocation of the Web service.

Generally, the system makes heavy use of tag clouds. Tag clouds are used to visualize unstructured user feedback and to support `Feature` selection. For the actual invocation of selected Web services the prototype incorporates the Dynvoker [25] framework, which is able to invoke both SOAP-based and RESTful Web services. Dynvoker allows to generate easy-to-use graphical user interfaces from WSDL or WADL service descriptions. In Figure 5 we have depicted the interfaces to select non-functional properties using tag clouds (Figure 5(a)) and to select services using structured feedback (Figure 5(b)). Users can select tags by dragging-and-dropping tags from the tag cloud onto the "+" and "-" icons. Furthermore, our prototype allows to manage user trust relationships (as described in Section 4), by defining "buddies" and blocking users. Tag merging following one of the three tag merging strategies described in Section 4 has been implemented as a server-side background process. This is because tag merging is computationally expensive, and cannot be done dynamically at request time. Therefore, the tag merging strategy used cannot be selected by the user of the system, but needs to be defined globally by a system administrator.

4. http://msdn.microsoft.com/en-us/netframework/aa663324.aspx

6. Case Study

To illustrate the usefulness of our approach we will now exemplify the usage of our system based on an illustrative case study. For this, we will use feedback on hotels gathered from TripAdvisor[5]. We collected feedback on 40 popular hotels located in Vienna, which includes data about roughly 1750 interactions.

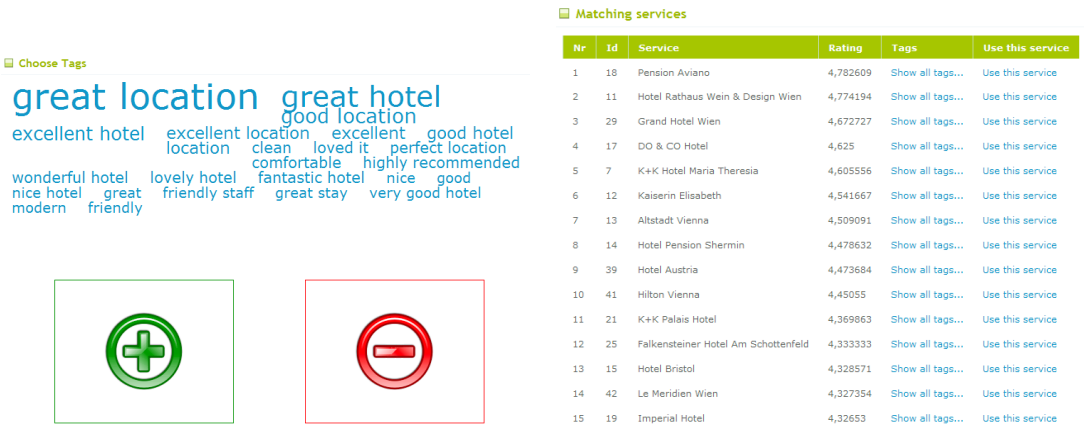
Figure 5(b) shows the top 15 hotels ranked according to their average numerical rating. This is akin to the traditional Web 2.0 way of listing services based on reviews. However, we can see that the difference of the average rating among the top 10 is only marginal. From this numerical ranking alone it is hard to judge which hotel is actually the best to choose. We can now use the unstructured feedback provided by previous users to get a clearer picture. Figure 5(a) shows the tag cloud visualization of the available feedback to the hotels. We have now a more detailed control over which properties we would like the service to exhibit. For instance, we are able to define that a good location is important for us, and that we want the hotel staff to be very friendly. These properties are specified by selecting the tags `good location` and `friendly staff` as required. The list of matching hotels is now significantly shortened to 5 (arguably more relevant) services. Additionally, among these 5 results, the difference in rating is more significant, allowing us to make a more informed decision.

Merging Strategy	Threshold	# Distinct Tags
None	–	1666
Co-Occurence	0.45	1590
Co-Occurence	0.20	1479
Co-Occurence	0.15	1469
Cosine Similarity	0.003	1591
Cosine Similarity	0.001	1503
Cosine Similarity	0.0005	1483

Table 1. Distinct Tags Per Merging Strategy

However, as we can see in Figure 5(a), there are currently many tags with very similar semantics (e.g., `great location`, `excellent location` and `good location`), which are blurring the picture. This can be resolved to some degree by enabling tag merging. In Table 1 we have summarized how various tag merging configurations influence the total number of tags contained in the system. Note that the thresholds used for cosine similarity are much lower than for co-occurence merging. This is due to the fact that cosine similarity values are by definition much lower than the co-occurence values (the arithmetic mean of all co-occurence similarities in our data set is 0.0532, and only 0.0011 for cosine similarity). From Table 1 we can see that cosine similarity merging

5. http://www.tripadvisor.com/



(a) Non-Functional Property Selection Using Tag Clouds

(b) Services Ranked by Structured Feedback

Figure 5. Prototype Screenshots

is in tendency more "conservative", in the sense that tags are merged less frequently. In our case study using co-occurence merging with a threshold of 0.25 produces the most useful results, leading to a rather clear tag cloud (see Figure 6). For reasons of brevity we do not discuss the consequences of including context information and trust in this illustrative example.

Figure 6. Tag Cloud After Merging

To draw some conclusions from this illustrative example, we can state that the general ideas of this paper seem valid when applied to real-world data. We argue that our approach significantly improves selection quality as compared to e.g., UDDI specifically in scenarios where the quality of services is not easily described using typical QoS metrics such as response time alone. However, two limitations should not be left undiscussed. Firstly, it is obvious that our way of filtering for non-functional properties is heuristic: just because no user ever used the tag `clean` or a synonym to describe a hotel it does not necessarily mean that it is dirty. However, this problem decreases with an increasing number of interactions stored, which means that our approach inherently relies on a certain scale of the system (regarding number of users and service alternatives). Secondly, when examining our example data set it became obvious that most user feedbacks are rather positive. In our sample, only about $\frac{1}{10}$ of the tags can be considered negative. After tag merging, some negative tags are contained, but they are still relatively scarce as compared to the positive ones.

7. Conclusion

The IoS is a global service ecosystem, with the two most prominent features being the large scale and the strong involvement of humans, also as service clients. Therefore, novel service discovery methods are necessary which take these characteristics into account. In this paper we have presented one approach which satisfies these requirements. We have explained the notion of QoE, and how QoE can be modeled using feedback from past user transactions. In VRESCo we have integrated a feedback-based service selection mechanism, which uses simple string tags to capture unstructured user feedback, and numerical ratings for structured feedback. Additionally, we use a trust model to prevent service providers from spamming the system, and digital signatures to verify feedback integrity. The context of interactions of users and services is incorporated using an existing context model. Finally, we allow to merge tags with similar semantics. We detailed the implementation of an end-to-end prototype system, and explained some aspects of the system based on an illustrative example.

Our current implementation can be seen as a first step towards a full selection infrastructure. To that end future work has to be carried out in some directions. Firstly, we plan to extend our model to incorporate implicit user feedback besides the explicit feedback discussed here. Such implicit feedback includes monitoring of user behavior and preferences. Secondly, more thought needs to be put into the way context information is used. Currently, contexts can either be "equivalent" or "not equivalent", ignoring the possibility of partially matching contexts. Finally, even though the first experimental results as discussed in Section 6 are promising we still need to conduct some real-life experimentation to further validate the ideas presented.

This is especially relevant to further "fine-tune" the various parameters of the system (e.g., parameters for tag merging, or the numerical values for trust levels as discussed in Section 4).

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement 215483 (S-Cube). We would like to express our gratitude to Christoph Dorn, Daniel Schall, Florian Skopik, Josef Spillner, Jordan Janeiro and Markus Jung for help on various aspects of the paper.

References

[1] C. Schroth and T. Janner, "Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services," *IT Professional*, vol. 9, no. 3, pp. 36–41, 2007.

[2] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," *IEEE Computer*, vol. 40, no. 11, pp. 38–45, 2007.

[3] J. Soriano, D. Lizcano, J. J. Hierro, M. Reyes, C. Schroth, and T. Janner, "Enhancing User-Service Interaction through a Global User-Centric Approach to SOA," in *Proceedings of the Fourth International Conference on Networking and Services (ICNS)*, 2008.

[4] A. van Moorsel, "Metrics for the Internet Age: Quality of Experience and Quality of Business," HP Labs, Tech. Rep., 2001.

[5] A. Mathes, "Folksonomies – Cooperative Classification and Communication Through Shared Metadata." [Online]. Available: http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html

[6] E. Michlmayr, "A Case Study on Emergent Semantics in Communities," in *Proceedings of the Workshop on Social Network Analysis, International Semantic Web Conference (ISWC)*, 2005.

[7] A. Michlmayr, F. Rosenberg, C. Platzer, and S. Dustar, "Towards Recovering the Broken SOA Triangle – A Software Engineering Perspective," in *Proceedings of the International Workshop on Service Oriented Software Engineering (IW-SOSE'07)*, 2007.

[8] L.-H. Vu, M. Hauswirth, and K. Aberer, "QoS-based Service Selection and Ranking with Trust and Reputation Management," in *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, 2005.

[9] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma, "A QoS-Aware Selection Model for Semantic Web Services," in *Proceedings of the 6th International Conference on Service-Oriented Computing (ICSOC)*, 2006.

[10] S. A. Golder and B. A. Huberman, "Usage Patterns of Collaborative Tagging Systems," *Journal of Information Science*, vol. 32, no. 2, pp. 198–208, 2006.

[11] S. Tai, N. Desai, and P. Mazzoleni, "Service Communities: Applications and Middleware," in *Proceedings of the 6th International Workshop on Software Engineering and Middleware (SEM)*, 2006.

[12] H. Meyer and M. Weske, "Light-Weight Semantic Service Annotations through Tagging," in *Proceedings of the 6th International Conference on Service-Oriented Computing (ICSOC)*, 2006.

[13] I. Silva-Lepe, R. Subramanian, I. Rouvellou, T. Mikalsen, J. Diament, and A. Iyengar, "SOAlive Service Catalog: A Simplified Approach to Describing, Discovering and Composing Situational Enterprise Services," in *Proceedings of the 6th International Conference on Service-Oriented Computing (ICSOC)*, 2008.

[14] P. Resnick and H. R. Varian, "Recommender Systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[15] U. S. Manikrao and T. V. Prabhakar, "Dynamic Selection of Web Services with Recommendation System," in *Proceedings of the International Conference on Next Generation Web Services Practices (NWESP)*, 2005.

[16] A. Birukou, E. Blanzieri, P. Giorgini, and N. Kokash, "Improving Web Service Discovery with Usage Data," *IEEE Software*, vol. 24, no. 6, pp. 47–54, 2007.

[17] R. Jurca, B. Faltings, and W. Binder, "Reliable QoS Monitoring Based on Client Feedback," in *Proceedings of the 16th International Conference on World Wide Web (WWW)*, 2007.

[18] S. Börzsönyi, D. Kossmann, and K. Stocker, "The Skyline Operator," in *Proceedings of the 17th International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 421–430.

[19] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar, "End-to-End Support for QoS-Aware Service Selection, Invocation and Mediation in VRESCo," TUV-1841-2009-03, Vienna University of Technology, Tech. Rep., 2009.

[20] F. Rosenberg, P. Leitner, A. Michlmayr, and S. Dustdar, "Integrated Metadata Support for Web Service Runtimes," in *Proceedings of the Middleware for Web Services Workshop (MWS'08), co-located with the 12th IEEE International EDOC Conference*, 2008.

[21] C. Cattuto, D. Benz, A. Hotho, and G. Stumme, "Semantic Analysis of Tag Similarity Measures in Collaborative Tagging Systems," May 2008. [Online]. Available: http://arxiv.org/abs/0805.2045

[22] M. Baldauf, S. Dustdar, and F. Rosenberg, "A Survey on Context-Aware Systems," *International Journal of Ad Hoc Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, 2007.

[23] G. Palla, I. J. Farkas, P. Pollner, I. Derenyi, and T. Vicsek, "Fundamental statistical features and self-similar properties of tagged networks," *New Journal of Physics*, vol. 10, 2008.

[24] D. Schall, C. Dorn, S. Dustdar, and I. Dadduzio, "VieCAR – Enabling Self-adaptive Collaboration Services," in *Proceedings of the 34th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, 2008.

[25] J. Spillner, M. Feldmann, I. Braun, T. Springer, and A. Schill, "Ad-Hoc Usage of Web Services with Dynvoker," in *Proceedings of the 1st European Conference Towards a Service-Based Internet (ServiceWave)*, 2008.