Integrating Preferences into Service Requests to Automate Service Usage[*]

Michael Klein[1] and Birgitta König-Ries[2]

[1] Institute for Program Structures and Data Organization, Universität Karlsruhe, 76128 Karlsruhe, Germany, kleinm@ipd.uni-karlsruhe.de
[2] Institute of Computer Science, Friedrich-Schiller-Universität Jena, 07743 Jena, Germany, koenig@informatik.uni-jena.de

1 Introduction

Today, web services are often used as a technology to integrate functionality of different entities. However, one important potential of service oriented computing is not exploited: the ability to form agile networks. Here, service requestors and service providers are not fixedly tied together, rather, bindings to inefficient or unavailable service providers are transparently replaced by bindings to more appropriate providers at runtime. In such an architecture, the robustness and efficiency would be increased dramatically.

The main reason why these networks are not a reality today is that current technologies do not allow for automatic service selection and invocation; rather, they require human interaction to decide on an appropriate service provider. Obviously, this approach is not feasible in a system where service selection needs to be carried out repeatedly at run time.

The most challenging prerequisite for automatic service description is an appropriate service description language. This language needs to be able to capture service offers and requests in sufficient detail to allow for automatic matchmaking. In this paper, we argue that such a language needs to explicitly incorporate user preferences into service requests.

2 Problems with the State of the Art

Many existing languages for service description use the same technique for describing requests and offers: the requesting application describes its desired functionality by specifying an instance of the "perfect" service. A generic matchmaker compares this request to the published offer descriptions and calculates a similarity as a value from the interval$[0.0, 1.0]$ by using heuristical structural and/or semantical similarity metrics. The service offer with the highest similarity is invoked directly by the service requestor.

Much effort has been put into the development of intelligent similarity calculators. The most important approaches perform a comparison of the functional parameters, perform a structural comparison of the description graphs, use logic subsumption or use combined approaches [2–5].

[*] This work is partially funded by the Deutsche Forschungsgemeinschaft (DFG) within SPP 1140. Some of the ideas in this paper have been published in [1].

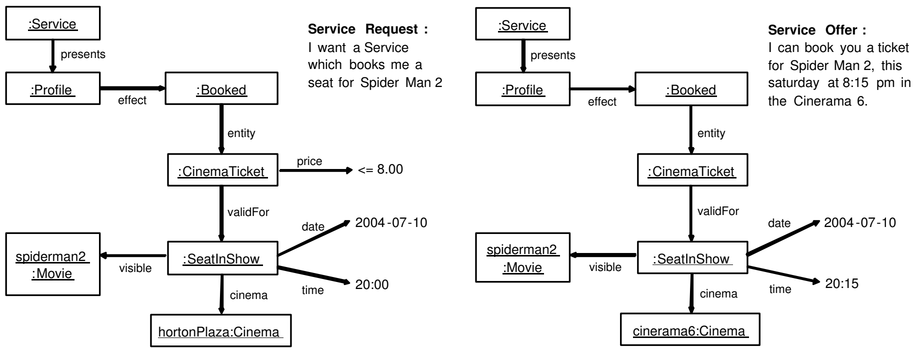

Fig. 1. Request description as perfect service (left) and offer description (right).

However, these approaches only work well, if offer and request description are exactly equal, so the matcher returns 1.0, or obviously different, so the matcher returns 0.0. However, in intermediate situations, in which the offer differs somewhat from the request, it becomes very difficult for the matcher to assign the value from $(0, 1)$ that is appropriate, i.e., that reflects the requestor's perception of the usefulness of the service offered.

An example shall illustrate this. The request on the left of Figure 1 is specified as one single instance[3], which represents the requestor's ideal service. He wants to invoke a service reserving a seat for Spiderman 2. He also gives some information about his perfect reservation: It should be in the cinema Horton Plaza, at a given date and time and the ticket's price should be 8 Euro or less.

On the other hand, we have a service provider which offers a nearly matching service offer (see Figure 1 (right)). He offers to book a ticket for Spiderman 2, but differs in some of the requested attributes[4]. The matcher now has to decide:

– The requestor wanted 20:00 as starting time, but the offered service can only reserve a ticket for 20:15. Is this still a match or only a 90% match?
– The requestor wanted the Horton Plaza cinema, but the offer is a about the cinema Cinerama 6. Is this still ok because they are in the same city[5]?
– The requestor wanted a price below 8 Euro, but the offer didn't mention the price. Is this a matching value of 0.0, or should some other value be assigned?
– What is more important for the offerer: A good price, a good time, a near cinema? The matcher has to decide whether to take the average of the individual matching values, their minimum or another function.

As we can see, the main problem lies in the fact that the preferences of the requestor are not clear as they are not explicitly specified anywhere. This matchmaker has to either use general, domain- and user-independent deviation heuristics or simply perform a very strict, conservative matching. Each of these

[3] We use a graphical, UML-like notation of the description.
[4] Realistically, the provider would offer a more generic service like the booking of arbitrary movie tickets. Our approach can handle this by introducing variables [1].
[5] This information could be provided in the underlying ontology.

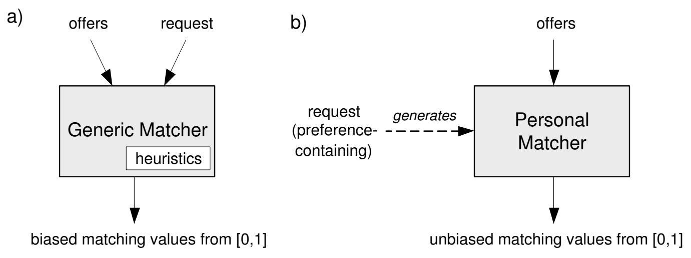

Fig. 2. Generic vs. personal matcher.

approaches leads to a biased matching process, i.e. the result of the match depends on the matchmaker used. Thus, the requestor typically will not blindly rely on its result but will want to choose one of the proposed services manually. Thus, automatic service invocation is prevented.

3 Approach: Preference Integration

As shown in Section 2, only an unbiased matching process could be accepted within an automatic service usage process. Such an unbiased matcher can only become a reality, if the service request contains enough information for the matcher to decide in deviation cases. This means, that the service request has to include the client's preferences.

Figure 2 illustrates this idea. We have to overcome the approach of a generic, all-purpose matcher (left side). Such a matcher is biased. It is not able to calculate reasonable matching results for service descriptions from arbitrary application domains that can be used to automatically choose an appropriate service for the requestor. Instead, we need *preference-containing* request descriptions that can be used to generate a highly specialized, personal matcher (right side). Such a matcher would be unbiased so that the requestor would agree to automatically invoke the best matching service.

To achieve this goal, we propose to express requests as fuzzy declarative sets of suitable services rather than as one specific instance representing the perfect service. The degree of membership of a service offer to the fuzzy set described in the service request expresses the requestor's preference for this offer and also its matching value. Requests are build up by using a limited set of well-defined constructors, which leads to structured and computationally feasible service requests. Consider as an example the request shown in Figure 3. Here, sets are depicted as rectangles with a small cross line in the left upper corner.

Again, the requestor is looking for a ticket for Spiderman 2. In her request, she specifies that she is not willing to see another movie or to see the movie on another than the specified date. However, she is willing to accept a slightly later or earlier time than her preferred starting time (expressed by the `~==[15min]20:00` condition for the time property of SeatInShow) and is also willing to attend a show in a theater close to the one that is her first choice (expressed by the similarity function `near`. Here, either a predefined function from the ontology

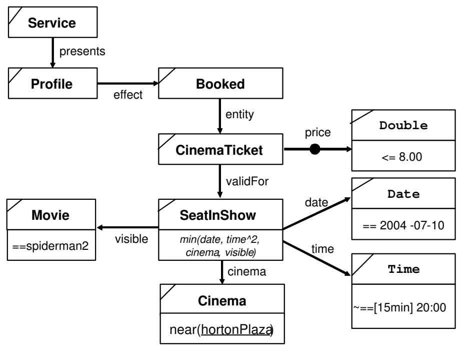

Fig. 3. Preference-containing request.

can be used or the user specifies her own function). While she would prefer not to pay more than 8 Euros, offers that do not specify a price should also be considered as possible matches (based on the real world experience of the user that cinema tickets seldom cost more than eight Euros). This is expressed by the *missing strategy* `assume_fulfilled`, which is depicted by the circle. The *connecting strategy* `min(cinema,visible,date,time^2)` expresses how the individual matching values should be combined. Here, the requestor has stated that the result has to be calculated by minimizing the results from the single conditions where the `time` attribute is emphasized by the exponent 2. As a result, this is a conjunctive connection. Given all this information, it is straightforward to generate a personalized matcher that will be able to determine exactly how well a service offer fulfills the requestor's needs [6].

Our service description language, DIANE Service Description (DSD) [1, 7], offers the means to express such requests. DSD and the corresponding matcher have been implemented.

References

1. Klein, M., König-Ries, B.: Combining query and preference - an approach to fully automatize dynamic service binding. In: Short paper at IEEE International Conference on Web Services (ICWS 2004), San Diego, CA, USA (2004)
2. Paolucci, M., Kawmura, T., Payne, T., Sycara, K.: Semantic matching of web services capabilities. In: Proc. of the Semantic Web Conf., Sardinia, Italy (2002)
3. Trastour, D., Bartolini, C., Gonzalez-Castillo, J.: A semantic web approach to service description for matchmaking of services. In: Proc. of the Intl. Semantic Web Working Symposium (SWWS), Stanford, CA, USA (2001)
4. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: Proc. of the Intl. WWW Conference, Budapest, Hungary (2003)
5. Sycara, K., Widoff, S., Klusch, M., Lu, J.: Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. Autonomous Agents and Multi-Agent Systems **5** (2002) 173–203
6. Klein, M., König-Ries, B.: Coupled signature and specification matching for automatic service binding. In: Proc. of ECOWS 2004, Erfurt, Germany (2004)
7. Klein, M., König-Ries, B., Müssig, M.: What is needed for semantic service descriptions? Intl. Journal on Web and Grid Services (2005). Submitted for publication.