A Conversational Approach to Semantic Web Service Selection

Friederike Klan and Birgitta König-Ries

Institute of Computer Science, Friedrich-Schiller-University Jena
{friederike.klan,birgitta.koenig-ries}@informatik.uni-jena.de

Abstract. Service consumers typically have no clear goal in mind when looking for service functionality and are not able to formulate their service needs in a formal or semi-formal language. We approach those issues by proposing a mechanism that implements semantic service selection as an incremental and interactive process alternating phases of intermediate service recommendation and requirements refinement by critiquing the presented alternatives. It thus facilitates the incremental construction of service requirements and their specification at an informal level. Our evaluation results demonstrate the effectiveness and efficiency of the proposed approach in an e-commerce domain.

Keywords: semantic web service selection, incremental, interactive.

1 Introduction

Semantic Web Services (SWSs) are an active area of research and are at the focus of numerous EU funded research projects . However, up to now virtually no real-world applications that use this technology are available [1]. This is particularly bad, since SWS technology and its advanced semantic description and discovery mechanisms have the potential to significantly improve existing retrieval-based applications such as required in e-commerce or web search. While Web Services (WSs) were originally envisioned for B2B applications, early on first ideas to also use them in B2C settings were developed [2, 3]. However, at this stage there are some serious barriers to the realization of this vision. In current SWS research, the focus is on the support of application developers and service providers. The end-users, i.e. service consumers, who require assistance in expressing their service needs and support in the subsequent process of service selection, are only marginally addressed. Hence, though SWS approaches provide adequate means to semantically describe service capabilities and in particular service needs, they require the user to do this at a formal, logic-based level that is not appropriate for many service consumers, e.g. in an e-commerce setting. Basic tools that support the task of requirements specification exist, but mainly address WS developers. Virtually, no end-user support for specifying service requirements is available. Moreover, existing approaches to SWS selection typically assume that service consumers have a clear goal in mind when looking for service functionality. However, as research results from behavioral decision theory indicate [4], this is often not true. Particularly, in service domains that are complex and unfamiliar, consumers have no clear-cut requirements and preferences. People rather construct them instantaneously when facing choices to be made [4]. Current SWS solutions do not account for those facts.

In this paper, we approach the addressed issues. We propose a mechanism that implements service selection as an incremental and interactive process with the service consumer. Inspired by conversational recommender systems [5] and example critiquing [6], it alternates phases of intermediate service recommendation and requirements refinement by critiquing the presented alternatives. It thus facilitates the incremental construction of service requirements. Moreover, it enables the user to specify his requirements at an informal level, either directly via a graphical representation of the requirements or indirectly by critiquing available service offers. Finally, the proposed solution considers the system's uncertainty that arises from incomplete and inaccurate knowledge about the user's true service requirements. We will show that by leveraging this information, the system is able to effectively direct and focus the requirements elicitation and specification process. As already argued, these are key requirements for the realization of SWS-based end-user applications. Our evaluation results will demonstrate the effectiveness and efficiency of the proposed approach in an e-commerce domain.

The remainder of the paper is structured as follows. As a basis for the further discussion, Sect. 3 briefly introduces the semantic service description language DSD [7] that underlies our approach. Sect. 4 constitutes the main part of the paper. It provides a detailed description of our approach to incremental and interactive service selection. After that, we present our evaluation results (Sect. 5), briefly comment on existing approaches and conclude with (Sect. 6).

2 Basic Idea

We suggest a solution that implements service selection as an iterative and interactive process that alternates phases of intermediate service recommendation and requirements refinement. During that process, the user incrementally develops his service requirements and preferences and finally makes a selection decision. To effectively support him in that tasks, the system maintains an internal model of the consumer's requirements, which we call *request model.* Uncertainty about the service consumer's true requirements is explicitly represented within this model. During the refinement process the request model is continuously updated to accurately reflect the systems's growing knowledge about the user's service requirements and preferences. Starting with a request model constructed from the user's initially specified service needs, the system determines the set of service alternatives that fit to these requirements. The service alternatives are determined by transforming the internal request model into a semantic service request that reflects the requirements specified in the model, but also the system's uncertainty about this model. We will demonstrate that standard matchmaking with a minor extension can be applied to retrieve matching service results sorted by their expected overall preference value. The user may then critique

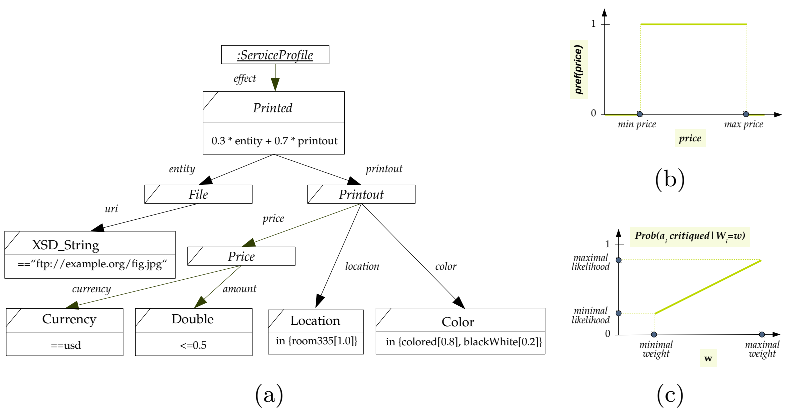


Fig. 1. DSD service request (a), preference function (b), linear likelihood function (c)

the presented service alternatives and thereby indicate desirable service characteristics. He can also specify new service requirements by directly modifying the internal request model via a graphical representation (not described in this paper). This will allow him to correct the system if necessary, will help him to become aware of his requirements and will enable him to actively develop them. All user interactions trigger appropriate model changes. Once modifications to the request model have been made, the user may decide to see service results fitting to the updated requirements. The process continues until the user finds an appropriate service among the presented alternatives or until he decides to stop without making a selection.

3 Diane Service Description

As a basis for the further discussion, we introduce the semantic service description language DSD (DIANE Service Description) [7] and its mechanisms for automatic semantic service matchmaking that underlie our approach. Similarly to other semantic service description approaches, DSD is ontology-based and describes the functionality a service provides as well as the functionality required by a service consumer by means of the precondition(s) and the possible effect(s) of a service execution. For instance, in the service request in Fig. 1(a), the desired effect is that something is printed after service execution. A single effect corresponds to a particular service instance that can be executed. While service offer descriptions describe the individual service instances that are offered by a service provider, e.g. possible printing jobs offered by a printer, service request descriptions declaratively characterize service instances that are acceptable for a consumer. In the service request in Fig. 1(a), acceptable instances are printing jobs that print the file located at "ftp://example.org/fig.jpg", that cost at most 0.5$ and where the printout is either colored or black-and-white and located at room 335.

As can be seen in the example, DSD utilizes a specific mechanism to declaratively and hierarchically characterize (acceptable) service effects. Service effects are described by means of their attributes, such as price or color. Each attribute has an ontological type and may be constrained by direct conditions on its values and by conditions on its subattributes. For instance, the attribute printout is constrained by a condition on its subattribute location, which indicates that only printouts located at room 335 are acceptable. The direct condition $<= 0.5$ on the price amount in Fig. 1(a) indicates that only prices lower than 0.5\$ are acceptable. Attribute conditions induce a tree-like and more and more fine-grained characterization of acceptable service effects, where possible attributes as well as their required type are defined in an ontology. A DSD request does not only specify which service effects are acceptable, but also indicates to which degree they are acceptable. In this context, direct conditions specify a preference value from $[0, 1]$ for each considered attribute value (the default is 1.0 (totally acceptable)). DSD also allows for the specification of connecting strategies that indicate how the preference values of a certain attribute's subattributes shall be combined (see e.g. the effect-attribute in Fig. 1(a)).

4 Incremental and Interactive Service Selection

Request Model. We propose a request model that builds on DSD request descriptions (Sect. 3). In particular, it inherits its tree structure with the typed service attributes as its nodes. This will later on allow us to use the DIANE matchmaker to compare the user's (uncertain) requirements with the offered service functionality. The request model supports three types of direct conditions on attribute values: range conditions, in- and not-in-conditions[1]. A range condition defines a range of acceptable values for a certain attribute, e.g. a range of acceptable prices, and a preference function that assigns a preference value to each possible value of this attribute. The preference function is parameterized with the minimum and the maximum value of the range. We do not make any assumptions about the type of this preference function. However, it should appropriately model the user's preferences. Fig. 1(b) shows a simple example of a preference function for the attribute price. In-conditions allow the user to specify attribute values that are acceptable for him as well as a preference value for each of those values, e.g. 0.8 for colored printouts. At this time, the request model implementation supports only weighted sum as a connecting strategy. Though DSD descriptions are well suited for modeling service requirements and preferences, they are not capable of representing uncertainty associated with the model. To compensate for that, we propose the following extension that allows to represent uncertainty about direct conditions and connecting strategies. We do not model uncertainty about the structure of a request. Uncertainty about range conditions is modeled by probability distributions $prob_{Min}(x)$ and $prob_{Max}(x)$, which provide the likelihood of attribute value x being the minimum/maximum of the range. In-conditions are modeled as a set of probabilities $\{Prob_{in}(x)\}$ over possible values x of an attribute, where the probability $Prob_{in}(x)$ provides the likelihood of attribute value x being

[1] Not considered in this paper.

acceptable for the user. The preference values $\{pref(x)|Prob_{in}(x) \neq 0\}$ for acceptable attribute values are user-provided. We do not consider uncertainty about a user's preference for a certain attribute value. Analogously, we cope with uncertainty related to connecting strategies. While their type is fixed to weighted sum, the parameters of the strategy, i.e. the weights are unknown. Uncertainty about those parameters is modeled by probability distributions $prob_{W_1}, \ldots, prob_{W_n}$, where $prob_{W_i}$ is a probability distribution over the possible weights of attribute a_i. Weights are absolute and taken from $[0, 1]$. The probability $prob_{W_i}(w)$ provides the likelihood of the weight for attribute a_i being w.

Uncertain Matchmaking. To understand how uncertain matchmaking based on the request model can take place, we first have to look at how certain, i.e. standard DSD service requests, are matched against available service offers. In the DIANE matchmaker [7], the comparison of the effect(s) described in the request and those described in the offer descriptions is recursive and proceeds as follows. Starting from the effect attribute of the request, the matchmaker checks in each step, whether the service effect(s) described in the offer fulfill(s) the conditions in the request. Proceeding to the leaves of the request results in a preference value for each of those attributes. In a final pass, those values are aggregated to an overall preference value ($\in [0, 1]$) for each offered service instance, i.e. to a preference value for the effect attribute a of the request. This value is recursively defined as the product of the matching degree $M_{type}(a)$ between the type of a and that of its corresponding offer attribute (the attribute lying on the same path), by the user's preference $Pref_{dc}(a)$ for the offered attribute values (specified in the direct conditions) and the aggregated preference value $Pref_{sub}(a)$ for the attribute's subattributes. The latter is determined by the connecting strategy specified for the attribute, i.e. a weighted sum of the subattributes' preference values.

The requirements specified in the request model are uncertain. In particular, there is uncertainty about the user's preference for the offered attribute values as well as uncertainty about the importance of attributes, i.e. the weights of the connecting strategies. Hence, uncertain matchmaking can only deliver an expected preference value for each offered service instance. We will show that only minor changes to the matchmaker are required to implement this. Using well-known properties of expected values, it can be easily shown that the expected aggregated preference value $\mathbb{E}(Pref_{sub}(a))$ of an attribute a's subattributes w.r.t. a given offer o, i.e. the expected weighted sum of its subattribute's preference values, is given by

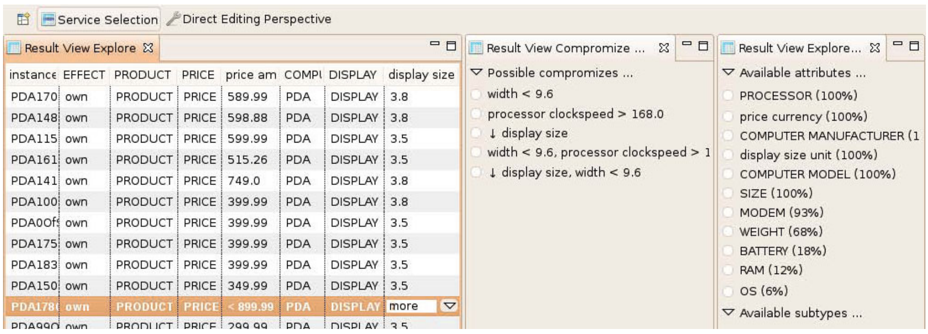
$$\mathbb{E}(Pref_{sub}(a)) = \mathbb{E}\left(\frac{\sum_{i=1}^{n}(W_i \cdot Pref(a_i))}{\sum_{j=1}^{n} W_j}\right) = \sum_{i=1}^{n}\left(\frac{1}{\sum\limits_{j=1, j\neq i}^{n} \frac{\mathbb{E}(W_j)}{\mathbb{E}(W_i)} + 1} \cdot \mathbb{E}(Pref(a_i))\right),$$

where $\mathbb{E}(W_i)$ is the expected weight of subattribute a_i and $\mathbb{E}(Pref(a_i))$ its expected preference value. Hence, assuming that the matchmaker is provided with the expected preference values for the offered attribute values (needed to compute $Pref_{dc}(a)$), it will return the expected preference value of the request model, when receiving a request whose attribute weights w'_i are defined by $1/\sum_{j=1,j\neq i}^{n} \frac{\mathbb{E}(W_j)}{\mathbb{E}(W_i)} + 1$ as input. This is convenient, since we achieve the desired matchmaking functionality by simply transforming the request model into a standard DSD request. We do not have to make any changes to the matchmaker's implementation. Unfortunately, the expected preference values for the offered attribute values cannot always be pre-computed. Provided that a direct condition has been specified for a given attribute, the expected preference value $\mathbb{E}(\mathit{Pref}(x))$ for an offered attribute value x is given by $\mathit{Prob}_{in}(x) \cdot \mathit{pref}(x)$. Consequently, it is sufficient to provide those expected values for all attribute values that are specified in an in-condition to the matchmaker. The expected preference value $\mathbb{E}(\mathit{Pref}(x))$ of an element x, when given a range condition with the distributions prob_{Min} and prob_{Max} and a preference function *pref(x)* as depicted in Fig. 1(b) is given by

$$\mathbb{E}(\mathit{Pref}(x)) = \mathit{Prob}_{(Min<x)\wedge(Max\geq x)} = \int\limits_{z=0}^{x} \mathit{prob}_{Min}(z)dz \cdot (1 - \int\limits_{z=0}^{x} \mathit{prob}_{Max}(z)dz).$$

Since we cannot pre-calculate this value for all attribute values that might potentially appear in an offer, we have to supply the matchmaker with a routine that computes this preference value to implement this. Summarizing, we can state that matching uncertain service requirements as specified in the request model can be implemented by generating a standard DSD service request with the properties detailed above and matching it with a slightly modified version of the standard matchmaker.

Adjusting the Service Results. Once the service offers, that match to the requirements that are specified in the request model, have been retrieved, the list of those offers sorted by their expected overall preference value is presented to the user (Fig. 2 left). This result table includes a column for each service attribute that is specified within the request model. The cells of a column show either the value of the corresponding attribute as specified in the depicted offer or the type of the attribute, if no value has been specified. Columns can be hidden and sorted to facilitate decision making. Besides viewing these service offers, the user may indicate desirable service characteristics based on the presented alternatives. The system supports three ways of doing this: (1) by adding a not yet specified attribute to the request model, (2) by refining, i.e. subtyping, an attribute's type and (3) by critiquing one of the listed service offers. To support the first two interaction opportunities, the system suggests the user a list of service attributes and attribute types that are specified in the matching service offers, but have not yet been included into the request model (Fig. 2 right). The suggested attributes are restricted to those that can be directly added as a subattribute to one of the service attributes that are already part of the request model. As soon as the user selects an attribute or a type, the request model is updated accordingly, matching offers are retrieved and presented to the user. In addition to these interaction opportunities, the user may select a service from



instance	EFFECT	PRODUCT	PRICE	price am	COMPU	DISPLAY	display size
PDA170	own	PRODUCT	PRICE	589.99	PDA	DISPLAY	3.8
PDA148	own	PRODUCT	PRICE	598.88	PDA	DISPLAY	3.8
PDA115	own	PRODUCT	PRICE	599.99	PDA	DISPLAY	3.5
PDA161	own	PRODUCT	PRICE	515.26	PDA	DISPLAY	3.5
PDA141	own	PRODUCT	PRICE	749.0	PDA	DISPLAY	3.8
PDA100	own	PRODUCT	PRICE	399.99	PDA	DISPLAY	3.8
PDA0Of	own	PRODUCT	PRICE	399.99	PDA	DISPLAY	3.5
PDA175	own	PRODUCT	PRICE	399.99	PDA	DISPLAY	3.5
PDA183	own	PRODUCT	PRICE	399.99	PDA	DISPLAY	3.5
PDA150	own	PRODUCT	PRICE	349.99	PDA	DISPLAY	3.5
PDA178	own	PRODUCT	PRICE	< 899.99	PDA	DISPLAY	more
PDA990	own	PRODUCT	PRICE	299.99	PDA	DISPLAY	3.5

Possible compromizes ...
- width < 9.6
- processor clockspeed > 168.0
- ↓ display size
- width < 9.6, processor clockspeed > 1
- ↓ display size, width < 9.6

Available attributes ...
- PROCESSOR (100%)
- price currency (100%)
- COMPUTER MANUFACTURER (1
- display size unit (100%)
- COMPUTER MODEL (100%)
- SIZE (100%)
- MODEM (93%)
- WEIGHT (68%)
- BATTERY (18%)
- RAM (12%)
- OS (6%)

Available subtypes ...

Fig. 2. Result view

the presented list of offers that fits reasonably well to his requirements. He may then indicate desirable service properties relative to this offer. For example, the user might indicate that the offer is fine, but too expensive (Fig. 2 left). This can be done by simply clicking on the referenced attribute value. Based on the indicated property and the properties of the available service alternatives that fulfill this requirement, the system produces a list of trade-off alternatives on other service aspects that the user has to accept when insisting on the indicated requirement. For example the system might indicate that the user has to accept a lower display size when critiquing on the price of a computer offer (Fig. 2 middle). After viewing the existing trade-off alternatives, the user can either decide to abandon his requirement, specify an additional requirement on the same service offer or he indicates that he is willing to accept one of the presented trade-off alternatives by clicking it. While the second option will lead to another set of trade-off alternatives that are produced by taking both requirements of the user into account, the third option will result in a model update reflecting the information provided by the user. In this context, trade-offs do not necessarily refer to service aspects that have been already considered in the request model, but may also refer to service attributes that have not yet been specified by the user. In this case, the value of the compromised attribute is depicted within the trade-off alternative. The presented feature encourages the user to make compromises where necessary and helps him in identifying yet unconsidered, but important service aspects. Presenting details on the implementation of this feature is out of the scope of this paper.

To effectively reduce uncertainty about the user's service requirements, we propose to direct and focus the process of requirements elicitation by suggesting those interaction opportunities to the user that have a high potential to increase the system's knowledge about the consumer's service requirements, i.e. his preferences for the available offers. Thereby, knowledge acquisition should concentrate on those aspects of the user's requirements that are relevant in light of the available service options and in light of the user's known requirements. Consider for example a flight booking scenario. If all available services offer food during the flight, then there is no need to know whether the user would also accept flight offers without this service. As well, if price is not relevant to a consumer's service selection decision, then it makes no sense to explore in detail which prices are more desirable for this user. In this paragraph, we will introduce a measure that covers this notion of uncertainty about the user's requirements and that can be leveraged to identify promising interaction opportunities.

Given a request model, that represents the user's known service requirements, we measure the uncertainty about the user's true preference for a service offer o as follows. Let a be an attribute of the request model, a' its corresponding attribute in o and $\sum_{i=1}^{n} W_i \cdot \mathit{Pref}(a_i)$ the connecting strategy defined over a's subattributes a_1 to a_n. If both, a and a' are specified, the system's uncertainty $U(\mathit{Pref}(a)) \in [0,1]$ about the user's preference value for o w.r.t. a is defined as the product of $M_{type}(a)$ and

$$U(\mathit{Pref}_{dc}(a)) \cdot U(\mathit{Pref}_{sub}(a)) \oplus \mathbb{E}(\mathit{Pref}_{dc}(a)) \cdot U(\mathit{Pref}_{sub}(a)) \oplus \mathbb{E}(\mathit{Pref}_{sub}(a)) \cdot U(\mathit{Pref}_{dc}(a)),$$

where $a \oplus b = a + b - a \cdot b$ and a higher value indicates higher uncertainty[2]. The intuition behind this definition is that the uncertainty about the user's preference value w.r.t. the attribute a is high, if and only if $M_{type}(a)$ is high and we either do not know much about both, $\mathit{Pref}_{dc}(a)$ and $\mathit{Pref}_{sub}(a)$, (first term), we are quite sure that the preference $\mathit{Pref}_{dc}(a)$ is high, but we do not know much about $\mathit{Pref}_{sub}(a)$ (second term) or we are quite sure that $\mathit{Pref}_{sub}(a)$ is high, but we do not know much about $\mathit{Pref}_{dc}(a)$ (third term). In case just a' is specified, we define $U(\mathit{Pref}(a)) = 1$. The uncertainty $U(\mathit{Pref}_{sub}(a))$ about the aggregated preference value for a's subattributes w.r.t. o is recursively defined by

$$U(\mathit{Pref}_{sub}(a)) = U(S_1) \cdot U(S_{2n}) \oplus \overline{\mathbb{E}(S_1)} \cdot U(S_{2n}) \oplus \overline{\mathbb{E}(S_{2n})} \cdot U(S_1),$$

where $S_i := W_i \cdot \mathit{Pref}(a_i)$, $S_{jn} := \sum_{i=j}^{n} W_i \cdot \mathit{Pref}(a_i)$ and $\mathbb{E}(S_{jn})$ and $\mathbb{E}(S_i)$ are the corresponding expected values. This means that the system's uncertainty about the aggregated preference value for a's subattributes is high, if either, the uncertainty about all the subattributes' preference values and weights is high (first term), the uncertainty about the preference values and weights of the subattributes a_2 to a_n is high and we are quite sure that S_1 is low (second term) or the uncertainty about the preference value and weight of subattribute a_1 is high and we are quite sure that S_{2n} is low (third term). The uncertainty $U(S_i)$ about the value of the product $W_i \cdot \mathit{Pref}(a_i)$ is similarly defined by

$$U(S_i) = U(W_i) \cdot U(\mathit{Pref}(a_i)) \oplus \mathbb{E}(W_i) \cdot U(\mathit{Pref}(a_i)) \oplus U(W_i) \cdot \mathbb{E}(\mathit{Pref}(a_i)).$$

The uncertainty $U(W_i)$ about the weight W_i is defined to be the Shanon entropy of the probability distribution $prob_{W_i}$ normalized to the interval $[0,1]$. Thanks to the tree-structure of DSD service offers and the request model, we can determine the system's uncertainty about the user's true preference for the service offer o w.r.t. the given request model by recursively computing $U(a)$ for the effect attribute a of o. Based on the proposed measure, we can determine those interaction opportunities, i.e. those subtypes, subattributes and trade-off alternatives, that, when

[2] We omit details about the definition of $U(\mathit{Pref}_{dc}(a))$.

selected, have the highest potential to reduce the system's uncertainty about the consumer's preferences for the offered services, and offer them to the user.

Model Update. Since the implementation of structural updates to the request model, e.g. when adding an attribute, is straight forward, we focus on the update of the probability distributions maintained within the model. We start with the weight distributions. Upon the addition of an attribute to the request model, a corresponding uniform probability distribution for its weight is created. This probability distribution is affected by two types of interactions, namely, either because the user directly adjusted the weight of the attribute via the graphical representation of the request model or because the user chose a compromise after critiquing one of the listed service offers. In both cases, a Bayesian update on the affected weight distributions is performed. The updated distribution is given by $prob_{W_i}(w|interaction) = c \cdot Prob(interaction|W_i{=}w) \cdot prob_{W_i}(w)$, where $prob_{W_i}(w)$ is the distribution before the update, $Prob(interaction|W_i{=}w)$ is the likelihood function indicating the likelihood of observing the interaction when the attribute's true weight is w and c is a normalizing constant. If a compromise was chosen by the user, the weight distributions of the critiqued and compromised attributes are adjusted. Weight distributions for attributes that have not yet been considered in the request model are created. In case of the critiqued attributes, we use a linear likelihood function increasing with the true weight w (see Fig. 1(c)). The intuition behind that is, that it is more likely that the user will critique an attribute, if it is important, i.e. it has a high weight. In case of the compromised attributes, we use the same update distribution, but reflected at its expected value. This is, because it is less likely that the user will compromise an attribute, if it is important, i.e. it has a high weight. Since the overall weight of an attribute is determined by the weights of all its parent attributes in the request model, we also adjust the weights of those attributes in a similar fashion. However, we reduce the impact of the update by decreasing *maxProb* the higher we get in the request model tree. We omit details on the direct model update via the graphical representation. The distributions related to range conditions are updated either when the user adjusts the minimum or maximum of the range via the graphical representation of the request model or when the user selected a compromise. In the latter case, the critiqued as well as the compromised attributes' range distributions are adjusted as indicated by the critiques and the chosen compromises. We omit details on this as well as on the update of the probabilities related to (not-)in-conditions.

5 Evaluation

In the evaluation, we wanted to find out, whether users that are not familiar with WSs and SWS descriptions were able to formulate their service needs by using our system, whether they were able to find the service functionality they desire and whether they felt supported in that task. To have a realistic set of services, we used information about computer items extracted from Amazon.com to generate semantic descriptions of services selling computer items.

We performed a preliminary evaluation with 5 test users. None of them was familiar with WSs. The test setting was inspired by [8]. Users were first asked to think about the type of service they would like to use. They were allowed to choose from 8 categories. Participants were then provided with a questionnaire containing questions related to their background and their initial service requirements. After a 5 minutes introduction into our system, the users were asked to use the tool to select the service offer that best suits to their requirements from a collection of 50 services. To make the choice more difficult, all offers presented to the user were taken from the selected service category. The users started with an empty request model, i.e. no specified attributes. Once a user made his final selection, he was asked to complete a second questionnaire comprising of questions about his (updated) service requirements, his confidence in the specified requirements and the selected service and questions related to the usefulness of the provided tool. To verify the suitability of the service that was selected by the user, we provided him with a list of all service offers and their properties. The participant was then asked to look through this list and check whether there is an offer other than the selected that fits better to his requirements. During the test, the user's interactions with the tool as well as the state of the internal request model was logged. Questions in the questionnaires were formulated as statements, where the users had to indicate their level of agreement on a scale from 1 (strongly disagree) to 5 (strongly agree). The presented evaluation results refer to the mean of the judgments for each question that have been provided by the users (indicated in brackets).

The test users indicated that the tool proposed in this paper was easy to use (3.8) and that they felt guided by it (4.0). They even preferred it over the e-commerce platforms they typically use (4.0). A comparison of the initial requirements specified by the participants and those they provided after having chosen a service offer by using the tool shows that the proposed system succeeded in stimulating the test users to develop and specify their requirements. In average, 8 attributes where specified by the users. The critiquing tool was averagely used 2.6 times per user. After having made their final choice, the test users had specified requirements on averagely 0.8 service aspects, they did not consider before. All of the participants changed the relative importance of their requirements and 80% changed their preferences related to the values of the considered service aspects. However, we were also interested in whether the participants did not just specify requirements, but also in whether they actually had the feeling that they learned more about their requirements and whether they felt confident about them. Moreover, we wanted to find out whether the test users actually made a good selection or whether they were just convinced of having made a good selection. As it turned out, the respondents indicated that they learned more about their requirements by using the tool (4.0) and felt confident about them (4.2). To evaluate the quality of the selection made by the participants, we compared the requirements they indicated after using the tool with those covered by the final request model maintained by the system. We also recorded the number of test users that switched to another service offer after having seen all available offers and their properties. As a result of our evaluation, we found that the conformance between the user specified requirements and those modeled by our system was high. In all cases, the internal request model covered all service aspects that were important to the user. Also the conformance between the relative importance of those aspects as indicated by the test person and that documented in the model, was high (at most one aspect's rank differed by one position). However, we found that 40% of the participants switched to a slightly different offer after having seen the complete list of available service offers. As it turned out, the reason for this was, that the finally selected offer was in fact better with respect to most of the service aspects that were important to the user, but did not provide information about some of them. As a result, the matchmaker did not mark those offers as match and hence did not present them to the user. However, it seems that this restriction is to strict and that consumers are willing to accept the risk that is associated with a selection that is based on incomplete information. Finally, we were interested in the amount of time that the test participants spent to select an offer via our tool. As it turned out, the users averagely required about 17 minutes to make a final selection. The respondents indicated that this amount of time was acceptable for them (4.2). These preliminary results show that the tool effectively supports potential service consumers in making a well-founded selection, even if they are unfamiliar with the concept of WSs and WS descriptions. For the future, we plan a follow-up study in another service domain and with more participants.

6 Related Approaches and Conclusion

Typically, approaches to SWS selection assume that application providers create generic request templates, that cover frequent service needs in a certain application domain at design time [9]. Since consumer requirements are various, even for a single application domain, it is unlikely, that a predefined template exists that can be instantiated to build a service request that accurately describes the user's service needs. Moreover, template-based approaches do not enable potential service consumers to develop their requirements. However, a number of approaches that provide advanced assistance with the specification and refinement of service requirements have been suggested. Colucci et al. [2] propose a visual interface for assisted and incremental refinement of OWL-based service requests. Though their approach provides advanced user support for service selection, it neither considers uncertainty about consumer requirements nor accounts for consumer preferences. Moreover, the process of requirements refinement is not directed to promising directions and does not encourage service consumers to make compromises between service aspects. With MobiXpl, Noppens et al. [3] propose a mobile user interface for personalized semantic service discovery, that facilitates the specification of service requirements as a set of preferences over service aspects and utilizes ontology-based preference relaxation techniques to avoid empty result sets. Unfortunately, the proposed solution implements a single shot approach, where preferences cannot be refined after viewing the matching results. Balke et al. [10] propose an approach that accounts for the fact that consumer requests might be incomplete. As a solution they suggest to automatically rewrite and expand service requests to retrieve additional services that might potentially fit to the user's needs. However, the user is not involved in that process and thus cannot actively construct his requirements.

The ideas of letting users critique presented alternatives and encouraging them to make compromises by clustering available alternatives by common trade-off properties, that have been presented in this paper, are not new and have been previously proposed in the area of recommender systems [11, 8]. Our work is inspired by those approaches, but largely differs from them. In particular, our system supports the user in the critiquing process by providing him with immediate feedback about the consequences of his critiquing wishes. This is in contrast to the solution of [8], where users can specify self-initiated critiques, which directly lead to model changes and of which they do not know whether they are reasonable in light of the available offers. Moreover, our solution allows to identify yet unconsidered, but important service aspects by selecting suggested compromises. A major difference to the mentioned solutions is that, by explicitly modeling the system's uncertainty about the consumers requirements, our system is able to effectively direct and focus the requirements elicitation process into promising directions. Finally, our approach enables the user to correct the requirements model maintained by the system by providing an intuitive graphical representation of the internal model. The mentioned approaches do not offer this opportunity and hence do not allow for model adjustments, if necessary.

References

1. Bachlechner, D., Fink, K.: Semantic web service research: Current challenges and proximate achievements. Intl. J. of Comp. Sci. and App. 5(3b), 117–140 (2008)
2. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F.M., Ragone, A., Rizzi, R.: A semantic-based fully visual application for matchmaking and query refinement in b2c e-marketplaces. In: Harper, R., Rauterberg, M., Combetto, M. (eds.) ICEC 2006. LNCS, vol. 4161, pp. 174–184. Springer, Heidelberg (2006)
3. Noppens, O., Luther, M., Liebig, T., Wagner, M., Paolucci, M.: Ontology-supported preference handling for mobile music selection. In: Workshop on Advances in Preference Handling (2006)
4. Payne, J.W., Bettman, J.R., Johnson, E.J.: The adaptive decision maker. Cambridge University Press, Cambridge (1993)
5. Smyth, B.: Case-based recommendation. The Adaptive Web, 342–376 (2007)
6. Burke, R.D., Hammond, K.J., Young, B.C.: The findme approach to assisted browsing. IEEE Expert 12, 32–40 (1997)
7. Küster, U., König-Ries, B., Klein, M., Stern, M.: Diane - a matchmaking-centered framework for automated service discovery, composition, binding and invocation. In: WWW (2007)
8. Chen, L., Pu, P.: Hybrid critiquing-based recommender systems. In: IUI (2007)
9. Agre, G.: INFRAWEBS designer – A graphical tool for designing semantic web services. In: Euzenat, J., Domingue, J. (eds.) AIMSA 2006. LNCS (LNAI), vol. 4183, pp. 275–289. Springer, Heidelberg (2006)
10. Balke, W.-T., Wagner, M.: Towards personalized selection of web services. In: WWW (2003)
11. Reilly, J., McCarthy, K., McGinty, L., Smyth, B.: Dynamic critiquing. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 763–777. Springer, Heidelberg (2004)