Towards Personalized Selection of Web Services

Wolf-Tilo Balke
Computer Science Department
University of California
Berkeley, CA, USA
balke@eecs.berkeley.edu

Matthias Wagner
Future Networking Lab
DoCoMo Communications Laboratories Europe
Munich, Germany
wagner@docomolab-euro.com

ABSTRACT

Web services have gained an increasing popularity, however, also shown problems like their automatic discovery or deficiencies in interoperability. Because of today's wide variety of services offered to perform a specific task, it is essential that users are supported in the eventual selection of appropriate services. In this paper we present an extensive study of different useful techniques towards advanced personalization of Web service selection. We propose the partitioning of the user profile to support different steps of interaction with services and present techniques to personalize each subsequent step. Our main contribution is an algorithm featuring the expansion of service requests by user-specific demands and wishes. Services not matching a certain profile are discarded on the fly and equally useful results of alternative services can be compared with respect to user provided strategies and preferences. We also present a case study to exemplify the application of our personalization techniques.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval– *information filtering, retrieval models, selection process.*

H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *Web-based services.*

General Terms

Algorithms, Design, Human Factors.

Keywords

Web Services Selection, Registries and Semantic Discovery, Personalization, Collaborative Filtering, Service/ Usage Patterns.

1. INTRODUCTION

Web services are Internet-based, distributed modular applications that provide standard interfaces and communication protocols aiming at efficient and effective service integration. They have started to show their usefulness in a wide variety of domains. Typical Web service applications include business-to-business integration, business process integration and management, e-sourcing and content distribution. Web service interfaces and bindings are defined, described and discovered by XML artifacts, supporting direct XML message-based interactions with other services and applications via Internet-based protocols like SOAP [30]. Standards for service lookup and discovery such as the Universal Description Discovery and Integration (UDDI) [33] specification are designed to function in a fashion similar to white pages or yellow pages, where businesses and services can be looked up by name and/or by a standard service taxonomy. UDDI provides a first framework for the description of basic business and service information and offers a simple, yet extensible mechanism to provide detailed service access information on the basis of the Web Services Description Language (WSDL) [8].

WWW 2003, May 20-24, 2003, Budapest, Hungary.
ISBN 963-311-355-5.

Initially Web services were mainly intended to engage in dynamic business-to-business interactions with services deployed on behalf of other enterprises or business entities. However, with the evolution of Web service technology networked services will not only become increasingly sophisticated, but also move into the area of business-to-consumer or even peer-to-peer interactions. In [35] we have already pointed out that user interaction with Web services will rarely be on a single per-service basis. Instead, the ultimate goal in personalized service provisioning has to be the fulfillment of individual user needs expressed as complex tasks which are typically further divided into simpler sub-goals and subsequently matched to different services. With the number and diversity of services expected to grow, adequate techniques for user-centric and preference-based service discovery and selection will be needed. Even though UDDI and WSDL are commonly used today to implement service catalogs they still essentially lack strong concepts for service personalization which are crucial for advanced usability. In this paper we try to address some of the most important challenges arising from the personalized use of Web services. We study how service cataloging can be enhanced with concepts from cooperative databases [9], [26] and collaborative filtering [22] to discover, select and combine services according to the special needs and preferences of an individual user. In this context, service usage patterns and patterns modeling typical user needs together with the explicitly or implicitly given preferences of single users or user groups can be used for an efficient and effective Web service matchmaking.

The rest of this paper is organized as follows: In the following section we sketch our ideas towards an advanced selection of Web services and present an architecture for the personalized discovery and selection. In section 3 we briefly survey the existing state of the art in Web service cataloging and advertising and discuss related work concerned with the enrichment of standard lookup services. Section 4 illustrates the use of fundamental concepts from cooperative databases and collaborative filtering: cooperative answering and preference-based modeling. We present a basic algorithm for personalized service selection and composition in detail together with a sample usage scenario in section 5 and close with a summary and brief outlook on our future work.

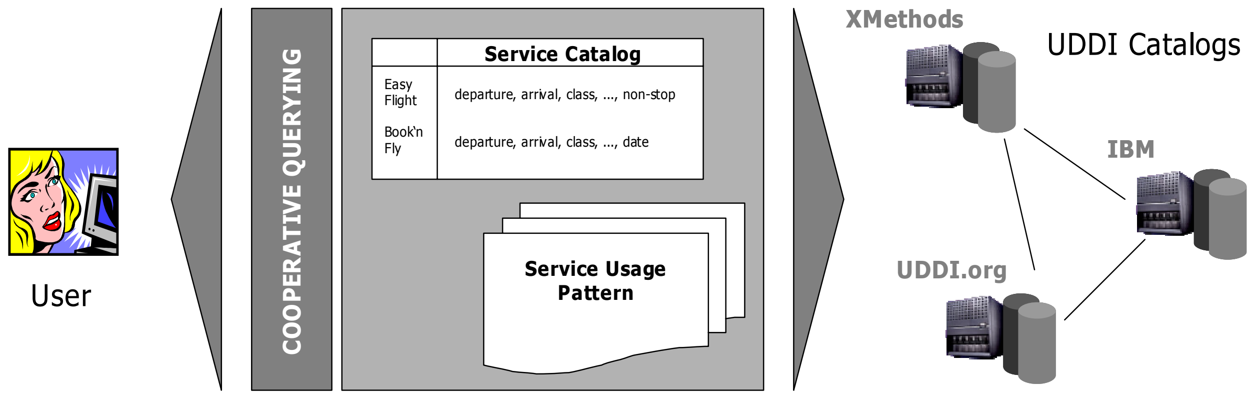

Figure 1: Personalized Service Discovery and Selection

2. PERSONAL SERVICES SELECTION

Today Web service platforms are beginning to offer technology that already supports simple forms of personalization. For instance, Microsoft's .NET Passport authentication service [23] enables users to enter and store simple profile information (e.g., e-mail address and password, mailing address, etc.) to enable users to sign-on once with a member of an affiliated group of organizations and subsequently use various sites offered by other group members without the need for signing-on times and again. In addition, services like e.g. the .NET wallet service will store credit card information so that users do not need to re-enter it for every transaction, either. In parallel, the Liberty Alliance Project [31] was formed as an organization of major information technology and telecommunication companies to deliver and support a similar federated network identity. This strongly emphasizes the importance that is assigned to personalization as a basic necessity for successful service provisioning.

However, although technologies such as .NET passport or the Liberty Alliance herald an essential paradigm shift in Web service deployment from mere B2B automation of business processes to more consumer-oriented network services, up to now they provide virtually no advanced personalization concepts. Especially the discovery and/or selection of services regarding the personal requirements of a user are not yet considered. In the following we consider the essential steps in selecting adequate Web services for individual users.

2.1 User-centered Web Services

Initially Web services were mainly intended to engage in dynamic business-to-business interactions with services deployed on behalf of other enterprises or business entities. Broad interest in standardization efforts was aimed at reducing the necessary user interaction. However, with the evolution of Web service technology the complexity of possible tasks and the availability of services anytime anywhere, e.g. through powerful mobile client devices, will strongly increase. Thus networked services will not only become increasingly sophisticated, but also move into the area of business-to-consumer interactions bringing back the user as active participant during interactions. The 'user in the loop' will enhance the service offerings and increase the competing variety that can be sensibly provisioned, but due to the information overload will also demand a great deal of personalization. The development of Internet portals like Yahoo.com already demonstrates the usefulness of a user-centered approach for e.g. e-commerce and related areas. But also here the necessity to provide simple personalization mechanisms or a support for browsing available services has become apparent to deal with the number of different offerings that may be more or less relevant for each individual user. For instance, MyYahoo.com allows individual users to define their own portal entry page and determine which personal content to display or which services to use on startup. With Web services becoming an integral component of the future Web, a user-centered approach to personal Web services is easily conceivable, e.g. through Web service technology in the backend of Internet portals. In the following we will deal with Web services in the context of being user-centered Services offered via the Internet.

The problem of how to interactively deal with Web services of course will have some impact on the areas of human computer interaction and user interface design (e.g. for mobile devices with limited capabilities) like it has already been the focus of research activities in setting up traditional portal pages. But since dealing with these issues is not the focus of the work presented here, we will assume the existing graphical user interfaces also as a basis for our necessary customer - service interaction. Trying to model the usage of Web services along the lines of traditional portal pages, but using the services' full capabilities and the powerful technologies, interaction with a truly user-centered service can thus be divided into three major phases (see fig. 2):

- the discovery of services that are basically able to perform a task based on a user's service request,
- the querying of services for subsequent selection based on user preferences for deliverable objects,
- and the final execution of a service, after a decision for one of the available objects.

Of course not every Internet service that is deployed as a Web service will be considered user-centered and needs personalization. We basically distinguish between simple services that can always be executed and just return a simple result, e.g. a currency converter or a weather forecast, and complex services, e.g. flight booking or restaurant reservation. Unlike for a currency conversion service that converts say US Dollars to Japanese Yen, for a flight booking service it is not obvious that the execution of the service will deliver the expected result and fulfill a user's needs. Its sensible execution may be dependent on some information, e.g. whether a flight with a certain airline can be booked on a certain date. Portals like Expedia.com [15] therefore pose the user's request to a variety of hard-wired services and present the collected results to the user. However, the hard-wiring of the Expedia portal does not allow for the flexibility of the Web services paradigm and thus does not really meet the challenges of the Web. We thus have to consider the questions, if a service generally has the capability of accomplishing a task and if the desired result is always available, if a certain service is executed. Of course, if we need a service like currency conversion that always is able to deliver the expected result, the process of selecting an adequate service for execution is rather straightforward, e.g. depending on the execution costs, or guaranteed accuracy, etc. The selection process will, however, be a lot harder in the case of e.g. flight booking where the quality of deliverable objects -in this case certain flight offerings- have to be compared. We will deal in detail with this problem in section 4.

Web services thus can be divided into business-to-business and business-to-consumer services and we can further subdivide the latter ones into always applicable services and those whose applicability in each specific case still depends on a specific piece of information. From a provisioning point of view the last class of services poses the most problems with respect to personalization, because:

- they usually model complex user-centered tasks consisting of different components like advertising, information and execution
- the final decision which service to execute almost always has to be performed directly by the user, because different services may offer various advantages beyond the specified user preferences.

2.2 Selection Steps

Let us now deal with the several steps that are necessary in selecting adequate services for subsequent execution. Again we will focus on more advanced services, where the general capability and the availability of suitable objects has to be considered. A successful user interaction with a service will generally consist of a complex goal statement followed by different discovery/interaction steps resulting in an optimal service selection. In detail we distinguish:

- **The user's intentional goal**: The purpose of interacting with a Web service is always a certain task a user wants to perform. The definition of this goal can be quite complex depending on the user's intention. There may be single time goals like ordering a book or arranging for a business trip. But there can also be permanent goals like managing communication tasks according to a user's current device or situation.
- **The discovery of available services**: The service discovery will show users which services can be possibly used to perform a certain task. It depends on the reachability of services and the device that a user plans to perform the service with. Sometimes services are offered for a variety of devices automatically adapting the content involved. The discovery is mostly an automatic step performed by the user's device.
- **The service composition**: Especially for complex goals there may be no adequate services discovered that perform the entire task, but only parts of it. The decomposition of a complex task into sub-goals which can be performed by simple services is a difficult matter. It strongly depends on the strategies that users choose to fulfill the entire task.
- **The service selection**: The service selection is the step of deciding which service(s) to use to finally perform the task. For instance there may be services performing similar goals and users can select the one that is closest to his/her intention or there may be different services performing the same goal, however offering different objects, at different costs, quality levels, etc.

We will revisit all these steps in the course of our paper and comment on possibilities to personalize each step. As can be easily seen, personalization will play a key role in the selection and composition of services: whereas the automatic discovery of unnecessary or unsuitable services is just a matter of inefficiency, the flooding of users with irrelevant services in the selection process can cause severe problems in usability. What is more, the sophisticated adaptation of decomposition strategies to the specific user's notion of utility will result in higher satisfaction and thus in higher quality of service.

2.3 A Usage Scenario

Consider a sample user called Michael, who is working in a Los Angeles Company and has to attend a business meeting in Boston during the next week. Setting up all necessary preparations is a complex matter and finding adequate services can already be quite time-consuming. But communicating personal requirements and preferences to many services that even might not be suitable for Michael's specific task will definitely get tedious. Basically the task may include, but is not restricted to setting up:

- The necessary transportation
- Reservations for accommodation
- Arranging meeting times and locations with business partners

Depending on Michael's personal preferences it may also involve other tasks as for instance arranging entertainment for the evenings or discovering sporting or sightseeing possibilities.

Throughout this paper we will consider Michael's typical tasks to show how different personalization techniques can be applied to improve the overall quality of service. Figure 1 shows how we envision the collaborative support of personalized service discovery and selection for our sample user Michael: existing UDDI-based service catalogs are integrated into a meta repository enabled for cooperative search. Beside the integrated Web Service catalog the meta repository also hold service usage patterns associated with the typical usage of certain services by particular users or user groups.

3. STATE OF THE ART AND RELATED WORK

Regarding our approach to personalized service discovery and selection we briefly survey ongoing Web service standardization activities and relate them to other work concerned with enhanced service description and advertisement:

UDDI (Universal Description, Discovery and Integration) [33] is an initiative proposed by Microsoft, IBM and Ariba to develop a standard for an online registry of Web services. UDDI enables the publishing and dynamic discovery of networked services and allows developers to locate services for direct invocation or integration into new complex services. A Web service provider registers its advertisements along with keywords for categorization. A potential service's user will retrieve advertisements out of the registry based on keyword search. It is assumed that user and provider use the same set of keywords for service characterization. The search mechanism relies on pre-defined keyword categorization and does not refer to the semantic content of the advertisements.

WSDL (Web Services Description Language) [7] is an XML vocabulary, closely associated with UDDI as the format for specifying interfaces to Web services registered with a UDDI repository. WSDL attempts to separate services – defined in abstract terms – from the concrete data formats and protocols used for implementation. It therefore describes a binding scheme between the abstract service description and its specific implementation. Note that the abstraction of services is at a comparatively low level in terms of abstraction from message protocols, service bindings and communication ports. WSDL has a concept of input and output types but, like UDDI, does not support semantic description of services.

E-Speak [13] is an initiative driven by Hewlett-Packard to enable enhanced service discovery. E-speak and UDDI have similar goals in that they both facilitate the advertisement and discovery of services. E-speak is comparable to WSDL in that it supports the description of service and data types and features a matching service that compares service requests with service descriptions. This is done primarily on the basis of input-output and service type matching. E-speak describes services as a set of attributes within several different vocabularies which are sets of attributes common to a logical group of services. Lookup requests are matched against service descriptions with respect to these attributes. Currently, there is no semantic meaning attached to any of the attributes. Any matching which takes place is done over the service description's keywords and does not distinguish between any further subtypes.

ebXML [14] is primarily developed by OASIS and the United Nations. It approaches the handling of service descriptions from a workflow perspective. ebXML uses two views to describe business interactions, a business operational view (BOV) and a functional service view (FSV). In ebXML the BOV deals with the semantics of business data transactions between Web service providers. The FSV deals with the supporting services themselves, i.e. their capabilities, interfaces and protocols. It has the concept of a collaboration protocol profile (CPP) which allows a trading partner to express their supported business processes and business service interface requirements by other ebXML compliant trading partners. A business process is a set of business document exchanges between trading partners. CPPs contain industry classification, contact information, supported business processes, interface requirements etc. They are registered within an ebXML registry, which facilitates the discovery of other trading partners and the business processes they support. In this respect, ebXML has some similarities with UDDI.

DAML-S [1] is an ontology-based approach to the description of Web services developed as part of the DARPA agent markup language program. DAML-S aims at providing a common ontology of services and is inspired by other research in the area of the so-called semantic Web that encompasses efforts to populate the Web with content and services having formal semantics. Build on top of DAML+OIL [10], the design of DAML-S follows the layered approach to semantic Web markup languages. The ultimate goal of DAML-S is to provide an ontology that allows agents and users to discover, invoke and compose Web services. Currently the structure of the DAML-S ontology is threefold and consists of a service profile for advertising and discovering services, a process model which gives a detailed description of a service's operation and a service grounding which provides details on how to interoperate with a service via message exchange.

WSMF (Web Service Modeling Framework) [16] provides another concept based on semantic Web technology for developing and describing Web services and their composition. WSMF describes the pre-condition and post-condition of services together with a service model. WSMF aims at strongly de-coupling the various components that implement a Web service application while at the same time providing a maximal degree of mediation between the different components. WSFM builds on comprehensive ontologies such as DAML-S and provides the concepts of goal repositories and mediators to solve complex service requests.

4. COOPERATIVE SERVICE SELECTION

The concepts and standards outlined above enable the basic discovery of Web services. However, after discovery Web service developers and individual still users have to select one or a composition of services that best serve their specific task. Given the growing diversity of services users can't be expected to browse all service offers until they find an adequate match. Thus, the task of service selection and composition has to be supported by leading to a recommendation of a set of possible services or combinations between which users can subsequently choose.

So far we have only spoken about personalization techniques to find or compose the appropriate service to suit a specific user's goal. But how does a user know from the service description, if the specific object he or she wants to access or deal with is available? Assume that a user has discovered a service for booking flights. Stating that she/he needs a flight booking service may not be enough – if a user cannot be sure that the specific flight on a specific date he needs can be booked by the service. Hence, after services have been chosen, a user's profile has to perform another important task. While using the service, it has to supply preferences of users with respect to the values for parameters that are transferred to the service.

Thus Preferences and Profiles are needed for:

- **Decomposing goals:** preferred strategies, disliked decompositions, etc.
- **Choosing services:** costs of execution, accuracy, trustworthiness, etc.
- **Supplying service parameters:** preferred input values, hard or soft constraints, ordering of relaxation, etc.

Expanding the query for a service with these preferences can essentially improve the quality of selection by ruling out inadequate services. However, by adding too many query terms a search might easily become too specific and return empty results. Thus for the personalized selection of services based on input parameters we need a cooperative service behavior that can also handle soft constraints. Cooperative retrieval techniques have been introduced by the research area of cooperative answering [26] based on the experience that finding the right query to pose against a database is a difficult matter. In contrast to traditional SQL databases – which use a perfect match retrieval model – any constraint in the query can be fulfilled or is violated by the objects in the database. The result set contains all those objects that fulfill every constraint of the query, i.e. all perfect matches. Thus a too specific query will return no results at all, while a too general query swamps the user with result objects.

However, at the time of posing a query a user can't be sure about the content of the database. Hence a typical behavior in querying databases is starting with a more general query and getting more specific, if too many results are returned or starting with a quite specific query an generalizing it by dropping attributes, if no results are returned. In traditional databases this task has to be performed by the user. Enabling a user to decide which parts of a query should never be dropped and which parts (and in what order) can be gradually relaxed is a helpful technique. If a user for instance didn't specify an airline in his query to a flight booking service, the choice of carriers can, according to this query, be relaxed to any value. Thus – as shown above – the possible booking services can be generalized to all those that have or don't have a carrier input parameter. Nevertheless, a user might have certain preferences for a specific airline. The semantics of leaving it out of the query means that a user does not insist on this point (in contrast to e.g. a specific arrival date or the destination of the flight). If many possible alternatives to choose from are returned, applying more specific user information can considerably reduce the answer set to the most desirable ones.

Those preferences can be gathered from previous interactions in form of a long-term profile or directly be specified by the user in the form of soft constraints. Unlike hard constraints those parts of the query can be relaxed, if the query gets to specific and does not return some suitable answers anymore. In our example from above we would add a preferred carrier attribute to our query (if the chosen service supplies such an attribute) and pose the query. Note that the expected result set will due to the more specific query generally contain less objects avoiding the flooding effect. But since the additional attributes have been chosen as soft constraints, they can be relaxed if necessary, thus avoiding empty result sets.

Cooperative answering techniques and their integration into database systems is an active field of research: the first cooperative systems [22] relaxed attributes step by step in a round robin fashion, if a query was too specific and did not return results. More user-centered approaches like [9] allowing users to specify also the order and steps in which to relax each attribute lead to results of higher quality. Recently the management of complex preference patterns and the translation into declarative database languages has gained more and more attention. Languages like PreferenceSQL [18] or the skyline-operator [6] already allow to pose declarative database queries with soft constraints filtering irrelevant result objects.

5. INTERACTING WITH WEB SERVICES

Discovering services using existing standard technology (cf. section 3) is mainly a task of keyword-based searches in more or less complex verbal service descriptions. It is beyond the scope of this paper to comment on all possible problems that occur when trying to discover services for specific tasks. Using today's techniques the user has for instance to use the exact phrases and ontology that is provided by standards like WDSL. Improvements for this situation are given by several techniques for matching of ontologies [12] or expansion by suitable thesauri [2]. In the following we will deal with the task of selecting services, after a discovery has yielded services matching the task a user is interested in.

5.1 Service Selection

In contrast to the task of composing Web services the selection of the right services strongly depends on the costs of a specific service or the objects or information that a service can provide. For instance there may be a variety of services revealing the location of a certain person, but at different costs or with different accuracy. The same applies for more complex service like e.g. flight booking, where a user has to know if the service offers a certain flight that suit his or her needs, before using the service. Therefore two aspects are necessary for service selection. The service has to:

- be generally able to perform the desired task
- offer the specific object/information the user is interested in

Thus users first have to query all possible services that generally perform their task, if they will also satisfy their specific need. Assuming that a set of services has been discovered, we will now discuss those personalization techniques that lead to a eventual choice of services offering the best quality for each specific user. We will first revisit our motivating example, then state the algorithm and perform a sample interaction.

Assume that our user Michael has to book a flight from Los Angeles for his business trip to Boston. Since the business meeting starts at 3 p.m. on the 10th of November, he has to arrive till noon to have some time left to get to the business location. Besides, he wants to fly business class. The SQL-like query

```
SELECT flights FROM service
WHERE departure = 'LAX' AND arrival = 'BOS'
  AND arrival_date <= '10-11-2002 12:00'
  AND class = 'business'
```

would express his needs. Please note that all sample values specified in the where clause will be considered hard constraints. Neither would Michael at this stage fly to New York instead of Boston, nor would he consider a flight that does not arrive in time to get to the meeting.

Our algorithm shows the necessary steps that have to be performed in order to choose adequate services and get the maximum quality results. It assumes that a goal is provided that can be managed by Web services, a long-term user profile and/or general common knowledge exists and that a common vocabulary is used. For ease of understanding we will abstract from techniques for ontology matching, etc. that could be applied in cases with no shared vocabulary.

Algorithm: Selecting a Service

1. Perform a keyword-based search on the semantic service descriptions to find services for the specified goal.
2. Group all discovered services by signature parameters and discard all services that do not allow querying with all user-provided query terms (explicit hard constraints).
3. Get the service parameters that are offered beyond those covered by hard constraints.
4. If existent, get preferred values for the parameters collected in step 3 from user profiles, domain knowledge, etc. (implicit soft constraints).
5. Expand the user's service request with these soft constraints and query the services from step 2 with their respective signature.
6. Collect all services' results and order by their utility, e.g. assign highest utility to those results meeting the most soft constraints.
7. If no results are returned, let the user reconsider the hard constraints and start over with step 2.

Let us consider how this algorithm works continuing our example: Michael wants to book a flight. First we have to discover all different services that allow to perform the task of booking a flight and check, if flights meeting Michael's criteria are available (step 1). A keyword search on service descriptions might result in a list of four services for flight booking given in table 1.

Table 1: Discovered Services for Flight Booking

service	**input parameters**	**semantic description**
Book'n Fly	departure, arrival, class, departure date, arrival date	flight booking
Flights On-line	departure, arrival, class, departure date, arrival date, airline	flight booking, flight information
Air Travel Economy	departure, arrival, price, departure date, arrival date, airline	economy class flight booking
Easy Flights	departure, arrival, class, departure date, arrival date, price, non-stop	flight booking

The next step is sorting out what parameters are needed to guarantee the observance of the hard constraints (step 2). Since our query specifies departure, arrival, arrival date and class, we have to discard the Air Travel Economy service, because no class input parameter is provided. Step 3 checks for additional parameters beyond the hard constraints and gets departure date, airline, price and non-stop. Now in step 4 Michael's long-term profile has to be queried for preferences with respect to these additional parameters and reveals that Michael generally uses to fly Delta Airlines and prefers non-stop flights. Some general preferences from the domain could also be applied like "everyone would prefer a short traveling time" (i.e. a departure date with maximum proximity to the arrival date is preferred). However, since it is a business trip, we may get no information about preferred prices from Michael. Thus no parameter for the price will be specified in the queries. Now we are ready to build the queries with respect to the services (step 5). Using again the declarative notation of [18] we get three queries that can be posed to the services:

```
SELECT flights FROM Book'n_Fly
WHERE departure = 'LAX' AND arrival = 'BOS'
  AND arrival_date <= '10-11-2002 12:00'
  AND class = 'business'
PREFFERING MAX(departure_date)

SELECT flights FROM Flights_On-line
WHERE departure = 'LAX' AND arrival = 'BOS'
  AND arrival_date <= '10-11-2002 12:00'
  AND class = 'business'
PREFERRING MAX(departure_date)
       AND airline = 'Delta'

SELECT flights FROM Easy_Flights
WHERE departure = 'LAX' AND arrival = 'BOS'
  AND arrival_date <= '10-11-2002 12:00'
  AND class = 'business'
PREFERRING MAX(departure_date)
       AND non-stop = true
```

Table 2 shows the sample results from our Web services. Please note that though some services could not be queried on airlines all services may be capable of stating the airline in the result set. Easy Flight delivers two possible flights, because one is a non-stop flight whereas the other one has a better departure date. Since Michael didn't specify an ordering those flights can be considered equally good.

Table 2: Sample Service Results

service	**result** name airline	 departure depart. date	 arrival arrival date	 class non-stop
Book'n Fly	UA908	LAX	BOS	Business
	United	10-11-2002 05:55	10-11-2002 11:35	---
Flights On-line	BA87	LAX	BOS	Business
	British Airways	10-11-2002 06:45	10-11-2002 10:45	---
Easy Flights	D765	LAX	BOS	Business
	Delta	10-11-2002 05:15	10-11-2002 11:25	True
Easy Flights	LH737	LAX	BOS	Business
	Luf-thansa	10-11-2002 06:35	10-11-2002 11:55	False

Performing step 6 of our algorithm we could order the result set proposing Easy Flight's D765 as best result since it optimizes two soft constraints (airline and non-stop), whereas all other flights only meet one of the constraints. However, also every other suitable utility assessment can be used: Michael might for instance insist on the shortest traveling time. Returning a non-empty result set the selection is finished and our user can eventually decide

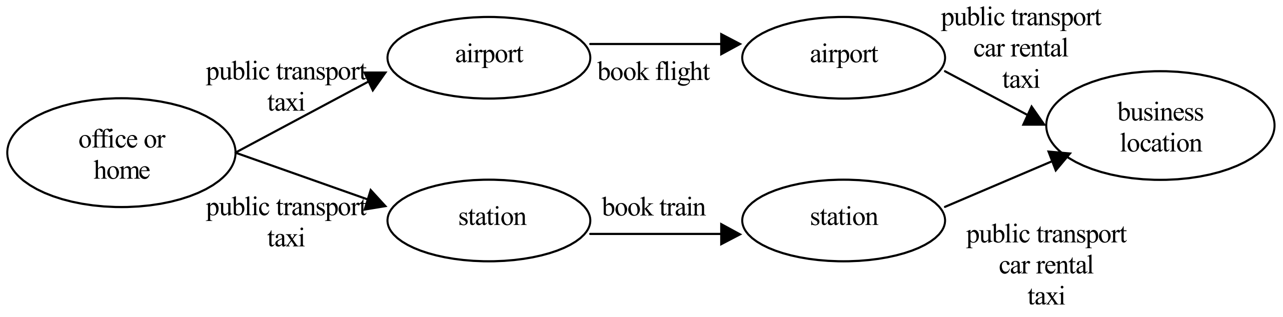

Figure 2: Decomposition of Services for Business Trips

which service should be used for the booking. However, if no results had been returned, Michael would have had to reconsider his constraint and might e.g. have decided to drop the business class constraint. Then we would have to start all over and please note that in this case even the previously discarded Air Travel Economy service would have to be considered again.

Our assumption that all user supplied query terms have to be hard constraints is only for the ease of understanding. The generalization to a case in which a user also explicitly provides soft constraints or a order of relaxation is straightforward. These soft constraints would have to be seen as supplied for this special service request and could be more specific than or even conflicting with general long-term preferences and would thus have to be evaluated after the provided hard constraints, but before the query is expanded with terms from the long-term profile.

5.2 Service Composition

So far we have considered concepts for a simple service discovery and shown how to select single services in a personalized manner. But whereas the discovery of services can often be performed using simple keyword-based searches on the semantic service description, a combination of several services still poses difficulties. First there are technical problems dealing with the ways to invoke services and the flow of data between different services. Due to the heterogeneity and autonomy of different services it is not possible to rely on the current models to build and coordinate compositions of web services. Complex interactions need high-level support like mediator services or transaction models. These rather technical problems have already gained broad attention and languages like WSFL [20], XLANG [32] or WSCL [2] have defined primitives for composing services and automated service coordination. But recent work has also already considered some high level interaction concepts. For instance [7] proposed dedicated combining services setting up a process model for each user's selection of services, [29] presented a XML-based transaction model and [27] addressed the logic-based automated validation and composition of web services in the area of semantic web.

On the other hand we have again personalization problems in choosing services for reasonable combinations. Consider for instance our sample user Michael having the goal of attending a business meeting in a different city. Depending on the characteristics of the starting point and the meeting place several ways of achieving the goal can be applied possibly involving a variety of different services. If for instance the journey spans over a rather long distance a service for booking a flight will probably be involved. However, Michael could also decide to travel by train. Since flights can only supply transportation between airports, the goal is decomposed into several pieces such as: finding possible or most convenient airports for departure and arrival, transporting our business person from the office or apartment to this airport, booking the flight to the chosen destination and transporting the business person from this airport to the business location.

A sample graph-based decomposition is shown in figure 2. This decomposition will need a flight booking service as single service for one of the components (i.e. traveling from airport to airport) and involve other services in the other steps. For instance the transportation from our business person's office to the airport can involve a taxi-service or public transportation which can itself be decomposed into a bus or a subway service. The transportation from the destination airport to the business location may again be of the simple kind like getting a taxi, but can also involve complex tasks like renting a car. Complex tasks will again generally consist of several services that can be decomposed. Even in this simple example we can already identify twelve different ways of doing a business trip. Each way states a usage pattern that is categorized by the respective location and the time constraints, e.g. if you decide to take a flight and use the subway to the airport a subway service providing a line to the airport has to be chosen and a flight has to be booked that departs some time after the subway has arrived allowing you to check in. Such decompositions have been thoroughly investigated in planning algorithms in AI and knowledge representation [34]. Standard approaches to similar problems have been found to be very complex. Thus deriving knowledge from service provided usage patterns, instead of relying on huge knowledge bases to segment every possible task seems advisable [5].

Though there is no standard solution to this problem, companies like e.g. travel agencies have certain general patterns or heuristics to deal with it. Starting with a general approach based on simple attributes like distance or duration of a journey they choose appropriate means of transportation or decide if booking a hotel will be necessary, then working out the details. Web services are in this respect very similar. When designing a certain Web service, designers have quite specific ideas what capabilities the service should provide. Designers anticipate different possibilities for usage of the service and in well-defined services useful decompositions for certain typical queries or service requests will generally exist. These typical interactions for different user groups are often referred to as usage patterns. A usage pattern depends on the intention of users. Generally speaking different intentions will need different patterns.

Since different user groups will have at least some typical strategies to achieve the same kind of goal, these groups can be clustered and assigned to different usage patterns. Consider for example a service for traveling. Of course the locations for departure and destination always have to be supplied and will differ from usage to usage, but possible decompositions would be taking the train or booking a flight. Generally speaking the train will be slower, but cheaper than the flight. Distinguishing only two groups of travelers we can assign some simple strategies. There are business travelers for whom time is probably a more important good than money. This can trigger some default business travel patterns as for example the choice of avoiding the train or choosing business class tickets, if a minimization of flight time or changing of planes during the trip can be achieved. Also different general rules for domain knowledge apply. For instance the relaxation of constraints like arrival date and time is far more difficult than a service would have to be with a usage pattern for tourists. Typical tourist patterns would probably be more concerned about money and hence try to minimize the costs, e.g. taking the train would be an adequate choice, whereas flying business class could be considered less desirable. Besides, using a tourist pattern a service might be more flexible in relaxing time constraints.

When composing different services, usage patterns can be seen as a representation of the different possibilities or strategies that users apply to decompose their task. What sub-goals are chosen has obviously a direct impact of the subsequent services that are performed to accomplish the complex task. Consider again our sample usage scenario from above. The complex transportation problem from the destination airport in Boston to the business location can be broken down into several sub-goals. One possibility would be renting a car at the airport, another one would be taking a taxi or public transportation. Whatever way is subsequently chosen strongly depends on the individual user and thus is a matter for adequate personalization techniques using personal profiles stating preferred decomposition strategies or dislikes. These strategies typically override more general terms of the pattern. For example, if a user suffers from fear of flying his or her personal profile would definitely state the dislike of flying though he or she might be part of the business travel pattern that prefers fast ways of traveling. In this case the specific personal profile would override the more general profile.

6. SUMMARY AND OUTLOOK

In this paper we have raised the question of personalization for Web service selection. Today Web service discovery and execution is performed using standards like UDDI, WSDL or ebXML, where only limited semantic meaning can be used in services description. Approaches from the Semantic Web like DAML-S or WSMF are first efforts that try to provide semantically rich service descriptions. But even letting aside the problems of different ontologies or missing concepts of interoperability the human interaction with services poses severe problems due to the lack of personalization. Today's service variety offers users to choose between different services to perform their specific task. With the seamless integration of networks this variety can be expected even to grow by orders of magnitude. The task of selecting an adequate service, however, can quickly grow tedious, if all services that are listed under a certain description have to be compared manually for the final selection. And what is more, the final selection does not only depend on service parameters like execution costs or accuracy, but also depends on the usefulness of objects or information that a service offers.

Interacting with a Web service a user brings in lots of expectations. Besides a (complex) goal there will be strategies to compose different services, certain preferred ways to achieve (sub-) goals, some hard constraints that have to be met and also a lot of preferences or knowledge that is implicitly given by previous service uses or knowledge about the domain. We have addressed the steps of service discovery, selection and composition. Finding useful means of personalization for each step we argued to divide a user's profile into three different parts:

- The first part to decompose goals relies on typical usage pattern anticipated by the service provider and helps to break down a complex task into preferred sub-goals that can be solved by simple Web services
- The second part consists of preferred service parameters and helps to choose between discovered services that advertise to perform the same task, but may differ in typical parameters like e.g. execution costs.
- The third part deals with the preferred characteristics of objects or information that a service claims to provide. This part helps to choose between objects from different services and to discard services that can in principle handle the task, but don't provide any desirable objects.

Considering a sample interaction we have shown, how to deal with the different profiles and have presented an algorithm for the subsequent selection of appropriate services. This algorithm features an expansion of the service request by user-specific demands and wishes. Services not matching a certain profile are discarded on the fly and equally useful results of alternative services can be compared with respect to user provided strategies. Using techniques from cooperative answering we also showed how to get a better service selection including users' long-term profiles without risking to run into empty result sets.

Aiming at improved usability of Web services our future work will focus on the subsequent refinement and integration of the presented ways of personalization into standard interaction techniques. Repositories for user profiles, domain knowledge or usage patterns and the generation of new patterns have to be investigated. We will also consider specific characteristics of our personalization techniques and heuristics when dealing with different client devices. In the case of for instance mobile devices with limited capabilities the extension of our basic techniques could pave the way to improved usability enabling a user-specific and context-aware selection of appropriate services.

7. REFERENCES

[1] A. Ankolenkar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, and K. Sycara. DAML-S: Web Service Description for the Semantic Web. . In *Proc of Int. Semantic Web Conf. (ISWC'02)*, Sardinia, Italy, LNCS 2342, Springer, 2002.

[2] A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K. Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, S. Sharma and S. Williams. Web Services Conversation Language (WSCL) 1.0. http://www.w3.org/TR/wscl10/, 2002

[3] L. Ballesteros, B. Croft. Phrasal Translation and Query Expansion Techniques for Cross-language Information Retrieval. In *Proc. of the Int. Conf. on Research and Development in Information Retrieval*. Philadelphia, USA, 1997

[4] T. Berner-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[5] G. Beydoun, A. Hoffmann. Monitoring Knowledge Acquisition, Instead of Evaluating Knowledge Bases. In *Proc. of the European Knowledge Acquisition Conference (EKAW2000)*. Juan-les-Pins, France, LNCS 1937, Springer, 2000.

[6] S. Börzsönyi, D. Kossmann, K. Stocker. The Skyline Operator. In *Proceedings of the 17th International Conference on Data Engineering*. pp. 421-430, Heidelberg, Germany. 2001.

[7] F. Casati and M. Shan. Dynamic and Adaptive Composition of E-Services. In *Journal of Information Systems*, 6(3): 143-163, 2001

[8] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/2001/NOTE-wsdl-20010315, 2001.

[9] W. Chu, H. Yang, K. Chiang, M. Minock, G. Chow, and C. Larson. CoBase: A Scalable and Extensible Cooperative Information System. *Journal of Intelligent Information Systems (JIIS)*, 6(3):223-259, 1996.

[10] D. Connolly et al. DAML+OIL Reference Description. W3C Note, December 2001.

[11] T. G. Dietterich. Learning at the Knowledge Level. *Machine Learning*, 1(3):287–316, 1986.

[12] A. Doan, J. Madhavan, P. Domingos, A. Halevy: Learning to map between ontologies on the semantic web. In *Proc. of the Int. World Wide Web Conf. (WWW 2002)*, pp. 662-673, Honolulu, Hawaii, USA, 2002

[13] E-Speak. E-Speak Architectural Specification Release A.0. http://www.e-speak.hp.com/media/a0/architecturea0.pdf.

[14] ebXML. ebXML Web Site. http://www.ebXML.org.

[15] Expedia Travel Portal. http://www.expedia.com

[16] D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. To appear in *Electronic Commerce Research and Applications*. http://www.cs.vu.nl/~dieter/wese/wsmf.paper.pdf

[17] T. R. Gruber. A Translation Approach to Portable Ontology Specification. *Knowledge Acquisition*, 5:199–220, 1993.

[18] W. Kießling, G. Köstler: Preference SQL - Design, Implementation, Experiences. In *Proc. of the Int. Conf. On Very Large Databases(VLDB)*. Hong Kong, China, 2002.

[19] O. Lassila and R.R. Swick. Resource Description Format: Model and Syntax Specification. W3C Recommendation, 1999.

[20] F. Leymann. Web Services Flow Language (WSFL 1.0). http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf, 2001

[21] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine Learning: An Artifical Intelligence Approach*, volume 1. Morgan Kaufmann, 1986.

[22] A. Motro. VAGUE: A User Interface to Relational Databases that Permits Vague Queries. *ACM Transactions on Office Information Systems (TOIS)*, 6:187-214, 1988.

[23] .NET Passport. http://www.passport.com, 2002.

[24] T. M. Mitchell. Generalization as Search. *Artifical Intelligence*, 18(2):203–226, 1982.

[25] S. McIlraith, T. Son, H. Zeng: Semantic Web Services. In *IEEE Journal on Intelligent Systems*, IEEE, pp. 46-53, 2001.

[26] J. Minker. An Overview of Cooperative Answering in Databases. In *Proceedings of the 3rd International Conference on Flexible Query Answering Systems (FQAS)*, pp. 282-285, Roskilde, Denmark, LNCS 1495, Springer, 1998.

[27] S. Narayanan and S. McIlraith. Simulation, Verification and Automated Composition of Web Services. In *Proc of Int. World Wide Web Conf. (WWW 2002)*, pp. 77-88, Honolulu, Hawaii, USA, 2002

[28] F. Patel-Schneider and D. Fensel. Layering the Semantic Web: Problems and Directions. In *Proc of Int. Semantic Web Conf. (ISWC'02)*, Sardinia, Italy, LNCS 2342, Springer, 2002.

[29] P. Pires, M. Benevides and M. Mattoso. Building Reliable Web Services Compositions. In *Proc. of the Int. Workshop on Web Services: Research, Standardization and Deployment (WS-RSD),* pp. 551-562, Erfurt, Germany, 2002.

[30] SOAP Protocol. http://www.w3.org/2000/xp/Group.

[31] Liberty Alliance. http://www.projectliberty.org.

[32] S. Thatte. XLANG: Web Services for Business Process Design. http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm, 2001

[33] UDDI. The UDDI Technical White Paper. http://www.uddi.org.

[34] M. Vilain. Getting Serious about Parsing Plans: a Grammatical Analysis of Plan Recognition. In *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI-90)*, pp. 190-197, Boston, USA, 1990.

[35] M. Wagner, W.-T. Balke, R. Hirschfeld and W. Kellerer. A Roadmap to Advanced Personalization of Mobile Services. In *Proc. of the Int. Federated Conferences DOA/ODBASE/CoopIS (Industry Program).* Irvine, CA, 2002.