10-2005

Towards Semantic Service Request of Web Service Composition

Qianhui (Althea) LIANG
Singapore Management University, althealiang@smu.edu.sg

Jen-Yao CHUNG
IBM T. J. Watson Research Center

Steven MILLER
Singapore Management University, stevenmiller@smu.edu.sg

DOI: https://doi.org/10.1109/ICEBE.2005.121

Published in IEEE International Conference on e-Business Engineering: ICEBE 2005: 12-18 October 2005, Beijing, China: Proceedings. (pp. 705-712). Los Alamitos, CA: IEEE Computer Society, 2005.
http://doi.org/10.1109/ICEBE.2005.121

Towards Semantic Service Request of Web Service Composition

[1]Qainhui Althea Liang, [2]Jen-Yao Chung and [1]Steven Miller
[1]*School of Information Systems, Singapore Management University, Singapore*
althealiang@smu.edu.sg, stevenmiller@smu.edu.sg
[2]*IBM T. J. Watson Research Center, USA*
jychung@us.ibm.com

Abstract

When meeting the challenges in automatic and semi-automatic Web service composition, capturing the user's service demand and preferences is as important as knowing what the services can do. This paper discusses the idea of semantic service requests for composite services, and presents a way to model the elements of a composite service request as user preferences and constraints. The model is based on an interactive and iterative strategy meant to obtain the exact requirements from potential service consumers. The markup vocabularies and associated inference mechanism of OWL-S are used as a means to bring semantics to service requests. Language constructs are added to OWL-S as uniform representations of possible aspects of the requests. Using this model to represent the semantics of service requests enables the discovery agents to unambiguously understand the service need and precisely produce the desired composition. An application scenario is presented to illustrate how the proposed model can be applied to the real business world.

1. Introduction

Sharing and reusing resources is made much easier with the Web services model and through service-oriented software development. This approach is expected to simplify software development by allowing a new system to be composed out of existing components that have been made very easy to describe, publish, find, bind and invoke by the technology itself. Two related concepts in service-oriented computing are composite (complex) services and service composition. A complex (composite) Web service is conceived as a combination of simpler Web services over a designated flow structure [1]. The primitive services are referred as member services or component services. Service composition is the construction of composite services.

Semantics of Web services [2] [3] [4] are the key to service composition, especially in automating the composition process. To satisfy the complex service needs, the discovery agents will have to pick out "service components" from existing services that can accomplish certain jobs, and to organize them into a larger service where the component services work as a collaborative team and provide the acceptable service. Understanding what the services actually provide to its users is the key to identifying the required component services. The agent must have enough knowledge to decide whether the service can perform a certain job. [5][6] and [7] emerged as languages for marking up Web services. They allow service providers to present the description of their services and to mark up their service descriptions for knowledge sharing and inference.

Service composition or composite service discovery is based on matching service request with available services. Service request can from a user looking for services or a computer program representing its human clients and discovering services automatically. Among what makes automatic service composition difficult is the non-straightforwardness and incompleteness of the service requests. Without good semantics on services requests, what the client really wants is unclear, how the user is capable of interacting is unspecified, and what the client prefers and can not accept is not known. Just as the physical service worlds, only with the client's need clearly in mind, services can be composed correctly. This has suggested to us that semantics of service request is an integral part of semantic Web services and has to be addressed properly.

Further, we realize the fact that for service composition, semantics of the service request has to be obtained in a way that allows the requestors to develop it incrementally. Interactions or interventions from

human beings are necessary and request can be modified in various ways. Preferences of users must be captured and uncertainty of the request must be considered. Our model is designed to satisfy these requirements.

In Section 2, we briefly review the composite Web service model and the service composition approach reported in the previous work. In Section 3, we discuss in detail the semantics of service requests for service composition. In Section 4, we present a new method to model semantic service requests based on OWL-S that includes language construct extensions for representing the service needs. In Section 5, we provide an example of using the model to represent the semantics of a request of a complex service. We discuss related work in Section 6. Our conclusions are stated in Section 7 and a list of references is given in Section 8.

2. Composition in Extended Web Services Model

The idea of the service composition system is to respond to the service requests that can only be satisfied with results of a composition of services. This is best achieved by enhancing the current Web services model with the capability to discover and invoke composite Web services in some automated manner. We extended the standard Web service model by introducing a role of an Intelligent Service Registry that is capable of answering complex queries by constructing or discovering composite services dynamically based on the semantic markup of registered services. We also introduced an add-on component called the Composite Service Processor. It is in charge of invoking composite services discovered according to the interaction policy semantics. Our belief is that the enhanced model offers an explicit mechanism allowing composite service processing to be studied in the same framework as simple Web services. It, therefore, eases the process of service-oriented application development. It also provides transparency to requesters who don't have to care whether the requested services are composite or simple services. This enhanced model is described in detail in our previous papers[8][1].

Previously, Liang et al [1] extended WSDL to include specifications of service restrictions in the descriptions of service providers. In this paper, we provide supplementary constructs to the OWL-S semantic Web service language that enables a service request to query the Intelligent Registry in a high-level declarative way. With these supplemental language constructs, described in Section 4.2, OWL-S can be used to make up a service request with the exact semantics required to construct a composite service.

The Intelligent Registry parses the semantic service request and then either identifies a simple service(s) or attempts to construct a composite service(s) that satisfies the user's request. The Intelligent Registry also ensures that the composite service discovered is a valid composite service against the requestor's particular requirements. It then generates a composite service specification, which can be registered with the Intelligent Registry. If the requestor would like the composite service to be invoked, he contacts a Composite Service Processor, which takes the composite service specification, calls the registered component services and returns the result.

Due to the incompleteness in understanding the complex service need of a service requestor and the non-determinism in resolving a requester's preferences on services, the Intelligent Service Registry is based on a semi-automatic approach [1][8]. This approach uses the AI AND/OR graph technique to automatically search for a possible composition. Complementary human critiques are also considered and modifications of service requests may explicitly or implicitly guide the next run of search. This process is iteratively carried out until an acceptable composition is produced.

3. Semantics in Service Requests

A service request is the only common contract that different roles in the Web services model can refer to when discovering the services. When semantics of services become explicitly available and understandable to machines, it is possible to use a piece of software to analyze the specifics of services. At the same time, when a Web service request specifies the desired service in such a way that the service demand is clearly understood by the matchmaker to make a choice for the requester, it is possible for a piece of software to select services and to compose the services automatically. Considering the fact that composite service is built on multiple services and involves multiple service providers, the importance of semantics of requests becomes more obvious. Two considerations when designing a semantic model of service requests are: how to make requests include information that helps reduce the complexity of automatic service composition; and how to make requests provide a mechanism that is nature and easy to use by the requestors. The rest of the section gives a briefing on these two considerations.

Given service requests, discovery of new composite services is defined through constructing the flow structure of the component services. In other words, the problem is to decide what component services shall be selected and in what manner they shall be organized and collaborating. Prior to a detailed functional or non-functional description of the desired service, the service composition agent can ask the requestor to identify the domains that the requested service falls in. We believe a specific service request can be narrowed down to one or several service domains under some service categorizations. This results from the composition always happens within the context of certain domains. Automatic discovery of composite services is made feasible if the services under consideration are limited to particular domains.

Discovery of composite service involves automatically selecting and composing appropriate Web services to perform some task. The service discovered must adhere to requested properties [2]. A user located in Shanghai might say, for example, "Make the travel arrangement for my conference trip. If morning flights are unavailable, I will go by train. " With such preferences, a composite service that uses an airplane as the transportation mode but flies the passenger by an evening flight is not acceptable to the user. Therefore, in addition to understand what the requester wants to do, we must also understand the requester's preference and constraints. The composition agent shall compare the constraints of service providers, such as "China airlines does not have morning services from Shanghai to Beijing" with the specification of the concerns of service requesters to achieve an acceptable composition of services. Further, if there are conflicts between the user preferences and the service products or between one preference and another, the composition agent will have to resolve the conflicts.

When considering the requester's preference and constraints in service composition, one objective is to have a mechanism that allows the user to elicit their preferences easily and that assists the user to achieve their goals to the greatest extend possible. In other words, the modeling of semantics of service request has to introduce an interactive and iterative mechanism for a flexible yet effective request elicitation. Most fully automatic approaches to services composition assume that the requester can put together their requirements and preferences all at once, without iterations. They do not allow the users to develop their preference model incrementally and make tradeoffs by adjusting their preferences. Such inflexibility prevents the discovery result from being acceptable.

4. Modeling Semantic Service Requests for Composite Web Services

In this section, we discuss the requirements on desired services and the modeling of the requirements of a service request. Since OWL-S is the most prevalent language used to encode Web service capabilities both for advertisement and for requests [4], we chose to model service requests using the OWL-S semantic framework. However, our modeling is not dependent on a particular semantic framework. Generally speaking, according to the OWL-S service ontologies [7], requirements on services can relate to the *profile* aspect, the *grounding* aspect or the *process model* aspect of the semantic services.

We see the semantic analysis of service requests as an integral part of the Web service process. As such, we augment [3]'s "Web service lifecycle in OWL-S ontologies" by adding a function for Request Semantic Analysis. The amended diagram is listed in figure 1.

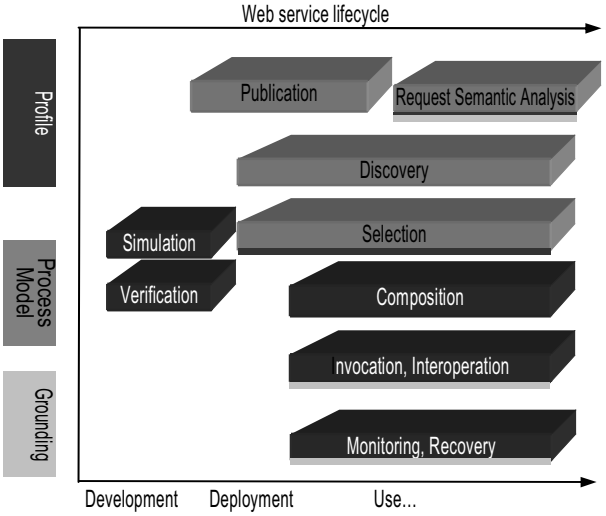

Figure 1. Web Service lifecycle with Request Semantic Analysis, enhanced from [3]

If the services are treated as objects, service properties can be represented in terms of service attributes. Service requesters can express their service requests by imposing requirements on the attributes of the services that they request. Depending on the nature of the service attributes that the requirements involve, we categorize requirements into two classes: functional and non-functional. A functional requirement consists of descriptions of functional properties of the service using ontology of service functions in OWL-S. An example is a "Bookselling service" or an "Airline ticketing service". A non-functional requirement consists of descriptions of non-functional properties of

the service, which are usually conditions or restrictions, such as "DepatureDate between June 3rd, and June 5th" or "Encrypt a header with an X.509 token".

We will discuss semantics of service requests at two levels. The first level is user requirements presented as constraints and preferences. The second level is iterative and interactive constraint (preference) revision. We also propose the OWL-S based service request language constructs to accommodate these semantics.

4.1. Multi-attribute utility and service requests

Multi-attribute utility theory and preference elicitation have been studied extensively in the literature of decision analysis [9]. If a user has preferences on the multiple attributes of a product or a system, the user needs a systematic method to make decisions about the kind of multi-dimensional tradeoffs that are best for his preferences. Without a supporting method, people or agents have problems with making multi-attribute decisions because the satisfaction of a preference along one dimension may result in a failure to meet preferences along other dimensions. In service-oriented computing, a service is an object that has its own attributes. A systematic and quantifiable method for satisfying the requirements can be based on the same techniques previously developed for decision analysis and for analyzing multi-dimensional user preferences in decision-support systems.

The following description of the standard multi-attribute decision analysis is taken from [9] and [10]. Based on the expected-utility paradigm, the set of the service alternatives, i.e., the outcome, S, is defined by a set of value dimensions, as in (1).

$$V = \{d_1, d_2, \quad, d_n\} \tag{1}$$

A utility function u is a real value function, all of whose arguments are in V, i.e.,

$$u(V): 2^V \rightarrow R. \tag{2}$$

The process of making decisions is modeled as identifying the best solutions from 2^V options. The following relationship among the outcome services, s, in S is defined: The preference relation is an asymmetric ($s \quad s' \Rightarrow \neg(s' \quad s)$) and transitive ($(s \quad s') \wedge (s' \quad s'') \Rightarrow (s \quad s'')$)[11]binary relation on the set of options. $s \quad s'$ indicates that the alternative s is preferred to alternative s'. s and s' are said to be indifferent, if ($\neg(s \quad s') \wedge \neg(s' \quad s)$). An utility function is indicating a unique preference order over the individual outcomes in S, i.e.,

$$s \quad s' \Leftrightarrow u(s) > u(s'), s \in S, s' \in S \tag{3}$$

Let's assume the probability distribution over V is denoted as $\Pr$. Let $Z_1, \quad, Z_k$ be a partition of V. $Z_1, \quad, Z_k$ are said to be additively independent regarding , if for any $\Pr_1$ and $\Pr_2$ that have the same marginal on Z_i for all i, $\Pr_1$ and $\Pr_2$ are indifferent. It has been proved [9] that $Z_1, \quad, Z_k$ is additively independent only if the utility function u, can be written as a sum of the utility functions of Z_i, i.e.,

$$u(V) = \sum_{i=0}^{k} u_i(Z_i) \tag{4}$$

To use the multi-attribute utility theory to model service requests, there are a couple of issues. First, in general cases, a dimension d is usually a function of the relevant simple attributes [12], i.e. $d = f(a_1, a_2, \ldots)$. In the case of Web services, the attributes of services in the service requests are heterogeneous, which makes it inconvenient to subsume them in a hierarchy. Also, it is sufficient to ask them to present the properties in higher-level ontologies of OWL-S with a flat structure. Based on the above consideration, when modeling the service requests, we use a simple model that directly operates on attributes avoiding a more complex model that involves a dimension hierarchy. We will use the following definition of the outcome service space: $V = \{v_1, v_2, \quad, v_n\}$, where v_i are service attributes that the requester can use to express their desired service.

Second, the requester's requirements on services can be both functional and non-functional, which could span all three classes of service descriptions (profile, process model and grounding) in OWL-S. We use the multi-attribute utility to handle all the requirements in the same framework. To achieve this, we uniquely model all requirements as "constraints" on services and use the concepts of hard constraints and soft constraints to differentiate constraints and preferences.

Third, to discover a service based on a higher-level description creates convenience for the service users, but at the same time introduces challenges in capturing the service requests. Denying the requestor's follow-up descriptions or preference changes, and enforcing "one-time" decisions or non-intervention are too inflexible. Besides, it limits the exploration of the user's true service needs and concerns, and makes the discovery harder. All these lead to the inapplicability of the discovery results. Therefore, the request

semantics must have mechanisms to allow incremental development of the request model.

4.2. Service request modeling

4.2.1. Service constraints and preferences on service attributes

In this section, we will show how we adapt multi-attribute utility theory to model service requests, especially for composite services, and describe our efforts towards resolving the three issues mentioned in Section 4.1.

Constraints are generally conceived as "hard", which mean that only services that meet conditions are acceptable. For example, the user may say that "A first-class seat is required" when requesting an air ticket booking service. Preferences are "soft", which means if there are choices the user will have a preference on one option over the other. For example, "A first-class seat is preferred" is a preference.

Constraints are able to, among other things, serve two purposes in service composition: to provide information for defining the composite service and to be matched with service providers when pinning down on suitable component service providers. Some constraints indicate control flows that must be satisfied by two or more component services in the requested service. When defining the flow structure of the composite service, we extract the control flow information from the requestor's concerns and establish links for the involved services or data in the solution space. Consequently, the discovered solution complies with the flows implying by the constraints. Further, other constraints can indicate possible usefulness of a component service that will not be considered otherwise.

In order to take into consideration of semantics providing information on the condition that has to be satisfied to make a solution acceptable and on the criterion that makes the solution a better one, we adopt an interactive user preference assessment model used in the Automated Travel Assistant (ATA), an iterative flight itinerary building prototype [13]. According to the multi-attribute preference theory, we can model service request over a set of attributes

$$A = O \cup AT \cup OD \tag{5}$$

where

$$O = \{o_1, \quad , o_n, \quad , o_N\} \tag{6}$$

$$AT = \{a_1, \quad , a_m, \quad , a_M\} \tag{7}$$

$$OD = \{r_1, \quad , r_l, \quad r_L\} \tag{8}$$

each of which takes on values from a set of domains

$$\begin{aligned} &D = \{D_1, \quad D_k, \quad , D_K\}, \\ &D_k = Dom(attribute_j), \\ &k = 1,...,K, K = N + M + L \; and \; attribute_j \in A \end{aligned} \tag{9}$$

$$D_k = \{v_{k,1}, \quad , v_{k,i}, \quad v_{k,I}\} \tag{10}$$

As indicated in (5), we allow requesters to impose three types of requirements. Each type of requirements are considered as constraints or preferences on one of the three types of attributes respectively, i.e., a) attributes identifying service categories and service operations of the requested service, b) attributes identifying particular interesting properties of the requested service, and c) attributes showing the control flow in the process model of the requested service. For an attribute of the first type, or $o \in O$, the domain is the service categories, and the operations that are alternatives to each other for achieving the requester's goal. For example, the values in the domain may be "the transportation service category", "to transport by a rental car service" or "to transport through an air flying service". For an attribute of the second type, or $a \in AT$, the domain is all the values that the attribute can take while satisfying the constraints or preferences. The third types of attributes are binary variables that are pairs of named operations, i.e., $r = (o_i, o_j), o_i \in O, o_j \in O, r \in OD$. Their domains are {1, 0}. It is set to 1, when this pair has a precedence relationship, meaning one operation has to be performed before the other. Otherwise 0, i.e., when there is not a precedence relationship.

According to ATA, a preference or a constraint is a function

$$P_k(v) : D_k - [0,1] \tag{11}$$

This function is actually a utility function scaled to the interval of 0 to 1.

The score of a particular composition regarding this preference is scaled to 0, if this preference or constraint is fully unsatisfied and is scaled to 1, if this preference is fully satisfied. Same as the assumption made in ATA, we also assume that the preferences are additive independent [9]. Additive independent means that if a set of attributes' values are fixed, the preference score on the solutions with varying values of its complementary variable set will not be different. For an example, the user's preference on air ticket price does not depend on whether or not the ticket booking is processed before or after the hotel reservation.

With additive independence, the user's preference on the requested service is formulated by ATA as a weighted sum of preference functions as in (12).

$$error(v_1, \quad, v_k, \quad) = \sum_{k=1}^{K} P_k(v_k) * w_k \qquad (12)$$

Our approach is to adapt the above equation to model the overall preference value of a candidate composite service. Our model, as listed in (13), is based on a probabilistic estimation on the value of a possible composite service in the service requestor's eyes. The basic idea is that a service provider, under many circumstances, show non-deterministic or probabilistic characteristics. Some attributes, therefore, represent certain aspects of services that are probabilistic in nature.

$$score(v_1, \quad, v_k, \quad) = \sum_{k=1}^{K} P_k(v_k) * w_k * \Pr(v_k) \qquad (13)$$

The above formula models the overall preference score on a candidate composition by a set of triplets. The elements in a triplet represent an itemized score on a particular attribute instance, $p_k(v_k)$, the probability that the attribute takes on that value instance, $\Pr(v_k)$, and the weight, or how important the service requester thinks of the satisfaction of the preference, w_k. The weights and preference scores can both be adjusted for different runs of composition to produce the most preferred service. We show an example of the weighted preference score of a composition in table 1. The fourth row shows the product of the satisfaction level, the probability and the weight. The overall score is 0.72.

We accommodate the above user preference model for composite service requests into our service request language by introducing the language constructs. Due to space limit, only a few snippets of the constructs are in list 1.

Table 1.

	SUV Rental	Air Ticket Booking	Hotel Reservation	Evening Flight (Morning=1, Afternoon=.5, evening =.2)	(Air Ticket Booking, Hotel Reservation)
Score	1	1	1	0.2	1
Weight	0.1	0.2	0.2	0.2	0.2
Probability	1	1	1	0.5	1
	0.1	0.2	0.2	0.02	0.2

List 1.

```
<owl:Class rdf:ID = "UserConstraint">
...
</owl:Class>
<owl:Class rdf:ID = "UserSoftConstraint">
...
</owl:Class>
<owl:Class rdf:ID = "UserHardConstraint">
...
</owl:Class>
<owl:ObjectProperty rdf:ID = "hasScore">
...
<rdfs:domain rdf:resource = "#UserSoftConstraint"/>
...
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID = "hasWeight">
...
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID = "hasProbability">
...
</owl:ObjectProperty>
```

4.2.2. Iterative and interactive constraint revision

We argue that iterative and interactive composition is an effective way in building new services by composition. The first reason is that users' service needs are not straightforward to the composition system, due to their cognitive limitation when querying the services. Automated generation of a service process structure can only be based on an 'estimation' of the customer's service need. The automation process relies on the request's critiques to make the decision. Second, a user has preferences and constraints. According to the SWSL group's requirement document [14], when composing a service, relaxation or tightening the constraints shall be allowed. To take advantage of this flexibility, requesters tend to iteratively revise their preferences in achieving their main objectives. Third, the services over the Web show very varying, dynamic and non-deterministic features. Users' constraints may be in conflict with the existing services or even themselves. These conflicts are not obvious until the composition system builds the user preference model and tries to resolve them. Resolving the conflicts is an iterative process and gives opportunities for the user to adapt their needs to the exact available services in the real world.

We introduce language constructs to make the iterative process easy both for the requesters to make changes on their previous requests and preferences, and for the composition system to produce the acceptable composition. Due to space limitation, details are not listed here.

5. Application in Real World Scenario

In this section, we demonstrate how the proposed approach of modeling service requests is applied to a

real-world business scenario. The case is about a service request for a composition of a travel arrangement service. Assume the request wants a package of a round trip flights, 2-day hotel accommodation service and 2-day SUV rental service. This could be from an agency or an individual. The scenario of the service client is as follows:

- Need Air ticket booking service.
- Need hotel reservation.
- Need SUV Rental. But a rental car is not a big deal.
- Travel on May 22nd or May 23rd. Prefer morning flights. Evening flights are not preferable but still acceptable.
- Book a seat first, before reserve a hotel room.
- All payment will be made using a credit card. Get the cheapest deal possible.
- Total budget limit is $1000.

What is available through the registered services is as follows:

> EZ Airlines, Royal Hotel, TrainTrans Online, and Enter-Car are the companies that provide travel services and are registered with the service registry. EZ Airlines provides a service called AirTicketProcessing. One of its operations, AirTicketBooking, allows clients to book air tickets. Royal Hotel provides a service called roomEPassIssue to issue electronic passes and a promotional rate, through an operation called PromotionalEPass. Enter-car provides SUV reservation through its SUVRental operation. TrainTrans Online provides TrainTicketBooking for ticket booking.

The list of the constraints of the different players in this scenario in the service registry is as follows:

> EZAirline: Evening seats on 23 May and morning seats on 22 May may be available. Payment must be made by a VISA or MASTER card. Fare is between $200-$250.
> RayalHotel: PromotionalEPass is only available between May 23th , and June 24th, which may provide a special rate at $125.00 each night. Regular price is $250 each night. Accept all major credits. The operation requires that the message must use a reliable messaging protocol and encrypt a header with WS-Security using a X. 509 token.

We model the service request as constraints on the service attributes. The requests are presented here as a set of constraints in following standard format:

(*attribute category* : *attribute type v* : *preference*, *probability*, *score*, *weight*: *preference*, *probability*, *score*, *weight*:)

The request in the scenario above is presented as follows:

1. (Functional & Profile: OperationName n1: n1= AirTicketBooking, 100%, 1, 18%)
2. (Functional & Profile: OperationName n2: n2= PromotionalEPass, 100%, 1, 18%)
3. (Functional & Profile: OperationName n3: n3= SUVRental, 100%, 1, 10%)
4. (Functional & ProcessModel: Precedence p1: p1 = (AirTicketBooking, HotelReservation), 100%, 1, 18%)
5. (non-Functional & Profile: FlightDepartureTime t: t = morning of 22 May, 50%, 1, 18%: t = evening on 23 May, 25%, 0.2, 18%)
6. (non-Functional & Profile: Cost c, $650<=c<=$1000 and has to be minimized, 50%, 0.5, 18%: c<$650 and has to be minimized, 50%,1, 18%)

If we discover all possible solutions, the alternatives are what time to travel and whether to use the non-promotional or promotional hotel booking. The matchmaker has to decide if it is better to fly on 22nd by a morning flight or fly on 23rd evening and enjoy the discount hotel rate. If we list the preferences and scores of all the attributes as in Table 1, a weighted overall score can be calculated. The overall weighted score of flying on 22nd by a morning flight is 0.775 and that of flying on 23rd, 0.739. The former with a morning flight is considered a better solution to the requester.

6. Related Work

The concept of Semantics Web services was first proposed by [2]. McIlraith et al showed the semantic Web technology, which makes information on the Web better understood by computers, can also be applied to Web services and also how the markups of Web services benefits the Web service discovery, execution and composition. They also pointed out how users' constraints and preferences are the main thing that makes service discovery, execution and composition difficult. In that paper and in related papers [15], they describe a semantic Web service composition system based on a LP language and AI planning techniques. They believe that requests of complex services are often for a limited number of common services with different personalized preferences. Therefore, most of requests can be fulfilled by making use of pre-built general templates. We argue that sometimes requests are not able to be represented by general templates. Automatic or semi-automatic composition should make it possible to construct the service process structure given a request, without assuming whether the request can be represented by a limited number of general templates. At the same time, the incompleteness of knowledge on the service need makes discovering the service process structure even more challenging. We propose the interactive and interactive semi-automatic approach for service composition, also keeping in mind the needs of interaction in resolving user constraints and preferences.

User preference elicitation has been extensively studied in the literatures of interactive decision systems [16][13][17][18]. The literature reports a number of interactive ways to help the user to establish a preference model and to make decisions on tradeoffs among multiple preferences. Reported research on user preference elicitation focuses on the design of effective interaction interfaces, and on developing appropriate models of the preference or utility function. Some of the work has been targeted towards applications in certain vertical domains such as the airline and travel business. In this paper, we consider multi-dimensional preferences in the semantic Web services context, and study the applicability of multi-attribute decision

theory for Web service composition to provide a mechanism in service requests that enables composing complex composite services.

The research to date on Web service requests for service composition mostly comes from the AI community. In [19] [20], the authors pointed out that [21] lacks the flexibility in responding to the unforeseen situation. They reported a request language called XML Service Request Language (XSRL) that integrates AI planning and constraint satisfaction techniques, and a planning architecture that accepts requests in XSRL. The planning strategy is on an interleaving of planning and execution. We tried to take into consideration the user preferences into service composition, and to present them in the semantic Web framework. We perceive that the semantic in the request has to be analysis on the whole composite service instead of its component services and therefore, see it beneficial to compose the service first before actually invoking the service.

7. Conclusions

The contribution of the paper is to demonstrate the applicability of multi-attribute utility techniques to the problem of modeling semantics in service requests for composing composite services. We enhance OWL-S with language construct extensions that make possible a clear and uniform representation of semantics in service requests to be used in service composition. The key to the idea is to model the request semantics as revisable user constraints and preferences and apply multi-attribute utility techniques to resolve the user preferences iteratively and interactively. We also work out an example to demonstrate the process of modeling the service requests and of comparing various solutions according to the preferences.

Based on the result reported in this paper, we suggest the following ways to continue extending the models and methods for semi-automatic and automatic composition of Web services. 1) There are uncertainties and risks involved when discovering and composing Web services. Extensions can be made on an appropriate uncertainty model to complement the request processing. 2) There are usually criteria regarding the services that are common to all customers. Extension work can look into the criteria on certain aspects that can be treated as default.

8. References

[1]. Q. Liang, L. N. Chakarapani, S. Y. W. Su, R. N. Chikkamagalur and H. Lam, A Semi-automatic Approach to Composite Web Service Discovery, Description and Invocation, IJWSR, 1(4): 64-89, 2004.

[2]. S. McIlraith, T. C. Son, and H. Zeng, “Semantic Web Services,” IEEE Intelligent Systems. Special Issue on the Semantic Web. 16(2):46--53, March/April, 2001.

[3]. S. McIlraith and D. Martin, "Bringing Semantics to Web Services," IEEE Intelligent Systems archive, 18 (1), 2003.

[4]. D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. R. Payne, M. Sabou, M. Solanki, N. Srinivasan and K. Sycara, "Bringing Semantics to Web Services: The OWL-S Approach," First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), San Diego, 2004.

[5]. Web Services Architecture, http:// www.w3.org/TR/ws-arch/.

[6]. DAML-S, http://www.daml.org/services/daml-s/0.9/.

[7]. OWL-S: Semantic Markup for Web Services, http://www.daml.org/services/owl-s/1.1/overview/.

[8]. S. Y, W. Su, Q. Liang, L. N. Chakarapani, R. N. Chikkamagalur and H. Lam, “A Web Service Composition Framework: Discovery, Description and Invocation,” ICECR-6, Texas, 2003.

[9]. R. L. Keeney and H. Raiffa, “Decisions with Multiple Objectives: Preferences and Value Tradeoffs,” Wiley and Sons, New York, 1976.

[10]. D. von Winterfeld and W. Edwards, “Decision Analysis and Behavioral Research,” Cambridge, England: Cambridge University Press, 1986.

[11]. D. M. Kreps, “Notes on the Theory of Choice,” Under-ground Classics in Economics. Westview Press, Boulder, 1988.

[12]. R. Schafer, http://www.kbs.uni-hannover.de/~henze/ ABIS_Workshop2001/final/Schaefer_final.pdf.

[13]. G. Linden, S. Hanks, and N. Lesh, “Interactive assessment of user preference models: The automated travel assistant,” Proceedings, User Modeling '97, 1997.

[14]. SWSL, http://www.daml.org/services/swsl/.

[15]. S. McIlraith, and T. C. Son, “Adapting Golog for Composition of Semantic Web Services,” Proceeding of 8th Conference on Knowledge Representation and Reasoning (KR2002), April 2002.

[16]. P. Pu, “User-Involved Preference Elicitation,” Eighteenth International Joint Conference on Artificial Intelligence, Workshop on Configuration, Mexico, 2003.

[17]. C. Gonzales and P. Perny, "Graphical Models for Utility Elicitation," DIMACS/LAMSADE Workshop on Computer Science and Decision Theory, France, 2004.

[18]. C. Boutilier, F. Bacchus, and R. I. Brafman, "UCP-Networks: A directed graphical representation of conditional utilities," UAI 2001.

[19]. M. Papazoglou, M. Aiello, M. Pistore, and J. Yang, “Planning for requests against web services,” IEEE Data Engineering Bulletin, 25(4):41-46, 2002.

[20]. A. Lazovik, M. Aiello, and M. Papazoglou, “Planning and monitoring the execution of web service requests,” Technical Report #DIT-03-049, University of Trento, 2003.

[21]. BPEL, http://www-106.ibm.com/developerworks/library/ws-bpel/.

Acknowledgement

Special thanks to Dr. Ramayya KRISHNAN, Professor of Management Science and Information Systems, Carnegie Mellon University for very helpful advices on the paper.