

Singapore Management University Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

11-2007

Modeling Semantics in Composite Web Service Requests by Utility Elicitation

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/10.1007/s10115-006-0052-4>

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

 Part of the [Computer Sciences Commons](#), and the [Management Information Systems Commons](#)

Citation

LIANG, Qianhui (Althea); CHUNG, Jen Yao; and MILLER, Steven. Modeling Semantics in Composite Web Service Requests by Utility Elicitation. (2007). *Knowledge and Information Systems*. 13, (3), 367-394. Research Collection School Of Information Systems. **Available at:** https://ink.library.smu.edu.sg/sis_research/1193

This Journal Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email libIR@smu.edu.sg.

Qianhui Althea Liang · Jen-Yao Chung ·
Steven Miller

Modeling semantics in composite Web service requests by utility elicitation

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 multi-attribute utility theory (MAUT) based model of composite service requests. Service requests are modeled as user preferences and constraints. Two preference structures, additive independence and generalized additive independence, are utilized in calculating the expected utilities of service composition outcomes. The model is also based on an iterative and incremental scheme meant to better capture requirements in accordance with service consumers' needs. OWL-S markup vocabularies and associated inference mechanism are used as a means to bring semantics to service requests. Ontology conceptualizations and language constructs are added to OWL-S as uniform representations of possible aspects of the requests. This model of semantics in service requests enables unambiguous understanding of the service needs and more precise generation of the desired compositions. An application scenario is presented to illustrate how the proposed model can be applied in the real business world.

Keywords Web service composition · MAUT · Semantic service request · Expected utility · Preference structure · Additive independence · Generalized additive independence · Iterative and incremental request elicitation

Q. A. Liang (✉) · S. Miller
School of Information Systems, Singapore Management University, Singapore
E-mail: althealiang@smu.edu.sg, stevenmiller@smu.edu.sg

J.-Y. Chung
IBM T. J. Watson Research Center, NY, USA
E-mail: jychung@us.ibm.com

1 Introduction

Sharing and reusing resources is made much easier with the Web services model and through service-oriented software development. A Web services framework simplifies software development by allowing a new system to be composed out of existing components which have been made very easy to describe, publish, find, bind and invoke in the framework. 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 [18]. The elemental services are referred to as member services or component services. Service composition is the construction of composite services. Similar ideas, such as compositional verification [7, 21] have been used in the design of complex knowledge-based systems before service composition evolved into a separate research topic.

Ontological techniques, including static ontology and dynamic ontology [14] have been used as representations of semantic Web knowledge. Semantics of Web services [22, 23, 25] are the key to service composition, especially to 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. Behind the scene, the component services work as a collaborative team and in combination provide the requested service. Understanding what the registered services actually provide to its users is important in identifying the required component services, because the composition system must have this knowledge to tell whether a service can perform a certain job within the larger service. WSDL [37], DAML-S [8], and OWL-S [27] have emerged as marking up languages of Web services. They allow service providers to present descriptions of their services and to mark up their service descriptions in order to share knowledge of services and to perform knowledge inferences.

Service composition or composite service discovery also relies on good understanding of the service needs and matching the available services with the service needs. Service needs of service requestors are presented in the form of service requests. Service requests can be issued by a user looking for services or by 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, which keeps service needs from being well understood by machines. Lacking good semantics, service requests cannot make it clear what client really wants, How to interact with the user, and what the client prefers and cannot accept. Just as in the physical service world, only with the client’s need clearly in mind can services be composed correctly. This suggests to us that semantics of service requests make up an integral part of semantic Web services and have to be addressed properly.

Further, for service composition, semantics of service requests have to be obtained and developed in an incremental and iterative way. Interactions from human beings are unavoidable, especially when requests need to be modified for various reasons. Preferences of users must be captured and non-determinism of service providers and requestors must be considered. Our model is designed to satisfy these requirements.

In Sect. 2, we review the composite Web service model and the service composition approach reported in the previous work. In Sect. 3, we discuss the semantics of service requests for service composition. In Sect. 4, we present a new multi-attribute utility theory (MAUT)-based method to model semantic service requests using the OWL-S semantic framework. Two preference structures, i.e. additive independence and generalized additive independence, are applied to possible service requests and discussed in detail. OWL-S construct extensions for incremental request elicitation are introduced in this section too. In Sect. 5, we provide an example of using the model to represent the semantics for a request for a complex service and comment on the evaluation of the incremental scheme. We discuss related work in Sect. 6. Our conclusions are stated in Sect. 7 and a list of references is given in Sect. 8.

2 Composition in extended Web services model

One purpose of service composition systems is to respond to such service requests that can only be satisfied with results of compositions of services. Therefore, such systems can be mutually beneficial with an enhanced Web services model that has the capability to discover and invoke composite Web services in some automated manner. We extended the standard Web service model by introducing a Service Registry that, in addition to the well-known functions, is capable of answering complex queries by constructing or discovering composite services dynamically based on the semantic markups of registered services. We also introduced an add-on component called 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 then do not have to care whether the requested services are composite or simple services. This enhanced model is described in detail in our previous papers [18].

Previously, in Su et al. 2003 [33], WSDL has been extended to include specifications of restrictions on services in service descriptions of service providers. In this paper, we provide supplementary constructs to the OWL-S semantic Web service language that enable a service requestor to query the (Intelligent) Service Registry in a high-level declarative way. With these supplemental language constructs (to be described in Sect. 4.2), OWL-S can be used to make up a service request with the exact semantics required to construct a composite service.

After the (Intelligent) Service Registry receives a semantic service request, it looks for a registered simple service(s) in its database of services. If necessary, it may have to construct a composite service(s) on the fly. While constructing the service, the (Intelligent) Service Registry also ensures that the composite service is valid against the requestor's particular requirements. It then generates a composite service specification, which can be registered with the (Intelligent) Service Registry. If the requestor would like to have the composite service invoked, he contacts a Composite Service Processor, which takes the composite service specification, calls the registered component services and returns the result.

Because the understanding of the complex service need of a service requestor is incomplete in most cases and there is non-determinism in resolving a requester's preferences on services, the (Intelligent) Service Registry is designed based on a semi-automatic approach [18]. Besides automatic AI search techniques, 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

In the context of service composition, the role and responsibility of service users is to provide a request to describe the service need precisely. Considering the fact that, in most cases, a complex service need cannot be described using one word or by one parameter, users want to present the description information to the composition system in a well-structured and easy-understandable way. The issue of how to make composition systems understand the exact service need goes hand in hand with the issue of how to allow users to specify their service needs unmistakably. Due to the limitations in both human cognition and machine representation, both of these tasks have proven to be major obstacles in automating service composition. The purpose of the paper is to develop such a semantic model for both the user and the composition system to communicate and improve the composition goal in the form of a service request.

A service request serves as the common contract that different roles in the Web services model can refer to when discovering the services. *Semantics of service requests* refer to the conceptualizations of the types of information required to describe a request for services. Here, we are only concerned of composite services, because requests for simple services are trivial. The conceptualizations of the information types and their mapping into the elements of the MAUT model we established for service composition will be discussed in Sects. 4.2.1 and 4.2.2. When semantics of service descriptions become explicit and understandable to machines, it is possible to use a piece of software to analyze the specifics of services. Similarly, when a Web service request specifies the desired service in such a way that the service demand is clearly understood by machines, it is also possible for a piece of software to select and compose services automatically. Furthermore, since the construction of a composite service involves selecting and organizing multiple services and dealing with multiple service providers, the importance of semantics of requests becomes more obvious. Semantics of service requests can be captured by designing a semantic model of service requests. A good semantic model must include information that makes automatic service composition possible and feasible. A good semantic model shall also provide a mechanism that is natural and easy to use by the requestors. The rest of the section gives a briefing on these two considerations.

Given service requests, discovering new composite services firstly means 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 and non-functional description of the desired service, the service composition system can ask the requestor to identify the domains that the requested service falls in. Au-

automatic discovery of composite services is made feasible if the services under consideration are limited to particular domains. We believe a specific service request can be narrowed down to one or several service domains under some service categorizations. Therefore, the composition always happens within the context of certain domains.

Discovery of composite services also means automatically selecting appropriate constituent Web services to perform the sub tasks and to make sure these services comprise a valid and useful service as requested. The service discovered must adhere to requested properties [25]. A user located in Singapore may say, for example, “Make the travel arrangement for my conference trip. If morning flights are unavailable, I will go by sea.” 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 preferences and constraints. The composition agent shall compare the constraints of service providers, such as “Singapore Airlines does not have morning services from Singapore 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 getting the requester’s preference and constraints, and resolving the conflicts if needed, one requirement is a mechanism that allows the user to elicit service requests with preferences easily and that assists the user to achieve their goals to the greatest extent possible. A good thing to do when modeling the semantics of requests is to introduce an iterative and incremental 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, within one iteration. They do not allow a user to develop the preference model incrementally and make tradeoffs by adjusting the preferences. Such inflexibility prevents the discovery result from being acceptable.

Please note that although users are obligated to provide description information of their needs, not all description information shall be compulsory as user input for composition. The composition system following the model makes an informed guess if it is not provided with the complete information, and displays a list of proposed composite services to the user to verify and select from.

4 Modeling semantic service requests for composite Web services

In this section, we discuss the modeling of the requirements of a service request with the mentioned considerations. Since OWL-S is the most prevalent language used to encode Web service capabilities both for advertisement and for requests [22], 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 [27], 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 [23]’s “Web service lifecycle in OWL-S

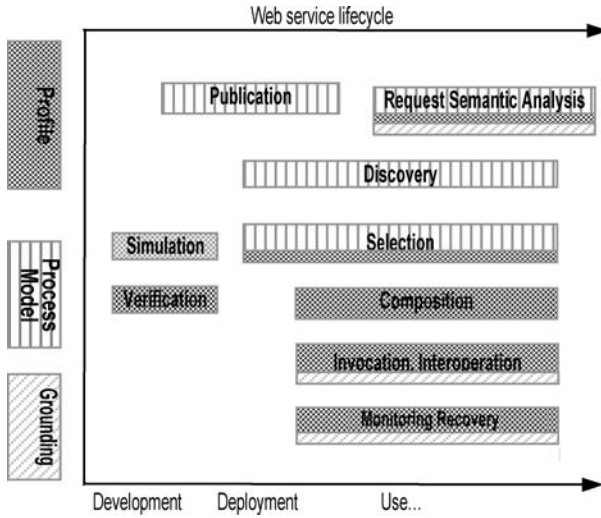


Fig. 1 Web service lifecycle with request semantic analysis, enhanced from [23]

ontologies” by adding a function for Request Semantic Analysis. The amended diagram is listed in Fig. 1.

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 incremental constraint (preference) revision. We also propose the OWL-S based service request language constructs to accommodate these semantics.

4.1 Multi-attribute utility theory and service requests

Multi-attribute utility theory (MAUT) and preference elicitation have been studied extensively in the literature of decision analysis [15]. 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 of meeting service requests using available services can be based on the same techniques developed for decision analysis and for analyzing multi-dimensional user preferences in decision-support systems.

The standard multi-attribute decision analysis is described in [15] and [36]. Applying it to the (Web) service domain, the set of the service alternatives, i.e., the outcome, S , is defined by a set of value dimensions, as in (1).

$$V = \{d_1, d_2, \dots, d_n\} \quad (1)$$

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

$$u(V) : 2^V \rightarrow R \quad (2)$$

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 ($s > s' \Rightarrow \neg(s' > s)$) and transitive ($(s > s') \wedge (s' > s'') \Rightarrow (s > s'')$) [16] binary relation on the set of options. $s > s'$ indicates that the alternative s is preferred to alternative s' . s and s' are said to be indifferent, if ($\neg(s > s') \wedge \neg(s' > s)$). A utility function is indicating a unique preference order over the individual outcomes in S , i.e.,

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

Let's assume the probability distribution over V is denoted as Pr. Let Z_1, \dots, Z_k be a partition of V . Z_1, \dots, 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 V , Pr_1 and Pr_2 are indifferent. It has been proved [15] that Z_1, \dots, 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.,

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

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 [31], i.e. $d = f(a_1, a_2, \dots)$. In the case of Web services, the variables of services in the service requests are heterogeneous, which makes it inconvenient to subsume them in a hierarchy. Also, it is sufficient to ask a requestor to present the required properties in higher-level ontologies of OWL-S with a flat structure. Based on the above considerations, when modeling the service requests, we use a simple model that directly operates on variables avoiding a more complex model that involves a dimension hierarchy. We will use the following definition of the service outcome space: $V = \{v_1, v_2, \dots, v_n\}$, where v_i are service variables that the requestor can use to express their desired service.

Second, the requestor's requirements on services can be either functional or non-functional, which could span all three classes of service descriptions (profile, process model and grounding) in OWL-S. 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 "FlightDepatureDate between June 3rd, and June 5th" or "Encrypt a header with an X.509 token". We use the multi-attribute utility to handle all the requirements in the same framework. To achieve this, we uniformly model all requirements as "constraints" on services and differentiate hard constraints and soft constraints.

Third, to discover a service based on a higher-level description creates convenience for service users, but at the same time introduces challenges in capturing the service needs. Denying the requestor's follow-up descriptions or preference

changes to enforce “one-time” decisions or non-intervention are inflexible and restricted. Besides, it limits the exploration of the user’s true service needs and concerns, and makes the discovery harder. All these may 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

This section discusses the model of service requests through explaining what do we mean by service constraints and preferences in Sect. 4.2.1., highlighting the related ontology conceptualizations of OWL-S and illustrating the service attribute structure to use when they are all additive independent in Sect. 4.2.2 and when they are not additive independent in Sect. 4.2.3, and showing the accommodation of iterative and incremental constraint elicitation in Sect. 4.2.4.

4.2.1 Service constraints and preferences

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 attacking the three issues mentioned in Sect. 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. Since constraints can be seen as a special type of preferences, in this paper, constraints and preferences are used interchangeably to refer to both hard and soft constraints.

Constraints are able to, among other things, serve two purposes in service composition: to provide information in defining the structure of the composite service and to describe identifying properties that validate suitable component service providers. Constraints serving the first purpose are usually functional requirements. Some of such 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 control flow information from the requestor’s concerns and establish various links for the involved services in the solution space. Consequently, the discovered solution complies with the flows implied by the constraints. Such constraints may also indicate possible usefulness of a component service that will not be considered as required otherwise. On the other hand, a majority of constraints are so-called attribute constraints, which are defined upon attributes or parameters. They describe interesting properties and tell “qualified” providers of a certain service from “unqualified” ones. They are used to filter out providers that are in conflict with the requesters’ interests or concerns. In service-oriented paradigm, attributes are defined for each service object to describe its identifying properties. Therefore, attribute constraints can be defined as restrictions imposed on single or multiple attributes of one or more services that must be complied with. Attribute constraints on a single attribute are referred to

as single-attribute constraints and attribute constraints on more than one attribute are referred to as inter-attribute constraints.

4.2.2 Additive independence request structure

In order to take into consideration semantics about a condition that has to be qualified to make a solution acceptable for the request and about the criterion that makes a solution a better one, we highlight in the paper the following four ontological concepts related to composite services, i.e. *Service Category*, *Service Operation/Atomic Process*, *Service Parameter* and *Control Construct*. All four concepts are from OWL-S. The concept of Service operation is in WSDL, which corresponds to the most canonical grounding of atomic process in OWL-S. We will use service operation in the rest of the text. With these ontological concepts, we ask requesters to impose different requirements on a requested composite service. Accordingly, we model service requests over a set of variables A of MAUT [15]. Each requirement is considered as a constraint defined as a relation on any combination of variables of one of the four variable types corresponding to the four ontological concepts, indicated in (5) and (6):

- a) The single attribute c of type C corresponds to the service categories of the requested service based on a service categorization scheme or service taxonomy such as [32, 35]. It is a multi-value attribute and each value refers to an identified service category.
- b) The single attribute o of type O corresponds to the service operations of the requested service. It is a multi-value attribute and each value refers to a service operation.
- c) Attributes of type AT define the parameters on the concerned and differentiating properties of the requested service.
- d) Attributes of type OD link operations with one another and are used to define control constructs of the process model of the requested service.

$$A = A_C \cup A_O \cup A_{AT} \cup A_{OD} \quad (5)$$

where

$$\begin{aligned} A_C &= \{c\}, A_O = \{o\}, \quad A_{AT} = \{a_1, \dots, a_m, \dots, a_M\}, \\ A_{OD} &= \{r_1, \dots, r_l, \dots, r_L\} \end{aligned} \quad (6)$$

each of which takes on values from a set of domains respectively:

$$\begin{aligned} D &= \{D_1, \dots, D_k, \dots, D_{M+L+2}\}, \\ D_k &= \text{Domain}(\text{var}_k), \quad k = 1, \dots, M + L + 2 \text{ and } \text{var}_k \in A \end{aligned} \quad (7)$$

The domain of attribute c is all subsets of registered service categories. For example, c can take {transportation service category} as its value. Within the identified service categories, there are multiple alternative services for achieving the requester's goal. Useful service operations will be referred to by attribute o . For example, the operation attribute can take the value of {transport by a rental car service}, or alternatively {transport through an air flying service}. The domain of

an attribute of type AT , or $a \in A_{AT}$, is all the values that a can take while satisfying the constraints or preferences. All attributes of type OD are binary variables that are defined on pairs of named operations, i.e. $r_l = \gamma(o_i, o_j)$, $o_i, o_j \in o$. The function γ maps operation pairs to $\{1, 0\}$. It is mapped to 1, when the 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. Note that this definition of OD attributes only reflects one specific type of such constructs defined in OWL-S, because we do not expect the user to provide complicated control structures. This definition can be expanded to include more complicated structures if there's a need.

A preference or a constraint is a function

$$u_k(v) : D_k \rightarrow [0, 1] \quad (8)$$

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

The score of a particular composition regarding the preference is scaled to 0, if the preference or constraint is fully unsatisfied and is scaled to 1, if the preference is fully satisfied. In this section, we assume that the service variables are additive independent [15]. Additive independence means that if a set of attributes' values are fixed, the preference scores 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.

Automated Travel Assistant (ATA), an iterative flight itinerary building prototype [20], proposed an interactive user preference assessment model. Assuming additive independence, the user's preference on the requested service is formulated by ATA as a weighted sum of preference functions as in (9). With this assumption, there are no interactions among any variables of any of the above four types and thus the preference of a requestor only depends on each variable.

$$\text{error}(v_1, \dots, v_k, \dots) = \sum_{k=1}^K C_k(v_k)w_k \quad (9)$$

We can adapt the above equation to model the overall preference score of a candidate composite service. Assuming additive independence, preferences over service compositions can be formulated as probabilistic estimations on the values of possible service compositions defined over all its service variables. Service providers, under many circumstances, show non-deterministic or probabilistic characteristics. Some variables, therefore, represent certain aspects of the service composition process that are probabilistic in nature. For this reason, only the likelihood of each possible outcome can be known instead of which outcome will appear for certainty. This is the case for online businesses serving a large amount of transactions at the same time. The customer cannot be guaranteed by 100% the availability of the merchandizes or services. In another scenario, the business model, such as buyer auction as successfully adopted by companies like price-line.com, determines that the service requester will not be notified of the exact output. Our model is listed in (10).

$$\text{score}(v_1, \dots, v_k, \dots) = \sum_{s \in S} u(V) \Pr(s) = \sum_{s \in S} \left(\sum_{k=1}^K u_k(v_k)w_k \right) \Pr(s) \quad (10)$$

Table 1 Example of the weighted preference scores of a composition

	SUV rental	Air ticket booking	Hotel reservation	Flight departure time	Air ticket booking, hotel reservation
Score	1	1	1	AM = 1 PM_E = 0.5 PM_L = 0.2	1
Weight	0.1	0.225	0.225	0.225	0.225
Probability	Uniform distribution in the outcome space				

Formula (10) models the overall preference score on a candidate composition by two nested summations. The inner summation sums up the products of the itemized score of a particular variable instance, $u_k(v_k)$, and the weight, w_k , or how important it is to satisfy the constraint on that variable. The outer summation is the expected value of the utility functions on various outcomes, or EU, as a dot product of the outcome utility vector and the particular lottery over the outcome space, $\Pr(s)$. 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 scores of a composition in Table 1. PM_E means afternoon flights and PM_L means evening flights. (Air Ticket Booking, Hotel Reservation) means that air ticket booking proceeds hotel reservation. Since the weight of SUV rental is only 0.1, relatively lower than all other weights, SUV rental service is not considered as important as other requirements.

One candidate composition is preferred to the other, if the first induces greater expected utility. The candidate compositions are represented by probability distributions over the outcome space, S . (11) shows the preference relationships among candidate compositions.

$$\Pr_1 \succ \Pr_2 \quad \text{iff} \quad \text{score}_1(v_1, \dots, v_k, \dots) > \text{score}_2(v_1, \dots, v_k, \dots) \quad (11)$$

We accommodate the above user preference model of 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.

In order to apply this model, there must be prepared statistic data about service providers for the composing system to use. Such service data can be obtained from marketing research companies that constantly conduct customer surveys to collect service data and observe service provider behavior or from the customer feedback provided to the organization that runs the composing system. Data from the latter source may be limited to a small number of businesses. However, most businesses in the same domain usually share common types of business rules in handling customer demands when they do not have the capacity to meet all individual customers. We believe it is reasonable to assume at this stage that data obtained from a small sample of businesses is likely to be applicable to other businesses in the same business domain. An alternative way to obtain this information is for service providers to provide such statistics to the service composition system, which uses it only for composition purpose without releasing the business confidential information to the public.

List 1. Snippet of language constructs for user preference model of composite service requests

```

<owl:Class rdf:ID = "UserConstraint">
...
</owl:Class>
<owl:Class rdf:ID = "UserSoftConstraint">
...
</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>
...

```

4.2.3 Generalized additive independence request structure

In some circumstances, the assumption of variables being additively independent, as Sect. 4.2.2., may not be true to the complete set of individual variables. For example, how much an operation is preferred by the requestor as necessary will change the structure of the preference relation of outputs. At the very least, those outputs with different satisfaction levels on that particular operation will be affected. In this case, we need to generalize the assumption of additive independence.

Generalized Additive Independence (GAI) is a generalization of additive utilities. Unlike additive independence, it allows interactions between variables and decomposes the utility function into a sum of sub-utilities on the sets of interacting attributes. If there exists a not necessarily disjoint partition of the original variables set, such that the utility function, $u(\cdot)$ can be decomposed as a summation of the utility functions of the partitioned variable sets [10], these variables are referred to as generalized additive independent for the preference relation defined by $u(\cdot)$. GAI is a less stronger independence assumption than additive independence and provides more flexibility in the structure of utility function [1, 10–12]. Since GAI is reasonably applicable to the service composition domain, we model the user utility function as GAI by rewriting (10) into (13) through the substitution of (12). Variable set $\{v_k\}$ are decomposed into $\{V_{I_{k'}} | V_{I_{k'}} \subset \{v_k\}\}$, where $I_{k'}$ is an index set to v and $\bigcup_{k'=1}^{K'} I_{k'} = \{1, \dots, K\}$. $I_{k'}$ and $V_{I_{k'}}$, may overlap with each other

respectively. The union of all $V_{I_{k'}}$ is the original variable set.

$$\begin{aligned} u(V) &= \sum_{k'=1}^{K'} \bar{u}_{k'}(V_{I_{k'}}) \\ &= \sum_{k'=1}^{K'} w_{k'} u_{k'}(V_{I_{k'}}), \quad \text{where } \{v_1, \dots, v_k, \dots\} = \cup_{k'=1}^{K'} V_{I_{k'}} \end{aligned} \quad (12)$$

$$\begin{aligned} \text{score}(v_1, \dots, v_k, \dots) &= \sum_{s \in S} \left(\sum_{k'=1}^{K'} u_{k'}(V_{I_{k'}}) w_{k'} \right) \Pr(s), \quad \text{where } \cup_{k'=1}^{K'} V_{I_{k'}} \\ &= \{v_1, \dots, v_k, \dots\} \end{aligned} \quad (13)$$

Referring to four types of variables in service requests discussed in the previous section, different types of variables can correlate to other variables in different ways. Studying whether variables interact with each other and what variables interact with each other is necessary to decide the partition of the variables. Variable c identifies the service categories involved. It represents what the service requestor wants to do, which shall not be affected by the distribution of other aspects of the outcome. Therefore, this variable forms the first participation. Variable o determines the service operations that are believed to be useful constituency of the complex service. It may correlate with the variables of type AT . For example, whether a traveler considers taking a train or a flight may depend on whether a nonstop-flight is likely or whether the discount ticket fare is likely. Variable o may also interact with the variables of type OD . The traveler preference of a flight or a train may affect his preference of a taxi pick-up before checking into the hotel, given the common sense that airports always locate far away from the city while train stations may be close to the city area. A variable of type AT may correlate with other variables of its own type. For example, the preference over price may depend on whether a flight departs during the preferred timeframe.

The algorithm listed below establishes service variable partitions $\{V_{I_i}\}$ for decomposing the composition utility. It starts by instantiating service variables c and o . c is assigned all service categories selected by the user as in step (b) and o is assigned all service operations picked from the selected categories as in step (c). The algorithm then goes to set the binary variables of type OD as in step (d) through to step (e). For each operation pair in o , if the user has defined a precedence over it, the corresponding variable in OD is set to 1, otherwise 0. Since there are $J \times J$ possible ordered pairs (where J is the number of operations in o), the variables can be indexed by a linear order indicated in the algorithm. After these initializations, algorithm starts creating partitions of service variables. It firstly creates a partition for the single variable of type C as in step (f). Each inter-attribute constraint is then examined and a new partition is created as in step (g) through to step (k). The algorithm repeatedly does the following two things: Whenever an attribute that is not already in this partition is seen, it is added to the current partition. Whenever an operation is seen, o is added to the current partition if not already in. After inter-attribute constraints, the algorithm checks each single-attribute constraint. If the attribute is not in any of the created partitions, a new singly partition is created

for that attribute as in step (h). The last two jobs of the algorithm are to create a partition for each non-zero variable of OD if that variable is not part of the created partitions as in step (i), and to create a partition for variable o if it is not in any partitions as in step (j).

Algorithm 1. Algorithm to establish service variable partitions for decomposing the composition utility

```

/* service variable establishment and instantiation */
Initialize service variable set by adding service variable  $c$  and  $o$ ;                                a)
Instantiate  $c$  by the requested service categories;                                          b)
Instantiate  $o$  by the possible operations picked from the categories in  $c$ , i.e.  $o_1, o_2, \dots, o_j$ ; c)
for ( $l = 1; l++; l < J$ )                                                                    d)
    for ( $m = 1; m++; m < J$ )
        if there is a precedence constraint  $(o_l, o_m)$ 
             $r_{(l-1)*J+m} = 1$ ;
        else
             $r_{(l-1)*J+m} = 0$ ;                                                                e)
/* variable partitioning */
 $V_{l_1} = \{c\}$ ; /* first partition for category attribute */                                f)
 $i = 1$ ;
for each inter-attribute constraint { /* variable partitioning based on inter-attribute constraints*/ g)
     $V_{l_{i++}} = \{\}$ ;
    for each distinct item in the constraint {
        if it is an variable  $a_{s_{i,k}} \in A_{AT}$ 
             $V_{l_i} = V_{l_i} \cup \{a_{s_{i,k}}\}$ ;
        If it is an operation  $o_{s_{i,k}} \in o$ 
             $V_{l_i} = V_{l_i} \cup \{o\}$ ;
    }
}                                                                                              k)

/* variable partitioning based on single-attribute constraints*/
for each single-attribute constraint on attribute  $a_{s_i}$                                        h)
    if  $a_{s_i} \notin \bigcup_{m=2}^i I_m$ 
         $V_{l_{i++}} = \{a_{s_i}\}$ ;
for ( $l=1; l++; l < J$ ) /* partitions for  $OD$  variables */                                    i)
    if  $r_l = 1, r_l \notin \bigcup_{m=2}^i I_m$ 
         $V_{l_{i++}} = \{r_l\}$ ;
if  $o \notin \bigcup_{m=2}^i I_m$  /* a partition for  $O$  variable  $o$  */                                    j)
     $V_{l_{i++}} = \{o\}$ ;

```

After partitioning the variable set, we need to calculate the sub-utility function of each variable sub set in the partition, or $u_{k'}(V_{l_{k'}})$ as in (12) and (13), so that we

can build the utility function of the complete variable set, or $\text{score}(v_1, \dots, v_k, \dots)$ as in (13).

We adopt what was proposed by Fishburn [10] to construct the sub-utility functions for each sub set of the variable set. We continue to use the same notation of V and V_{I_i} . V denotes the complete set of service variables and V_{I_i} denotes the sub set of variables indexed by the numbers in I_i . To simplify the notation, we use V_i to denote V_{I_i} . Under GAI, the sub-utility functions for each sub set can be shown as in (14). Sub-utility $u_{k'}(\cdot)$ on subset $V_{k'}$, or $u_{k'}(V_{k'})$, is calculated as a sum of utilities of certain several other outcomes, $u(V[I])$. These outcomes have the same value as $V_{k'}$ on some variables and have the default values on the rest variables. For example, outcome $V[I_1]$ has the same values as in V for variables that are index by the numbers in I_1 and has default values for all other variables. There are two important related notions. The notion of a default outcome, denoted by $V^0 = (v_1^0, v_2^0, \dots, v_k^0)$, refers to an arbitrary assignment to the set of variables. $V[I]$ refers to a (complete) outcome where variables in I remain the same value as in V and all others are set to the corresponding arbitrary value in V^0 . The second formula in (14) tells that the sub-utility function $u_{k'}(\cdot)$ on the local sub set of variables, $V_{k'}$, is defined as the sum of the utility function of outcome $V[I_{k'}]$ and a two-level nested summation. In the outer summation, it switches the sign of the inner summation between 1 and -1 to indicate either adding or lessing the result of the inner summation. The inner summation iterates through a certain number, j , of sub sets to find out the intersection between each sub set V_{i_s} iterated through and the local sub set $V_{k'}$. j intersecting sets are then intersected again to get the utility function on $V[\bigcap_{s=1}^j \dots]$. All utilities with different iteration numbers j , ranging from 1 to $k' - 1$, are then added together and multiply by 1 or -1 .

$$u_1(V_1) = u(V[I_1]),$$

$$u_{k'}(V_{k'}) = u(V[I_{k'}]) + \sum_{j=1}^{k'-1} (-1)^j \sum_{1 \leq i_1 \dots < i_j < k'} u \left(V \left[\bigcap_{s=1}^j I_{i_s} \cap I_{k'} \right] \right) \quad (14)$$

Now the elicitation of the utility of an outcome is simplified to specifying utilities of certain several outcomes. However, notice that $V[I_{k'}]$ is still a complete outcome over all variables in the complete variable set. This means that to calculate sub-utilities, the user still has to assess the tradeoffs in domains of many variables, which may be hard for the user. To further reduce the complexity, we can restrict elicitation to utilities of subsets of variables instead of the complete variable set. Braziunas et al. suggested separating the elicitation process into local elicitation and global scaling [6], which we adopted.

For each subset I_i , a top and bottom anchor outcome, $V[I_i]^T = (V_{I_i}^T, V_{I_i^C}^0)$ and $V[I_i]^\perp = (V_{I_i}^\perp, V_{I_i^C}^0)$ are chosen. $V_{I_i}^T$ and $V_{I_i}^\perp$ are the top and bottom values of I_i respectively. C_i is the conditioning set of I_i , defined as the union of all subsets that overlap with I_i excluding the variables in I_i , or $C_j = \bigcup_{j \in I_i} (\bigcup_{k: j \in I_k} I_k) - I_i$.

To give an example, if $V = \{\text{flightStops}, \text{flightDepartureTime}, \text{Airlines}\}$, $V_1 = \{\text{flightStops}, \text{Airlines}\}$ and $V_2 = \{\text{flightDepartureTime}, \text{Airlines}\}$. $V^0 = \{v_1^0, v_2^0, v_3^0\} = \{1, 8:00 \text{ AM}, \text{UnitedAirlines}\}$. $V_{I_1}^T = \{0, \text{Northwest}\}$. $V_{I_1}^\perp = \{3, \text{Budget}\}$. $C_1 = \{v_2\} = \{\text{flightDepartureTime}\}$ and $V_{I_1}^0 = \{8:00 \text{ AM}\}$. Therefore,

the top anchor outcome of I_1 is $V[I_1]^T = (V_{I_1}^T, V_{I_1^C}^0) = \{0, 8:00 \text{ AM, NorthWest}\}$ and the bottom anchor outcome of I_1 is $V[I_1]^\perp = (V_{I_1}^\perp, V_{I_1^C}^0) = \{3, 8:00 \text{ AM, Budget}\}$.

It was proven in [6] that local elicitations with respect to local anchors can be used to evaluate local outcomes, because the rest variables are irrelevant to the local outcome preference levels. Using local value functions, which are only locally calibrated value functions, sub-utility functions can be defined on $v(\cdot)$, local value functions, and $u_i^T = u(V[I_i]^T)$ and $u_i^\perp = u(V[I_i]^\perp)$, utilities of the anchor outcomes. For details, please refer to [17].

4.2.4 Iterative and incremental request elicitation

Iterative and incremental request elicitation improves the effectiveness of composite service building. Such process gives service requestors more flexibility when describing their needs and allows decision-support systems to improve the utility model to deliver better payoff to the user. We would like to discuss briefly why we need iterative and incremental service request elicitation.

The first reason is that users' service needs are not straightforward to the composition system, due to their cognitive limitation in describing the service needs. 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 requestor's critiques to make the decision. Research on incremental utility elicitation [3, 13, 30] has also provided proof that an incremental and interactive theoretic framework can benefit the decision making by updating the incomplete model based on user feedback.

Second, a user has preferences and constraints. According to the SWSL group's requirement document [34], 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, 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, which gives opportunities for the user to adapt their needs to the available services in the real world.

The need for an incremental elicitation scheme can be interpreted as the need for feedbacks on service attribute constraints provided by the requestors such that the request can be modified into a more complete or non-conflict one. Feedbacks can take different forms depending on the problems in the requests. If a requested data entity cannot be generated due to not enough input data entities provided, possible operations and the data entity input(s) that may have to be provided to produce the requested output shall be reminded using attribute o . If the failure of service composition is due to a particular non-satisfiable constraint, the constraint will be prompted to the user, along with a suggestion for how to relax that constraint via the attributes of type AT .

4.2.4.1 *Iterative elicitation*. *Iterative elicitation* refers to user changing their previous specifications on the weights and scores, possibly in repetition. This provides the system with the adaptability of changes of both users and the service providers. This also provides opportunities for the decision making process to retreat from unsuccessful efforts in solving unsolvable problems.

Several ontology conceptualizations need to be introduced for this purpose including *version*, *validity*, *weight* and *score*. We introduce corresponding 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. Please refer to List 2 for details. We tend to keep the history of all incremental changes making the service requests stateful. *UserConstraint* encloses language constructs that capture the version evolution of the user constraint and the validity of the constraint. Each elicitation of user preference is labeled with a version number, which takes on non-negative integer numbers and is incremented by 1 for every change on the preference. In this case, the user is allowed to change six times. At the same time, every time the user weight or score is revised, the new weight or score is appended to the end of the weight list or score list. In order to support history recording, the ranges of *hasWeight* and *hasScore* are declared as container resources.

List 2. Language constructs to make iterative process

```

<owl:DatatypeProperty rdf:ID = "hasVersion">
  <rdfs:label> Which is the current version of this constraint</rdfs:label>
  <rdfs:domain rdf:resource = "#VersionedThings"/>
  <rdfs:range rdf:data = "&myTypes;nonNetativeIntegerLT6"/>
</owl:ObjectProperty>
<owl:Class rdf:ID = "VersionedThings">
  <rdf:unionOf owl:parseType = "Collection">
    <owl:Class rdf:about = "#Parameter"/>
    <owl:Class rdf:about = "#Process"/>
    <owl:Class rdf:about = "#Precedence"/>
    <owl:Class rdf:about = "#Category"/>
  </rdf:unionOf/>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasVersion"/>
    <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    <owl:maxCardinality rdf:datatype = "&xsd;nonNegativeInteger">1</owl:maxCardinality>
  </owl:Restriction>
</owl:Class>
<owl:DatatypeProperty rdf:ID = "isValid">
  <rdfs:label> 0 if the preference is canceled by the user, otherwise 1</rdfs:label>
  <rdfs:domain rdf:resource = "#UserConstraint"/>
  <rdfs:range rdf:data = "&xsd;boolean"/>
</owl:ObjectProperty>
<owl:Class rdf:ID = "UserConstraint">
  <rdfs:comment> A user preference model </rdfs:comment>
  <rdfs:subClassOf owl:resource = "&expr;Expression;#Condition"/>
  <rdfs:subClassOf owl:resource = "#VersionedThings"/>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#isValid"/>
    <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
  </owl:Restriction>

```

```

<owl:Restriction>
  <owl:onProperty rdf:resource="# hasWeight"/>
  <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
  <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
</owl:Restriction>
</owl:Class>
<owl:ObjectProperty rdf:ID = "hasWeight">
  <rdfs:label > Weight on this constraint </rdfs:label>
  <rdfs:domain rdf:resource = "#UserConstraint"/>
  <rdfs:range rdf:resource = "weightSequence"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID = "hasScore">
  <rdfs:label> Score on this constraint, only softconstraints have score </rdfs:label>
  <rdfs:domain rdf:resource = "#UserSoftConstraint"/>
  <rdfs:range rdf:resource = "scoreSequence"/>
</owl:ObjectProperty>

```

...

4.2.4.2 Incremental elicitation. *Incremental elicitation* refers to user adding new preferences to give more complete descriptions of the utility model of service compositions. Incremental elicitation helps to achieve better utility function estimations, when the utility functions cannot be precisely derived based on the limited number of user elicitations. In the case that the utility function estimation does not turn out to be satisfying, incremental elicitation drives the system towards a better solution.

In the case of additive independence, the utility function can be seen as a linear estimation from a number of single-variable sub-utility functions as shown in (4). Each of the sub-utility function is then factored into a local value function $u_k(v_k)$ and a weighting factor w_k , as in (10). When a new preference on two outcomes, $a \succ b$, is added, we get $\vec{w} \cdot a > \vec{w} \cdot b$, where \vec{w} is the weighting vector. So $\vec{w} \cdot (a - b) > 0$. We can now transform the problem of introducing the new elicitation into the old utility model to the problem of solving the following set of constraints, as suggested in Bylthe [3]:

Maximize $c \cdot \vec{w}$

$$A \cdot \vec{w} > 0, \quad -1 \leq w_i \leq 1, \quad 1 \leq i \leq n$$

In the above constraints, c can be assigned the previous utility estimates, leading to a conservative updating scheme [3]. \vec{w} is an n -dimension weighting vector. A has rows of preference vectors $\vec{a} - \vec{b}$.

5 Application scenario and scheme evaluation

In this section, we demonstrate how the proposed approach of modeling service requests is applied to a real-world business scenario. We will first give an example for additive independence. We then revise this example to show how GAI works for composite service requests. We also propose a possible way to evaluate the iterative and incremental scheme of the model.

5.1 Additive independence

The case is about a service request for a composition of a travel arrangement service. Assume the requestor wants a package of a round trip flights, 2-day hotel reservation service and preferably 2-day SUV rental service. Air ticket booking must be done before hotel reservation. The cost must be between \$650 and \$900 and cost must be minimized. Travel on 22nd May or 23rd May. Morning flights are preferred. This could be from an agency or an individual. The scenario of the service client is as follows:

- Travel Arrangement category.
- Need Air ticket booking service.
- Need hotel reservation.
- SUV Rental is preferred.
- Cost $< \$650$ preferred and Cost $> \$900$ not acceptable.
- FlightDepartureDateTime < 12 PM on 22nd May or 23rd May is preferred.
- Air ticket booking precedes hotel reservation.

What is available through the registered services is as follows:

EZ Airlines, Royal Hotel, Budget 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. Both Royal Hotel and Budget Hotel provide 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 May 23rd, and morning seats on May 22nd, may be available. Payment must be made by a VISA or MASTER card. Fare is between \$200 and \$250.

RoyalHotel: PromotionalEPass is only available between May 23rd, and June 24th, which may provide a special rate as low as \$125.00 each night. Regular price is \$250 each night. Accept all major credit cards. The operation requires that the message must use a reliable messaging protocol and encrypt a header with WS-Security using a X. 509 token.

Enter-Car: SUVrental provides SUV rental for \$100 daily.

BudgetHotel: PromotionalEPass is only available between May 23rd, and June 24th, which may provide a special rate as low as \$175.00 each night. Regular price is \$200 each night. Accept all major credit cards.

To help understand the request elicitation and decision making process with this model, we will walk through how the process proceeds and how tradeoffs in different aspects of possible service compositions are assessed. We ask the user

Table 2 Royal Hotel's lottery

	SUVrental	AirTicket Booking	PromotionalEPass	FlightDepartureDateTime		Cost		(Air Ticket Booking, PromotionalEPass)
				Morning 22 May	Evening 23 May	(650,900]	[0,650]	
Score	1	1	1	1	0.2	0.5	1	1
Weight	10%	18%	18%	18%		18%		18%
Product (S, W)	0.1	0.18	0.18	0.18	0.036	0.09	0.18	0.18
Probability				0.5	0.5	0.5	0.5	

Table 3 Budget Hotel's lottery

	SUVrental	AirTicket Booking	PromotionalEPass	FlightDepartureDateTime		Cost		(Air Ticket Booking, PromotionalEPass)
				Morning 22 May	Evening 23 May	(750,800]	[0,750]	
Score	1	1	1	1	0.2	0.6	0.8	1
Weight	10%	18%	18%	18%		18%		18%
Product (S, W)	0.1	0.18	0.18	0.18	0.036	0.108	0.146	0.18
Probability				0.3	0.7	0.3	0.7	

to select service categories and pick service operations. We then ask the user to specify all constraints, which are what we have in the first part of the scenario description at the beginning of this section. The above information will be used to establish service variables and populate some of the variables. We will then build the partitions of the variables. In the case of the additive independent request structure, all partitions are singly partitions. We will then get the utility of certain values. For example, $u_1(a_1 = \$650) = 1$ and $u_1(a_1 = \$901) = 0$, if a_1 represents Cost. The user also has to explicitly give the weights of all the variables.

In order to build the probabilistic distribution functions, we can collect statistic data of certain values. For example, $\Pr_{\text{RoyalHotel}}(a_1 < \$650) = 0.5$, $\Pr_{\text{RoyalHotel}}(a_1 \in [\$650, \$900]) = 0.5$. In the rest of this section, we will assume that the statistics of taking on different data values can be collected and the probabilistic distributions can be induced from the data.

The above process is exactly the same for both additive independent request and GAI requests.

The decision making process can be put in tables. We can list the preference scores of all the attributes, weights and probability distribution of different outcomes in tables against all various lotteries. In our case, Table 2 is for Royal Hotel's lottery and Table 3 is for Budget Hotel's lottery. In both tables, we use different shades to show different sub-outcomes. Probabilities of outcomes are also given in cells with corresponding shades. We calculate the expected utilities of two options and select the one has a large expected utility.

Utility estimation of Royal Hotel

$$= 0.5 * (0.1 + 0.18 + 0.18 + 0.18 + 0.09 + 0.18) \\ + 0.5 * (0.1 + 0.18 + 0.18 + 0.036 + 0.18 + 0.18) = 0.883$$

Utility estimation of Budget Hotel

$$= 0.3 * (0.1 + 0.18 + 0.18 + 0.18 + 0.108 + 0.18) \\ + 0.7 * (0.1 + 0.18 + 0.18 + 0.036 + 0.146 + 0.18) = 0.8538$$

Royal Hotel seems to be the better choice. Below is an intuitive explanation. If we sketch out possible solutions, the alternatives are between Royal Hotel and Budget Hotel with tradeoffs between different travel time and the availability of promotional hotel booking. For both hotels, the only two possibilities are to fly on 22nd by a morning flight without the discount hotel rate or to fly on 23rd evening and enjoy the discount hotel rate. Budget Hotel has a lower regular rate and a higher discount rate compared to Royal Hotel. Understandingly, its utility on regular rate is higher and the utility on discount rate is lower compared to those of Royal Hotel. We need also take into consideration the probability distributions of both hotels, which is 30–70% for Budget Hotel and 50–50% for Royal Hotel. Because the latter shows dominance in higher utility, it turns out to be a better choice.

5.2 Generalized additive independence

To make the previous scenario a bit more complicated, we add in two constraints i.e. “If the flight is a morning flight (arrival during working hours), a rental car is more preferred”, and “If the flight is a morning flight, the user is more willing to pay a higher price.”

- FlightDepartureTime <12 PM → Rental car service is more preferred.
- FlightDepartureTime <12 PM → Preference on a relatively higher cost increases.

Using Algorithm 1, the variables are defined and initialized as follows:

$A_C = \{c\}$, $c = \{\text{travel arrangement}\}$;
 $A_O = \{o\}$, $o = \{\text{Air ticket booking, RoomEPassIssue, SUV Rental}\}$;
 $A_{AT} = \{a_1, a_2, a_3\}$, $a_1 = \text{FlightDepartureTime}$, $a_2 = \text{Cost}$, $a_3 = \text{FlightDepartureDate}$
 $A_{OD} = \{r_1, r_2, r_3\}$, $r_1 = \gamma(\text{Air ticket booking, RoomEPassIssue}) = 1$, $r_2 = \gamma(\text{Air ticket booking, SUV Rental}) = 0$, $r_3 = \gamma(\text{RoomEPassIssue, SUV Rental}) = 0$

Using Algorithm 1, the variables can be partitioned as follows:

$$\begin{aligned}
 V_{I_1} &= \{c\}; \\
 V_{I_2} &= \{a_1, o\}; \\
 V_{I_3} &= \{a_1, a_2\}; \\
 V_{I_4} &= \{a_3\} \\
 V_{I_5} &= \{r_1\}
 \end{aligned}$$

Given the above partition of the variables, we have the following utility function decomposition.

$$u(c, o, a_1, a_2, a_3, r_1) = u_1(c) + u_2(a_1, o) + u_3(a_1, a_2) + u_4(a_3) + u_5(r_1)$$

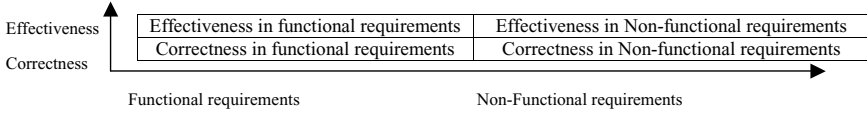


Fig. 2 Simple two-dimensional evaluation matrix against various types of composition tasks

Following formulas in [17], we then define sub-utility functions by local value functions and utilities of the anchor outcomes as:

$$\begin{aligned}
 u_1(c) &= (u_1^T - u_1^\perp)v_1(c) + u_1^\perp, \\
 u_2(a_1, o) &= (u_2^T - u_2^\perp)[v_2(a_1, o) - v_2(a_1^0, o^0)], \\
 u_3(a_1, a_2) &= (u_3^T - u_3^\perp)[v_3(a_1, a_2) - v_3(a_1^0, a_2^0)] + [u_3^T - 2u_3^\perp + 2u_3^T] \\
 &= (u_3^T - u_3^\perp)[v_3(a_1, a_2) - v_3(a_1^0, a_2^0)] + u_3^T, \\
 u_4(a_3) &= (u_4^T - u_4^\perp)[v_4(a_3) - v_4(a_3^0)] + [u_4^T - 3u_4^\perp + 3u_4^T - 3u_4^\perp] \\
 &= (u_4^T - u_4^\perp)[v_4(a_3) - v_4(a_3^0)] - 2u_4^\perp, \\
 u_5(r_1) &= (u_5^T - u_5^\perp)[v_5(r_1) - v_5(r_1^0)] + [u_5^T - 4u_5^\perp + 4u_5^T - 4u_5^\perp + 4u_5^T] \\
 &= (u_5^T - u_5^\perp)[v_5(r_1) - v_5(r_1^0)] + u_5^T
 \end{aligned}$$

5.3 Proposed effectiveness measurement

We have walked through the travel scenario to prove the applicability of the model by showing how the model works for both additive and generalized additive request structures. The other important issue is to evaluate the effectiveness of the iterative and incremental scheme of the model and show how this scheme helps the process of composition. Intuitively, for complex services, most users are not able to put together all their requirements at once nor are they able to guarantee the requirements they have are not in conflict. Such an iterative and incremental scheme will be needed under most circumstances to fix the tentative problems of the users communicating service needs to the system.

In order to take a quantitative measurement, one alternative is to identify a group of human subjects as service users and to design some experiments on composing services for their service requests. We propose a simple two-dimensional evaluation matrix against various types of composition tasks submitted by the group of subjects, as shown in Fig. 2. The upper bound of the interactions allowed for between the system and the users is also assumed.

On one dimension of the evaluation matrix are functional requirements versus non-functional requirements. On the other dimension are effectiveness and correctness. First, how effective the scheme is in terms of helping satisfy the functional requirements of the users is evaluated. The frequency that the functional requirements of service requests are met by one-time composition effort can be measured. One minus this frequency gives us the *effectiveness* of the scheme on this dimension. The frequency of unsatisfied composition results despite of the

allowed number of interactions is observed, one minus which gives us the *correctness* of the scheme on this dimension. Second, how helpful the scheme is in terms of framing out the reasonable non-functional requirements of the users and satisfying the requirements is evaluated. For the same functional requirements, the frequency that the non-functional requirements of service requests are met by one-time composition effort can be measured. One minus this frequency gives us the *effectiveness* of the scheme on the non-functional dimension. The frequency of various non-functional requirements of the same functional requirements that are failed by the system with the model despite of the maximum number of allowed interactions is also recorded. An average of such frequencies over all functional requirements is taken. One minus the average gives the *correctness* of the scheme on the non-functional dimension.

6 Related work

The concept of Semantic Web services was first proposed by McIlraith et al. [25]. These researchers initiated the discussion on the semantic Web technology and on how it makes information on the Web better understood by computers. McIlraith et al. applied Semantic Web to Web services and developed markups of Web services that benefit Web service discovery, execution and composition [25]. They also pointed out that user constraints and preferences are the main thing that makes service discovery, execution and composition difficult. In that paper and in related papers [24], they described 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 facilitate construction of 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 interactions in resolving user constraints and preferences.

Medjahed et al. reported in their paper “Composing Web services on the Semantic Web” an ontology-based framework for automatic Web service composition [26]. They use a high-level declarative language to describe services and “composability rules” to check the composability of services from both the syntactic and semantic perspectives. The proposed ontology-based descriptions of Web services are modeled as a graph composed of nodes and edges representing the WSDL and extended service description concepts and relationships of these concepts respectively. Based on the ontology, they define the concepts of mode, message, operation and Web services, on top of which, they demonstrate syntac composability (mode and binding composability) and semantics composability (message composability, operation semantics composability, qualitative composability and composition soundness). The composition algorithm is composed of four phases including the specification phase using a CSSL language, matchmak-

ing phase using the composability rules defined, selection phase based on quality of composition and the generation phase to generate a detailed composite service description. Their work is similar to ours in the high-level declarative descriptions of services and their concept composability is very similar to service constraints. Our MAUT model serves as a mapping between the description of the service need and the complete process of service composition in a streamlined manner, which irons out a better-formalized systematic approach to service composition.

User preference elicitation has been extensively studied in the literatures of interactive decision systems [4, 11, 12, 20, 29]. Literatures have reported a number of interactive and incremental 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 [6, 10–12]. Some of the work has been targeted towards applications in certain vertical domains such as the airline and travel business. On the other hand, there are a few efforts in making use of decision theories in selecting service providers. For example, Benatallah et al. presented a model-driven service composition system and peer to peer service orchestration with multi-attribute provider selection policies [2]. They suggested a set of predefined attributes, such as execution price, execution duration, reputation and etc., and their corresponding score functions. Liang et al. models service attribute constraints by MAUT and solves the composition problem by decision-support system techniques [19]. As another example, Fakas et al. developed a multi-context information based intelligent navigation system to incorporate user profile including their preferences and interests [9]. In this paper, we consider multi-dimensional preferences in the semantic Web service request context, and study the applicability of MAUT as the enabling model for both functional and non-functional requirement satisfaction on complex services. In our model, MAUT serves a more general role in selecting and composing the desired services, compared to quality comparisons in Benatallah et al. [2].

Research to date on Web service requests for service composition mostly comes from the AI community. In [17, 28], the authors pointed out that BPEL [5] 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 user preferences for service composition, and to present them in the semantic Web framework. We perceive that the semantics of the request have to be analyzed against 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 composite services. 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. In this model, non-deterministic aspects in service

composition are captured in expected utilities. Two preference structures, i.e. additive independence and generalized additive independence, are discussed in detail for the model. We enhance OWL-S with language construct extensions that make possible a clear and uniform way of representing the semantics of both functional and non-functional aspects of service requests. We also work out an example to demonstrate the process of modeling the service requests and of comparing various solutions according to the preferences.

The significance of the contribution is that it strengthens the service request processing capability of the Web services model as it eases the interfacing of the service demand descriptions with service discovery and composition. Benefits of Web services framework favor a roll out of the request processing capability that adds value to the automatic service discovery, invocation and integration. Better request processing relies on a model where semantics of service requests can be captured by machines. This model must handle a range of requests. Requests can range from very brief ones that have only a single operation to very complicated ones that provide sophisticated structure of operations with constraints on correlated multiple service variables. Our work provides a design of such request models that facilitate the request processing and promote better service discovery and integration.

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 involved when discovering and composing Web services. In some cases, probability distributions cannot be derived. Extensions towards an appropriate uncertainty model can be made to complement the request processing. (2) Some criteria regarding services may be common to all customers. Extension work on identifying both the aspects that can be treated as common and their criteria is valuable. (3) As part of the future work, we will look into deploying the implementation of the model/system into the public domain in order to carry out evaluation based on real usage and user experience.

References

1. Bacchus F, Grove A (1995) Graphical models for preference and utility. In: Proceedings of the UAI-95, Montreal, pp 3–10
2. Benattallah B, Dumas M, Sheng QZ (2005) Facilitating the rapid development and scalable orchestration of composite web services. *Distrib Parallel Dat* 17(1):5–37
3. Blythe J (2002) Visual exploration and incremental utility elicitation. In: Proceedings of the 18th national conference on artificial intelligence
4. Boutillier C, Bacchus F, Brafman RI (2001) UCP-networks: a directed graphical representation of conditional utilities. In: Proceedings of the 17th conference on uncertainty in artificial intelligence (UAI), University of Washington, Seattle, Washington, USA
5. BPEL (2005) <http://www-106.ibm.com/developerworks/library/ws-bpel/>
6. Brazianus D, Boutillier C (2005) Local utility elicitation in GAI models. In: Proceedings of the IJCAI-05 multidisciplinary workshop on advances in preference handling, Edinburgh, Scotland
7. Cornelissen F, Jonker CM, Treur J (2003) Compositional verification of knowledge-based task models and problem-solving methods. *Knowl Inf Syst J* 5(3):337–367
8. DAML-S (2001) <http://www.daml.org/services/daml-s/0.9/>
9. Fakas G, Kakas AC, Schizas C (2004) Electronic roads: intelligent navigation through multi-contextual information. *Knowl Inf Syst J* 6(1):103–124

10. Fishburn PC (1967) Interdependence and additivity in multivariate, unidimensional expected utility theory. *Int Econ Rev* 8:335–342
11. Gonzales C, Perny P (2004a) GAI networks for utility elicitation. In: Proceedings of the 9th international conference (KR2004) on principles of knowledge representation and reasoning, pp 224–234
12. Gonzales C, Perny P (2004b) Graphical models for utility elicitation. In: Proceedings of the DIMACS/LAMSADE workshop on computer science and decision theory, France
13. Ha V, Haddawy P (1997) Problem-focused incremental elicitation of multi-attribute utility models. In: Proceedings of the 13th conference on uncertainty in artificial intelligence, pp 215–222
14. Jurisica I, Mylopoulos J, Yu SKE (2004) Ontologies for knowledge management: an information systems perspective. *Knowl Inf Syst J* 6(4):380–401
15. Keeney RL, Raiffa H (1976) Decisions with multiple objectives: preferences and value tradeoffs. Wiley and Sons, New York
16. Kreps DM (1988) Notes on the theory of choice. Under-ground classics in economics. Westview Press, Boulder
17. Lazovik A, Aiello M, Papazoglou M (2003) Planning and monitoring the execution of web service requests. Technical report #DIT-03-049, University of Trento
18. Liang Q, Chakarapani LN, Su SYW, et al. (2004) A semi-automatic approach to composite web service discovery, description and invocation. *Int J Web Serv Res* 1(4):64–89
19. Liang Q, Chung J, Miller S (2005) Towards semantic service request of web service composition. In: Proceedings of the IEEE conference on e-business engineering (ICEBE 2005)
20. Linden G, Hanks S, Lesh N (1997) Interactive assessment of user preference models: the automated travel assistant. In: Proceedings of the user modeling '97
21. Long DE (1993) Model checking, abstraction and compositional verification. PhD thesis, Carnegie Mellon University
22. Martin D, Paolucci M, McIlraith S, et al. (2004) Bringing semantics to web services: the OWL-S Approach. In: Proceedings of the first international workshop on semantic web services and web process composition (SWSWPC 2004), San Diego
23. McIlraith S, Martin D (2003) Bringing semantics to web services. *IEEE Intell Syst Arch* 18(1):90–93
24. McIlraith S, Son TC (2002) Adapting golog for composition of semantic web services. In: Proceedings of 8th conference on knowledge representation and reasoning (KR2002)
25. McIlraith S, Son TC, Zeng H (2001) Semantic web services. *IEEE Intell Syst, Spl Issue Semantic Web* 16(2):46–53
26. Medjahed B, Bouguettaya A, Elmagarmid AK (2003) Composing web services on the semantic web. *VLDB J* 12(4):333–351
27. OWL-S: Semantic Markup for Web Services (2006) <http://www.ai.sri.com/daml/services/owl-s/1.2/>
28. Papazoglou M, Aiello M, Pistore M, et al. (2002) Planning for requests against web services. *IEEE Data Eng Bull* 25(4):41–46
29. Pu P (2003) User-involved preference elicitation. In: Proceedings of the 18th international joint conference on artificial intelligence, Workshop on Configuration, Mexico
30. Pu P, Faltings B (2000) Enriching buyers' experiences: the smartclient approach. In: Proceedings of the ACM CHI conference on human factors in computing systems, pp 289–296
31. Schafer R (2001) http://www.kbs.uni-hannover.de/henze/ABIS_Workshop2001/final/Schaefer_final.pdf
32. Strikeiron, <http://www.strikeiron.com/StrikeIronServices.aspx>
33. Su SYW, Liang Q, Chakarapani LN, et al. (2003) A web service composition framework: discovery, description and invocation. In: Proceedings of the ICECR-6, Texas
34. SWSL (2005) <http://www.daml.org/services/swsl/>
35. UNSCSP, <http://www.unspc.org/>
36. von Winterfeld D, Edwards W (1986) Decision analysis and behavioral research. Cambridge University Press, Cambridge, UK
37. Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001, <http://www.w3.org/TR/wsdl>

Author Biographies



Qianhui Althea Liang received her Ph.D from the Department of Electrical and Computer Engineering, University of Florida in 2004. While pursuing her Ph.D, she was a member of Database Systems Research and Development Center at the University of Florida. She received both her bachelor's and master's from the Department of Computer Science and Engineering, Zhejiang University, China. She joined the School of Information Systems at Singapore Management University, Singapore, as an assistant professor in 2005. Her major research interests are service composition, dynamic service discovery, multimedia Web services, and applied artificial intelligence.



Jen-Yao Chung received the M.S. and Ph.D degrees in computer science from the University of Illinois at Urbana-Champaign. Currently, he is the senior manager for Engineering and Technology Services Innovation, where he was responsible for identifying and creating emergent solutions. He was Chief Technology Officer for IBM Global Electronics Industry. Before that, he was program director for IBM Institute for Advanced Commerce Technology office. He is the co-founder of IEEE technical committee on e-Commerce (TCEC). He has served as general chair and program chair for many international conferences, most recently he served as the steering committee chair for the IEEE International Conference on e-Commerce Technology (CEC06) and general chair for the IEEE International Conference on e-Business Engineering (ICEBE06). He has authored or coauthored over 150 technical papers in published journals or conference proceedings. He is a senior

member of the IEEE and a member of ACM.



Steven Miller is founding Dean of the School of Information Systems (SIS) at Singapore Management University, and also serves as Practice Professor of Information Systems. Since 2003, he has led efforts to launch and establish the undergraduate, graduate and professional programs of the SIS. Immediately prior to joining SMU, Dr. Miller served as Chief Architect Executive for the Business Consulting Services unit of IBM Global Services in Asia Pacific. He held prior industry appointments with Fujitsu Network Systems, and with RWD Technologies. Dr. Miller started his professional career as an Assistant Professor at Carnegie Mellon University, conducting research and teaching related to Computer-Integrated Manufacturing and Robotics applications and impacts. He has a Bachelors of Engineering Degree in Systems Engineering (Magna Cum Laude) from the University of Pennsylvania and a Masters of Science in Statistics and a Ph.D in Engineering and Public Policy from Carnegie Mellon University.