Dynamic Selection of Web Services with Recommendation System

Umardand Shripad Manikrao
Indian Institute of Technology, Kanpur
shripad@cse.iitk.ac.in

T.V.Prabhakar
Indian Institute of Technology, Kanpur
tvp@cse.iitk.ac.in

Abstract

The realization of the Semantic Web is underway with the development of an arena of services providing similar properties, capabilities, interfaces, and effects. To pick one of such similar services that matches the user's requirements is a difficult task and necessitates the use of an intelligent decision making framework. This paper addresses precisely this component. We present the design of a Dynamic Web service selection framework that makes use of a semantic matcher to support matching and composition of software services. The framework also uses a recommendation system which helps a user to select the best service that matches his requirements. This recommendation system results in evolution of the framework to dynamically adapt to users requirements.

1. Introduction

Classical web services suffer from the limitations of the Web Service Description Language (WSDL). WSDL is not strong enough to address the semantics of the interface. A semantic web service [1] however supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in an unambiguous, computer-interpretable form. To add to the advantages of the semantic web services, the proposed dynamic web service selection framework not only addresses the limitations of a classical web service by extending WSDL and UDDI capability with semantic descriptions of the service [2] but also provides an additional recommendation system to select a service from a set of similar services.

1.1. Dynamic Web Service Selection Approach

The proposed scheme works as follows:

a) A web service user provides his requirements using a semantic document. The requirements may vary from description of service and quality of service (QoS) parameters like max execution time, average execution time, max response time, average response time etc.

b) The service providers of web services would need to register their services using service descriptions [3]. These service descriptions will contain the semantic service profile and QoS parameters like max execution time, average execution time, max response time, average response time etc. The service provider would also be required to specify the location of a WSDL document describing a web service.

c) The semantic matcher will match the user request with registered service descriptions and provide a list of available services matching with requirements. This list will be given to the recommendation system. The recommendation system based on its learning through users' feedback, orders the list and presents to the user. Each component of the list, finally provided to the user, may be a single service or a composition of registered services.

The user can select a service from this list. After the execution is over, the user may provide a rating to this service using given metric. This rating indicates user's satisfaction level. It is stored in a repository and used as an input to the recommendation.

2. Related Work

This section describes work that is going on related to web service selection and compares it with our approach.

Digital Business Ecosystem is a concept for open source distributed environment that can support spontaneous evolution and composition of software services, components and applications. It learns user's needs over time and optimizes itself. They have used an MDA (Model Driven Architecture) approach. The user requests a service using business models using BML[4][5] (Business Modeling Language). This request has the functional requirement and the non functional requirements and agreements, contracts. This request is compiled into a service specification that includes interface, behavior and semantic of the service. The user request in form of a service description is given to a matching block, which gives matching services to an automatic composer for composition. It has some memory

where it stores different models and service descriptions. It does mutation, replication of these models. This along with selection provides evolution. Here the evolution means, when there is a chain of web services; it can dynamically replace one set of web services with another set of web services. This replacement is done considering the quality of services.

Our Approach to dynamic web service selection uses a DAML document to specify the functional and the non functional requirements, agreements, contracts. It has some memory where it stores these DAML descriptions of web services. To describe an interface of a service (technical description of service i.e input and output parameters to service, location of service), WSDL document is used. It matches the DAML description of user requirements with the DAML description of a web service to find a matching service. Our framework also supports evolution of a web service chain. The user feedback of a web service plays role in a web service composition. So the service with better quality will be considered in a composition and hence resulting in an evolution.

Evren Sirin and his team have developed a prototype system [6] that can compose the web services deployed on the internet and provide filtering capabilities where large number of similar services might be available. They use a semantic matching algorithm for filtering. Composition process is user driven. At each step in a composition, it asks the user to select one of web services that can be used at this step.

Our approach to web service selection is fully automated. At each step in a composition, we have a set of services that we can use for the composition; we select a service or a service composition with most popularity (highest rating). The popularity of a service composition will be same as of least popular web service in a composition. Our approach is better because it is fully automated and works based on user experience of a web service.

The instance composition approach is to generate a composite service plan out of existing services. A composition path is proposed by Z.M.Mao [7]. This path is a sequence of operators that computes data and connectors that provide data transport between operators. The search for possible operators to construct a sequence is based on the shortest path algorithm on the graph of operator space. However, they have considered only two kinds of services - operator and connector with one input and one output parameter. SWORD uses a rule based expert system to determine if a plan of composite service can be built using existing services.

Ruoyan Zhang [8] has proposed a web service composition approach which is based on an interface matching. It uses modified shortest path algorithm to find a composition with less execution time. Our approach selects a composition with less execution time as it's rating will be higher than other compositions.

3. Architecture

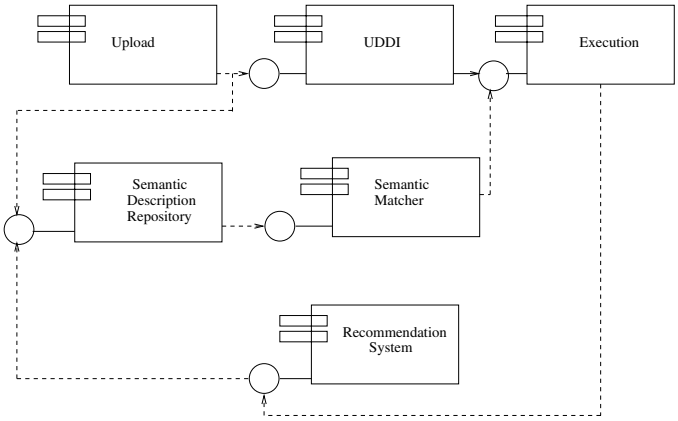

Figure 1. Architecture of the Dynamic Web Service Selection Framework

Figure 1 shows different components involved in a Dynamic web service selection Framework. The upload component uploads semantic description and WSDL parameters of a web service. The information from WSDL document is extracted and stored in UDDI repository. The semantic matcher matches semantic descriptions of services with user requirements and proposes a list of services matching with his requirements. The user can execute any of matching services using execution environment. The recommendation component asks the user to rate the executed service, so it will be used for recommendation purpose.

4. Semantic Matcher

Service providers publish DAML-S [9] descriptions of services to a Semantic Description Repository. A service user gives his requirements using DAML-S description. The semantic matcher finds the match between user requirement and all published service descriptions using a Semantic Matching Algorithm. It along with Recommendation System gives matching services in an order.

Figure 2 shows the detailed architecture of a Semantic Matcher [10][11][6]. The Ontology Inference Engine creates a knowledge base from the specified ontology in a DAML-S description and a request description. Web Service Description parser parses the Web Service Descriptions to find out different parameters to be matched. The criteria table specifies service attributes to be compared and the

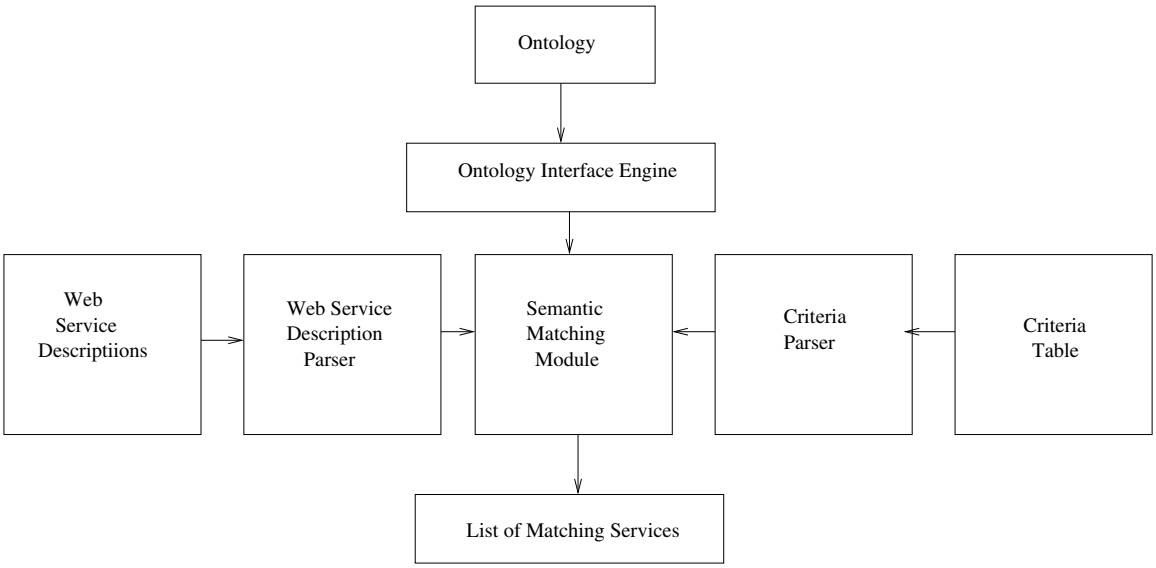

Figure 2. Architecture of Semantic Matcher

least preferred similarity measures for those attributes. The similarity measure can be exact, plug-in, subsumption, container, disjoint, part of.

If the two conceptual annotations are syntactically identical, the mapping is called an Exact map. If the second conceptual annotation specializes the first, the mapping is called Plug-in. If the first conceptual annotation specializes the second, the mapping is called Subsumption map. If the first conceptual annotation contains the second, the mapping is called a Container map and if first conceptual annotation is part of the second, the mapping is called Part of map. Otherwise the mapping is called disjoint map.

4.1. Matching Algorithm

This subsection describes the matching algorithm [10] using following definitions.

Service: A service S is defined as a set of attributes that defines the service. Let S.A defines a set of attributes of the service S and S.A_i defines each member of this set. Let S.N denotes the number of attributes of service S.

Similarity Measure: The Similarity Measure (μ) of two service attributes is the mapping that measures semantic distance between the conceptual annotations associated with the service attributes.

μ: A $\times$ A $\longrightarrow$ {Exact, Plugin, Subsumption, Container, part of, disjoint}

Where A is set of all attributes.

Similarity Measure Preference: Preference amongst similarity measures is strictly governed by the following strict order.

Exact $\succ$ Plug in $\succ$ Subsumption $\succ$ Container and

Container $\succ$ part of $\succ$ Disjoint

Here a $\succ$ b means a is preferred over b.

Sufficient Service Match: Let S^R be a service that is requested and S^A be a service that is advertised. Let S^R.A be the set of attributes to be utilized for matching. It may include both a service capability (subset of service attributes directly related to its working) as well as quality attributes(the service attributes other than service capability). Let μ_i be the desired similarity measure for each service attribute $S^R.A_i$. A sufficient match exists between S^R and S^A if the values of attributes satisfy desired similarity measure.

Formally,

$\forall_i \exists_j$ $(S^R.A_i = S^A.A_j) \wedge \mu(S^R.A_i, S^A.A_j) \succeq \mu_i \Longrightarrow$ SuffMatch(S^R,S^A) 1$\leq$i$\leq S^R$.N ....(1)

Let the Criteria table be denoted by C. C is a relation consisting of two attributes C.A (service attributes to be compared) and C.μ (least preferred similarity measure for that attribute). Let C.A_i and C.μ_i denotes the attribute value and desired similarity measure in the i^{th} tuple of the relation and C.N denote total number of tuples in C.

A sufficient match exist between S^R and S^A if for every attribute in C.A, there exist an identical attribute of S^R and S^A and values of attribute satisfy the desired similarity measure as specified in C.μ.

$\forall_i \exists_{j,k}$ (C.A_i =$S^R.A_j$ =$S^A.A_k$) $\wedge$ $\mu(S^R.A_j, S^A.A_k) \succeq$ C.$\mu_i \Longrightarrow$ SuffMatch(S^R,S^A) 1$\leq$i$\leq$C.N ..... (2)

The Matching algorithm is implementation of Equation (2).

5. Recommendation System

The Dynamic Web Service Selection Framework has a recommendation system, which recommends the best service satisfying the user's requirements. When a user uses a web service, it asks user to rate a web service; so that users can help each other to find a better web service. This is especially important when there are more than one web services which have same functionality but their quality of service is different. We provide the user, a metric to help him decide the rating of a web service. It will be a comparison matrix of runtime behavior of a web service and the user's expected QoS parameters like max execution time, average execution time, max response time, average response time etc. Web service with better quality of service will get more rating than other service which offers same functionality but poor service quality.

The recommendation system uses the item based collaborative filtering approach [12]. As users rate web services, it is possible to predict how a given user will rate a particular web service. Once it knows prediction of ratings to each web service satisfying user requirements, it can recommend web services in order of their ratings. This approach looks at the set of web services the target user has rated and computes how similar they are to the web service for which user rating is to be predicted. Once the similar web services are found, the prediction is computed by taking a weighted average of the target user's ratings on these similar web services.

The item based collaborative filtering approach has two aspects namely similarity computation and prediction generation.

5.1. Similarity Computation

The similarity [12][13] between two web services is computed by subtracting the average rating of the two web services. Considering only users who have rated both web service A and web service B, say that there are 10 such users, we sum the ratings that A and B got, say 65 and 85. Clearly B is ranked higher than A by 2 on average. The similarity between web services is computed whenever users rate a web service. The result of similarity computation is stored in a similarity matrix.

5.2. Prediction Generation

The prediction function [12][13] predicts how a particular user will rate a web service. It computes prediction on a web service i for a user u by computing the sum of ratings given by the user on the web services similar to i. Each rating is weighted by the corresponding similarity $S_{i,j}$ between web services i and j.

$$P_{u,i} = \frac{\sum_{\text{all similar items, j}}(s_{i,j} * R_{u,j})}{\sum_{\text{all similar items, j}}(|s_{i,j}|)}$$

Basically it tries to capture how the active user rates the similar web services. The weighted sum is scaled by the sum of the similarity terms to make sure the prediction is within the predefined range.

If the user has used a similar service, it predicts his likely satisfaction index for this service/service chain. If no similar service has been used before, it considers the average rating of all the users for similar services.

5.3. Evolution

The web service recommended by the system can be a single service or composition of web services. As user ratings of web services changes, the recommendation system evolves.

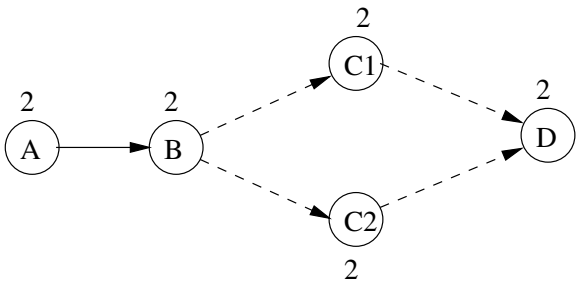

Figure 3. Initial Configuration of Recommendation System

As shown in Figure 3, there is a single user and five web services A, B, C1, C2, D in the recommendation system. Each of theses services has rating 2. The user request matches with service chains A, B, C1, D and A, B, C2, D.

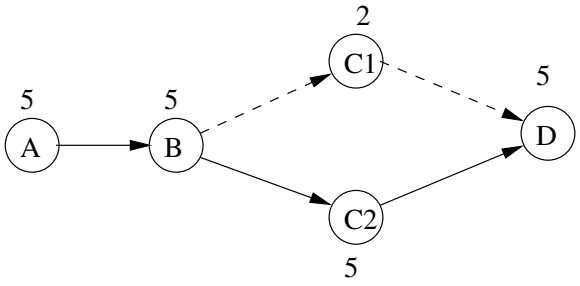

Figure 4. Evolution of Recommendation System

The user selects a service chain A, B, C2, D for execution and rates this service composition as 5. The rating given

to a service chain is assigned to individual components in the service chains. When a service chain is considered for recommendation, the rating of a service chain is same as lowest rating of components in a chain. So rating of service chain A, B, C1, D is 2 and the rating of service chain A, B, C2, D is 5. As shown in Figure 4, when the user gives the same request, the recommendation system recommends service chain A, B, C2, D as first recommendation and service chain A, B, C1, D as it's second recommendation.

6. Conclusion and Future work

This paper proposes a dynamic web service selection framework which combines a recommendation system with semantic matching of service requirements. The matching service can be a single service or composition of registered services. The recommendation system is based on user feedback and collaborative filtering techniques. It helps the user in selecting a web service from a set of similar services.

Our system does not provide facility for generation of DAML descriptions of a web services. It would be helpful to have this facility. The Model Driven Architecture (MDA) approach is useful to ease code generation for web service providers and to specify user requirements. The requirements can be specified using technology independent models of business concepts and then those can be mapped to technology dependent service specifications. Similarly web service providers will specify business models of their services and then those can be mapped to platform specific code.

References

[1] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng. "Semantic Web Services", *IEEE Intelligent Systems. Special Issue on the Semantic Web. 16(2):46–53*, March/April, 2001.

[2] Rama Akkiraju, Richard Goodwin, Prashant Doshi, Sascha Roeder. "A Method for Semantically Enhancing the Service Discovery Capabilities of UDDI". *In the Proceedings of IJCAI Information Integration on the Web Workshop, Acapulco, Mexico*, August 2003.

[3] Anupriya Ankolekar, Mark Burstein, Jerry Hobbs J., et al. DAML-S. "Web Service Description for the Semantic Web", *In the Proceedings of First International Semantic Web Conference. (ISWC02)*, 2002

[4] Paolo Dini, Neil Rathbone, Tuija Helokunnas, Angelo Corrallo, Pierfranco Ferronato. "Towards semantically rich business language for the automatic composition of web service", *eBRF 2003 Conference*, September, 2003.

[5] Thomas Heistracher, Thomas kurz, Angelo Corallo, Paolo Dini, "Pervasive Architecture for a Digital Business Ecosystem", *First International Workshop on Coordination and Adaption Techniques for Software Entities(WCAT04)*, June, 2004.

[6] Evren Sirin, Bijan Parsia, and James Hendler. "Filtering and Selecting Semantic Web Services with Interactive Composition Techniques", *IEEE Intelligent Systems*, 19(4):42-49, 2004.

[7] Z.M.Mao, E. R Brewer, and R.H.Kartz. "Fault tolerant, Scalable, wide area Internet service composition", *U.C. Berkeley Technical Report UCB//CSD-01-1129*, January, 2001.

[8] Ruoyan Zhang, I Budak Arpinar, and Boanerges Aleman-Meza. "Automatic composition of semantic Web Services", *ICWS 2003: 38-41*, June, 2003.

[9] DAML Technical Committee. DARPA Agent Markup Language- DAML. http://www.daml.org

[10] Prashant Doshi, Richard Goodwin, Rama Akkiraju. "Parameterized Semantic Matching for Workflow Composition", *IBM Research Report,RC23133(W0403-026)*, March, 2004.

[11] M. Paolucci et al. "Semantic Matching of Web Services Capabilities", The Semantic Web-ISWC 2003: *1st Int'l Semantic Web Conf., LNCS 2342, Springer-Verlag*, 2002, p.333.

[12] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. "Item-based Collaborative Filtering Recommendation Algorithms". *In the Proceedings of the 10 International World Wide Web Conference. Hong Kong*, 2001.

[13] Daniel Lemire, Sean McGrath. "Implementing a Rating-Based Item-to-Item Recommender System in PHP/SQL", *Technical Report D-01*, January, 2005.