Semi-automatic Composition of Web Services Using Semantic Descriptions

Prof. Haresh R. Parmar
Assistant Professor in Computer Engineering Department
Silver Oak College of Engineering & Technology, Ahmedabad.

Abstract.
The emergent Semantic Web community needs a common infrastructure for testing the scalability and quality of new techniques and software which use machine process able data. Since ontologies are a centerpiece of most approaches, we believe that for an accurate evaluation of tools for quality, scalability and performance, the research community needs a freely available ontology with a large description base. If the use of tools is to be for advanced semantic applications, such as those in business intelligence and national security, then instances in the knowledge base should be highly interrelated connected. Thus, we propose and describe a Semantic Web Technology Evaluation Ontology (SWETO) test-bed. In particular, we address the requirements of a test-bed to support research in semantic analytics, as well as the steps in its development, including, ontology creation, semi-automatic data extraction, and entity disambiguation.

Introduction
Web services are designed to provide interoperability between diverse applications. The platform and language independent interfaces of the web services allow the easy integration of heterogeneous systems. Web languages such as Universal Description, Discovery, and Integration (UDDI) [13], Web Services Description Language (WSDL) [4] and Simple Object Access Protocol (SOAP) Standards for service discovery, description and messaging protocols. However, these web service standards do not deal with the dynamic composition of existing services. The new industry initiatives to address this issue such as Business Process Execution Language for Web Services (BPEL4WS) focus on representing compositions where own of the process and the bindings between Services are known a priori. A more challenging problem is to compose services dynamically, on demand. In particular, when a functionality that cannot be realized by the existing services is required, the existing services can combined together to full the request. The dynamic composition of services requires the location of services based on their capabilities and the recognition of those services that can be matched together to create a composition as described in. The full automation of this process is still the object of ongoing research activity, but accomplishing this goal with a human controller as the decision mechanism can be achieved. The main problem for this goal is the gap between the concepts people use and the data computers interpret. We can overcome this barrier using Semantic Web Technologies. The Semantic Web is an extension of the current web in which information on is given well-dennned meaning, better
enabling computers and people to work in cooperation. This is realized by marking up Web content, its properties, ditz relations, in a reasonably expressive markup language with a well-defined semantics. The Web Ontology Language (OWL) [7] is a forthcoming W3C specification for such a language which will supersede the earlier DARPA Agent Markup Language (DAML+OIL). OWL is an extension to XML and the Resource Description Framework (RDF) enabling the creation of ontologies for any domain and the instantiation of these ontologies in the description of resources. The DAML+ services language (DAML-S) is a set of language features arranged in these ontologies to establish a framework within which the web services may be described in this semantic web context. Our work uses OWL and DAML-S to provide the semantics needed for service altering and composition. The remainder of this paper is organized as follows: In we explain the examples of web service composition problems we address and then in section 3 we describe how semantic service descriptions are used in these examples. The details of our prototype and the algorithms used in service composition.

**Motivating Examples**

Our work focuses on the composition of web services that have been previously annotated with semantics and discovered by a system. As an example of composition, suppose there are two web services, an on-line language translator and a dictionary service, where the one translates text between several language pairs and the second one return the meanings of English words. If a user need as French Dictionary service, neither of these can satisfy the requirement. However, together they can {the input can be translated from French to English, fed through the English Dictionary, and then translated back to French. The dynamic composition of such services is difficult using just the WSDL descriptions, since each description would designate strings as input and output, rather

Than the necessary concept for combining them { that is, some of these input strings must be the name of languages, others must be the strings representing user inputs and the translator's outputs. To provide the semantic concepts like language or French, we can use the ontologies provided by the Semantic Web. Service composition can also be used in linking Web (and Semantic Web)concepts to services provided in other network-based environments. One example is the sensor network environment which includes two types of services; basic sensor services and sensor processing services. Each sensor is related to one web service which returns the sensor data as the output. Sensor processing services combine the data coming from deferent sensors in some way and produce a new output. These sensors have properties that describe their capabilities, such a sensitivity, range, etc., as well as some non-functional attributes, such as name, location, etc. These attributes, taken together tell whether the sensor's service is relevant for some specific task. An
Example tasks in this environment would involve retrieving data from several sensors and using relevant fusion services to process them via SOAP calls. As an example, the data from several acoustic and infrared sensors can be combined together and after applying and special functions, this data may be used to identify the objects in the environment. In this setting, we need to describe the services that are available for combining sensors and the attributes of the sensors that are relevant to those services. More importantly, the user needs a exile mechanism for altering sensor services and combining only those that can realistically be fused (for example the set representing a particular geographic area shown as a latitude/longitude box).

Service Composition Architecture

We have developed a service composition prototype that has two basic components: a composer and an inference engine. The inference engine stores the information about known services in its Knowledge Base (KB) and has the capability to match services. The composer is the user interface that handles the communication between the human operator and the engine. The inference engine is an OWL reasoner built on Prolog. Ontological information written in DAML is converted to RDF triples and loaded to the KB. The engine has built-in axioms for OWL inferencing rules. These axioms are applied to the facts in the KB to and all relevant entailments such as the class inheritance relation between two classes that may be not be directly encoded in the subclass relationships. The composer lets the user create a work-up of services by presenting the available choices at each step. The user starts the composition process by selecting one of the services registered to the engine. A query is sent to the KB to retrieve the information about the inputs of the service, and for each of the inputs, a new query is run to get the list of the possible services that can supply the appropriate data for this input. The composer also shows the different service classes available in the system and alters the results based on constraints which the user may specify on the attributes of a service. These functionalities are explained in detail in the following subsections.
WSC
The IEEE WSC (Web Services Challenge) encourages both industry and academic researchers to participate. These include the groups that develop software components or intelligent agents. These applications should have the ability to discover relevant web services and also generate composite services. The sixth competition, which was held in 2010, focused exclusively on semantic composition of web service chains, whereas in the early editions, it was a syntactic-based contest. Rather than XML Schema, it incorporates the use of OWL ontologies to define services and their relationships to each other. The participants were required to determine relations between different types during the process of service composition. The IEEE WSC has its own test set generator. This tool generates an arbitrary number of services using any number of concepts that the user likes. These concepts are also randomly generated and saved in an OWL taxonomy file.

Pattern Discovery from Web Transactions

Effective marketing strategies and optimizing the logical structure of the Web site. Because of many unique characteristics of the client-server model in the WorldWide Web, including differences between the physical topology of Web repositories and user access paths, and the difficulty in identification of unique users as well as user sessions or transactions, it is necessary to develop a new framework to enable the mining process. Specifically, there are a number of issues in preprocessing data for mining that must be addressed before the mining algorithms can be run. These include developing a model of access log data, developing techniques to clean/filter the raw data to eliminate outliers and/or irrelevant items, grouping individual page accesses into semantic units (i.e., transactions), integration of various data sources such as user registration information, and specializing generic data mining algorithms to take advantage of the specific nature of access log data.

Matching on Functional Properties

The composer only presents as options for composition those services whose output could be fed to a selected service as an input. The matching of two services is done using the information in the service profiles. Each service profile describes its inputs, outputs, and the range of these parameters. The parameter descriptions in the profile allow defining two different types of matches between services: an exact match and a generic match. An exact match is defined between two parameters which are restricted to the same OWL class in their Service Problem descriptions. The services that supply an output of an exact match are more likely to be preferred in the composition and these services are displayed at the top of the matching services list. The match between the services whose output type is a subclass of the other service's input type is called a generic match. When the output of a service is subsumed by the input, the output type can be viewed as a specialized version of the input type and these services can still be chained together. The generic matches are put at the end of the list since they are less likely to be chosen for the composition. The inference engine also orders the generic matches such that the priority of the matches are lowered.
when the distance between the two types in the ontology tree increases.

Filtering on Non-Functional Attributes

The number of services displayed in the list as possible matches can be extremely high in many cases. For example, a power grid or telephone network might have many thousands of sensors each providing several services. This will make it infeasible for someone to scroll down a list and choose one of the services simply by name. Further even if the number of services is low, the service names themselves may not be mnemonic enough to let a user know what they do, or the short text descriptions from UDDI or other services descriptions would not be enough to fully describe the services. When the name of the service does not help to distinguish the services, other non-functional attributes of the services such as location will be useful to determine the most relevant service for the current task. Thus, a sensor description, linked to a particular service, can be queried as to the sensors’ locations, type, deployment date, sensitivity, etc. In our prototype, altering is provided based on the problem descriptions of the services. The problem hierarchies mentioned in section 4.3.4 each subclass inherits some attributes from its container class and extends them with other attributes that apply to its category. These attributes are presented to the user and the constraints entered for these properties constitutes the second level of altering.

Consider an example in the sensor network where we want to select a service whose input will be retrieved from a sensor service. With no other restriction, the system will present as many possible matches as the number of sensor services in the environment. If the user chooses to alter the results to the services of type AcousticSensorService that decreases the number of matches significantly. The composer then queries the inference engine about the non-functional parameter of the selected service type. Based on the answer returned from the engine, the composer creates a GUI panel in which the user can enter constraints for the properties of the services. The user’s constraints are translated to Prolog queries and sent to the inference engine. The engine simply applies the new query to the previous result set and removes from consideration those services that do not satisfy the current constraints.

Execution of Composed Services

The current implementation of the system executes the composition by invoking each individual service and passing the data between the services according to the constructed by the user. This method is primarily dictated by the DAML-S and WSDL specification which both describe the web services as an interaction of either a request/response or as a notification messaging between two parties. As a consequence of this design, the client program serves as the central control authority that handles all the RPC calls to invoke individual services. However, this centralized coordination users from scalability and availability problems. It also can require passing redundant messages between the coordinator and other parties causing a quite nescient use of the bandwidth which is a more severe problem when you consider the output of a such as the sensor readings of an acoustic sensor that may provide large waves. For the efficient
execution of a dynamically created composite process, we need a special framework where each node abides by a set of system rules to conduct the execution process by directly passing its result to the next service. In the prototype, we address this by adding the functionality of generating an XML workup description that can be passed to the non-centralized system in SOAP (and forwarded as necessary). As the standards in this area of web services are settled, it will be easy to adapt the system to the new interface.

**Semantic Association Discovery Search Engines**

Usually one of the user’s interests is finding semantic relations between two input terms. Old search engines handled these requests using learning and statistical methods, but semantic web standards and languages have provided more effective and precise methods. Semis is a real sample for these systems, its goals is finding and ranking semantic associations. There are different types of semantic association but most known of them is a sequence of classes and relations between two classes. In fact we talk about just two terms because as said in average length for users’ queries term. With respect to our definition for semantic association, two terms may have one of these associations: Null (both of them are instances of one concept), Direct (when there is a direct relation between them) and Indirect (chain of relations instead of single direct one). In the Bayesian network, it was applied in order to discover semantic association. Our reference ontology forms the graph of this network and logs of user’s queries are used to computing its parameters. In general manner, for finding semantic association between more than two terms some techniques have been proposed, for example in Spread Activation Technique is used to expand an initial set of instances to contain most relative instances to them. The initial set is populated by extracting important terms from user’s query, then with respect to the meta data Repository corresponding instances is retrieved and after expanding them an instances graph is produced which each of its edges has correctness weight in addition to usual semantic label. Technically speaking, after discovery phase often we have numerous semantic association, therefore a ranking policy must be used. In some criteria for these ranking algorithms are introduced:

- **Context:** special part of reference ontology that is interested by user
- **Subsumption:** low level classes in hierarchy have more information than their parents
• Path Length: having a shorter path between two terms indicates that they have near meaning.
• Trust: obtained results from trusted resources is more valuable in final ranking results.

Conclusion

In this work, we have shown how to use semantic descriptions to aid in the composition of web services. We have developed a prototype system and shown how it can compose the actual web services deployed on the internet as well as providing altering capabilities where a large number of similar services maybe available. Our prototype is the first system to directly combine the DAML-S semantic service descriptions with actual invocations of the WSDL descriptions allowing us to execute the composed services on the Web. The ontology-driven Semantic Freedom toolkit has been used for graphical creation of the ontology schema, as well as for automated population of the ontology with extractors. Additionally, Freedom was used for entity disambiguation. Lastly, we provided a summary of the statistics that make up for the current population of over 800,000 entities and over 1,500,000 explicit relationships among them.

We have investigated some major test data collections in today's semantic service research field. Some of these test collections are publicly available, and there is also another test collection that was specifically used for a research experiment. We looked into all these test collections to Find, in particular, their ability to be used for world-altering category of services. Furthermore, we presented the contests and the challenges of semantic services that use these test collections.

References