Semi-automatic Composition of Web ServicesUsing Semantic Descriptions

Prof. Haresh R. Parmar

Assistant Professor in Computer Engineering Department
Silver Oak College of Engineering & Technology, Ahmadabad.

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 interred 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

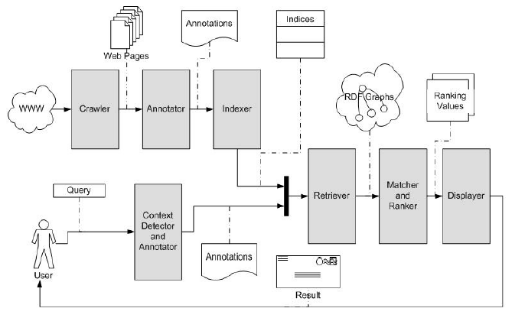
Web services are designed to provide interoperability between diverse applications. The platform and language independent interfaces of the web services allow the easy integration of heterogeneous systems. Web languages such as Universal
Description, Discovery, and Integration (UDDI) [13], Web Services 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 has 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-denned 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-denned 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 wouldinvolve retrieving data from several sensors and using relevant fusion services toprocess them via SOAP calls. As an example, the data from several acoustic andinfrared sensors can be combined together and after applying and specialfunctions, this data may be used to identify the objects in the environment. Inthis setting, we need to describe the services that are available for combiningsensors and the attributes of the sensors that are relevant to those services. Moreimportantly, the user needs a exile mechanism for altering sensor servicesand combining only those that can realistically be fused (for example the setrepresenting 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

Evolutionary Search Engines

The advanced type of search is something like research; infact as mentioned in this kind of searches aim at gatheringsome information about specific topic. For example if wegive the name of a singer to the search engine it should beable to find some related data to this singer like biography,posters, albums and so on. These engines usually use oneof the commercial search engines as their base componentfor searching and then augment returned result by these baseengines. This augmented information is gathered from somedata-insensitive web resources. In we showed overallarchitecture for such engines.As it can be deduced from the figure this architecture hassome similarities with what we discussed in previous subsection;here we crawl and generate annotation just for somewell-known informational web pages i.e. CDNow, Amazon, IMDB as mentioned in and. After this phase we collectannotations in a repository. Whenever a sample user poseda query two processes must be performed: first, we shouldgive this query to a usual search engine (usually Google) toBase (KB) and has the capability to and matching 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 areapplied to the facts in the KB to and all relevant entailments such as the classinheritance relation between two classes that may be not be directly encoded inthe subclass relationships.The composer lets the user create a workup of services by presenting theavailable 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 toto 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 supplythe appropriate data for this input. The composer also shows the different service classes available in the system and alters the results based on constraintswhich the user may specify on the attributes of a service. These functionalitiesare explained in detail in the following subsections.

WSC

The IEEE WSC (Web Services Challenge)encouragesboth industry and academic researchers to participate. Theseinclude the groups that develop software components orintelligent agents. These applications should have the abilityto discover relevant web services and also generate compositeservices.The sixth competition, which was held in 2010, focusedexclusively 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 OWLontologies to define services and their relationships to eachother. The participants were required to determine relationsbetween different types during the process of service composition.The IEEE WSC has its own test set generator. This toolgenerates an arbitrary number of services using any numberof concepts that the user likes. These concepts are alsorandomly generated and saved in an OWL taxonomy file.

Pattern Discovery from Web Transactions

Effectivemarketing strategies and optimizing the logicalstructure of the Web site. Because of many uniquecharacteristics of the client-server model in the WorldWide Web, including differences between the physicaltopology 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 todevelop a new framework to enable the mining process.Specifically, there are a number of issues in preprocessingdata for mining that must be addressed beforethe mining algorithms can be run. These includedeveloping a model of access log data, developing techniquesto clean/filter the raw data to eliminate outliersand/or irrelevant items, grouping individual page accessesinto semantic units (i.e. transactions), integrationof various data sources such **as** user registration Information, and specializing generic data mining algorithmsto take advantage of the specific nature ofaccess 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 isdone using the information in the service proles. Each service prole describesits inputs, outputs and the range of these parameters. The parameter descriptions in the prole allow denning two different types of matches between services,an exact match and a generic match. An exact match is denned between twoparameters which are restricted to the same OWL class in their ServiceProblemdescriptions. The services that supply an output of an exact match are morelikely to be preferred in the composition and these services are displayed at thetop of the matching services list.The match between the services whose output type is a subclass of the otherservice's input type is called a generic match. When the output of a service issubsumed by the input, the output type can be viewed as a specialized versionof the input type and these services can still be chained together. The genericmatches are put at the end of the list since they are less likely to be chosen forthe composition. The inference engine also orders the generic matches such thatthe priority of the matches are lowered when the distance between the two typesin the ontology tree increases.

Filtering on Non-Functional Attributes

The number of services displayed in the list as possible matches can be extremelyhigh in many cases. For example, a power grid or telephone network mighthave many thousands of sensors each providing several services. This will makeit infeasible for someone to scroll down a list and choose one of the servicessimply by name. Further even if the number of services is low, the service namesthemselves may not be mnemonic enough to let a user know what they do, orthe short text descriptions from UDDI or other services descriptions would notbe enough to fully describe the services. When the name of the service does nothelp to distinguish the services, other non-functional attributes of the servicesuch as location will be useful to determine the most relevant service for thecurrent task. Thus, a sensor description, linked to a particular service, can bequeried as to the sensors' locations, type, deployment date, sensitivity, etc.In our prototype, _altering is provided based on the problem descriptions ofthe services. The problem hierarchies mentioned in section

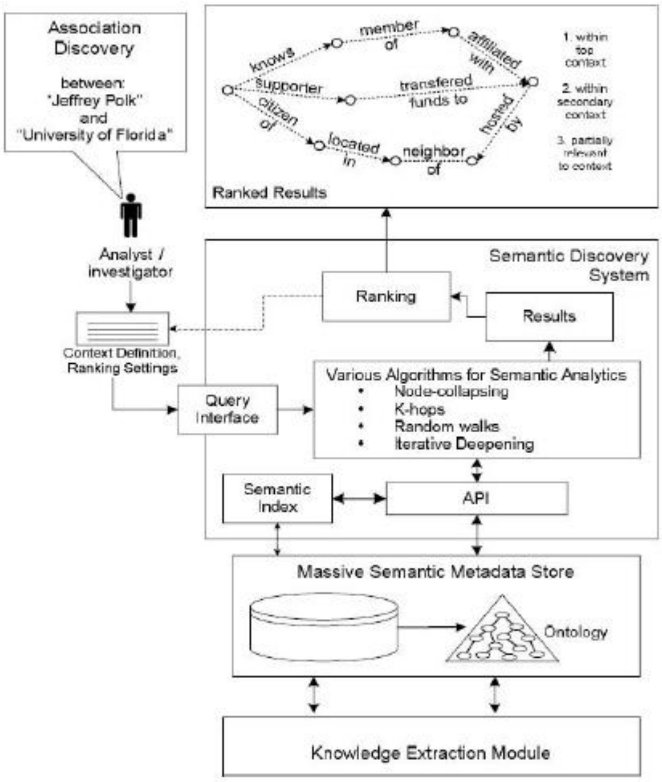
Each problemsubclass inherits some attributes from its container class and extends them withother attributes that apply to its category. These attributes are presented to theuser and the constraints entered for these properties constitutes the second levelof _altering.Consider an example in the sensor network where we want to select a servicewhose input will be retrieved from a sensor service. With no other restriction, thesystem will present as many possible matches as the number of sensor servicesin the environment. If the user chooses to alter the results to the services of typeAcousticSensorService that decreases the number of matches significantly. Thecomposer 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 forthe properties of the services. The user's constraints aretranslated to Prolog queries and sent to the inference engine. The engine simplyapplies the new query to the previous result set and removes from considerationthose services that do not satisfy the current constraints.

Execution of Composed Services

The current implementation of the system executes the composition by invokingeach individual service and passing the data between the services according to the constructed by the user. This method is primarily dictated by the DAML-Sand WSDL speciation which both describe the web services as an interactionof 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 controlauthority 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 thecoordinator and other parties causing a quite nescient use of the bandwidthwhich is a more severe problem when you consider the output of a such asthe sensor readings of an acoustic sensor that may provide large waves. Forthe efficient execution of a dynamically created composite process, we need aspecial framework where each node abides by a set of system rules to conductthe execution process by directly passing its result to the next service. In theprototype, we address this by adding the functionality of generating an XMLworkup 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 aresettled, 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 semanticrelations between two input terms. Old search engines handledthese request using learning and statistical methods, butsemantic web standards and languages have provided moreeffective and precise methods. Semis is a realsample for these systems, its goals is finding and ranking semanticassociations. There are different types of semantic associationbut most known of them is a sequences of classes and relationsbetween two classes. In fact we talk about just two terms because as said in average length for users' queriesterm. With respect to our definition for semantic association, two terms may have one of these association: Null (both ofthem are instances of one concept), Direct (when there is adirect relation between them) and Indirect (chain of relationsinstead of single direct one). In the Bayesian networ kswas 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. Ingeneral manner, for finding semantic association betweenmore than two terms some techniques have been proposed,for example in Spread Activation Technique is used toexpand 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 usualsemantic label. Technically speaking, after discovery phaseoften we have numerous semantic association, therefore aranking policy must be used. In some criteria for theseranking algorithms are introduced:

Context: special part of reference ontology that is interestedby user

• Subsumption: low level classes in hierarchy have more information then 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 thecomposition of web services. We have developed a prototype system and shownthat it can compose the actual web services deployed on the internet as wellas providing altering capabilities where a large number of similar services maybe available. Our prototype is the _rest system to directly combine the DAML-Ssemantic service descriptions with actual invocations of the WSDL descriptionsallowing us to execute the composed services on the Web.
The ontology-driven SematicFreedom toolkit has been used for graphical creation ofthe ontology schema, as well as for automated populationof the ontology with extractors. Additionally, Freedomwas used for entity disambiguation. Lastly, we provided asummary of the statistics that make up for the currentpopulation of over 800,000 entities and over 1,500,000explicit relationships among them.
We have investigated some major test data collections intoday's semantic service research field. Some of these testcollections are publicly available, and there is also anothertest collection that was specifically used for a researchexperiment. We looked into all these test collections to Find, in particular, their ability to be used for world-alteringcategory of services. Furthermore, we presented the contestsand the challenges of semantic services that use these test collections.

References

[l] R. Agrawal and R. Srikant. Fast algorithms **for** miningassociation rules. In *Proc. of the 20th VLDB Conference,*pages **487-499,** Santiago, Chile, **1994.**
[2] S. Agrawal, R. Agrawal, P.M. Deshpande, A. Gupta,J. Naughton, R. Ramakrishna, and S. Sarasangi. Onthe computation of multidimensional aggregates. In*Proc. of the 22nd VLDB Conference,* pages **506-521,**Mumbai, India, **1996.**
[3] R. Armstrong, D. Fkeitag, T. Joachims, andT. Mitchell. Webwatcher: A learning apprentice forthe world wide web. In *Proc. AAAI Spring Symposiumon Information Gathering from Heterogeneous, Distributed Environments.* **1995.**
[4] M. Balabanovic, Yoav Shoham, and ***Y.*** Yun. Anadaptive agent for automated web browsing. *Journalof Visual Communication and Image Representation,*
6(4), 1995.
[5] A. **Z.** Broder, S. C. Glassman, M. S. Manasseh, andG Zweig. Syntactic clustering of the web. In *Proc. Of6th International World Wide Web Conference,* **1997.**
[6] A. Sheth, C. Bertram, D. Avant, B. Hammond, K. Kochut,and Y. Warke. (2002). Managing semantic content for the Web. IEEE Internet Computing, 6(4), 2002. pp 80-87
[7] R. Mihalcea, and S. I. Mihalcea: Word Semantics forInformation Retrieval: Moving One Step Closer to theSemantic Web. ICTAI 2001: 280-287.
[8] P. Resnik, "Semantic Similarity in a Taxonomy: AnInformation-Based Measure and its Application toProblems of

Ambiguity in Natural Language”, Journal ofArtificial Intelligence Research, 1999.

[9] V. Kashyap, and A. P. Sheth, Semantic and schematicsimilarities between database objects: A context – basedapproach. VLDB Journal, 5(4):276—304, 1996.

[10] M. Rodriguez, and M. Egenhofer, Determining SemanticSimilarity among Entity Classes from Different Ontologies, IEEE Transactions on Knowledge and Data Engineering,Vol. 15, No. 2, March/April 2003.

[11] S. Handschuh, S. Staab. CREAM - CREAting Metadata forthe Semantic Web. Computer Networks. 42, pp. 579-598,Elsevier 2003.

[12] B. McBride. Jena: A semantic Web toolkit. IEEE InternetComputing, 6(6), 55-59, 2002.