An Approach for Semantic Web Services Automatic Discovery and Composition With Similarity Metrics

Ivo Calado
Embedded Systems and Pervasive Computing Lab
Federal University of Campina Grande
Campina Grande, Paraíba, Brazil
ivo@dsc.ufcg.edu.br

Heitor Barros
GrOW– Group of Optmization of the Web
Computing Institute
Federal University of Alagoas
Maceio, Alagoas, Brazil
rotieh@gmail.com

Ig Ibert Bittencourt
GrOW– Group of Optmization of the Web
Computing Institute
Federal University of Alagoas
Maceio, Alagoas, Brazil
ig.ibert@gmail.com

ABSTRACT

With the emerging growth of web services tecnology, the Web is now evolving to become a service provider. In this context, the semantic web services discovery and composition are hard tasks because injecting context into adaptive service integration raises a number of significant challenges, i.e. defining effective metrics to manage adaptation. This paper proposes an algorithm for Semantic Web Services discovery and compositon which makes use of similarity metrics.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*
; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process*
; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Plan execution, formation, and generation*

General Terms

Design, Experimentation, Algorithms

Keywords

Semantic Web Services, discovery, composition

1. INTRODUCTION

As Web Services rely on open standards for interfaces and protocols definitions, its main use is as basic blocks of software construction on service-oriented architecture [2]. The prosperity of such approach has gained attention and the community has invested on two major areas: Web Services discovery and composition.

SAC'09 March 8-12, 2009, Honolulu, Hawaii, U.S.A.

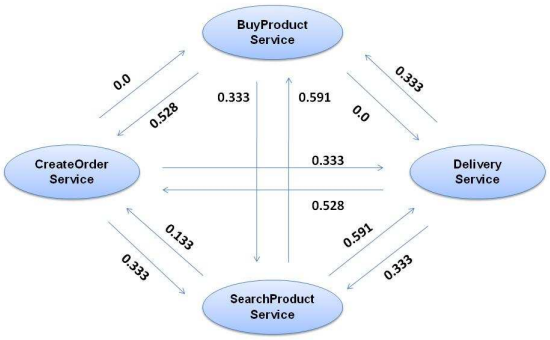

Figure 1: A graph representation of a services set

However, the addition of semantic annotations to services, through ontologies, enables the development of new techniques and tools that improve the services' discovery and composition processes. In this context, this paper presents an algorithm for such tasks that makes use of similarity metrics.

2. THE PROPOSED ALGORITHM

Our proposal allows the automatic discovery and composition of OWL-S described Semantic Web Services and uses similarity metrics in order to provide a more accurate discovery. This work was developed as an extension of [1], from which the following features can be listed:

- Direct and indirect matching of services;
- Weighted directed graph representation for the services relationships, as presented in Figure 1;
- Shortest path algorithms to locate the best path in the weighted graph, increasing composition accuracy.

The algorithm implementation was conceived using Java with the *MindSwap*[1] API for semantic services creation and manipulation.

[1]http://www.mindswap.org/2004/owl-s/api/

2.1 The developed algorithm

The algorithm has two execution steps. The first one, called 'preprocessing step', is executed during the service insertion into the search directory. This step is also responsible for maintaining the weighted graph structure up to date. The second one, or 'request processing step', is performed during discovery requests.

2.1.1 Preprocessing step

This step includes the insertion and removal of tasks from the service directory. The update process is kept alive search tool keeps updating to always provide consistent information about the stored services.

Every relation that is established between two services has a determined weight that is equivalent to the compatibility of the first service's input parameters to the second one's output parameters. Such compatibility measurement is defined through similarity metrics.

From the graph-weighted model, it is possible to construct paths and consequently enable services composition. For each newly inserted service, similarity metrics between this service and the other ones will be calculated according to the adopted one.

To illustrate, suppose a service 'A' with 3 output parameters and a service 'B' with 3 input ones. In this case, the algorithm does an interaction among B's input parameters to find the best possible combination with A's output parameters.

2.1.2 Request processing step

The format of a request is a 'perfect' service described in OWL-S. This service presents all the desired input and output parameters of the real service to be discovered.

The execution return is a Java object (class *Service*) representing the desired service. This object can represent both atomic and composite services. This feature creates an abstraction layer to the client application since the service composition becomes transparent.

Summarising, here are the service discovery process steps.

1. Create two empty sets: the first one contains the services whose inputs match the request, called I, while the other set contains the services whose output match the request, this one is called O;
2. Scan all nodes of the graph, and taking the compatibility filter as basis, perform a search for services that fit for I and O sets;
3. Try stablishing an intersection between the sets I and O through selecting the request's most compatible services. If such operation is successful, then there is a direct matching;
4. Stablish a cartersian product between I and O;
5. Input and Ouptut Weights are assigned to the graph edges as defined by the adopted similarity metric;
6. A graph search algorithm is applied to find the shortest path between the cartesian product nodes. The Dijkstra algorithm is being used in this implementation, but that is not a restriction. Other similar and improved algorithms could be used instead;
7. The results of direct and indirect macthings are compared and the service, or chain of services, with the highest similarity value, according to the request, is selected;
8. If a chain of services was chosen, then a composite service is created. A bind between the input and output parametes is done and the sequential execution of all bound services is automatically performed;
9. The service is returned;
10. If neither a direct matching nor an indirect one satisfies the request (the sets I and/or O are empty or there is no path at all between an element of I and an element of O) then nil is returned;

2.2 Extensions of the original algorithm

The changes to the [1] proposal were implemented to improve the discovery process. A list of the new features is as follows:

- Direct and indirect macthing are defined, then the best set of services is identified. Since that would depend on the services characteristics, a composite service can be more suitable to the request than a single service;
- Weight measurements were added to the graph edges. This allows verifying services relationship strength, which means semantic proximity. The weight in each edge is based on the use of similarity metrics among the related services' parameters. The original algorithm does not handle weighted graphs;
- A cartesian product is created between both set of services (see 2.1.2). This allows a more refined service composition.
- A graph search algorithm has been used to find the shortest paths in a weitghted graph.

3. CONCLUSIONS AND FUTURE DIRECTIONS

As an evolution of a former algorithm for the same purposes, this new approach relies on similarity metrics to help improving the quality of composite-services. As future work, we quote the importance of studying and comparing the available similarity metrics' results. This way, subsidy for new improvements is expected to arise shortly. Examples of such investment areas are the automatic selection or simultaneous use of all available similarity metrics to help software decide and reach adaptation goals on a composition scenario.

4. ADDITIONAL AUTHORS

Evandro Costa, Elvys Soares, Marlos Silva, Marcelo Arruda and Tarsis Toledo. Federal University of Alagoas.

5. REFERENCES

[1] R. D. Giv, B. Kalali, S. Zhang, N. Zhong, and A. Lopez-Ortiz. Algorithms for direct and indirect semantic matching of web services. Technical report, University of Waterloo, 2004.

[2] J. Ma, Y. Zhang, and J. He. Efficiently finding web services using a clustering semantic approach. In *CSSSIA*, page 5, 2008.