QoS Driven Semantic Based Grid Service Composition Using AND/OR Trees

Xhemal Zenuni
Faculty of Contemporary Sciences and Technologies
South East European University
Tetovo, FYR of Macedonia
e-mail: xh.zenuni@seeu.edu.mk

***Abstract*—Composition oriented service discovery is an important requirement and research challenge in Service Oriented Architectures (SOA), such as in the third generation of Grid. It provides added value services, more rapid application development and improved reusability of existing services. This paper proposes and formulates a composition approach where semantic information is used to determine Grid services dependencies in form of AND/OR tree associated with semantic QoS information, thus transforming the issue of service composition discovery to constrained AND/OR tree search problem. Different constraint forms and QoS aggregation patterns in such trees are analyzed, and some major constrained searching techniques that can be applied to such trees are discussed. Our findings show that AND/OR trees are expressive in addressing QoS – aware semantic Grid service composition and able to employ different discovery searching techniques for that purpose.**

Keywords-Grid service composition; QoS; AND/OR tree; Semantic Web; Ontology;

I. INTRODUCTION

The Web Service Resource Framework (WSRF) [1] set documents of specifications defines a new formal framework for building current and future Grid applications based on Service Oriented Architecture (SOA) principles. In such Grid, services are becoming the fundamental building blocks and the basic collaboration element which can be used to build grid applications and resolve complex scientific problems.

In this conceptually new approach, composition oriented service discovery brings multiple benefits. In many situations, individual services in isolation are limited to respond to more complex user demands. However with the combination of several ones together, new solutions not anticipated in individual services can be implemented and more complex problems can be solved effectively. In addition, the same service can be combined in many composite ones, thus enabling better service re – usability.

On the other hand, when the composition process is automated, new services can be constructed faster and with less effort, thus accelerating a rapid application development in Grid. Moreover, a good service composition middleware can hide the composition details, by making visible to users only the available interfaces. In effect, this black – box encapsulation can simplify their usage.

Service composition system is part of a larger lifecycle development in Grids and imposes a list of requirements that can not be definitive. However, in order to achieve enhanced service composition process, some fundamental requirements and challenges need to be addressed timely, especially faced with the service proliferation. The composition middleware should effectively and efficiently discover service dependencies and coordination rules of different services in repositories, and conducting this in an automatic manner. Secondly, Grid systems are dynamic, with services created and destroyed on the fly. Service composition system should be adaptable and must detect those changes, and make quality decisions at run – time. Moreover, in large repositories and for a given problem, more than one solution may exist. The system should allow the users to define extra non – functional properties and preferences as a discriminating and/or ranking factor. Furthermore, QoS becomes a common model for narrowing the list to best discovered solutions.

Considering all these factors, this paper presents a holistic approach to service composition based on three fundamental elements. Semantic Web technology has been used to model and describe Grid Services functionalities and QoS features and as enhanced background to determine service interdependencies. Then, from functional point of view, these dependencies are expressed in form of AND/OR tree. Finally, composition services are discovered by QoS constrained search in such AND/OR trees.

The rest of the paper is organized as follows. Section 2 briefly presents the related work. Section 3 explains the semantic model developed to describe Grid service features, and explains how this model enhances the discovery of services. Section 4 introduces AND/OR trees and how service dependencies can be expressed through them. Section 5 explains the different forms of QoS preferences that a user may express, the aggregation patterns that occur in AND/OR tree structure and how to calculate end – to –end QoS dimension of composition services in such situations. Section 6 investigates some major constrained searching algorithms that are applicable in AND/OR trees. Finally, Section 7 concludes the work and gives the future directions for improvements.

II. RELATED WORK

Many works on service discovery and automated service composition have been reported, especially as Artificial Intelligence planning problem [2]. However, the focus here is more on graph approaches based on I/O data and semantic information of services, which closely relates to our work. Liang [3] proposes a semi – automated method for service composition based on AND/OR graphs and applies the

 ISBN: 978-1-61208-116-8

REV* searching algorithm to find composite services. However, in this approach, the role and usage of semantic information in such model is not comprehensible. In addition, it didn't consider the scale of services. On the other hand, Gu [4] advances the work by presenting a faster service composition method, where indexing of services plays critical role for acceleration of composition algorithm. It proposes also a method how to handle semantic relationship between I/O data, but yet the method doesn't fully explore advantages and conversion of semantic information. Yan [5] goes further, especially by improving the searching algorithm to support the recognition, conversion and usage of semantic information described in OWL format. However, in this approach, as with all the other above, service composition is mainly seen from functional aspect of services. They have been used to determine the service dependency graph and then a searching algorithm is applied to find composite services. Our work is distinguished at least in two aspects, which we consider as contributions. First, we developed QoS ontology for describing the non – functional aspects of services, allowing users to define QoS preferences as well. This becomes necessity with service proliferation, and when many solutions may be anticipated. Secondly, we extended some searching algorithms for AND/OR trees to find composite services that fits to user constraints.

III. SEMANTIC GRID SERVICES

Industry standards for Grid Services, such as UDDI (Universal Description, Discovery and Integration) [6], WSDL (Web Service Description Language) [7] and SOAP (Simple Object Access Protocol) [8] focus on operational and syntactical details, which in turn make service publication, discovery and composition process very restricted. To overcome the limitations of keyword oriented searching with such standards, Semantic Web technologies have gained momentum as an approach that can provide better background and enhanced service discovery and composition mechanisms based on semantic information.

To this point, the ontologies for service discovery and composition can be defined at two main levels. The first level consists of domain – specific ontology, which describes specific domain concepts, in form of class and sub – class hierarchy, individuals of such classes and other relationships in them, such as synonyms, etc. The second level consists of an upper ontology, which provides uniform description of provided services. Several upper ontologies [9][10] for service modeling have been reported, mainly to describe them in terms of IOPE (Inputs, Outputs, Preconditions, Effects) which drive the composition process, but as discussed in [11], such ontologies have two main drawbacks when applied to Grid Services. Grid services usually act upon some resource, and the semantic information of Grid "resource" is absent. And secondly, the QoS features of services are not represented. To overcome these limitations, a new ontology model that includes these two aspects was presented in [11].

This ontology model allows semantic description of different aspects of Grid Services. The robustness of this model can be seen in many directions. First, a search algorithm can recognize different relationships defined in domain specific ontology, such as instance and concept relationships, other relationships such as synonyms, and use the semantic information to better recognize the relationships of different services and their coordination rules.

Secondly, service providers can describe multiple dimensions of any arbitrary QoS parameter for the services they provide, such as its overall impact, if it is measurable, the metric, the unit used and so on. This features become important ranking and/or discriminating factor in service provisioning.

Finally, service requestors and providers can express their QoS preferences in different forms, especially using different measuring units, and a search algorithm can support their equivalence if the relationships of different units if they are priory defined.

IV. AND/OR BACKGROUND

An AND/OR graph (and AND/OR tree as special case) [13] is a structure commonly used in automatic problem solving where the solution involves decomposing the problem into smaller problems, and then the solution is found by solving this small tasks. It is a generalization of directed graphs, consisting of two types of nodes (connectors), namely AND connectors if there is a logical AND relationships in such nodes, and OR nodes if there is such logical relationship. In such graphs, the terminal nodes are solved nodes. If non – terminal node has OR successors, then this node is considered as solved only if at least one of its successors is solved. Contrary, if non – terminal node is AND node, then it is solved only all of its successors are solved.

These characteristics place AND/OR graphs as powerful formalism to express service dependency graph of services, because it can handle *n – to – m* relationships of I/O. Such dependency graphs are created by analyzing inputs and outputs of available services.

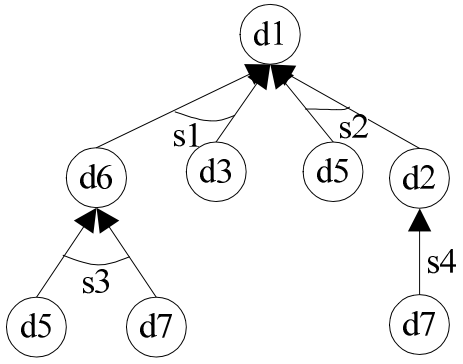

Figure 1. AND/OR tree representing service dependencies

For example, if four services are available in a service repository, such as *s1={In(d3,d6}, Out{d1}}, s2={In{d2,d5}, Out{d1}}, s3={In{d5,d7}, Out{d6}}* and *s4={In{d7} ,Out{d2}},* then the service dependency graph can be represented in AND/OR tree like in Figure 1.

For example, if a user specifies the request in form of desired outputs (or resource) *O={d1}* and the available inputs *I={d3, d5,d7}* then a solution can be found in two ways. The first solution is using a chain of services: *s1* and *s3*. The second solution is the service chain: *s2* and *s4*. Indeed, in large repositories more then one solution can be anticipated. In this case, QoS features become a common model to discriminate and distinguish the different solutions. Therefore, service composition model should allow users to define non – functional demands as well as an important requirement faced in huge service repositories.

V. MEASURING QOS OF COMPOSITE SERVICES

In large repositories with services that overlap in their functionality, QoS is becoming natural discriminating and ranking factor. Quality of service is important in composite services as well, as they should not respond to business complex needs only, but they must perform within the limits of given QoS constraints.

User preferences over quality of service parameters can take different forms. In certain cases, user may express certain QoS requirements over a single variable only, such as the response time solely. But in many situations, users have more complex demands, and the quality of composite service is evaluated based on multiple variables, like response time, cost, throughput, and so on. Moreover, preferences may come in form of constraint satisfaction or constraint optimization problem. In former situation, given the AND/OR tree with QoS data, the quest is to find solutions that satisfy the given constraints. In later case, the composition system should be able to find "the best" solution that maximize or minimize the given objective function.

Based on this, different QoS constraints over composite services can be applied, ranging from single variable as constraint satisfaction problem to multiple variable as constraint optimization problem. The later is considered especially difficult situation. In presence of multiple QoS parameters, it is a difficult task to find the optimal solution and there must be tradeoffs among different quality criterion. It is not always possible to find a solution with minimum execution time, minimum cost and highest availability rate of services.

Moreover, not all QoS parameters follow the same aggregation pattern when doing end – to – end planning. The solution of AND/OR tree is rather sub tree than a path, and this in turn involves combination of parallel and sequential vertices. This implicates different aggregation patterns when calculation the global QoS of the composite service. Indeed, for the computation of the global QoS, four different aggregation functions have been identified, and brief explanation is provided in what follows:

1. No aggregation can be applied: certain QoS can not be aggregated. This is especially true for non – measurable ones, such as the requirement that each service has to support "SOAP v2.0" or "X.509" digital certificate for communication. In this case, such QoS are used on the level of local planning, even when end – to – end QoS analysis is performed.
2. Critical Path Calculation: in parallel structures, for some QoS is taken into consideration the maximum value as valid. Such example is the execution time of services. There is no point to further minimize the value of the lower execution time in parallel structure, since has no affect to global QoS.
3. Sum function: some QoS are aggregated using pure sum function. For example, the price of the composite service is calculated as the sum of all involved services in solution, regardless in parallel or in sequential manner.
4. Average sum: some QoS parameters, such as reputation, are aggregated as average sum of all involved services in solution.

VI. SEARCHING COMPOSITE SERVICES

Different searching techniques can be applied to AND/OR trees, depending on the form of QoS preferences. We have mainly considered multiple QoS parameters, and three fundamental searching approaches that can be applied in such situations are discussed in what follows. First approach can be used for QoS constraint optimization solution, and the other two algorithms for multiple QoS constraint satisfaction solutions.

First, in presence of multiple QoS values, we can transform them in a single value using the equation described in:

$$tw = \sum_{i=1}^{N} w_i * QoS_i \quad (1)$$

where N is the number of different QoS taken into consideration, w_i is the weight that the user gives to QoS parameter i. In addition, the following condition should hold: $w_i = [0,1]$ and $\sum_{i=1}^{N} w_i = 1$. Thus, the multidimensional QoS is transformed into one single value. In this situation, the AO* algorithm [10] can be applied directly to find composite services.

The limitations of first method is that not every QoS can be expressed using the Equation 1. Many QoS are not measurable and are not numbers. In this case, the first approach is not applicable. In addition, we may want to express some specific QoS boundaries that we do not want to be exceeded. Thus, the issue is transformed to constrained satisfaction problem, and not optimization one.

A critical issue when attempting to find the solution tree at run time is the ability to calculate QoS aggregate values of the multiple parameters that are presented in Section 5, and how to eliminate nodes that exceed the preset QoS threshold from further expansion. Non – measurable QoS parameters

 ISBN: 978-1-61208-116-8

are easy to use, because those services that violate these criteria's are automatically discarded during AND/OR tree expansion. For other patterns, we modify the QoS parameters values of all related nodes in the solution path recursively using the modification formulas as follows:

- For Critical Path Calculation (e.g. execution time):

$$ET(n) = \begin{cases} ET(n) + ET(p), \text{ n is AND node} \\ ET(p), \text{ n is OR node} \end{cases} \quad (2)$$

where *p* is parent node of *n*.

- For sum pattern(e.g. price):

$$\Pr ice(n) = \begin{cases} \sum_i \Pr ice(c_i), \text{ n is AND node} \\ \max_i \Pr ice(c_i), \text{ n is OR node} \end{cases} \quad (3)$$

where c_i are children's of node *n*.

- For average sum (e.g. reputation) the situation is more complex, and we keep two data, i.e. the reputation of node and how many services contributed to that reputation *(N)*:

$$R(n/N) = \begin{cases} \dfrac{R(n) + \sum_i R(c_i)}{N + \sum_i R(N_i)}, \text{ n is AND node} \\ \min_i R(c_i / N_i), \text{ n is OR node} \end{cases} \quad (4)$$

where c_i are children's of node *n*.

Then, two major approaches could be applied directly: breadth – first constrained searching and depth – first constrained searching. The pseudo – code for constrained breadth – first searching in bottom – up fashion is presented in Algorithm 1.

Algorithm 1: Pseudo code for breadth-first AND/OR tree constrained searching

1:	Put the start node *s* (*s* points to desired outputs) on a list called OPEN
2:	Remove the first node on OPEN and put it on another list, for example called CLOSED and call this node *n*.
3:	Update QoS aggregate values recursively in expanded tree using Equations 2, 3 and 4.
4:	Check constraints.
5:	If QoS violated, label then the node *n* as UNSOLVABLE and continue. Otherwise go to step 9:.
6:	Apply the unsolvable – labeling procedure to the search tree.
7:	If the start node is labeled unsolvable, exit with failure; otherwise continue.
8:	Remove from OPEN any nodes having unsolvable ancestors and their influence in the overall QoS values and go to step 2:.
9:	Expand node n, generating all its successors. Put these successors at the end of OPEN and provide pointers back to n. If there are no successors, label n as UNSOLVED and go to step 6:, otherwise continue.
10:	If any of the successors are terminal nodes (desired inputs of services), label them as SOLVED and continue; otherwise go to 2:.
11:	Apply the solve labeling procedure to the search tree
12:	If the start node is labeled SOLVED, exit with the solution tree that verifies that the start node is solved; otherwise continue;
13:	Remove from OPEN any nodes that are solved or that have ancestors that are solved
14:	Go to 2:.

We first create an auxiliary node *s* and connect it to user desired outputs. In breadth – first fashions node expansion, the solution tree is incrementally enlarged by adding more nodes to AND/OR tree structure, and then we continuously update the QoS aggregate values of nodes using Equations 2, 3 and 4. If the QoS contribution of the last node violates the user's preset QoS threshold then this node is removed from further expansion, including its influence to the possible solution tree. We stop the whole procedure when the start node *s* is marked SOLVABLE or when there are no further nodes to expand.

Indeed, the AND/OR tree contains two types of nodes. Data nodes (input and output of services) which are of type OR nodes, and they do not contribute directly to the aggregate QoS values. On the other side, services are AND type of nodes, because all their inputs must be available for their successful invocation. These nodes directly contribute to the overall QoS of composite services.

In following paragraphs we provide an example that illustrates the way the algorithm proposed in the previous section works. Assuming the repository presented in Table I, a simple request given by an imaginary client would be as follows:

- Output: ZipCode, PriceDollar,.
- Input: Book, GoalCurrency, City.
- Constraints: execution time to be less then 7 millisecond, price to be less then 10 dollars ($), and reputation to be greater then 0.80.

TABLE I. SAMPLE SERVICE REPOSITORY

Service	Input	Output	QoS
CurrencyConverter (CC)	PriceEuro GoalCurrency	PriceDollar	[2,3,0.85]
BookpriceFinder (BF)	ISBN	PriceEuro	[1,2,0.87]
ISBNFinder (IF)	Book	ISBN	[3,2,0.80]
ZipCodeFinder (ZCF)	City	ZipCode	[2,2,0.82]
CompositeService (CS)	Book GoalCurrency	PriceDollar	[7,1,0.80]

In Table I, the QoS parameters of services are expressed in form of a vector. The first element denotes its execution time in milliseconds (ms), the second parameter denotes its invocation price in dollars ($) and the third element gives information about its reputation. Figure 2 illustrates fragments of trace of breadth – first searching in AND/OR tree, solving the given problem.

The final solution is found using services ZCF, CC, BF and IF. Another alternative solution can be obtained using services ZCF and CS. However, during the tree expansion, CS exceeds the threshold of execution time to be less than 7 milliseconds.

The best – first constrained search can be applied in similar fashion, by expanding first the recently generated nodes firstly.

Although the implementations details are out of the scope of this paper, a prototype system that serves as proof – of – the concept discussed in this paper is developed. It consists

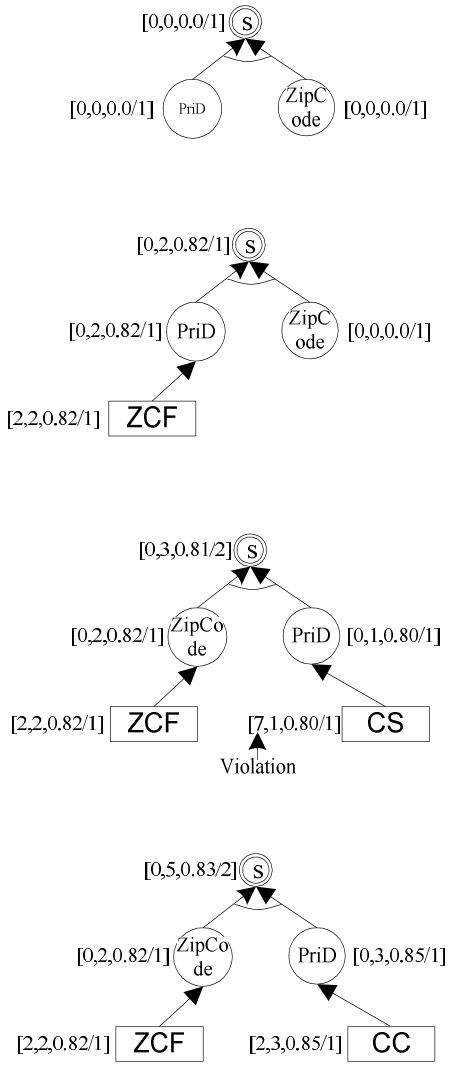

Figure 2. A trace of breadth –first AND/OR tree expansion

of four basic layers. The first layer consist of files and other semantic data used to describe functional and QoS features if available services using the ontology scheme presented in [7]. Second layer consist of the searching function that is able for semantic discovery of individual services that takes into account QoS constraints, and JENA API [14] was selected. The third layer consists in creation of search space representation and searching techniques for finding composition services. The Java API JGraphT [15] primitives and generic infrastructure for graphs has been adopted to represent AND/OR trees and to implement the searching approach explained earlier. Finally, the fourth layer is the user interface through which are entered the QoS constraints and displayed the result.

VII. CONCLUSION AND FUTURE WORKS

The AND/OR tree represent an elegant formalism to express the Grid service composition problem with QoS constraints. The model has high expressiveness, which consequently allows addressing QoS driven service composition from different perspectives. Constraints can be expressed in all shapes, sizes and flavors. In addition, different searching techniques can be applied to find composite services that fit to complex user requirements. Combined with semantic annotations used to describe functional and non – functional features of services, it provides flexible infrastructure for composition oriented Grid service discovery.

Service composition systems except being effective, they must be able to find composite services in a reasonable time. In effect, this depends on the underlying implementation details, such as the data structures used, searching techniques and cleverness how to combine them in an effective way. Moreover, the evaluation of the efficiency should be conducted on clear benchmarks. In absence of widely accepted benchmarks, the evaluation turns out to be difficult process. Therefore, our future work will be mainly focused on investigating and developing efficient, flexible data structures and searching techniques that address semantic composition discovery of Grid services based on AND/OR graphs not effectively but efficiently as well, and compare them with other approaches on clear benchmarks. The construction of a friendly user interface would also contribute as an improvement.

REFERENCES

[1] Web Service Resource Framework. Available online at: http://www.oasis-pen.org/committees/tc_home.php?wg_abbrev=wsrf (Last accessed: October 2010).

[2] J. Rao and X. Su. A Survey of Automated Web Service Composition Methods. *In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition*, SWSWPC 2004, San Diego, California, USA, July 6th, 2004.

[3] Liang, Q.A. and Su, S.Y.W. AND/OR Graph and Searching Algorithm for Discovering Composite Web Services. *International Journal of Web Services Research, vol.2, no.4, pp.* 48 – 67, 2005.

[4] Gu, Zh., Xu, B., and Li, J. Inheritance – Aware Document – Driven Service Composition. *CEC/EEE '07*, p. 513, Tokyo, Japan. IEEE Computer Society.

[5] Yan, Y., Xu, B. and Gu, Z. Automatic Service Composition Using AND/OR Graph. *10th IEEE Conference on E – Commerce Technology and The Fifth IEEE Conference on Enterprise Computing, E – Commerce and E – Service. p.* 335. *2008*

[6] UDDI Specification v.3.0.2. Avaliable online at: http://www.oasis-open.org/committees/uddispec/doc/spec/v3/uddi-v3.0.2-20041019.htm (Last accessed: October 2010).

[7] WSDL Spec. Avaliable online at: http://www.w3.org/TR/wsdl, March 2001. (Last accessed: October 2010).

[8] SOAP Spec. Available online at: http://www.w3.org/TR/2007/REC-soap12-part0-20070427/,

[9] OWL – S: Semantic Markup for Web Services. Available online at: http://www.w3.org/Submission/OWL-S/ (Last accessed: October 2010).

[10] WSMO: Web Service Modeling Ontology. Available online at: http://www.wsmo.org/ (Last accessed: October 2010).

[11] Zenuni, Xh., Ismaili, F. and Raufi, B. Ontology Design and Development for Grid Services. In Proceedings of the Fourth International Conference of Information Systems and Grid Technologies. pp. 105-114. 2010.

[12] Nilsson, N. Problem Solving Methods in Artificial Intelligence. McGraw – Hill 1971.

[13] Martelli, A., and Montanari, U. Optimizing Decision Trees Through Heuristically Guided Search. Commun. ACM, vol 21, no 12, pp. 1025 – 1039, 1978.

[14] JENA: A Semantic Web Framework for Java. Available online at: http://jena.sourceforge.net/ (Last accessed: October 2010).

[15] JGraphT: A Free Java Graph Library. Available online at: http://www.jgrapht.org/ (Last accessed: October 2010).

 ISBN: 978-1-61208-116-8