Efficient Intelligent Secure for Web Service Composition

Somayeh Karimi, Seyed Morteza Babamir
Islamic Azad University, Meymeh Branch, Department of Computer, Meymeh, Iran
University of Kashan, Department of Computer Engineering, Kashan, Iran
S_karimi@iaumeymeh.ac.ir, babamir@kashanu.ac.ir

Abstract—Service-Oriented Architecture (SOA) provides a flexible framework for web service composition. Using standard based protocols (such as SOAP and WSDL), composite services can be constructed by integrating atomic services developed independently. For each service in the composition, many service providers can offer the same function but may have different non-functional attributes (i.e. security, QoS parameters). These attributes have to be considered to make a decision selection more carefully and have suitable composition of web services composition. In this paper, we propose a broker-based framework for integration and adaptation of non-functional aware web services. We present the web service selection mechanism which selects the best (most suitable) web service based on the both requester and provider’s security issues and quality attributes.

Key words — Web service composition, Business process, Non-functional constraints, Security, QoS (Quality of Service), Broker.

I. INTRODUCTION

Nowadays the web is the main means by which companies and organizations are carrying on their business. This central role of the web has forced many companies to reorganize their businesses by using Web-based heterogeneous technologies in order to remain competitive in a business world. In such a scenario, the web service paradigm has emerged as the most promising approach for web-based application development [1]. A web service is a software system designed to support interoperable application-to-application interaction over Internet. Some benefits of adopting web services are that they are platform and vendor independent, since they are based on a set of standards, they provide a means for the convergence of disparate business functionalities, they make easier to deploy business applications for trading partners, thus resulting in a significant reduction in total cost of development. Web services rely on a set of XML standards such as Universal Description, Discovery and Integration (UDDI) [2], web services Description Language (WSDL) [3], and Simple Object Access Protocol (SOAP) [4]. One of the major goals of web services is to make easier their composition to form more complex services. To this purpose, many emerging languages (e.g., BPEL4WS [5], WSBPEL [6] and BPML [7]) have been proposed for coordinating web services into a workflow.

Discovering of web services for take parting to the composition are based on their functional properties. The increasing number of web service providers on the web supporting numerous web services having same or similar functionality caused using their non-functional properties to distinguish them and several tools and techniques was created to search suitable web services.

In this paper, we focus on security and QoS issues of web services compositions which may be specified by requesters or provider’s web services that must be taken into account when composing web services. For instance, a web service provider may not want to accept requests issued by a specific IP address, or a web service requester wants to use only web services with minimum response time in the composition. In this paper, we propose a brokered architecture to compose web services according to the specified security constraints and represent a mechanism to select the best web services with regards to QoS (i.e. response time, service cost, availability and reliability) properties. The main motivation of the proposed framework is to delegate brokers to make intelligent web service compositions. The main functions of a broker include:

- Service tracking: A broker has a service repository to record all feasible web services it is aware of.
- Service selection: This is the key function of our service broker. A broker match web services according to security constraints and select the best web services to execute a business process.

After each business plan transaction, a user should report the record QoS performance (from the services) to the broker so that the broker can update the statistical QoS data of each service.

The rest of paper is organized as follows. Next section discusses related work. Section 3 discusses security and QoS issues related to web service composition. Section 4 describes broker based architecture for web service composition and presents the security matchmaking and QoS Adapting. Section 5 presents the implementation details. Section 6 draws conclusions.

II. RELATED WORKS

Recently, business process or workflow proposals relevant to web services are proliferating in the business and academic world. Most of the proposals are XML-based languages to specify web services interactions and compositions. All of the proposed XML languages are based on WSDL service descriptions with extension elements, such as BPEL4WS (Business Process Execution Language for Web Services) and
BPML (Business Process Modeling Language). BPML is XML-based meta-language developed by the Business Process Management Initiative (BPII) as a means of modeling business processes. Advanced semantics such as nested processes and complex compensated transactions that are supported by BPML, are not addressed by BPEL4WS. BPEL4WS by combining Microsoft’s XLANG [8] with IBM ‘s WSFL (Web Service Flow Language) [9], provides a language for the formal specification of business processes and business interaction protocols. It can model the behavior of both executable and abstract processes. A detailed comparison between BPML and BPEL4WS is given in [10].

To describe the behavior of a business process based on interactions between the process and interfaces of its web service, BPEL4WS defines a model and a grammar. In short, a definition of BPEL4WS business process can be made up of as a template to create business process instances. Each of these activities in a routine model must be executed by an appropriate web service. In this scenario, the role of selection manager is to assign an appropriate web service to each activity. This assignment process is called matchmaking. Besides exploiting the UDDI registers, the matchmaking process can be performed also by means of semantic web service descriptions. In this context, DAML-S [11] provides capability to semantically annotate web services based on an ontology that provides classes and properties to describe content and capabilities of the web services.

The closest to our work is [12], in which authors present a security broker for the composition of web services. However, their work is more focused on security parameters, which are specified by the requester and provider of web services and must be taken into account when composing web services. A difference between the approach reported in [12] and the one proposed in this paper is that we consider both security and QoS constraints of web services composition but [12] identifies only security issues. We propose a broker-based framework to compose Web services based on the security requirements related to both Web service requestors and providers, and optimize the composition by considered QoS parameters. The main functionality of this broker is to match web services according to security constraints and select the best web services to execute a business process.

Some related work on QoS, such as the METEOR project has been done in the workflow area [13]. Four QoS attributes are: time, cost, reliability and fidelity. Instead of constructing a QoS workflow, this project focuses on analyzing and verifying a workflow QoS.

Other related works are those that exploit AI by planning techniques for web service composition. Among them, we recall the McIlraith [14] that extends the logic programming language Golog for automatic composition of web services, while the one by Medjahed [15] proposes a technique to generate composite web services from high level declarative description. This method uses compound rules to determine whether two services are compoundable.

### III. NON-FUNCTIONAL MODEL FOR WEB SERVICE COMPOSITION

Such as the user’s request, several non-functional properties may deeply affect on web services compositions that are satisfied by them. In this paper, we consider two classes of the non-functional parameters:

#### A. Security constraints

It is important to note that in order to verify whether a security constraint is satisfied by a web service or a web service composition, we need, in addition to a constraint language, also a language for specifying the security characteristics of a web service. For instance, a security constraint of a web service provider could require the adoption of a specific authentication mechanism. To verify this constraint, the broker needs to know which authentication mechanisms a web service supports. We call these two classes of information security capabilities and security compatibility, respectively. Security capabilities describe the security features of a web service, that is, strategies and techniques adopted by the service for ensuring security requirements. Clearly, it is essential that there exists one or more trusted entities in charge of validating and issuing security capabilities. Security Compatibility refers to those conditions that a web service could impose to another web service in order to cooperate with it. For instance, a web service may not want to accept requests from those web services that do not encrypt input parameters. We store these constraints in WSDL document. For more details refer [12].

#### B. QoS attributes

In this section, we consider quantitative non-functional properties of web services, which can be used to describe the quality criteria of a web service [16, 17]. These can include generic QoS attributes like response time, availability, service cost, reliability, whilst these attributes can be quantified and represented by real numbers. To represent the QoS attributes of service s, We use the vector $Q_s = (q_1(s), \ldots, q_r(s))$ in which the function $q_i(s)$ determines the value of the i-th quality attribute of s. The values of these QoS attributes can be directly collected either from service providers (e.g. price), recorded previous execution monitoring (e.g. response time) or user feedbacks (e.g. reputation) [18]. The set of QoS attributes can be divided into two subsets: positive and negative QoS attributes. For the values of positive attributes it is required to be maximized (e.g. Reliability and availability), whereas for the values of negative attributes it is required to be minimized (e.g. price and response time). The following four quality metrics of a web service are considered:

1. **Response Time** ($T_s$): the time interval between when a service is invoked and when the service is finished.
2. **Service Cost** ($C_s$): the price that a service requester has to pay for invoking the service.
3. **Availability** ($A$): the probability that the service is available at some period of time.
4. **Reliability** ($R$): the probability that a request is correctly responded within the expected time [19].
IV. BROKER ARCHITECTURE

We propose broker based architecture for web services with an objective of matchmaking security constrains and selecting the best web service according the QoS properties and preferences. Fig. 1 depicts the different component of the broker based web service architecture.

Figure1. Broker architecture

A. Service Table

Each web service in the Service Table (SR) includes several fields are represented in Table 1 (both functional and non-functional characteristics).

B. QoS manager

This component performs registration, updating, deleting QoS information related to web services. In addition, it obtains the performance for specific QoS property values of web service from the provider. Finally before registering the obtained QoS property values into RS registry, Broker verifies and certifies them[19] These numbers will be refined by QoS-Manager in RS according to the actual values received from user feedbacks to reflect more accurate values.

C. Composition manager

To generate a process plan, composition Manager (CM) takes a user's service request as input. Process plan is an abstract process that consists of service classes and the connection relationships between them (only the sequential connection is discussed in this paper). A process plan defines a flow of service classes without identifying the actual service to be invoked. Fig 2 shows a workflow with three tasks and a process plan to relate it with three service classes. To generate process plan, CM is responsible. Users create the workflow by selecting which tasks must be done and by specifying connection relationships between them. Then discover functional similar web services from the UUDI through functionality matching [20, 21].

D. Selection manager

We should know for each task, we have a list of web services able to perform it, without considering any security compatibility and QoS attributes during this selection. This is done by the selection manager. Indeed, given the process plan and web services candidates for each task by composition manager, the security matchmaker selects web services satisfying the specified security constraints and considering QoS attributes. The function of selection manager includes:

a) Verify the compatibility of web services with regard to security constraints.
Two web services $w_1$ and $w_2$ are compatible with regard to security if $w_2$ satisfy $w_1$'s compatibilities constraints. Thus, when selecting a web service to be associated with a task, the Security Matchmaker has to choose web services that are compatible with regard to security with the web Services already assigned to activities in the business process. To represent the web services and relationship between them, we use a graph direction that shows all possible security-aware web service compositions. In general, a level $j$ of the composition graph is related to the $j$-th task in the process plan, where each node at level $j$ represents a web service able to perform activity $j$ and compatible with regard to security with all its predecessor nodes. Therefore, each path in the graph denotes a security-aware composition (Fig 3).

Figure2. Workflow and business plan

Figure3. An example of composition graph
TABLE I. SERVICE TABLE

<table>
<thead>
<tr>
<th>Web service</th>
<th>Service class</th>
<th>Task</th>
<th>Ws capability</th>
<th>Compatibility constraint</th>
<th>Response time</th>
<th>Cost service</th>
<th>Availability</th>
<th>Reliability</th>
</tr>
</thead>
</table>
| s1          | S1            | T1   | authentication= SAML  
Privacy Access control=P3P | Signature=XML-SIG | 100 | 50 | 0.89 | 0.95 |
| s2          | S1            | T1   | authentication= X.509 | | 160 | 80 | 0.99 | 0.91 |
| s3          | S2            | T2   | authentication= SAML | Privacy Access control=P3P | 120 | 100 | 0.95 | 0.98 |
| s4          | S2            | T2   | authentication= X.509 | Signature=XML-SIG | 150 | 60 | 0.91 | 0.88 |

b) Select, for each task, a web service with the best quality.
In this step, for each task, select the best web service considering QoS parameters among web services which chosen in preview step by matchmaker for every level of graph.

Definition 1: utility function
In order to evaluate the multi-dimensional quality of a given web service a utility function is used. The function sorts and ranks service candidates by QoS values[19]. The utility computation involves scaling the QoS attributes’ values to allow a uniform measurement of the multi-dimensional service qualities independent of their units and ranges. The scaling process is then followed by a weighting process for representing user priorities and preferences. In the scaling process each QoS attribute value is transformed into a value between 0 and 1, by comparing it with the minimum and maximum possible value according to the available QoS information of service candidates. For instance, maximum QoS attribute value of each web service is defined by maximum value in the service class that the web service belong it.

\[
Q_{\text{max}}(j, k) = \max_{\forall s_j, s_k} q_k(s_j)
\]

\[
Q_{\text{min}}(j, k) = \min_{\forall s_j, s_k} q_k(s_j)
\]

Where \(Q_{\text{max}}(j, k)\) is the minimum value (e.g. minimum price) and \(Q_{\text{max}}(j, k)\) is the maximum value (e.g. maximum price) that can be expected for service class \(S_j\) according to the available information about service candidates of this class.

Then for each candidate web service in service class \(S_j\) execute function (1).

\[
F(S_j) = \sum_{k=1}^{m} \frac{Q_{\text{max}}(k) - q_k(S_j)}{Q_{\text{max}}(k) - Q_{\text{min}}(k)} \times W_k + \sum_{k=\text{max}+2}^{c} \frac{1 - \frac{Q_{\text{max}}(k) - q_k(S_j)}{Q_{\text{max}}(k) - Q_{\text{min}}(k)}}{W_k}
\]

With \(W_k \in R^+\) and \(\sum_{i=1}^{4} W_k = 1\) \(r = 1..4\) being the weight of \(q_k\) to represent user’s priorities. Positive attribute s transform into negative attributes by mines from 1. Thus, the web service with minimum utility value in every service class is selected to be executed.

V. IMPLEMENTATION AND EXPERIMENTS
The broker is implemented on Windows XP platform using Microsoft Visual Studio .NET development environment and Microsoft visual C# as a programming language. To store QoS values of web services, we use Microsoft SQL Server 2000 database. The four QoS and four security properties for eight web services are considered for the experimentation.

The broker receives a set of tasks as a business process contains several tasks and their relationship among them in its database. It has in information about web services that able performed the tasks, their QoS and security constraints and their priorities. After the execution of matchmaking and selection mechanism, the broker returns the best web services to the requestor.

VI. CONCLUSION
In this paper, we have considered the non-functional parameters (i.e. security and QoS attributes) during composing web services. We have proposed an approach to compose web services according to the security requirements of both web service requestors and providers and filtered it by QoS attributes. This paper presents a broker-based framework for dynamic web services composition with security and QoS parameters. The components in a broker implement service composition, adaptation and selection regard to security and QoS issues. We propose an approach to select the best web service among web services candidate to perform task that has suitable QoS values but the broker does not have any solution when a web service failed during composition. For future work we can propose an algorithm to finds another web service as replacement of the web service is chosen for every task in workflow when Occurrence a failure.
REFERENCES


