A QoS Broker Based Architecture for Secure Intelligent 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—A web service is a software system that can be independently deployed and invoked by other software over Internet. To this end and to model the composition, web services by using the set of XML standards, such as WSDL, SOAP and UDDI, some languages (e.g., WSBPEL, BPML) are emerged. For each service in the composition, many service providers can offer the same function but may have different non-functional attributes (i.e. security and QoS parameters). These attributes have to be considered to make a 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 presents the web service selection mechanism which selects the best (most suitable) web service based on the both requester and provider's security issues and QoS characteristics of web services to identify the optimal web service composition.

Key words — Web Services, security, QoS (Quality of Service), Optimization, Service Composition, Broker.

I. INTRODUCTION

The web service is a self-described application that uses standard internet technologies to interact with other web services. The interactions of them can be done through several standardized XML messaging [1]. Web services can be advertised, located and used across the internet using a set of standards such as Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI). The web service performs a specific task or a set of tasks and it can be used alone or with other web services to carry out a complex aggregation or a business process [1]. To this purpose, many emerging languages (e.g., BPEL4WS, WSBPEL and BPML) have been proposed for the orchestration of web services and the description of abstract business processes [2].

With increasing number of web service providers on the web, there are exist many web services having similar functionality and discovering of web services for participating to the composition becomes difficult. Thus, non-functional properties (i.e. security and QoS parameters) of web services are a criterion to distinguish them.

In this paper, we focus on security and QoS issues of web service 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 provider may not want its information to be processed by any Web service that does not adopt P3P to express privacy policies. These constraints must be taken into account when composing web services or a service requester asks a service with maximum reliability. We propose an approach to compose Web services based on the security requirements of related to both Web service requestors and providers, and optimize the composition by considered QoS parameters. In addition, we present a broker-based framework for security intelligent Web services composition and dynamic selection with QoS attributes. 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.

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 QoS values of each service.

The rest of paper is organized as follows. Next section we review some related work. Section 3 discusses security and QoS issues related to web service composition. Section 4 describes QoS Computation of Composite Services. Broker based architecture for web service composition is presented in Section 5. In section 6 the paper presents web service selection mechanism with an illustration. Section 7 draws conclusions.

II. RELATED WORKS

Many XML-based standards have been proposed for workflow specification as a coordinated execution of web services based on WSDL, 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 (BPMI) for modeling the business processes. BPML supports advanced semantics such as
processes and complex compensated transactions that are not addressed by BPEL4WS. BPEL4WS displace Microsoft’s XLANG [3] and IBM’s WSFL (Web Service Flow Language) [4] and 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. Other related standards are given in [5].

BPML is a draft standard proposed by BPMI.org. It provides an approach to modeling business process interactions through messages. BPEL4WS defines a model and a grammar for describing the behavior of a business process based on interactions between the process and its web service interfaces. Each of the activities in a flow model must be executed by an appropriate web service. In this scenario, the role of Selection manager is to assign a suitable web service for each activity. In [2] this assignment process is called “matchmaking”. The matchmaking process is used for using semantic web service descriptions. For example DAML-S [6](DAML-based Web Service Ontology) provides capability to semantically annotate web services to discovery, execution and composition of web services automatically.

The process model of DAML-S is developed separately from that of BPEL4WS and so they are different. It is interested if the two models can be related to each other, because BPEL4WS is widely supported in industry, but it does not have a formal semantic.

Some related work on QoS has been done in the workflow area, such as the METEOR project [7]. Four QoS attributes are defined: time, cost, reliability and fidelity. The project focuses on analyzing and verifying a workflow QoS instead of constructing a QoS workflow.

Other related works are those exploiting AI planning techniques to composite web services. Among them, we recall the work by McIlraith et. al. [8] that extends the logic programming language Golog for automatic composition of web services, and Medjahed [9] proposes a technique to composite web services with high level declarative description. A basic difference between the approach reported above and the one proposed in this paper is that we exploit a syntactic approach for modeling security information of a web service (i.e., the WSDL document).

The closest work to ours is [2], in which authors present a security broker for web services composition. However, their work is focused on security parameters which are specified by web services requestor and provider and must be taken into account when composing web services. A difference between the approach reported in [2] and the one proposed in this paper is that we consider both security and QoS constraints of web services composition but [2] only security issues.

III. NON-FUNCTIONAL MODEL FOR WEB SERVICE COMPOSITION

As well as, the user's request, several non-functional properties may deeply affect on web services compositions that to be satisfied by the web services. In this paper, we consider two classes of the non-functional parameters:

A. Security constraints

In this paper, we consider concept of security constraints of web service that must be satisfied by each atomic web service of composition or final service of composition. We classify these security constraints into two different classes: Security capabilities and Security compatibility. Security capabilities describe the security strategies and techniques specify by requester that must be supported by one or more web services participating to the composition, or only by resulting web service. For example, requester wants to only web services participate in the composition that using SAML authentication. The latter determines by provider which refers to those conditions that a web service cooperate with another web service in the composition. For instance, a web service may not want to accept requests from those web services that do not encrypt input parameters. To verify this constraint, the broker must be known which web services satisfy these constraints. To this propose we store these constraints in WSDL document. In our work, we focus on these security constraints that are specified by provider. For more details refer [2].

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 [10, 11]. These can include generic QoS attributes like response time, availability, service cost, reliability, as long as these attributes can be quantified and represented by real numbers. We use the vector $Q_s = \{q_1(s), \ldots, q_i(s)\}$ to represent the QoS attributes of service $s$, where the function $q_i(s)$ determines the value of the i-th quality attribute of $s$. The values of these QoS attributes can be gotten from service providers (e.g. cost), collected from previous execution monitoring (e.g. response time) or from user feedbacks (e.g. reputation) [12]. The set of QoS attributes can be divided into two subsets: increasing and decreasing QoS attributes. The values of increasing attributes need to be maximized (e.g. Reliability and availability), whereas the values of decreasing attributes need to be minimized (e.g. cost and response time). The following four quality metrics of a web service are considered:

1) Response Time: This is the amount of time between sending a request and receiving its response (i.e. the guaranteed average time required to complete a service request. It also referred to the execution duration and is computed in terms of the processing time and the transmission time. It is sometimes referred to latency.

2) Service Cost: The cost involved in requesting the service which can be estimated by operation or volume of data.
3) **Availability:** This is defined as the ratio of time period in which a web service exists or it is ready for use.
4) **Reliability:** Reliability represents the ratio of successfully returned messages after requested tasks are performed without errors [13].

### C. QoS Computation of Composite Services

The QoS value of a composite service is decided by the QoS values of each individual service as well as the composition model used. There are different ways individual services can be integrated to build a business process. The four basic relationships are (1) sequential; (2) parallel; (3) conditional; and (4) loop. In this paper, we focus on the sequential composition model. Other models may be reduced or transformed to the sequential model.

As the QoS vector of individual service, we assume for a composite service the composition QoS vector that defined as $Q_C = \{q_1(S), \ldots, q_r(S)\}$ where $q_i(C)$ denotes the estimated QoS values of a composite service $C$ and can be computed from the QoS values of individual services which participate in the composition. Table I shows the aggregate function for computing the QoS attributes of a business process that is consisted of $n$ web services sequentially. The reliability and availability aggregate functions are non-linear functions. Thus, to make all aggregate functions to be linear ones, we transform them by using a logarithmic function:

$$\log(q_{rel}(c)) = \log (\prod_{j=1}^{n} q_{rel}(s_j)) = \Sigma_{j=1}^{n} q_{rel}(s_j)$$

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Aggregate Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response time</td>
<td>$q_{res}(c) = \Sigma_{j=1}^{n} q_{res}(s_j)$</td>
</tr>
<tr>
<td>Cost</td>
<td>$q_{cost}(c) = \Sigma_{j=1}^{n} q_{cost}(s_j)$</td>
</tr>
<tr>
<td>Reliability</td>
<td>$q_{rel}(c) = \prod_{j=1}^{n} q_{rel}(s_j)$</td>
</tr>
<tr>
<td>Availability</td>
<td>$q_{av}(c) = \prod_{j=1}^{n} q_{av}(s_j)$</td>
</tr>
</tbody>
</table>

### IV. ARCHITECTURE COMPONENTS AND THEIR OPERATIONS

The broker is defined between Web UUDI and the requester which facilitates the requester and provider to specify security constraints and QoS preferences to select the desired web service and make optimal services composition as shown in Fig. 1. The select operation is defined between broker and the requester to select the best web service based on requester and provider’s security constraints and QoS values attributes. From an architectural perspective, the broker is a middleware which can be implemented as a web service. The broker registers and edits the web service non-functional property values through the Info-registry. The Info-registry provides the facilities to search security, cost, performance and requester’s response sensitive QoS information of web services.

![Broker Architecture](image)

**Figure 1. Broker Architecture**

**Component Interactions:**

1) **Publisher:** The publisher component facilitates the registration, updating and deletion of business process and web service related information. In addition to this, it gets specific QoS property values of web services and security constraints from the providers. The obtained QoS property values are finally verified and certified [20] by the broker before registering them into Info-registry. In addition, publisher estimates the requester’s response specific QoS property values through requester’s feedback on service execution. The requesters will return the QoS property values to the QoS broker when asked by it after the service consumption.

2) **Composer:** Composer takes a user’s service request as input to generate a process plan. Process plan is an abstract process that consisting of service classes and the connection relationships among them (only sequential connection is discussed in this paper). A Service Class is a group of services that provide similar functionalities with different nonfunctional parameters. A process plan defines a flow of service class without identifying the actual service to be invoked. Fig. 2 shows a workflow with three tasks and a process plan related it with three service classes. Composer is responsible for generating process plan. Users create the workflow by selecting which tasks must be done and specifying connection relationships among them. Then discover functionally similar web services from the UUDI through functionality matching [1, 16].
security compatibility and QoS attributes. This is done by the list of web services able to perform it, without considering any similar web services from the UDDI, for each task, we have a component edits the QoS properties of specific web service.

• Step 8. Based on requester’s feedback, the Publisher will send the feedback about the service execution to the provider.

• Step 7. After consuming web service from the provider, the composition with the best quality. Then it executes them.

• Step 6. The selector component now gets required security constraints with the publisher component of broker.

• Step 5. The composer component discovers functionally similar web services associated with task t \( t_1 \) and web services associated with task t \( t_2 \) (i.e., W\(_5\), W\(_3\)). The composition graph represented in Fig. 3 has been built by supposing that W\(_1\) is compatible only with W\(_8\), whereas web service W\(_2\) is compatible only with W\(_5\), and W\(_4\) with W\(_3\) and W\(_5\). Moreover, we have supposed that web service W\(_4\) is compatible with W\(_8\), and W\(_3\) whereas web service W\(_7\) with W\(_8\). Once the composition graph has been completed, the Security Matchmaker is able to determine the corresponding secure web service compositions, which, according to Fig. 3, are: \{W\(_1\), W\(_4\), W\(_6\), W\(_7\), W\(_8\)\}, \{W\(_2\), W\(_3\), W\(_5\), W\(_7\)\}, \{W\(_2\), W\(_4\), W\(_8\)\}, \{W\(_3\), W\(_4\), W\(_5\)\}, \{W\(_5\), W\(_7\), W\(_8\)\}, \{W\(_5\), W\(_8\)\}, \{W\(_6\), W\(_8\)\}.

V. THE WEB SERVICE SELECTION MECHANISM

After the composer component discovers functionally similar web services from the UDDI, for each task, we have a list of web services able to perform it, without considering any security compatibility and QoS attributes. This is done by the selector. Indeed, given the execution plan and web services candidates for each task by composer, the security matchmaker selects web services satisfying the specified security constraints. Then optimize the composition with QoS attributes. The function of selector 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.

The composition graph built by the Security Matchmaker for this sequence is represented in Fig. 3. In particular, the first level of the graph contains all the web services performing task \( t_1 \) (i.e., W\(_1\), W\(_2\) and W\(_3\)). To build the second level of the graph, the Security Matchmaker verifies the compatibility with regards to security among web services associated with task \( t_1 \) and web services associated with task \( t_2 \) (i.e., W\(_4\), W\(_5\)).
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 selected in preview step by matchmaker for every level of graph. Based on user’s request consider one of QoS property from Vs vector. For each level of graph we compute QoS aggregation function for every path. For example in level i execute following function,

\[ q_i(c) = \sum_{k=1}^{i} q(s_k) \]

If the considered attribute is an increasing (decreasing) parameter, we select the path with maximum (minimum) of the aggregation values. For example, suppose response time of composition is important for requester, thus selector component chooses the path of graph composition with minimum response time. The QoS parameters of each atomic service are shown in Table II. For each web service in different path compute aggregate function with response time values. The path with minimum response time value will be chosen which shows the best composition of web services for user’s request. Based this approach the optimal selection is \{W_3, W_5 \text{ and } W_8\} with response time of 330.

<table>
<thead>
<tr>
<th>task</th>
<th>Service class</th>
<th>Service candidate</th>
<th>Response time</th>
<th>Cost (€)</th>
<th>Availability</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_1</td>
<td>S_1</td>
<td>w_1</td>
<td>120</td>
<td>100</td>
<td>0.95</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td>w_2</td>
<td>150</td>
<td>60</td>
<td>0.91</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>w_3</td>
<td>100</td>
<td>50</td>
<td>0.89</td>
<td>0.95</td>
</tr>
<tr>
<td>a_2</td>
<td>S_2</td>
<td>w_4</td>
<td>160</td>
<td>80</td>
<td>0.99</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>w_5</td>
<td>130</td>
<td>90</td>
<td>0.96</td>
<td>0.94</td>
</tr>
<tr>
<td>a_3</td>
<td>S_3</td>
<td>w_6</td>
<td>220</td>
<td>70</td>
<td>0.90</td>
<td>0.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td>w_7</td>
<td>110</td>
<td>60</td>
<td>0.91</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td>w_8</td>
<td>100</td>
<td>50</td>
<td>0.98</td>
<td>0.90</td>
</tr>
</tbody>
</table>

The selection result produced by selector includes: the optimal executable business process that contains a sequence of services that are compatible with regard to security. The selection result is sent back to the service requester. A business process execution engine (such as a BPEL engine) is then initiated in the service requester. Based on the optimal business process generated, the execution engine orchestrates the component services to execute the instance of the business process. At run time, the execution engine monitors the execution at every step, such as response time at each service, network transmission delay, etc. If everything goes well, after successful complete executions, the execution engine reports the actual recorded QoS to publisher in broker. Publisher will put them into the QoS Statistics Table in Info-registry.

VI. AN ILLUSTRATIVE EXAMPLE

In this section, we show an example to clarify how a suitable composition is generated by broker. Assume that a user asks a ‘travel plan’ service which consists of three tasks such as plane, hotel and car reservations. The user’s request has the following security and QoS constraints:

a) All web services participating to the composition must use SAML for authentication.
b) Final service reliability > 0.98

User submits the request to the composer. Composer constructs the Process plan as shown in fig.2. Composer has to find out, for each of tasks, one or more web services able to perform them. Composer can do it by WSDL documents of those web services. Then each web services in Service Class1 is evaluated to whether they satisfy security constraint. This task perform by Selector. It has to remove those web services that do not use SAML to authenticate.

In the second step, the Selector evaluates compatibility constraints. To do that , it extracts compatibility constraints from the WSDL documents of the selected web services. Assume W_2 in Service Class1 cooperate with only web services that use XML-SIG standards as signature mecanism. Selector searches in Service Class2 to find web services that have this capability. This task, for each web services which selected in preview step is performed. Fig. 3 shows a graph that presents the web services and relationship between them. Each relationship between two nodes represents security compatibility of them, each path in the graph denotes a security-aware composition.

Now, Selector computes reliability aggregation function for every path and selects the path that its reliability aggregate value is more than 0.98. For doing this task, Selector uses QoS values that is maintained in the Info-registry.

We used Microsoft Visual Studio .NET development environment and Microsoft visual C# as a programming language to implement the broker on Windows XP platform. We implemented the broker as a Web service with a database that store QoS values. The broker reads QoS values of Web services form the Microsoft SQL Server 2000 database.

After the execution of selection mechanism, the broker returns the best Web service to the requester.

VII. CONCLUSION

In this paper, we consider on web service composition and focusing on security and QoS issues. We have proposed an approach to compose web services according to the security requirements of both web service requestors and providers and optimize the composition by considered QoS parameters. This paper presents a broker-based framework for Security intelligent web services composition and dynamic selection with QoS attributes. In our study, during composing and choosing of optimal path only one of the QoS parameters is considered. We are extending our research to handle more than one QoS attributes.
REFERENCES


