International Journal of Scientific Engineering and Applied Science (IJSEAS) - Volume-1, Issue-2, May 2015
ISSN: 2395-3470
www.ijseas.com

Automation of web services behavior using WS-Policy

Jayashree, K [1], Chithambaramani, R [2]
[1] Rajalakshmi Engineering College, Chennai, India
[2] Rajalakshmi Engineering College, Chennai, India

Abstract

Web service technology has emerged as the bridge between applications on heterogeneous platforms. Service providers publish their service description that contain information on the service interface and includes details of input, output, data types and binding information. Web services are dynamic in nature, properties and requirements will change at runtime and hence it is necessary to verify the behavior of web services at runtime. Runtime monitoring, accounting and verification uses the specified behavior of web services. To specify the web services behavior a tool can be used to generate the web services behavior. A tool is proposed to generate the web services behavior for the service provider using WS-Policy. It provide flexibility in specifying and updating changes in web service behavior

Keywords: Ws-Policy, Web services behavior.

1. Introduction

Web services have brought dramatic changes in IT system architectures and application paradigms. It provides the ability to seamlessly exchange information between business units, customers and partners which is vital for the successful management for organizations (Cavanaugh 2006). Web services use standard protocols like Simple Object Access Protocol (SOAP) (Mitra 2003) for the exchange of information in a decentralized, distributed environment, Web Services Description Language (WSDL) for describing Web services and how to access them and Universal Description Discovery and Integration (UDDI) (Bellwood et al 2004) for service discovery (Gottschalk et al 2002). It also includes business issues such as security, quality of web services and management. The Extended Markup Language (XML) (Bray, 2006) provides a common, platform neutral syntax for defining both web services standards as well as messages exchanged between services, hence supporting interoperability and platform-neutral communications.

The web service architecture (Booth, 2004) defines three component roles: service provider, service requester and service broker. A service provider sends a description of the featured service to a service broker, which publishes information in a service repository. The service broker manages the repository and allows service requesters to find services. A service requester sends a description of the desired service to the service broker, which returns the services matching the request. Then, the requester interacts with the respective service.

Runtime monitoring is a key factor in detecting problems that occur during execution of web service. Monitoring is a necessary component of a runtime validation facility. Monitoring consists of collecting data about a monitored service and checking that the data conform to the expected behavior of that service. Runtime monitoring takes place in parallel with the normal execution process to determine if web services execute correctly at runtime.

Web Service Accounting is the process of collecting web service usage information for the purpose of charging and billing. A comprehensive accounting framework for web services would have to provide and support flexible charging/pricing models. Accounting Management (AM) in general provides for metering, charging, accounting, billing, payment and auditing.

This paper focuses on generating the policies of expressing the specified behavior of web services from the service provider. The rest of the paper is organized as follows. Section 2 presents the related work. Section 3, presents the proposed work with respect to expressing constraints. Section 4, presents the implementation of the proposed work. Section 5 concludes the paper.

2. Related Work

Runtime monitoring is used to intercept the messages between the key players. Message request and replies have to be parsed and matched with the specified behavior. Robinson (2003) presented a methodology for monitoring

requirements expressed using KAOS and temporal logic. Events to be monitored at run time are derived from goals, which represent requirements. Patterns of events observable at run time are then assigned to an agent for monitoring.

Pistore et al. (2004) define a framework for planning the composition and monitoring the execution of BPEL Web services. The monitor component is synthesized by using planning techniques and observes interactions with the external services to detect whether the external partners behave inconsistently with the specified protocols.

Barbon et al. (2006) proposed Run-time Monitoring Specification Language to express temporal, boolean, time-related, and statistic properties of BPEL service compositions. These properties can then be monitored using a monitor engine run in parallel with the BPEL execution engine.

Li et al (2006) have developed a framework for monitoring the runtime interactions of web services. The runtime behavior is monitored and validated against pre-defined interaction constraints expressed using Finite state automata. The framework involves interception of service messages and conformance checking with the service constraints. Baresi et al (2004) describes a methodology for runtime monitoring of composed web services in BPEL using contracts, which are expressed as assertions and constrains. Monitoring rules, governing the degree of runtime checking, are expressed as WS-CoL assertions, in an explicit and external way.

Skene et al. illustrate a model and an analysis technique for reasoning on the monitorability of systems, whose service-level agreements are expressed using SLAng.

The expected behavior is specified formally via algebraic specifications, and the run-time behavior is then checked against them (Bianculli, 2007). The check is performed by using an interpreter of the formal specification, which performs symbolic execution based on term rewriting. The results of the interpreter are compared with the values yielded by the monitored service.
Prior work has primarily proposed about the specifying the excepted behavior using algebraic language, Finite state automata, etc. In this the intended behavior is specified as constraints in the format and content of request and replies expected.

3. Proposed Web Service Behavior using WS-Policy

It has been proposed to specify the behavior of web services from the service provider. The specified behavior during execution of web services can be expressed using constraints. The constraints can be expressed using WS-Policy.

3.1 WS-Policy

WS-Policy is a W3C recommendation that provides a general purpose model to describe and communicate the policies of web services. Policies support standard assertions and provide a simple method to express the capabilities, requirements and characteristics of web services. Web services are frequently modified to suit the changing needs of the users. Policies make it easy to make necessary changes in the specifications of web services and facilitate its dynamic update to the system. WS-Policy (Asir et al 2004), a web service standard, has been used for expressing the specifications of the web services.

A WS-Policy consists of policy alternative and policy assertions. A policy is an unordered collection of zero or more policy alternatives. A policy alternative is an unordered collection of zero or more policy assertions and represents a collection of behaviors or requirements or conditions for an interaction. A policy assertion identifies a domain specific behavior or requirement or condition and it has a QName that identifies its behavior or requirement or condition. The QName of the assertion element is the QName of the policy assertion. A policy assertion may contain assertion parameters and a nested policy.

Policies can be used to define interaction constraints with respect to the message exchanges between the main components of web services. The capabilities and constraints are expressed as a set of specifications using XML syntax. The constraints in specifying the message parameters have been given as XML tags in the policy.

3.2. Expressing Constraints

Constraints can be used to describe the count of parameters and whether the parameters are optional or mandatory. The parameters can be of data type numeric, alphanumeric or date. Constraints can be used to define the data type as well as their range of values.

3.2.1. Constraints with respect to Format and Content of Parameters

Format refers to the construction of the parameter and Content refers to the value of the parameter used in the message exchanges. Parameter can be alphanumeric, numeric, or date. A sample set of constraints used for alphanumeric, numeric data and date is given in Table1.

These constraints can also be used to specify the format of input output parameters used for the execution of web services.

Table 1. Constraints for Message/Data Parameters

S. No	Constraints	Description
1.	Length	Length is used to specify the exact number of characters allowed
2	minLength	minLength is used to specify the minimum number of characters allowed
3	maxLength	maxLength is used to specify the maximum number of characters allowed
4	Enumeration	Enumeration is used to define a list of acceptable values
5	Pattern	Patterns can be used to check for permitted combination of values
6	totalDigit	totalDigit constraint is used to specify the maximum number of digits allowed in numeric parameters. The value given must be greater than zero.
7	minInclusive	minInclusive is used to specify the lower bounds for numeric values of parameters; that is, the value must be greater than or equal to the given value
8	minExclusive	minExclusive is also used to specify the lower bounds for numeric values; however, the value must be only greater than the given value
9	maxInclusive	maxInclusive is used to specify the upper bounds for numeric values of parameters; that is, the value must be less than or equal to this value
10	maxExclusive	MaxExclusive is used to specify the upper bounds for numeric values; however, the value must be only less than the given value
11	fractionDigit	fractionDigit is used to specify the maximum number of decimal places allowed, which may be equal to or greater than zero
12	dateFormat	dateFormat constraint is used to specify a date. It is specified in the following form such as YYYY-MM-DD. YYYY indicates the year MM indicates the Month DD indicates the day

For instance, to bind a particular web service, the user has to provide the user name and the password. It is assumed that user name should be in length of 4 to 8 characters. Use of minLength and maxLength can be used.

Numeric constraints are explained with examples of input and output parameters used for executing a web service. For example, to book travel tickets, the number of tickets required would have to be specified as input parameter. Input constraints include minimum number of tickets and the maximum numbers of tickets have to be reserved. The number of tickets booked has to be at least 1 and cannot exceed 6 for one booking.

Use of numeric constraints fractionDigitis used while booking travel tickets, the total amount payable for the tickets is displayed by the web service application. The amount is a decimal number with 2 decimal places.

3.2.2 Constraints with respect to number and type of parameters

Constraint is used to express the count of parameters used in message exchanges. The constraint is expressed using **maxOccurs** attribute. The use of this constraint is explained for service publication by service provider. The service provider sends a message to service

registry giving details of the service to be published. An example of the constraint in the Number of Parameters as five in publishing a web service that can be present in such a message.

The parameters in the message may be mandatory or optional. To publish a service in the service registry, the provider provides details that include publisher name, service name, business name, Uniform Resource Locator (URL) of the web service WSDL and service description of the specific service. The elements that are considered mandatory include publisher name, service name, business name and URL of the web service WSDL. Service description is optional and can be used by the user to describe their service.

3.2.3 Constraints with respect to relations between parameters

Constraints can also be used to express relations between two parameters used in the message exchanges. For example, if a web service performs division of two numbers, the input given consists of dividend and divisor. Similarly relations can be expressed between input and output parameters. For example, orders for merchandise are booked, delivered and then invoice is raised. The invoice can be raised on the same day or later on. The invoice date, therefore, must be greater than or equal to the order date.

4. Implementation

A tool was developed using JAVA for capturing the constraint information from the service provider for the services which he wants to register. It generates the policy in the prescribed format. The tool provides option to accept the required behaviour of the web services, which were defined as constraints in the execution of web services. The policy tags for the defined web services were automatically generated by the tool. Some sample screen snapshots of the tool are given in Figure 1.

Menu

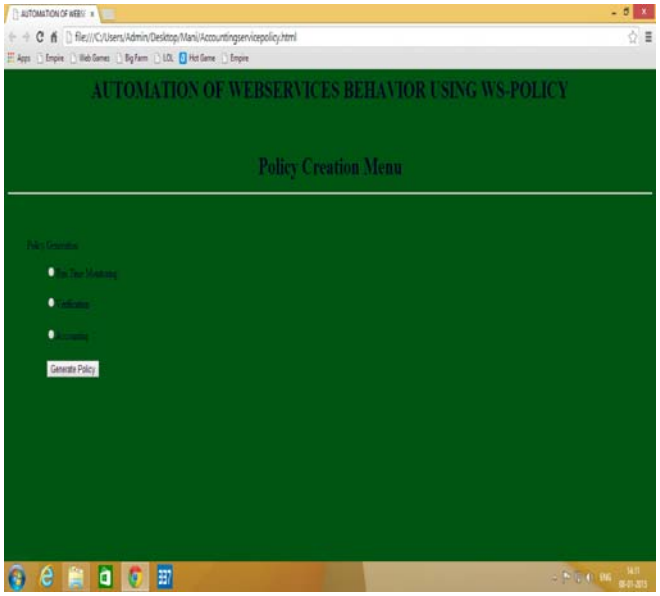

Figure 1 Tool Screen Snapshots

Figure 2 Tool Screen Snapshots

Figure 1 shows the menu containing web service details for runtime monitoring, accounting and verification. The service provider can select anything as of their choice. Figure 2 shows the details that the service provider has to provide the details such as the web service name, service provider name, WSDL URL and the service registry name and the constraints form. The service provider has to choose the constraints for a specific web

service such as minLength, maxLength, length, pattern value, enumeration. The tool was designed to provide option to copy and duplicate policies for web services. For example, if a service provider were to offer similar policies for multiple web services, then the policy can be generated once, and duplicated and modified for other services. As the service offerings generally change frequently, the tool makes it easy to carry out the modifications. The sample policy snippet generated by the tool for the data field "gender" is as shown in Table 2.

Table 2. Sample Policy Snippet for Gender Field

```
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy
" Name="http://TicketBooking">
<wsp:ExactlyOne>
<wsp:All>
<input>
<input>
<xs:element name="input" id="in">
<xs:element name="Gender"  type ="multi-value">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value ="male"/>
<xs:enumeration value ="female"/>
</xs:restriction>
</xs:simpleType>
</element>
</element>
</input>
</input>
</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>
```

A WS-Policy snippet expressing the constraint for "user_name" is given in Table 3.

Table 3. Representation of Input Parameter in WS-Policy

```
<xs:element name="user_name">
<xs:simpleType>
<xs:restriction base="xs:int">
<xs:minLength value="3" />
<xs:maxLength value="8" />
</xs:restriction>
</xs:simpleType>
</xs:element>
```

A sample WS-Policy snippet generated for expressing constraints in the input parameter "No_of_Tickets" is given in Table 4.

Table 4 Representation of Input Parameter in WS-Policy

```
<xs:element name="No_of_Tickets">
<xs:simpleType>
<xs:restriction base="int">
<xs:minInclusive value="1" />
<xs:maxExclusive value="4" />
</xs:restriction>
</xs:simpleType>
</xs:element>
```

5. Conclusion

This paper presents an automatic generation of web services specified behavior policies. The specified behavior during execution of web services has been expressed using constraints. The constraints have been expressed using WS-Policy. Policies facilitate making the necessary changes in the specifications of web services and provide for its dynamic update. The service provider can effectively utilize the tool for the business needs.

References

Barbon, F , Traverso, P, Pistore, M & Trainotti, M 2006, 'Run-Time Monitoring of the Execution of Plans for Web Service Composition', Proceedings of the IEEE International Conference on Web Services, pp. 63-71.

Baresi, L, Ghezzi, C & Guinea, S 2006, 'Towards Self-healing Service Compositions', Contributions to Ubiquitous Computing, vol. 42, pp.27-46.

Bigala, P & Ekabua, O, Implementation of Novel Accounting, Pricing and Charging Models in a Cloud-based Service Provisioning Environment ISBN: 978-0-9891305-3-0 ©2013 SDIWC

Bellwood, T, Clément, & Riegen, C, Available from <http://uddi.org/pubs/uddi-v3.0.1-20031014.htm>.

Bianculli, D & Ghezzi, C, 2007, 'Monitoring Conversational Web Services', Proceeding of the 2nd international workshop on Service oriented software engineering', pp.15-21.

Booth,D, Haas,H, McCabe,F, Newcomer,E, Champion,IM, Ferris,C & Orchard,D, W3C. Web Services Architecture, 2004. http://www.w3.org/TR/ws-arch/.

Bray, T , Paoli,, Sperberg-McQueen, Eve Maler, François Yergeau **W3C. Extensible markup language (xml) 1.0 (fourth edition), 2006. http://www.w3.org/TR/xml/**

Cavanaugh 2006, Handbook of Web services: Benefits, challenges, and a unique, visual development solution. Available from Altova. [pages 1-21, 2006].

Gottschalk, K, Graham, S, Kreger, H & Snell, J 2002, 'Introduction to Web services architecture', IBM Systems Journal, vol. 41, no.2, pp. 170-177.

Mitra, N, Available from: <http://www.w3.org/TR/2003/REC-soap12-part0-20030624/>

Li, Z, Jin, Y & Han, J 2006, ' A Runtime Monitoring and Validation Framework for Web Service Interactions', Proceedings of the 2006 Australian Software Engineering Conference, pp.70-79.

Pistore, M, Barbon, F, Bertoli, P, Shaparau, D & Traverso, P 2004, 'Planning and Monitoring Web Service Composition', Artificial Intelligence: Methodology, Systems, and Applications Lecture Notes in Computer Science, vol. 3192, pp. 106-115.

Robinson, WN 2003, 'Monitoring Web Service Requirements', In Proceedings of 11th International Conference on Requirements Engineering, pp. 65– 74.

Skene,J, Skene,A, Crampton,J & Emmerich W, 2007, The monitorability of service-level agreements for application-service provision. In WOSP '07: Proceedings of the 6th international workshop on Software and performance, pages 3-7.