DOI: http://dx.doi.org/10.26483/ijarcs.v9i2.5668

ISSN No. 0976-5697

Volume 9, No. 2, March-April 2018

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

CONSUMER CENTRIC ENDPOINT COMPUTATION FOR MULTIVERSIONED SOA BASED SYSTEMS

Paul Arokiadass Jerald. M
Assistant Professor,
Department of Computer Science
Periyar Arts College, Cuddalore, India

Vivekanandan. K
Professor, Dept. of Computer Science & Engg
Pondicherry Engineering College
Puducherry, India

Abstract: The tremendous expansion of internet based applications and programs have created a new dimension in the Service Oriented Architecture based systems. Change Management and the need for enhancing the existing systems have resulted in creation of multiple versions in Web Services. With the availability of concurrent multiple versions, it is necessary to provide a suitable method for computing the endpoint which suits the needs of service consumers based on consumers' infrastructure and usage. Since there is no standardized method for finding the endpoint and recommending appropriate version to the consumer, a method for determining the endpoint is proposed.

***Keywords*:** Service Oriented Architecture, Web Service, Version, Endpoint, SOAP, WSDL.

I. INTRODUCTION

Service Oriented Architecture (SOA) is an extension of distributed computing where applications are developed with the components that are loosely coupled. These software components are called "services".

Applications can be created based on these services, and the services can be shared among multiple applications and accessed by consumers. Services are developed, deployed and implemented in web that is accessed by service consumers. Simple Object Access Protocol (SOAP)[1] uses XML language for definition of message architecture and message format. An XML based language called, Web Services Description Language (WSDL)[2] is used to describe operations and interfaces of the web service. HTTP protocol is used for communication due to its wide usage and popularity. Due to the changes that occur because of upgradation of technology, it becomes necessary to create a new version during the life cycle of a service. To indicate a change in service or the methodology in which a service is described, versions are normally used. It becomes necessary to make simultaneous availability of the new version along with the existing versions, since the service consumer cannot be forced to use the new service immediately as and when a new version is introduced.

This paper proposes a methodology to compute the appropriate version that can be provided to the service consumer based on their previous usage statistics. The methodology considers single service with simultaneous availability of multiple versions and multiple services with multiple versions. The main advantage of having multiple versions is that changes and enhancements can be made to individual services and released as new versions without causing an impact on the existing consumer's applications.

The remainder section of this paper is formulated as follows. Section II briefly reviews the various versioning representation and endpoint computation mechanisms. Section III discusses how versions of Web Services are made available and how the details of the versions are incorporated in WSDL descriptions. Section IV proposes with a flow graph and pseudo code on how endpoint can be computed based on service consumer's constraints. Section V discusses the results obtained from the computed values and the versions available to the service consumers. Section VI concludes with the finding and future enhancements to the recommendations.

II. LITERATURE SURVEY

The concept of service versioning has been widely implemented in the IT industry, but not much versioning methods related to SOA have been published. We have discussed here a few versioning mechanisms that were published.

Martjaz B.Juric et al[3] [11]suggested versioning methods that can be defined at the URI level. The suggested method supports both version aware and version unaware system. The advantage of this method is that it can be embedded using WSDL and a version handler which provides control over scope versions, gradual deprecation, retirement of versions can be made available. The limitation of this method is that it is more suitable only to BPEL supported systems. As per this method, the URI is of the form http://www.uni-mb.si/book/1/0 where 1 stands for major version and 0 stands for minor version.

Rainer Weinreich[4] et al recommended a three digit versioning mechanism which is of the form major.minor.micro where major represents incompatible version, minor represents compatible changes and micro represents change of service implementation which does not affect the published interface. This recommendation has a URI of the form http://.../subsytem_U_V1_1/Service_X. The main feature of this recommendation is that it can be implemented at subsystem level so that each subsystem can work at different versions. This version recommendation was implemented on basis of EJB and can be accessed via RMI. The advantage of this model is that it defines a versioning model along with units of versioning, a

versioning schema and a schema for services. It supports asynchronous client updates. The disadvantage of this method is that no long term study was available to check the success of its implementation.

Ghosh[5] et al in his paper on Multi services versioning based on checkpoint has defined Checkpoint to mention a group of versions. Live read transactions are maintained by CMS through checkpoint. Based on older versions object versioning could be implemented. The advantage is that there is no need of universal check point for versioning and it has mechanism for garbage collection. The main drawback of this system was that Checkpoint has to be maintained for every consumer and every service.

Kenneth Laskey[6] defined a service versioning representation at the URI level which was of the form http://a.b.c/services1/20090601/.... The main feature of this versioning representation was that URI is used to identify the resources and it uses date at the end for describing the service number. Here version is done as part of the service description. The advantage of this versioning model is that it is easy to identify when the service was created and since URI is used, the browser will return the latest version. The main drawback is that from Business Perspective, it may not be suitable to associate a date from two years ago with the service and it cannot be used in version unaware systems.

III. SERVICE DESCRIPTION

A. AVAILABILITY OF MULTIPLE VERSIONS

Web Services are the best way to implement SOA services[7]. For concurrent availability of multiple versions of the same service, the concept of service versioning is followed by the service developers. The scenario of having multiple versions for a single service is shown in figure.1. On the other hand, it would not be complex to understand the concept of multiple active versions. Figure 2 shows multiple versions with a view of the services and the dependency with its service consumers. The UDDI registry contains details about the different versions of the same service which can simultaneously exist. The service consumer, based on his software and hardware limitations will be redirected to the optimal version of service as described in the UDDI.

For computing the availability of individual version MTBF (Mean time between failures) and MTTR (Mean time to repair) values are estimated for each version. Once MTBF and MTTR are known, the availability of the individual versions can be calculated as:

$$A = \frac{MTBF}{MTBF + MTTR}$$

The combined versions are operational if any one version is available. From this, it follows that the combined availability is 1 - (both versions are unavailable). The combined availability is shown by the equation below:

$$A = 1 - (1 - A_x)^2$$

B. DESCRIPTION OF VERSION IN WSDL

Whenever a new version is deployed the new service should be described through appropriate schema, WSDL and service changes. This step might include registration to a

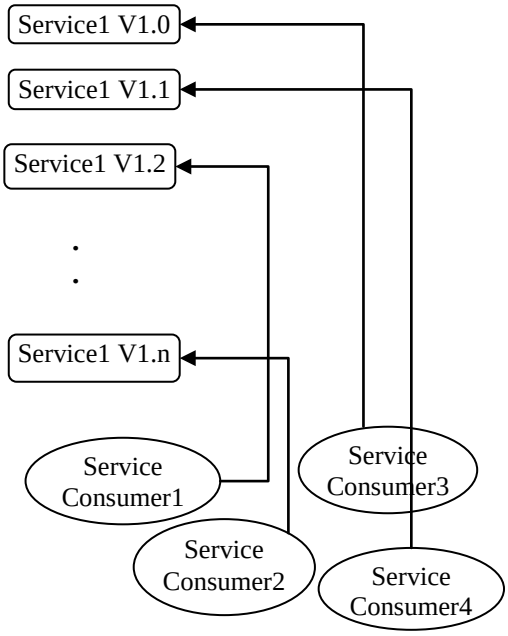

Figure 1. Scenario of Single Service with multiple versions

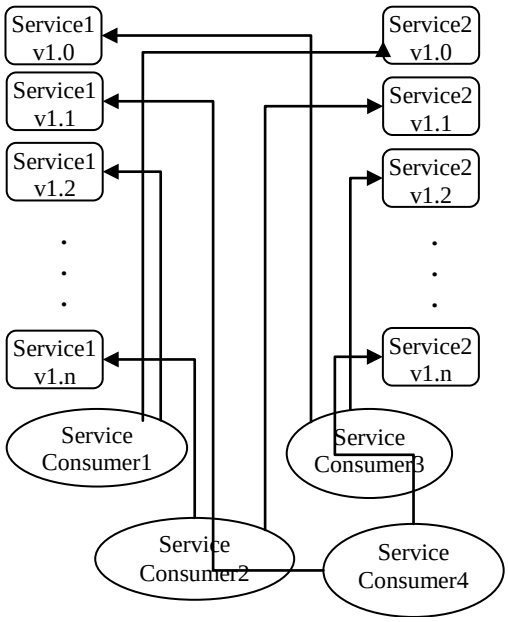

Figure 2. Scenario of Multiple services with multiple versions

UDDI registry or the Web services platform from where the services are searched. WSDL is the W3C standard for describing Web Service and in WSDL, it is necessary to define new messages, port types, and bindings, since the same set of messages/port types/bindings cannot be used to carry messages from two different schemas. In order to add descriptions for new versions in WSDL, it is necessary to define new or extended types in a new namespace, define new port types and bindings and finally add a new service or new ports to an existing service.

Figure 3 shows WSDL description existence of two versions designated by v1 and v2 for the same service are described. The bindings for every version should be separately described, and the SOAP will access the appropriate version of the web service based on the bindings described in WSDL.

```
<wsdl:definitions targetNamespace="service:v2"
xmlns:v1="service:v1"
xmlns:v2="service:v2"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2010/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
<import namespace="service:v1" location="v1_bindings.wsdl"/>
<import namespace="service:v2" location="v2_bindings.wsdl"/>
<wsdl:service name="ServiceManagerService">
<wsdl:port name="ServiceManagerPort" binding="v1:ServiceManagerBinding">
<soap:address location="http://localhost:4002/serviceManager/v1"/>
</wsdl:port>
<wsdl:port name="ServiceManagerPort_v2" binding="v2:ServiceManagerBinding">
<soap:address location="http://localhost:4002/ServiceManager/v2"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Figure 3. WSDL Description for multiple versions

IV. ENDPOINT COMPUTATION

At any given point of time, when a service is accessed by the service consumer, more than one version of the same service will be available. These versions would have been created due to the enhancement of current version in case of minor revision or change in features. In such cases, the liability of provisioning of suitable service to the service consumers could either be done by the service provider or automated suggestion based on the service consumer's previous use of service. [8]emphasizes the need for having a version recommendation with lesser computations.

Apart for having a version recommendation based on smaller computations it is also necessary to consider the service consumer's preference before recommending a version. We suggest a method for computing the endpoint, the version of the service to be provided to the service consumer based on the consumer's preference.

The endpoint consists of the binding information which contains the port information about what version of the service is to be loaded in which port. This information consists of the physical network address and specific URL. We propose a model by which endpoints are computed based on consumer preferred QoS parameters.

A. Flow Graph for Endpoint Computation

The flow graph for the proposed endpoint of the version that is to be recommended to the service consumer is shown in figure 4.

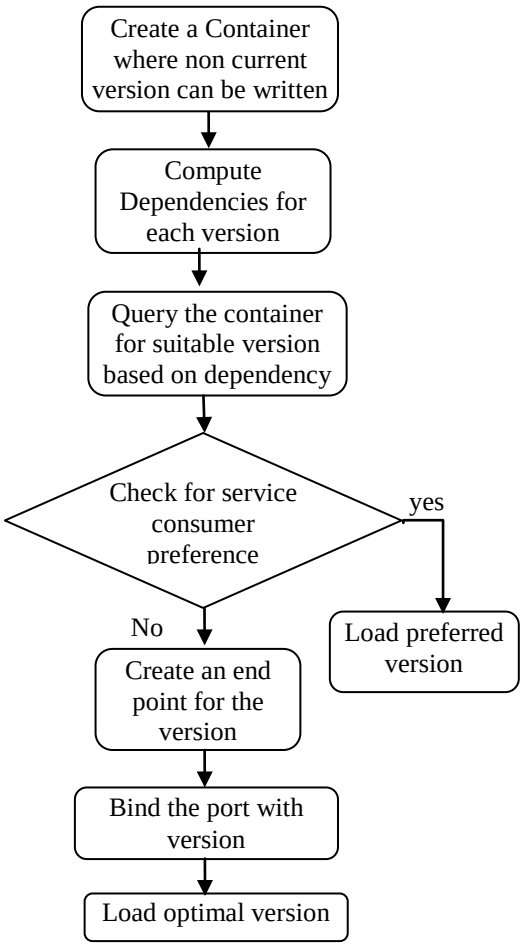

Figure 4. Proposed Endpoint Determination method

B. Algorithm for Endpoint Computation

The algorithm for computing the endpoint for availability of multiple versions is presented. Each service is considered to have multiple versions that are available concurrently. The description about the versions is described in WSDL in the xml file.

Input: xmlns file with version placeholders
xmlns file with v1,v2,v3....for S1, S2,,S3….
Vi represents versions
Si represents Services

Output : Recommendation for Suitable Version

Algorithm :
Step 1: Repeat till no more version is available
Step 2: Read xml version header tags
Step 3: Sort each version based on Consumer ratings
Step 4: Check for compatibility in user settings
Step 5: If Consumer rating does not match with user settings
Compute endpoint
Step 6: If Deprecated version
Suggest for New version/Unload current version
Step 7: If revision in minor version
Load current version
Step 8: If revision in major version
Remove previous version
Load new version

Once the versions are described in WSDL, the core issue would be to transform the messages according to the required version, bind the port and provide routing information. Each version of the service would have a separate schema and the computed endpoint has to be sent to the targeted ports. For example, if a Service has four versions, and the algorithm computes the optimal version for the service to be version2, then version2 has to be bound to the port that passes messages to the service consumer. This message transformation and routing is done whenever a service consumer seeks a service.

V. EXPERIMENTAL SETUP AND RESULTS

The proposed method was tested with similar services which had multiple versions. The version descriptions were made available in the WSDL descriptions. Four such similar services which had multiple versions including deprecated versions were considered for the test. The test focused on multiple services with multiple versions and also single service with multiple version scenarios. It is necessary to choose a tool for testing the service performance.[9] [12]has discussed the various web service testing tools and their performances. Based on the performance tools discussed in[9], Jmeter [10] an open source tool was used to predict the preferred version and the following results were inferred.

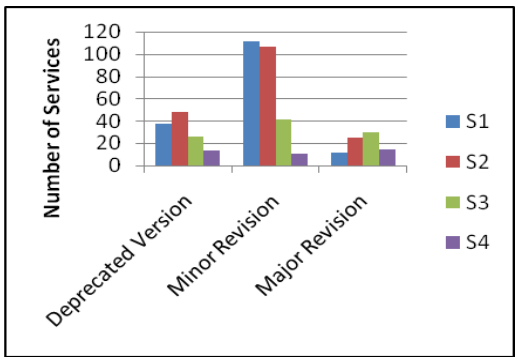

Figure 5 : Types of Version accessed by consumers

Figure 5 shows that the services which had minor revisions were preferred by the service consumers invariably for all the four services, whereas versions having major revisions were having least preference of access by the service consumers.

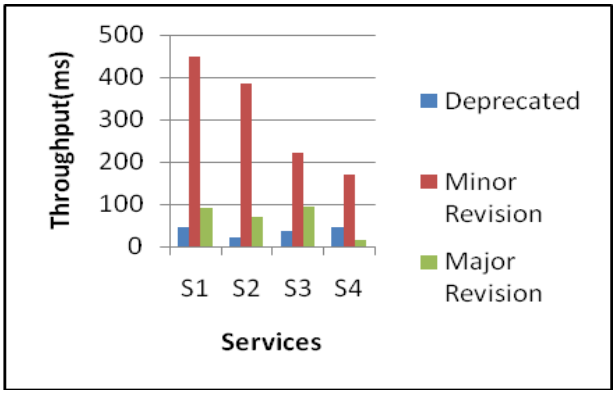

Figure 6: Throughput for different types of versions

Figure 6 shows the throughput for the different services. The throughput for the services which had minor revisions was high. This is due to the availability of the services with minor revisions and the compatibility of hardware and software available to the service consumers fits to the current version of the web service.

Table I. shows that when more than one version of the web services are available in parallel, the availability rate increases.

Table I. Availability of simultaneous versions

Number of Versions	Availability %
Version 1	97%
Version 1 and Version 2 available in parallel	98.3%
Version 1, Version 2 and deprecated version available in parallel	99.7%

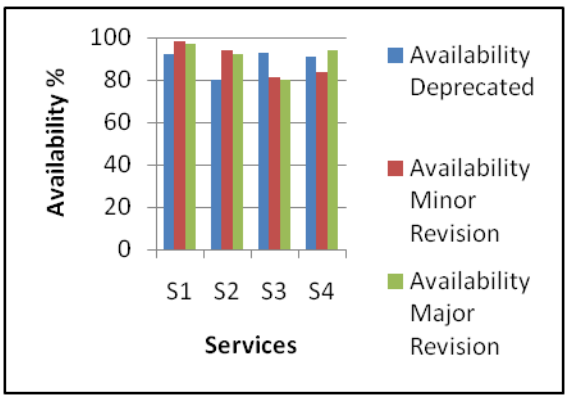

Figure 7: Availability of versions to the consumers

Figure 7 shows the availability of the different versions to the service consumers. It is necessary in a versioned service all the versions of a service minor version, major version and the deprecated version has to be made available to the service consumer. The results imply that the deprecated version will be of lesser preference to the service consumer than minor and major revisions.

VI. CONCLUSION

Change management is inevitable to all software. Web Service being the implementation of Service Oriented Architecture, when revisions to the existing services are done, Versioning of services has to be carried out to all services and it has to be backward compatible so that the service clients can have the flexibility of accessing older versions.

In this paper we have discussed a method to find the endpoint of multiple versions available for web services. We have considered Web Services which offers simultaneous availability of multiple versions. It is necessary that out of all the versions available, the service consumer should be provided with the optimal version based on his previous usage and availability of the versions. We have provided a method to find the appropriate endpoint based on consumer's preference. We have tested the proposed method for similar web services. The results obtained from the tool and the web service accessed by the service consumers' browser were analysed. The proposed method is found to be appropriate and justifies that the presence of multiple versions of a service improves the availability of the service to the service consumers and could be implemented in real-time.

REFERENCES

[1] "W3C - SOAP Version 1.2", http://www.w3.org/TR/soap12-part1/

[2] W3C - Web Services Description Language (WSDL) 1.1",http://www.w3.org/TR/wsdl

[3] M. B. Juric and A.Sasa, "Version Management of BPEL Processes in SOA," *2010 6th World Congress on Services*, Miami, FL, 2010, pp. 146-147.

[4] R. Weinreich, T. Ziebermayr and D. Draheim, "A Versioning Model for Enterprise Services," Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on, Niagara Falls, Ont., 2007, pp. 570-575.

[5] A. Ghosh, R. Chaki, and N. Chaki, "CMS: Checkpoint-based Multi-versioning System for Software Transactional Memory," in Progress in Intelligent Computing Techniques: Theory, Practice, and Applications, ed: Springer, 2018, pp. 471-482.

[6] Kenneth Laskey, 2008, "Considerations for versioning SOA Resources", In IEEE,

[7] Torry Harris Business Solutions, SOA – Service Versioning Best Practice, White paper from http://www.thbs.com/thbs-insights/soa-service-versioning-best-practices

[8] Almalki and H. Shen, "A Lightweight Solution to Version Incompatibility in Service-Oriented Revision Control Systems," in Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference, 2015, pp. 59-63.

[9] Ravi Kumar et al, "A Comparative Study and Analysis of Web Service Testing Tools" International Journal of Computer Science and Mobile Computing, Vol.4 Issue.1, January- 2015, pg. 433-442.

[10] Jmeter tool - https://github.com/apache/jmeter

[11] Matjaz B. Juric; Ana Sasa; Ivan Rozman; (2009): WS-BPEL Extensions for Versioning", Information and Software Technology, 51 pp 1261–1274.

[12] Jyoti Choudrie, Gheorgita Ghinea and Vishanth Weerakkody, "Evaluating Global e-Government Sites: A View using Web Diagnostic Tools", Electronic Journal of e-Government Volume 2 Issue 2 2004 pp 105-114.