International Journal of Engineering Science Invention (IJESI)
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org || PP. 54-58

A Survey of Approaches in Versioning Adapted for Service Oriented Architecture based Systems

Paul Arokiadass Jerald M [1]
[1]*(*Research Scholar, Manonmaniam Sundaranar University, Tirunelveli).*

Abstract: *Service oriented architecture is essentially a collection of services, where these collective services communicate with each other. Services have to be connected with each other based upon their requirements, which involve either simple data passing or involving two or more combined services.*
Change in business conditions and technological advancements forces service consumers to seek and adapt to newer versions. In legacy systems changes were dealt with on an application-by-application basis mostly based on the whole software. Since SOA is based on loosely coupled services, and the nature of the architectural style, adapting to changes is simple compared with legacy systems. As a result, business requirements changes can often be implemented by either changes to existing processes or creation of new enterprise business processes based on the existing services. Versioning is normally used to indicate a change in service or the methodology in which a service is described.
The aim of this paper is to provide an overview of the different versioning methods that are proposed and currently available with respect to the usage in service oriented design, development.
Keywords: *Services, Service Oriented Architecture, Versioning,*

I. Introduction

Service Oriented Architecture (SOA) is an evolution of distributed computing which is designed to allow the interaction of loosely coupled software components, called "services", across a network. Applications are created from a composition of these services, and the services can be shared among multiple applications and by many consumers.

If there is a constant in IT implementation, it is change. As in the case of all software, Versioning is an important aspect of web service development too, particularly in complex mission-critical systems and in service-oriented architectures (SOA). Versioning is important because web services evolve over time (Vinoski, 2004). It is always necessary to maintain more than one version of a single service interface.

In SOA, versioning is also related to the development of reusable services, which is one of the key objectives of SOA. Reusable services are usually not developed in a single phase but they are developed in several iterations. In each iteration, the service is enhanced to be better suited for reuse. Enhancements most likely require changes to the interface and/or to the behavior of the service.

"Versioning assumes simultaneous existence of multiple implementations of the same thing, with every implementation distinguishable and individually addressable. In the case of SOA, service versioning equates to coexistence of multiple versions of the same service, which allows each consumer to use the version that it is designed and tested." [1]

II. Motivation

Change is constant and as all software require maintenance and enhancement, so is the case for Service oriented architected based systems. As in the case of legacy systems it is necessary to maintain versions of the different services that evolve due to change in requirements or error detection. Versioning in SOA has received more attention from the research and vendor communities, because the stability of service interfaces is part of the agreement (formal or informal) between service providers and consumers.
Change Management and versioning becomes vital and important because:

- Deployed service-oriented systems will have to be maintained and evolved with respect to technology up gradation, service requirement, and
- Legacy systems will migrate to SOA environments to make their legacy functionality available to other systems.

The work usually involves the versioning of Services/Contract, and not other components of a service-oriented system. Proper versioning is necessary because, it is necessary to identify an old service and the new service, since a service consumer can still be using a older version, when more consumers would have started using a newly deployed service.

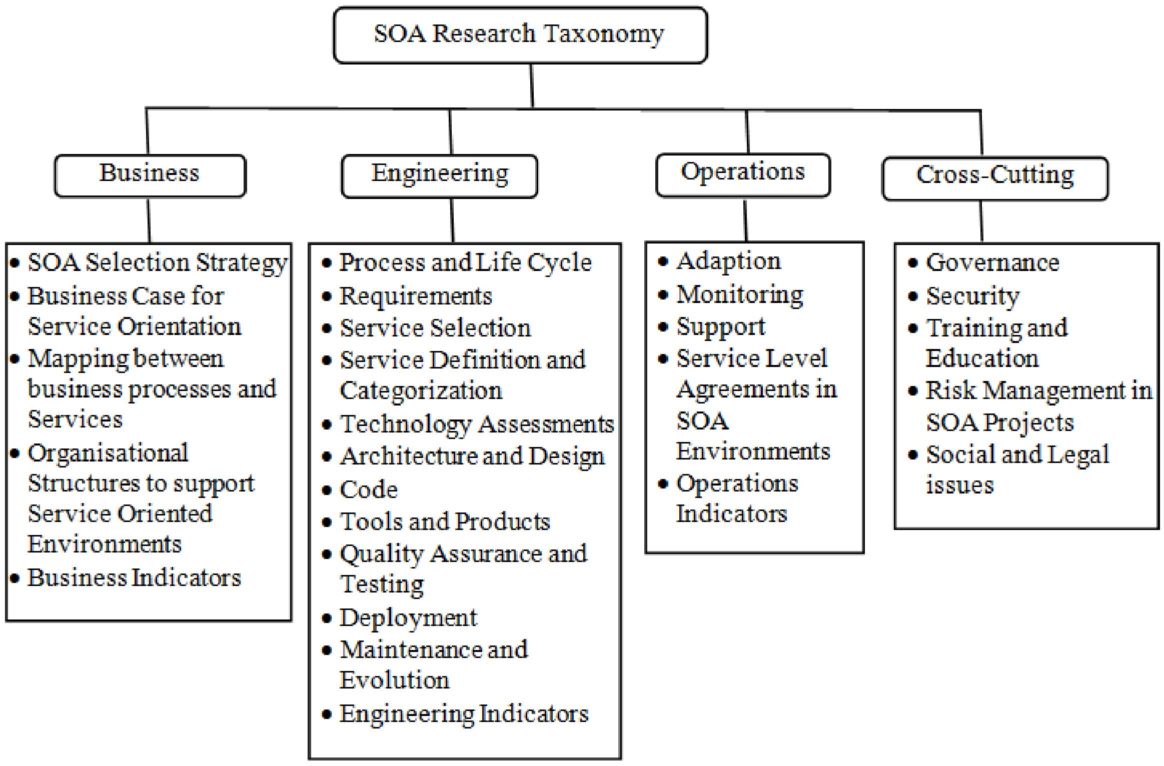

SOA Research Taxonomy
Business
Engineering
Operations
Cross-Cutting
• SOA Selection Strategy
• Business Case for Service Orientation
• Mapping between business processes and Services
• Organisational Structures to support Service Oriented Environments
• Business Indicators
• Process and Life Cycle
• Requirements
• Service Selection
• Service Definition and Categorization
• Technology Assessments
• Architecture and Design
• Code
• Tools and Products
• Quality Assurance and Testing
• Deployment
• Maintenance and Evolution
• Engineering Indicators
• Adaption
• Monitoring
• Support
• Service Level Agreements in SOA Environments
• Operations Indicators
• Governance
• Security
• Training and Education
• Risk Management in SOA Projects
• Social and Legal issues

Figure 1: Research Taxonomy in SOA

III. Soa And The Soa Research Taxonomy

Service-Oriented Architecture (SOA) (Newcomer and Lomow, 2005; Erl, 2009) is an architectural style for developing software applications that use services as their building blocks. Web services (Schmelzer et al., 2002; Josuttis, 2007) are application components which are self-contained, self-describing and are used by other applications. These are both platform and language

Web services extend the World Wide Web infrastructure to provide the means for software to connect to other software applications. Applications, access Web services via ubiquitous Web protocols and data formats such as HTTP, XML, and SOAP, with no need to worry about how each Web service is implemented. Web services combine the best aspects of component-based development and the Web.

As in traditional software development, SOA development has also been prescribed with a standardised development lifecycle. Versioning normally evolves as a result of a need in change or detection of deteriorating Quality of Service. Hence versioning normally falls in the maintenance stage of SOA development. Figure.1 shows the SOA research taxonomy[2], in which the Versioning of service is dealt within the Maintenance and Evolution of the Engineering phase.

The first paragraph under each heading or subheading should be flush left, and subsequent paragraphs should have a five-space indentation. A colon is inserted before an equation is presented, but there is no punctuation following the equation. All equations are numbered and referred to in the text solely by a number enclosed in a round bracket (i.e., (3) reads as "equation 3"). Ensure that any miscellaneous numbering system you use in your paper cannot be confused with a reference [4] or an equation (3) designation.

IV. Versioning Methods Used In Soa

Although versioning is important in SOA, not much versioning methods related to SOA have related with versioning in SOA have been analysed and their features have been listed below:

4.1 [5]**Process level and scope level versioning**

The versioning model used is :

http://www.uni-mb.si/book/1/0

where 1 stands for major version and
0 stands for minor version

The features, advantages and limitations of this versioning model is described in table 1.

4.2 [6] Three digit Versioning
In this versioning model Rainer Weinreich, Thomas Ziebermayr, Dirk Drahein,, have proposed model uses a 3 digit versioning method which is of the form:
major.minor.micro
Example : http://.../subsytem_U_V1_1/Service_X
Major - incompatible version
Minor - compatible changes
Micro - change of service implementation (does not affect published interface)
The features, advantages and limitations of the 3 digit versioning model is described in Table 1.

4.3 Ken Laskey URI Based Approach [7]
The proposed versioning is URI based. The attributes of the version are embedded in the URI which is of the form:
http://a.b.c/services1/20090601/....
The features, advantages and limitations of the URI based model is described in Table 1.

4.4 Compound Version Indentifier [8]
Michael Poulin in his model has proposed a Policy based version control for SOA services.
A Compound Version Identifier (CVI) discriminates between the version visible to the clients and the "assembly" of the versions of individual service's components, interfaces, and elements.
<srv>.<nbc>.<bwc>.<rel>
- srv is an element reflecting the major version of the service as a whole. Changes in this element represent significant changes in the service lifecycle that may not be backward compatible.
- nbc is an element that represents a version state of the major version that is not backward compatible.
- bwc is an element indicating an extension or modification in service functionality. It's strictly backward compatible.
- rel is an element showing little backward-compatible changes like bug fixes.

4.5 [9]**Greg Flurry, 2008, Service versioning in SOA, IBM WebSPhere TechJournal.**
The proposed model in this version is defined as an identifier, abbreviated as “Nx.y”
defined as <major.minor>
- Backward compatible changes cause the <minor> number to increment.
- Backward incompatible changes cause the <major> number to increment and the <minor> number to reset to 0.

4.6 Other relevant work
While implementing or controlling the versioning of web services, it is also necessary to follow the recommendations of standards organizations related to The World Wide Web Consortium (W3C). The W3C provides the beginnings of a semantically annotated WSDL document called SAWSDL, which would give explicit versioning capabilities to WSDL documents [10]. OASIS attempts to address the versioning problem in their WSDM MOWS document, although it is more of an exploratory paper [11]. IBM produced the WSLA specification that offered a way to represent QoS characteristics, but this work was not continued [12].

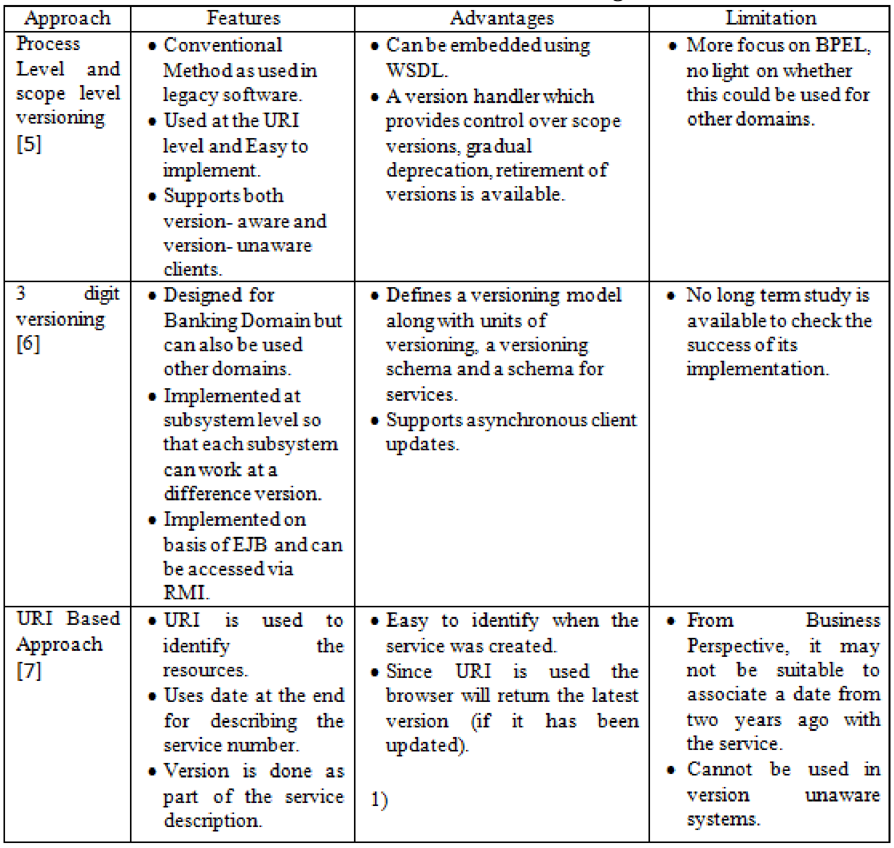
V. Factors To Be Considered During Selection Of Version

Whenever a new version of the service is available it is necessary that the new version has to be registered so as to enable other service seekers to access the new version. It is also necessary that the new version has to be published and adhere to the industry standards. While enforcing it to the service consumer it is necessary to ensure that it has the following properties :

- Should be backward Compatible.
- Should be loosely coupled.
- Should be used both by Version aware and Version unaware systems.
- Version placeholder which automatically redirects to new version to be introduced.
- Enabled with a version history tracker.

Should not be domain specific.

Table 1: Features of available Versioning Method



Approach	Features	Advantages	Limitation
Process Level and scope level versioning [5]	• Conventional Method as used in legacy software. • Used at the URI level and Easy to implement. • Supports both version- aware and version- unaware clients.	• Can be embedded using WSDL. • A version handler which provides control over scope versions, gradual deprecation, retirement of versions is available.	• More focus on BPEL, no light on whether this could be used for other domains.
3 digit versioning [6]	• Designed for Banking Domain but can also be used other domains. • Implemented at subsystem level so that each subsystem can work at a difference version. • Implemented on basis of EJB and can be accessed via RMI.	• Defines a versioning model along with units of versioning, a versioning schema and a schema for services. • Supports asynchronous client updates.	• No long term study is available to check the success of its implementation.
URI Based Approach [7]	• URI is used to identify the resources. • Uses date at the end for describing the service number. • Version is done as part of the service description.	• Easy to identify when the service was created. • Since URI is used the browser will return the latest version (if it has been updated). 1)	• From Business Perspective, it may not be suitable to associate a date from two years ago with the service. • Cannot be used in version unaware systems.

VI. Conclusion

Every service will have to undergo a change and a newer version becomes inevitable. While implementing newer versions, versioning methods are adapted. Every versioning method has a shortcoming and there is always a scope for improvement. Service versioning is a hot topic that has generated a broad range of curiosity from a variety of sources and above all versioning requirement will fall into new documents which extend existing data constructs and message enhancements. In loosely-coupled environments it is not possible to instantly upgrade all service consumers to use the latest version of a service interface. We have to maintain the older versions of the service interface as well.

This paper has compared a few popular versioning methods that were suitable for web service. This study will provide an insight and make a study on the versioning methods and propose a new versioning method which can be implemented on all services. A good versioning method will enhance the Quality of Service such as storage space, bandwidth, elimination of redundancy etc.

References

[1]. Boris Lublinsky "Versioning in SOA". *Microsoft Architect Journal*. msdn2.microsoft.com/en-us/arcjournal/bb491124.aspx

[2]. Grace A. Lewis, Dennis B. Smith, Ned Chapin, Kostas Kontogiannis, 2009, *Proceedings of the 3rd International Workshop on a Research Agenda for Maintenance and Evolution of Service-Oriented Systems (MESOA 2009)*

[3]. Philipp Leitner, Anton Michlmayr, Florian Rosenberg, Scharam Dustdar, 2008, *IEEE International Conference on Services computing.*

[4]. John Evdemon, 2005, Principles of Service Design:Service Versioning. Microsoft Corporation.

[5]. M.B. Juric, A. Sasa, I. Rozman, WS-BPEL Extensions for Versioning, *Information and Software Technology*, Volume 51, Issue 8, 2009, pp. 1261-1274.

[6]. Rainer Weinrich, Thomas Ziebermayr, Dirk Drahein, 2007, A Versioning model for Enterprise Services. *In 21st IEEE Conference on Advanced Information Networking and Applications Workshop.*

[7]. Kenneth Laskey, "Considerations for versioning SOA Resources", In *IEEE, Proceedings of the Enterprise Distributed object computing conference Workshop, 2008.*

[8]. Michael Poulin, 2006, Service Versioning For SOA : Policy-based version control for SOA services, *In SOA World Magazine. http://soa.sys-con.com/node/250503.*

[9]. Greg Flurry, 2008, Service Versioning in SOA, White paper from IBM Software group. http://www.ibm.com/developerworks/websphere/techjournal/0810_col_flurry/0810_col_flurry.html

[10]. World Wide Web Consortium. Semantic Annotations for WSDL and XML Schema. http://www.w3.org/TR/sawsdl (2007).

[11]. OASIS. Web Services Distributed Management: Management of Web Services (WSDM-MOWS) (Draft). http://www.oasis-open.org/committees/download.php/5664/wd-wsdm-mowsversioning (2004).

[12]. Ludwig, H., Keller, A., Dan, A., King, R. P., & Franck, R. Web Service Level Agreement (WSLA) Language Specification. IBM Corporation, 2003. http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.

[13]. P. Ciccarese, S. Soiland-Reyes, K. Belhajjame, A. J. Gray, C. Goble, and T. Clark, "PAV ontology: provenance, authoring and versioning," *Journal of biomedical semantics,* vol. 4, p. 37, 2013.

[14]. J. Mwebaze, D. Boxhoorn, I. Rai, and E. A. Valentijn, "Supporting dynamic pipeline changes using Class-Based Object Versioning in Astro-WISE," *Experimental Astronomy,* vol. 35, pp. 157-186, 2013.

[15]. V. Krishnamurthy, "Versioning Based Dynamic Reconfiguration for SOA Applications," *Research Journal of Applied Sciences, Engineering and Technology,* vol. 9, pp. 926-934, 2015.

[16]. A. Kant and S. Sharma, "Applications of vedic multiplier designs-A review," in *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on*, 2015, pp. 1-6.

[17]. Z. Feng, D. K. Chiu, and K. He, "A service evolution registry with alert-based management," in *Service Science and Innovation (ICSSI), 2013 Fifth International Conference on*, 2013, pp. 123-130.

[18]. W. Zuo, A. N. Benharkat, and Y. Amghar, "Change-centric model for web service evolution," in *Web Services (ICWS), 2014 IEEE International Conference on*, 2014, pp. 712-713.

[19]. K. Chiponga, P. Tarwireyi, and M. O. Adigun, "A version-based transformation proxy for service evolution," in *Adaptive Science & Technology (ICAST), 2014 IEEE 6th International Conference on*, 2014, pp. 1-5.

[20]. S. Sohan, C. Anslow, and F. Maurer, "A case study of web API evolution," in *Services (SERVICES), 2015 IEEE World Congress on*, 2015, pp. 245-252.

[21]. H. Cai and L. Cui, "MultiGranular: An effective Service Composition Infrastructure for Multi-tenant Service Composition," *International Journal of Multimedia and Ubiquitous Engineering,* vol. 9, pp. 171-182, 2014.