Change-centric Model for Web Service Evolution

Wei Zuo, Benharkat Aïcha-Nabila, Youssef Amghar

Change-centric Model for Web Service Evolution

Wei Zuo, Aïcha Nabila Benharkat, Youssef Amghar
Université de Lyon, CNRS INSA-Lyon, LIRIS UMR5205,
20 Avenue Albert Einstein 69621, F-69621 Villeurbanne
Cedex, France
{wei.zuo, nabila.benharkat, youssef.amghar} @insa-lyon.fr

***Abstract*—Web service is subject to frequent changes during its lifecycle. Web service evolution is a widely discussed topic. Many related problems have also been generated from Web service evolution such as Web service adaptation, Web service versioning and Web service change management. To treat with these issues efficiently, a complete evolution model for Web service should be built. In this paper, we introduce our change-centric model for Web service evolution and how we use it to design, execute, and adapt to the changes during Web service evolution.**

***Keywords*—Web service; evolution; version; changes; adaptation**

I. INTRODUCTION

Web service is subject to frequent changes during its lifecycle through versions. Web service evolution also generates several related problems such as Web service versioning [1], Web service change management, and Web service adaptation [3] from the different perspectives of stakeholders. In this work, we address these problems with a set of contributions such as the model, the framework, and the strategies on Web service evolution. Compared to our approach, current approaches of Web Service adaptation in [1, 3] do not deal with the ambiguity of Web Service changes from different stakeholders, current approach of programming Web Service evolution in [4] lacks of event mechanism to propagate changes. The goal of our work is to implement automatically Web service evolution, client adaptation, and Web service self-adaptation [2]. We propose a Web service evolution model that describes why, how, and when the Web service evolve, a Web service evolution programming model that provides the simple interface for Web service provider to design the Web service evolution [4], and a Web service adaptation strategy that manages Web service self-adaptation and client adaptation automatically to reduce the cost of human involvement.

II. CHANGE-CENTRIC MODEL

Current languages for Web service description such as WSDL and OWL-S for Web service are not designed to deal with the evolution. Consequently, different stakeholders can have different understanding of Web Service changes, which may cause errors when they react to the changes. Thus, a model for Web service evolution is needed. We introduce our change-centric model which overcomes this problem.

A. Web service information model

Firstly, we extract the Web service information model which is suitable for research on Web service evolution. In the service-oriented systems, several stakeholders roles are interested in Web service evolution. They are Web service provider, broker, and consumer. The Web service provider designs, develops, and publishes versions of Web services. Therefore, the related information about Web service evolution interested by the provider includes: 1) Web service description which presents the interface, semantics, and QoS of Web service for information exchange over network, and 2) underlying implementation which contains the byte code depending on platforms which is binded to Web service.

B. Change-centric model

Secondly, we introduce the change-centric model which describes the behaviors of the stakholders in Web service evolution.

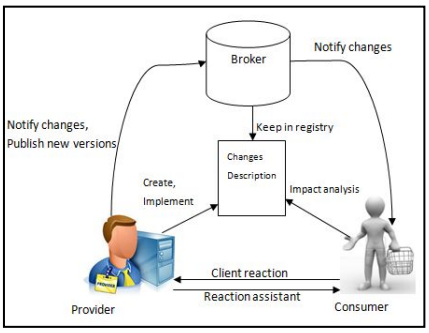

Fig. 1. Change-centric model

As shown in Fig. 1, in our model, the changes description is a set of formatted and standardized information which contains all the changing actions and changed elements from a version to its next version. The provider designs, executes and publishes the changes description. The consumer uses the changes description to analyze the impacts and adapt to. The broker maintains the registry which integrates the Web services from different providers with the changes descriptions and notifies the consumers who are interested in them. An example of representing Web Service changes is shown in Fig. 2. It described how to represent the changes "adding an operation named *ExampleOperation* when evolve the Web Service from version 1.0 to version 1.1".

```
<versionId="1.1" previousVersion="1.0" majorVersion="0"/>
    <change:interface type="add">
        <addSequence>
            <wsdl:operation name="ExampleOperation">
                <wsdl:input message="ExampleInput"/>
                <wsdl:output message="ExampleOutput"/>
            </wsdl:operation>
        </addSequence>
    </change>
```

Fig. 2. Example of representing Web Service changes

Changes description is not only used for presenting what has been changed. It is also used for Web service description independently as in Fig 3. We specify the major version as the one that contains complete changes in business process such as V1.0 and V2.0, and the minor version as the one that contains fixing modifications such as V1.1 and V1.2. Instead of complete Web service evolution, the minor versions only contain Web service change descriptions.

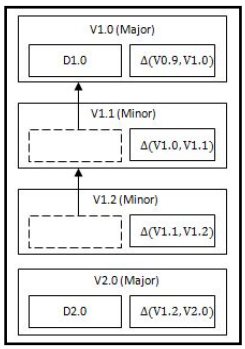

Fig. 3. Changes description in Web service description

III. PROGRAMMING IN CHANGE-CENTRIC MODEL

To provide the methodology for Web service evolution execution and support client adaptation, we also design the system architecture as shown in Fig. 4 to facilitate Web Service programming. The target of the system includes:

- Providing scripting APIs for Web service developers to modify Web service at runtime and design changes descriptions directly (changes description in Fig 3).
- Executing and publishing designed Web service changes descriptions at runtime automatically.
- Adapting the client applications to Web service evolution automatically (consumer in Fig 3).
- Providing the ability of self-evolution for Web services.

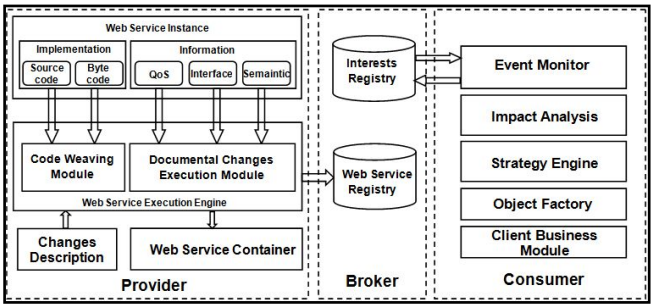

Fig. 4. System architecture under the programming model

A. Evolution execution

As shown in Fig. 4, the changes description accomplish as input of Web service execution engine. The execution engine generates the full instance of Web Service including implementation (underlying code) and description (WSDL or OWL-S). It also publishes the new instance to the Web Service container and to the registry of the broker. The consumer uses event monitor to obtain the changes. Then he analyzes the impacts and adapts to the changes basing on strategy engine and object factory.

B. Client adaptation

Client adaptation requires the client applications to adopt a loose-coupled design pattern which separates the client business module from the Web service invocation module as depicted in Fig 5. We use strategy engine to provide adaptation strategies. And the object factory will generate dynamically the proxy which refers to a new version of Web Service.

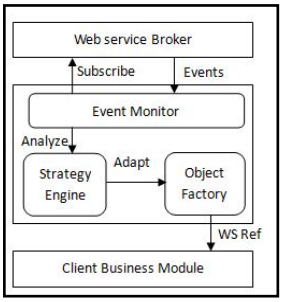

Fig. 5. Client adaptation

CONCLUSION

This paper presents our methodology to treat with Web service evolution in a change-centric way. We firstly build the change-centric model to describe the Web service evolution. Secondly we show our programming model which brings solutions to how to design, execute, and adapt to the changes during Web service evolution.

In future, there still exist several challenges that are not faced yet and should be taken into account such as:

- Web service execution is facing the resolution of dependency challenge.
- Impact analysis for the client lacks deep exploration with composed Web services.
- Web service self-adaptation needs more patterns.

REFERENCES

[1] Kaminski P, Müller H, Litoiu M. A design for adaptive web service evolution[C]//Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems. ACM, 2006: 86-92.

[2] Kajko-Mattsson M, Lewis G A, Smith D B. A framework for roles for development, evolution and maintenance of soa-based systems[C]//Systems Development in SOA Environments, 2007. SDSOA'07: ICSE Workshops 2007. International Workshop on. IEEE, 2007: 7-7.

[3] Fokaefs M, Stroulia E. WSDarwin: automatic web service client adaptation[C]//CASCON. 2012: 176-191.

[4] Treiber M, Juszczyk L, Schall D, et al. Programming evolvable web services[C]//Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems. ACM, 2010: 43-49.