Composition and Management System of Long Term Web Services

[1]B Manoj Kumar, [2]G Senthil Kumar
[1]M.Tech Student, [2]Assistant Professor
[1,2]Dept of Software Engineering, SRM University , Chennai, India
[1]manu.kumar2k12@gmail.com, [2]senthilkumar.g@ktr.srmuniv.ac.in

***Abstract*— Web services are gaining momentum in academia and industry as they hold the promise of developing loosely coupled business processes. This momentum is witnessed from the widespread adoption of Web services in multiple research projects and application domains. LCS is a dynamic collaboration between autonomous web services that collectively provide a value-added service .change reaction process will go through a set of steps, such as discovering Web services based on the requirement, and regenerating a collaboration among outsourced Web services. A short-term has a very limited lifetime. Once the goal is reached, the collaboration among its component services is then dissolved. A framework is presented to detect and react to the exceptional changes that can be raised inside workflow driven Web application. First, the provisioning of Web services drastically reduces the capital required to start a business. Web services are readily available for integration and orchestration. LCS enable a wide-integration of business entities for the business organization. Web services are having some different features like: global availabilities and standardization.**

Keywords –UDDI, SOC, web Service, Ontology, Change Management

1. INTRODUCTION

The last few years have witnessed an explosion of activity around Web services[1] . Web services are distinguished from other traditional applications by two major features: global availabilities and standardization. First, Web services take advantages of the powerful communication paradigm of the Web to provide global availabilities. Second, they are built upon XML[11]-based standards as a vehicle for exchanging messages across heterogeneous Web applications. The change optimization component selects the Web services that participate in a LCS to ensure that the change has been reacted to in the best way.

With the introduction of Service-Oriented Computing (SOC)[2], business organizations have been encouraged to deliver their functionalities via Web services. The functionalities can be programmatically accessed and manipulated over the Web, a global platform, in a standard way. Furthermore, this enables a collaboration among different organizations in a way that is larger scaled, more loosely coupled, distributed, and flexible than the traditional distributed applications.

The change management engine focuses on managing top–down changes in LCSs. The first role of the change management engine is to manage a change in an LCS orchestration (e.g., replacing a Web service, deleting a Web service, and adding a Web service) that may or may not be the result in a change in a LCS schema. This aspect of the framework is presented in detail in. The output of this process is always a new LCS instance[4] but may also include a new LCS schema. The new LCS schema[6] specifies the new functionality provided by the LCS after reacting to the change. In this work, we focus on optimizing the replacement and addition of Web services that results in new LCS instances. The second role of the change management engine is interfacing with the optimizer for selecting the appropriate Web services. The optimizer will be invoked only when the decision of the change management engine is to select one or more Web services to fulfill a functionality with desirable quality.

2. RELATED WORK

Change may occur to an individual service without consent of the enterprises that utilize the service. The changes may affect the invocation of other services in the SOE[3], which affects the entire performance of the SOE[3]. To manage these bottom-up changes, it first presents a taxonomy that classifies bottom-up changes into categories. Changes are distinguished between service level and business level: triggering changes that occurs at the service level and reactive changes that occur at the business level in response to the triggering changes.

For example, the free user navigation through Web pages may result in the wrong invocation of the expired link, or double-click the link when only one click is respected. The semantic exceptions are driven by unsuccessful logical outcome of activities execution. The exceptions that are present in system are driven by the malfunctioning based on workflow-based Web application[8], such as network failures and system breakdowns.

Adaptive Workflow System

Workflows are the popular means of composing enterprises. They provide the ability to execute business processes that span multiple organizations .Traditional workflows do not provide methods for dynamic change management. Workflows are geared towards static integration of components. Furthermore, workflows do not cater for the behavioral aspects of Web services[7]. For example, they do not distinguish between the internal and external processes of a Web service . Workflow management systems (WMS) aim at coordinating activities between different business processes (systems) so they can run in an efficient manner. This is done by automating the business processes and invoking appropriate resources in a sequence.

Change Management in Traditional Databases

Several change detection algorithms have been proposed to measure changes in Web content and also XML[11] documents. Most of the algorithms that deal with changes in Web content aim to maintain the consistency of data on the Web site. They do provide mechanisms to "understand" what the data represents and react appropriately. Research on detecting changes to XML documents has been mainly focused on the syntactic changes to a document. For example, provides an efficient algorithm that detects structural change to an XML document. It does not efficiently detect semantic changes to XML[11] documents such as WSDL[9] descriptions. Furthermore, deals with detecting changes after a move operation.

This data is primarily nested and is not represented by a unique identifier. It proposes techniques for comparison between two structured data versions. However, these techniques do not deal with the Web service environment. Therefore, change detection is only at the abstract layer and cannot be readily applied to Web services.

3. PROPOSED METHODOLOGY

The framework aims to identify the key components to introduce, model, and manage changes to ensure that the changes can be dealt with in a proper way. The bottom of the framework is a Web service space, which consists of the actual service providers in a certain domain, a.k.a., service instances. LCSs are on top of the service space. They outsource their functionalities from services in the service space. Above LCSs is a formal model, which is expected to provide the sufficient semantic support for the change management. Based on the formal model, change model is used to declaratively specify top-down changes that might occur in LCSs. The change management component modifies a LCS to react to the changes.

Change Enactment

Change enactment is to implement the changes defined by change models. It consists of two subcomponents: change reaction and change verification. Change reaction is the process of modifying a LCS to make its features conform to the requirement of the change. The features of a LCS include its functionality, performance, and context. They are determined by the services that a LCS outsources and the way that the services cooperate with each other. Therefore, during the process of change reaction, a LCS may change the partnership of a service, i.e., starting or stopping a partnership. A LCS schema [4]gives a high-level overview of a LCS which facilitates a strategic and operational analysis on how to react to changes. Change reaction should be performed in a top-down fashion. More specifically, the modification should be started at the schema level and then propagated to the service instance level.

Change verification is the process of checking whether a LCS schema still defines a correct configuration once it has been modified after the phase of change reaction. For example, if a service is added to a LCS, it needs to ensure that the service can be invoked properly. If not, the LCS will be considered as having an incorrect configuration. Once an error is detected, the incorrect configuration will be fixed by modifying the LCS schema. To do this, the correctness criteria of a LCS's configuration should be defined. The correctness criteria will then be used as a guidance for change verification.

Change Optimization

Change optimization is to optimize the result of change management. This phase focuses on the change that requires efficient selection and replacement of outsourced Web services. A change may trigger the modification of a LCS's functionality, performance, context, or all of them. It may then trigger the change of a LCS's instance[4] (i.e., re-orchestration of Web services). Thus, a LCS may add new services to fulfill a new functionality, deliver the desired performance, or replacing the existing service. Since the Web services are in demand, the process of modifying, that is new services are selected from the service space needs to be optimized, so that the "best" service from a huge amount of competitive services will be selected.

To conform to the criterion of QWS[10], Web service providers may make the promises about the QWS[10] they offer. The information about the QWS[10] of the services can be used for efficient service selection from list of services. Considering that Web services are provided by independent service providers by UDDI[9] registry, highly volatile (i.e., low reliability), and a priori unknown (i.e., new or no prior history), Web service providers may fail partially or fully to deliver the promises they make on QWS[10] they offer. The trustworthy services should be recognized and selected. Therefore, change optimization is taken out by two phases. During the first phase, the services that have bad reputation will be filtered out. The remaining services are all considered as trustworthy services. During the second phase, the ones that provide the best QWS[10] will be selected from the trustworthy services.

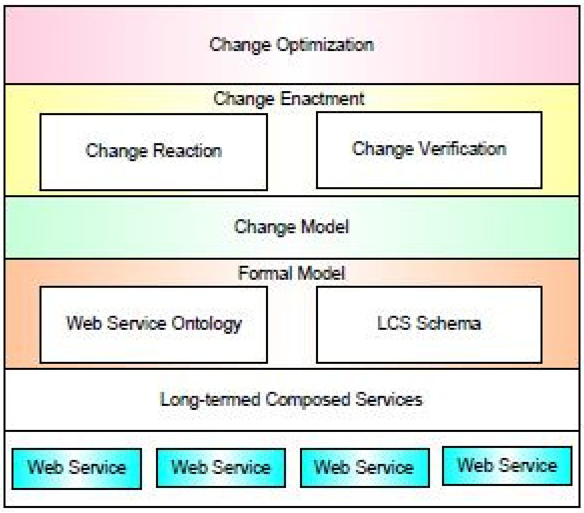

Fig 1 The change management framework

The underpinning of the proposed LCS architecture is a standard Service Oriented Architecture (SOA)[12]. The service providers use WSDL to publish Web services. The WSDL[9] description specifies the address and the arguments to invoke the service. Web service registries, such as UDDI[9], can be used as a directory for a LCS to look for Web services. After locating a Web service, SOAP[9] messages are exchanged between a LCS and the service providers for invoking the service

The Formal Model

The formal model is expected to provide the grounding semantics[4] to support change management. It includes a LCS schema and a Web service ontology[5]. A LCS schema gives a high-level overview of a LCS, such as the functionality it provides and the performance of the web services. It is used to capture the important information about a LCS, such as what the service it offers, how it works, the environment on which it works with, and how well the service will performs. It then helps software agents understand what a change is intended to make to a LCS and how to modify the LCS to react to the change.

Change Model

A change model is used to specify top-down changes. A change is always associated with a specific goal on a LCS. During the process of change management, the LCS will be managed and updated to reach goal ultimately. To ensure that a change is feasible, it is important that a change specification contains all the necessary information. To improve the automation of change management, it is important that the software agents understand what change is intended to make to a LCS. Therefore, a change specification should contain machine-understandable semantics, such as predefined keywords and logic-base expressions.

4. CONCLUSION

Web services are gaining momentum as a new computing paradigm for delivering business functionalities on the Web. They are increasingly regarded as the most promising backbone technology that enables the modeling and deployment of the Service-Oriented Architecture. Many service providers expose to move their business functionalities on the Web using Web services. This, in turn, has opened the opportunities for composing autonomous services on demand. Meanwhile, it has raised the research issues of managing changes during the lifetime of composed services.

We present a process of change enactment to implement the changes specified in terms of the proposed SCML language. Changes are reacted to at two levels: schema-level and instance level. During the schema-level change reaction, a LCS schema is modified to fulfill the functional requirement associated with a change. During the instance-level change reaction, a new LCS instance is generated by selecting and orchestrating Web services that both fulfill functional and non-functional requirement associated with a change. We also propose a process of verifying changes. We first check the correctness of a LCS schema from the semantic perspective. It is for the purpose of ensuring that the services in a LCS can be invoked gracefully. We then check the correctness of a LCS schema from the structural perspective. It is for the purpose of ensuring that there is no isolated services in a LCS.

REFERENCES

[1] "Web Service List" http://www.webservicelist.com/.

[2] Shan, T.C. ; Hua, W.W, Service-Oriented Computing Kit. In IEEE International conference on services computing,2006, SCC '06.

[3] Xumin Liu and Athman Bouguettaya. Managing top-down changes in service-oriented enterprises. In IEEE International Conference on Web Services 2007, UTAH, USA, 2007.

[4] Uminliu , Chunmei,Liu,Rege, M., Bouguettaya, A. Semantic support for adaptive long term composed services.In proceedings of: Web Services (ICWS),2010IEEE International Conference on Digital Object Identifier: 10.1109/ICWS.2010.34Publication Year: 2010.

[5] udakrpinar,I. ; AlemanMeza,B. ; Zhang,R.; Maduko,A. Ontology driven Web services composition platform, In proceedings of:e-Commerce Technology, 2004. CEC2004.IEEE International conference on digital object identifier : 10.1109/ICECT.2004.1319728 Publication Year:2004.
[6] Xumin Liu ; Bouguettaya,A. ; Wu,J. ; LiZhou,EvLCS:A System for the Evolution of Long Term composed services. Proceedingsof:Services Computing,IEEETransactionsonVolume:6, Issue:1DigitalObjectIdentifier: 10.1109/TSC.2011.40 Publication Year: 2013.
[7] Akram MS, Medjahed B, Bouguettaya A (2003) Supporting dynamic changes in web service environments. In: First international conference on service oriented computing, Trento, Italy.
[8] M. Brambilla, S. Ceri, S. Comai, and C. Tziviskou. Exception handling in workflow-driven web applications. In WWW '05: Proceedings of the 14th international conference on World Wide Web, New York, NY, USA, 2005. ACM Press.
[9] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI, IEEE Internet Computing, vol. 6, no. 2, Mar./Apr. 2002.
[10] M. Conti, M. Kumar, S.K. Das, and B.A. Shirazi, "Quality of Service Issues in Internet Web Services," IEEE Trans. Computers, vol. 51, no. 6, June 2002.
[11] W3C. Extensible Markup Language (XML). http://www.w3.org/XML, 2003.
[12] Service-Oriented Architecture (SOA) (2003) http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html