Dycsr: Dynamic Composition of SOAP Services and restful Services in e-Governance Applications

AKSHAYA G
PG Scholar
SSN College of Engineering
Chennai, India
akshaya1102@cse.ssn.edu.in

KANCHANA R
Assistant Professor
SSN College of Engineering
Chennai, India
rkanch@ssn.edu.in

CHITRA BABU
Professor
SSN College of Engineering
Chennai, India
chitra@ssn.edu.in

ABSTRACT
Composition of web services is a prominent feature of Service Oriented Architecture, for implementing business processes. Business processes may collaborate within or across organizations. During cross organizational collaborations, services are often heterogeneous thereby bringing challenges to composition. The two most popular types of web services are SOAP based web services and RESTful web services. Most of the existing solutions for service composition address SOAP based web services and RESTful web services separately. A few prior attempts to compose SOAP and RESTful web services together use semi automatic methods like converters which are inefficient. In this paper, "*DyCSR*" a dynamic composition technique for composing SOAP and RESTful web services has been proposed. The technique involves a middleware service that enables composition of SOAP and RESTful web services dynamically.

General Terms
Service, Composition, e-Governance

Keywords
Dynamic composition, SOAP, RESTful, Web Service, Service Composition, template.

1. INTRODUCTION

The two widely used types of web services are Simple Object Access Protocol (SOAP) [1] based web services and REpresentational State Transfer (REST) [2] web services. In SOAP based web services, web service interfaces are exposed through Web Service Description Language (WSDL) [3] and exchange message in SOAP format. SOAP based web services are suitable for applications requiring security and transaction due to the support of WS-* standards. They are also suitable when a transport protocol other than HTTP has to be used. The RESTful style of web service is a lightweight one as opposed to the heavyweight SOAP based web services. RESTful web services use existing standards such as Hypertext Transfer Protocol (HTTP) and Uniform Resource Identifier (URI) which make them simple and effective. However, RESTful services lack in standards for transaction and security. Both SOAP and RESTful services have their own advantages and disadvantages [4]. In case of RESTful web services, there is no necessity to register the service description in a service registry since the URI of the service is provided directly to the user. Hence, RESTful web services are more suitable for services with a single provider. Services provided by Government departments like Registration of Vehicle, Land etc., are examples of RESTful web services. On the other hand, SOAP based web services are mainly preferred for private enterprise applications where multiple providers exist for each service. Hence, SOAP based web services are deployed in the service registry and they are selected based on their functional and non functional capabilities.

There is an increased growth of e-Governance applications which require handling the transactions across various departments. The heterogeneity and loose coupling that characterizes the Service Oriented Architecture (SOA) make it the most preferred paradigm for e-Governance applications. For instance, a business process of applying for license must deal with several departments such as police, birth registration, banks, etc. Such applications require Government departments to collaborate with private departments. In several such scenarios of e-Governance applications, there is a high probability that different departments provide different types of web services which may result in a heterogeneous environment.

Composition of web services can be done in a static or dynamic way. In static composition, the concrete workflow is built during design time. Static composition can be performed only if the partners involved in composition are predetermined and are not

likely to change during execution. Significant number of solutions [5, 6, 7] have been proposed for dynamic composition of SOAP based web services. In dynamic composition, the abstract workflow is created during design time and the concrete workflow is created during runtime. A template based composition approach [8] is suitable for business processes where the workflow can be predetermined.

Most major industrial players use Business Process Execution Language (BPEL) standard [9] to compose SOAP based web services. With BPEL, process modeling can be done in a simpler and standard way. The web services involved in the composition are added as partner links to the BPEL process. The WSDL version 2.0 offers better support for description of RESTful web services by accepting binding to all the HTTP request methods. However, the latest version of BPEL only supports WSDL 1.1 and hence cannot accommodate RESTful services as partner links. Pautasso et al. proposed an approach named "BPEL for REST" [10] that extends WS-BPEL process modeling language to support the composition of RESTful Web services. A RESTful Web service API was implemented using "BPEL for REST" with the proposed declarative constructs for publishing resources. The interaction primitives (GET, POST, PUT, and DELETE) of RESTful web services can be directly used within a BPEL process as new service invocation activities. However, the approach does not address dynamic selection and composition of web services. Hence, a dynamic composition approach namely *DyCSR* has been proposed in this paper that composes SOAP as well as RESTful services at runtime.

The paper is organized as follows. Section 2 addresses the existing works. Section 3 describes the drawbacks of existing system and motivation for research in this paper. Section 4 describes the proposed approach and section 5 discusses a prototype application suitable for depicting the *DyCSR* approach.

2. EXISTING WORK

Peng et al. [11] proposed REST2SOAP framework to integrate SOAP based web services and RESTful services. A RESTful service is converted into a SOAP based web service semi automatically so that it can be integrated with other SOAP based web services to construct a BPEL composite service. REST2SOAP framework uses Web Application Description Language (WADL) [12] and wraps RESTful service into a SOAP service.

Another hybrid orchestration approach [13], for integrating SOAP and RESTful web services involves BPEL engine as well as a RESTful orchestration engine. A workflow process is divided into sub-workflows according to their type of web services. Each sub-workflow is passed to the respective orchestration engine so that SOAP based orchestration is performed in the BPEL engine and RESTful services are orchestrated in the REST orchestration engine. This approach is not effective when the interactions among heterogeneous types of services are significant. The above two approaches enable only design time composition of SOAP and RESTful services and do not support dynamic selection and dynamic composition of services.

An approach proposed by Giorgio et al. [14] involves static discovery of a set of services either SOAP based or RESTful, for functionality and replacing a failed service at runtime, with an alternate one, from this set. This approach is not truly dynamic as the services are selected at design time. New services cannot be selected on the fly, based on changing requirements.

3. MOTIVATION

The existing works that compose SOAP and RESTful web services deal either with static composition or impose certain restrictions on interaction between services. However, composition should be dynamic to adapt to frequent changes in requirements. Moreover, it is not possible to restrict the interactions between heterogeneous services as the business workflow is based on the business scenarios. Hence, *DyCSR*, a dynamic composition approach is proposed in this paper. The approach involves a middleware service that enables dynamic selection and composition adapting to different types of web services at runtime. Dynamic composition of the services enables high composition control and failure recovery at runtime.

4. PROPOSED APPROACH

DyCSR enables dynamic composition of SOAP and RESTful web services using the middleware services such as *WFParser, URIRetrieval,* and *WSSelection* as shown in Figure 1. The requirements for any business process are obtained from the end user through a portal which initiates the composition. The proposed approach uses template based composition method. In template based composition method, the predefined workflow template is fetched from the library based on the user requirements. The workflow template is an XML document that contains a set of activities to be carried out for a particular application scenario with their control flow represented using workflow patterns [15]. Each activity in the template is represented by its name and description.

The concurrent activities are specified using *WFParallel* tag. Sequential activities are placed in a *WFsequence* tag. For example, consider a simple workflow in Figure 2 wherein activities T2 and T3 are executed in parallel. Activities T1 and T4 precede and

succeed in sequence to the parallel composition of activities T2 and T3.

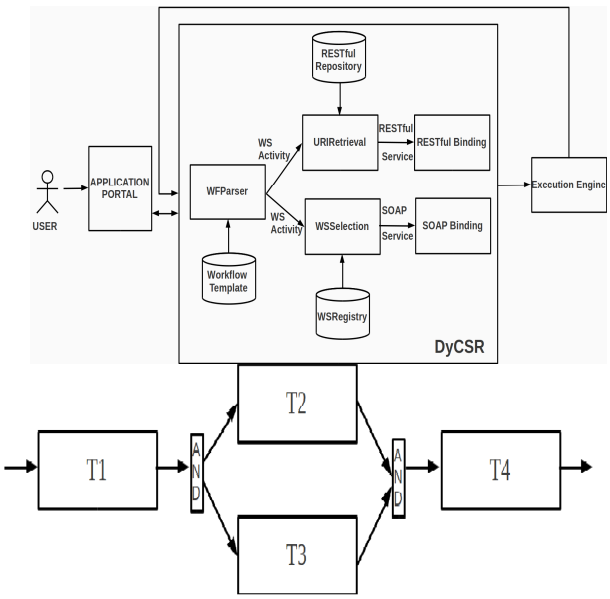

Fig 2: A Sample Workflow

The workflow template of this workflow is shown in Listing 1.

Listing 1: XML Listing of Workflow Template

```
<?xml version="1.0" encoding="UTF-8"?>

<WF:workflow
xmlns:xsi='http://www.w3.org/2001/XMLSchema-
instance'
xmlns:WF='http://xml.netbeans.org/schema/WF'
xsi:schemaLocation='E:\schema_files\WF.xsd'

WFId="SampleWorkFlow"
xmlns:xlink="http://www.w3.org/1999/xlink">
<WF:WFsequence>
<WF:activity id="T1">
        <WF:name>ActivityName</WF:name>
        <WF:desc>Description</WF:desc>
</WF:activity>
<WF:WFparallel>
<WF:activity id="T2">
        <WF:name>ActivityName</WF:name>
        <WF:desc>Description</WF:desc>
</WF:activity>
<WF:activity id="T3">
        <WF:name>ActivityName</WF:name>
        <WF:desc>Description</WF:desc>
</WF:activity>
</WF:WFparallel>
<WF:activity id="T4">
        <WF:name>ActivityName</WF:name>
        <WF:desc>Description</WF:desc>
</WF:activity>
</WF:WFsequence>
</WF:workflow>
```

The workflow template is parsed by *WFParser* middleware service to identify the activities in the workflow. The service can be a SOAP or RESTful service. RESTful web services are selected from the RESTful repository by using the *URIRetrieval* middleware service and SOAP based web services are selected from the registry of web service, *WSRegistry* using *WSSelection* middleware service. The *WSRegistry* maintains the WSDL files corresponding to all the published web services and a private RESTful repository contains the WADL files corresponding to all the RESTful services. The selected service is bound based on its type and executed. In the case of a SOAP based web service, a SOAP request message is constructed and in the case of a RESTful web service, a HTTP request is constructed. Then, the requests are passed to their respective execution engines for execution. The execution engine sends the response back to the composer. The procedure is repeated for each activity thereby enabling dynamic selection and composition. The approach of *DyCSR* is described using the algorithm 1.

Algorithm 1: DyCSR Composer

Input : User requirements

Output : Results based on the requirements

1. fetch *WorkflowTemplate*
2. repeat steps 3 to 5 until EOF(*WorkflowTemplate*)
3. *nextactivity* ← *WFParser*(*WorkflowTemplate*)
4. if *nextactivity* is found in *RESTfulRepository* then

   URI ← *URIRetrieval*(*nextactivity*)

   call *InvokeRESTService*(URI)
5. if *nextactivity* is found in *WSRegistry* then

   EndpointReference←*WSSelection*(*nextactivity*)

   call *InvokeSOAPService*(*EndpointReference*)

6. stop

5. MOTIVATING EXAMPLE

The proposed method of dynamic composition is illustrated using an e-Governance application of *SelectLand* as depicted in Figure 3. Each real estate provider publishes a *SearchLand* service which helps the users to search for the lands available for sale. Land registration department fixes the guideline value for a land according to the land type and location. The *GetGuidelineValue* service is provided by the registration department to fetch the guideline value details. *CalculateLandValue* service calculates the land value based on the area and the corresponding guideline value. The *SelectLand* application involves collaboration among the various real estate providers and the registration department. A business workflow for *SelectLand* shown in Figure 3

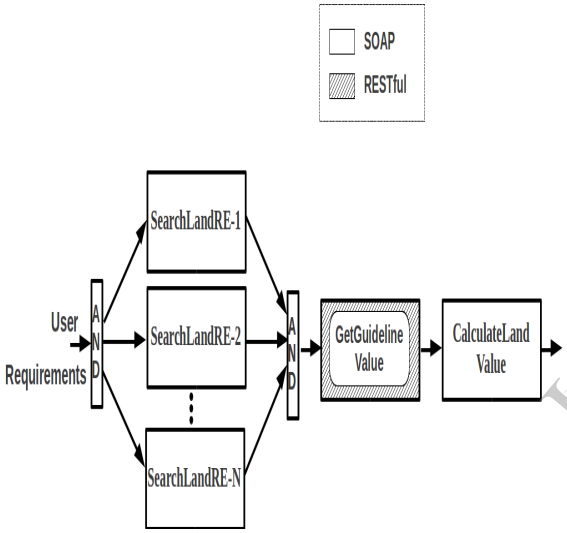

Fig-3.e-Governance application for LAN connection

involves three activities namely, *SearchLand*, *GetGuidelineValue* and *CalculateLandValue*. Based on the workflow activities, the suitable services are dynamically selected and executed. The *SearchLand* activity is repeated for the number of real estate providers available. *SelectLand* application is implemented using the technologies, JAX-WS for SOAP based web services and JAX-RS for RESTful services. The development environment consists of Netbeans IDE and OpenESB. The services are deployed in different web servers (Glassfish, Tomcat), different platforms (Linux, Windows) and different database servers (PostgreSQL, Derby) in order to realize the agility and interoperability of SOA.

The services involved in *SelectLand* are deployed in the local Intranet. The workflow template for *SelectLand* is designed confining to the workflow shown in Figure 3. In general, it is assumed that SOAP based web services are suitable for functionalities involved in private enterprise applications and RESTful services are suitable for Governmental functions. Therefore, *SearchLand* and *CalculateLandValue* are implemented as SOAP based web services and *GetGuidelineValue* service is implemented as a RESTful service. Figure 4 shows the log file of Glassfish server captured during the execution of the *SearchLand* service. Two service providers are assumed to be available for the SearchLand activity and both the services are executed.*SearchLand* activity and both the services are executed. The consolidated list of lands available with these two providers is displayed to the user. The user selects the land of his choice from the list provided. Figure 5 shows the server log captured during the execution of *GetGuidelineValue* and *CalculateLandValue* service. The guideline value is fetched for the selected land and is used in calculating the total cost of the land. The calculated land value is returned to the end user. Thus, the services involved in the workflow of *SelectLand* application are dynamically selected, composed, and executed using the proposed approach *DyCSR*.

6. CONCLUSION

In this paper, a dynamic composition method *DyCSR* is proposed to compose SOAP and RESTful services together. The significant strength of the proposed approach is that it can compose both SOAP as well as RESTful services at runtime. The proposed approach works for any kind of business workflow and does not require human intervention. DyCSR adapts to the frequent changes in user requirements during dynamic composition. An e-Governance application has been used to illustrate the proposed approach. The approach need to be extended to work for hierarchical business workflows in order to make the dynamic composition method more robust and mature. The future work also includes applying recovery mechanisms for failure that arise due to non-availability of services, to achieve failure tolerance.

7. REFERENCES

[1] Simple object access protocol (SOAP), W3C, 2007. www.w3.org/TR/soap/.

[2] Fielding, R., Representational state transfer (REST), Architectural styles and the design of network-based software architectures, Ph.D. thesis, University of California, Irvine, 2000.

[3] Chinnici, R., Moreau, J.-J., Ryman, A., and Weerawarana, S., Web Services Description Language (WSDL) version 2.0 part 1: Core language, W3C Recommendation, 2007.

[4] Pautasso, C., Zimmermann, O., and Leymnn, F., Restful web services vs. big web services: making the right architectural decision, Proceedings of the

17th international conference on World Wide Web, pp. 805–814, 2008.

[5] Mustafa, F., and Mccluskey, T., Dynamic web service composition, IEEE International Conference on Computer Engineering and Technology, ICCET'09, vol. 2, pp. 463–467, 2009.

[6] Zeng, L., Ngu, A. H., Benatallah, B., Podorozhny, R., and Lei, H., Dynamic composition and optimization of web services, Int. J. on Distributed and Parallel Databases, vol. 24, pp. 45–72, 2008.

[7] Agarwal, V., Chafle, G., Mittal, S., and Srivastava, B., Understanding approaches for web service composition and execution, Proceedings of the 1st Bangalore Annual ACM Compute Conference, pp. 1-8, 2008.

[8] Rajaram, K., and Babu, C., Template based SOA framework for dynamic and adaptive composition of web services, IEEE International Conference on Networking and Information Technology (ICNIT), pp. 49–53, 2010.

[9] WS-BPEL 2.0, Web Services Business Process Execution Language version 2.0, OASIS Standard, 2004. http://www.oasis-open.org/committees/tchome.php?wg abbrev=wsbpel

[10] Pautasso, C., Restful web service composition with BPEL for REST, Int. J. on Data & Knowledge Engineering, vol. 68, pp. 851–866, 2009.

[11] Peng, Y.-Y., Ma, S.-P., and Lee, J., REST2SOAP: A framework to integrate SOAP services and RESTful services. IEEE International Conference on Service-Oriented Computing and Applications (SOCA), pp. 1–4, 2009.

[12] Hadley, M. J., Web Application Description Language (WADL), World Wide Web Consortium Recommendation, 2006.

[13] He, K., Integration and orchestration of heterogeneous services, IEEE Joint Conferences on Pervasive Computing (JCPC), pp. 467–470, 2009.

[14] De Giorgio, T., Ripa, G., AND Zuccala, M., An approach to enable replacement of SOAP services and REST services in lightweight processes, Int. J. on Current Trends in Web Engineering, pp. 338–346, 2010.

[15] van der Aalst, W. M., Hofstede, A. H., Kiepuszewski, B., and Barros, A. P., Workflow patterns, Int. J. on Distributed and parallel databases, vol. 14, pp. 5–51, 2003.

Akshaya G is currently pursuing M.E (Computer Science and Engineering) at SSN College of Engineering. She received her B.Tech in computer science from SASTRA University, Thanjavur. She has worked as System Engineer in Infosys Ltd. for 2 years.

Kanchana Rajaram is an Assistant Professor at the Department of Computer Science, SSN College of Engineering, Chennai. She holds a M.E. in Computer Science from National Institute of Technology, Tiruchirapalli and currently pursuing her research in Service Oriented Architecture. She has more than 20 years of teaching experience.

Chitra Babu is a Professor and Head of the Department of Computer Science, SSN College of Engineering, Chennai. She obtained PhD in Computer Science from IIT, Madras, Chennai and M.S. in CIS from Ohio State University, USA. She is currently guiding 6 PhD research scholars.