Spring Framework in Smart Proxy Transaction Model

Sunam Pradhan
Graduate School of Information Technology and Mathematical Sciences
University of Ballarat
Ballarat, Victoria, Australia
s.pradhan@ballarat.edu.au

Arkady Zaslavsky
Faculty of Information Technology
Monash University
Caulfield, Victoria, Australia
Arkady.Zaslavsky@infotech.monash.edu.au

Zahir Tari
School of Computer Science
RMIT University
Melbourne, Victoria, Australia
zahir.tari@rmit.edu.au

***Abstract*— This paper explores adoption of open source application framework in Smart Proxy (sProxy) Transaction model for transaction support. An open source application framework – Spring Framework is plugged into the Smart Proxy (sProxy) Transactional model to support transactional properties. Spring Framework in the sProxy Transaction model increases the transactional interoperability in Web Services context.**

Keywords-Spring Framework; Web Services transaction; transactional interoperability

I. INTRODUCTION

The emerging Web Services technology opens a new paradigm in business transactions. Pervasive computing in the Web Services, is the next stage in Information Technology. Web Services are becoming pervasive in development of enterprise applications, especially in transaction services. Pervasive services add new dimensions to the Web. Service Oriented System integrates and composites multiple services transparently. Standards such as Business Process Execution Language for Web Services (BPEL4WS) [1], Extensible Markup Language (XML) [2], Simple Object Access Protocol (SOAP) [3] and Web Services Description Language (WSDL) [4] etc have been around to meet the business needs. Standard bodies are rapidly laying out the foundation to define consistent and concrete standards.

Pervasive devices like mobile phones, Personal Digital Assistant (PDA), Global Positioning System (GPS) are becoming more popular in business transactions. However, at the same time, reengineering legacy systems are very challenging.

In this paper, Smart Proxy (sProxy) Transaction Model Figure **1** [5] is presented to support transaction in Web Services context using an open application framework, Spring Framework. sProxy is a common interface to Web Services transaction and legacy systems. This approach provides significant advantages in the Web Services computing environment. An online Business-to-Business (B2B) or Business-to-Consumer (B2C) wine selling scenario is used to illustrate transactional properties in Web services.

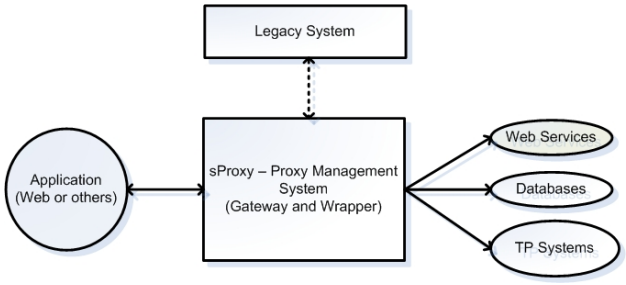

Figure 1. sProxy Transaction Model

In the proposed proxy based transactional model [5], proxy management system plays central role to manage and multithread transactions. The model consists of three-tier application, namely; client application as the first tier, sProxy as the second tier and Web Services and other services, as the third tier. A major advantage of our approach is a low cost, scalable and interoperable model since sProxy acts as the middleware and gateway between services and transactions.

II. FRAMEWORK FOR TRANSACTIONAL MODEL

The framework provides a structure for integration, and a foundation for protocols and services in Service-Oriented

environment. The main goal of this framework is to achieve autonomous, scalable and robust middleware system.

A. sProxy Transactional Framework

This section explains the framework for the architecture proposed in [5].

Web Services are set of protocols based on XML. Three important base protocols are SOAP, WSDL and Universal Description Discovery and Integration (UDDI). Figure **2** shows how Web Services work.

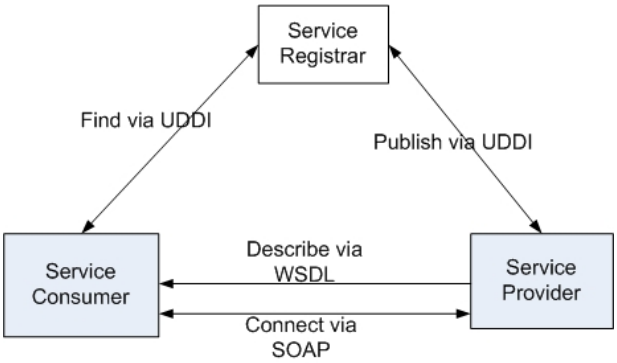

Figure 2. Base Web Services Protocol

To improve the transactional features of the Web Services, and to address more complex business scenarios, the following framework has been developed as an implementation of proxy management systems.

As shown in Figure 3, sProxy model enables Web Services transaction and composition by adapting multilayered architecture of several specialized components which are described below. These objects transactionalize the Web Services. In addition, the transacting Web Services also need to agree on a mutually trusted transactional coordinator to control the execution of a transaction.

A wrapper object encapsulates the transactional properties into a web service call, and provides its properties such as fault tolerance and security.

A getaway object contains data for real-time processing for the active object.

A transactional memory allows concurrent execution of a program. The main goal of this object is to achieve unlimited transaction, long-running transactions and unlimited closed/open nesting transactions while maintaining Atomicity, Consistency, Isolation and Durability (ACID) properties.

A Active Object Server is used for event notifications with a client module for network communication and object cache management. It communicates with other active objects located at different machines.

A transactional proxy implements all the same interface as its target. Transactional proxy object is created for the target object with transactional features. Transactional manager handles the transactional proxy object [6].

B. Scenario

In this scenario, let us consider people buying wines from an e-commerce site. There are many different tasks involved in this process. Considering the following tasks:

- Customer ordering wine
- Customer paying via Credit Card
- Wine review by judges
- Wine review by customers
- Getting wines from national and international vendors

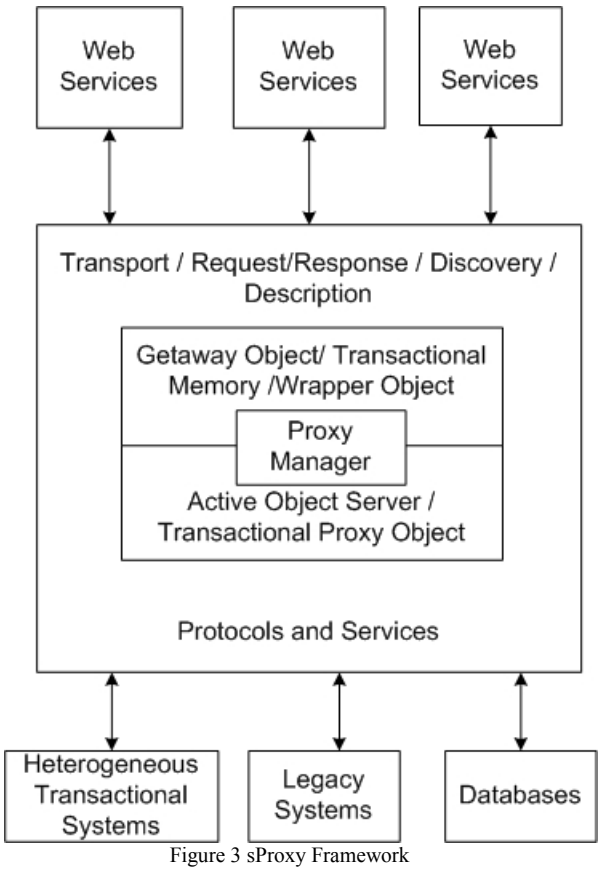

Figure 3 sProxy Framework

This example describes wrapping transactional behavior in a Web Services transaction where series of request are sent to the sProxy which in turn sends the transaction request to different web servers depending on their requirement and service availability.

The process is triggered by the registered customer login. Then the control flows through different stages of the process with customers' selection. This hierarchical sub model illustrates the relationship among various controls.

After the module receives the login details of a customer, the customer is verified and then given control access to order lists of wine. Wine list is obtained from different web services depending on the service availability. The process is followed by payment method, then logout.

All the wine listed on the Web site must maintain minimum standards. So, wines must be approved by panel of

judges. After customer buys a wine, they may provide a review on a selected wine that they purchase.

Seller purchases wine from national and international vendors after the review from wine judges.

III. IMPLEMENTATION

The sProxy provides the interface to connect open source application framework.

The Java 2 Platform, Enterprise Edition (J2EE) application development environment with the Spring Framework, the MySQL, the XML/SOAP protocol, and the NetBeans 6.5 Integrated Development Environment (IDE) are used for this project. It is believed that our application tool is more appropriate since XML and SOAP is the cornerstone of interoperability. It is a common protocol in Web technology for data transfer. Spring Framework is an open source, lightweight, inversion of control, and aspect oriented container framework. It provides better leverage and Plain Old Java Object (POJO) programming. Inversion of control simplifies Java Database Connectivity (JDBC). It uses the Dependency Injection (DI) design pattern, which greatly helps in unit testing. With Spring transaction management framework, transaction system is easily set up with configuration rather than Java Transaction API (JTA) or Enterprise Java Beans (EJB).

The basic approach is to wrap a web service call to a transaction. Solution is to build a new, lightweight Web Service which wraps existing Web Service which is exposed by WSDL, and can be consumed by other services.

First step is to create WSDL for the client services, then import WSDL for service implementation.

Structure needed for the WSDL is created using WSDL editor as shown in the Figure 4.

Second step is generating entity class from a database. Connect to the MySQL database, wineDB table using Services properties. From the wineDB, required tables are selected to create Entity classes from Database with Persistence Unit.

Next step is to import the URL of the WSDL created in the first step.

Then Web Service request is added to the Wrapper class. For the simplicity, local transaction is taken for example. Declarative transaction management is used. Interface for the services required are created. In this context, interface for the following services are created: wineService, customerService, reviewService, and paymentService etc. Then implementation file is written for each interface. In declarative transaction, configuration file is used for transaction setting.

In a declarative transaction, the configuration file plays an important role in the creation of a transactional proxy around the object created from the wineService bean.

Proxy is configured with the transactional advice and appropriate method is invoked as defined in the configuration file as in Figure 5. Different configuration file can be created for different transactional semantics.

Figure 6 shows in details on the functionality offered by the @Transactional annotation. In declarative programming, an annotation based approach to transaction configuration is used. Declaring transaction semantics directly in the concrete class of the source code has more control in programming [7].

Transactional properties available in the Spring Framework, are outlined in Table 1.

Characteristics of a transaction is specified by a transaction definition. TransactionDefinition is an interface to specify six propagation behaviour. They are required, supports, mandatory, requires new, not supported and never. Default transaction definition is required. Transaction definition also includes isolation levels and timeout. However, these are not supported by all resources.

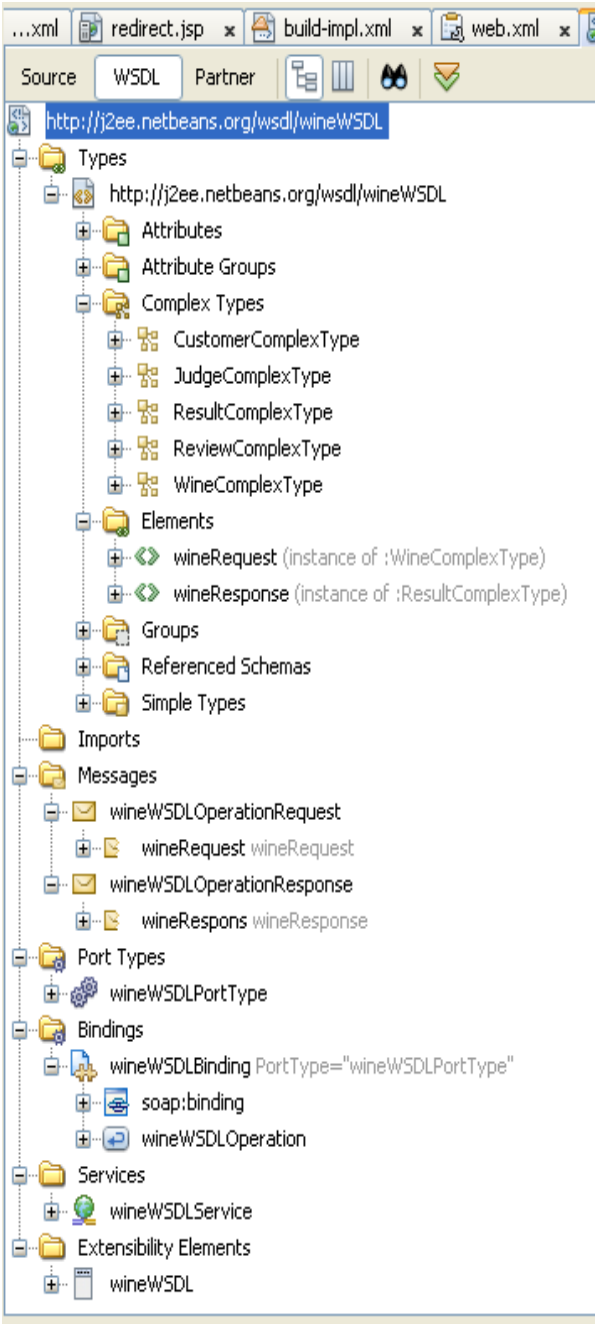

Figure 4 Wine Services structures for the WSDL

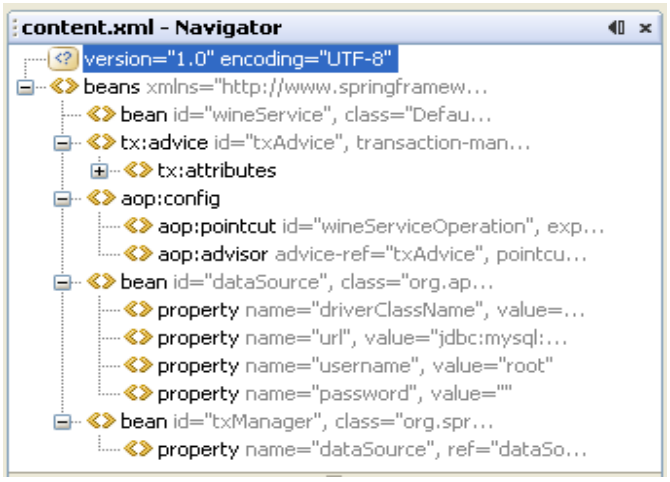

Figure 5 Configuration file

```
package wineService;

import java.util.logging.Level;
import java.util.logging.Logger;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.persistence.EntityManager;
import javax.transaction.UserTransaction;
import
org.springframework.transaction.annotation.Transa
ctional;
import winedb.Wine;


@Transactional

public class DefaultWineServices implements
WineService {


     public Wine findbyID(Short wineID) {
        Wine aWine = null;
        try {
             Context ctx = new InitialContext();
             UserTransaction utx =
(UserTransaction)
ctx.lookup("java:comp/env/UserTransaction");
             utx.begin();
             EntityManager em = (EntityManager)
ctx.lookup("java:comp/env/persistence/LogicalName
");
             aWine = em.find(Wine.class, wineID);
             utx.commit();
        } catch (Exception e) {

Logger.getLogger(getClass().getName()).log(Level.
SEVERE, "exception caught", e);
             throw new RuntimeException(e);
        }
        return aWine;
    }
…
\
```

Figure 6 Functionality @ Transactional

Table 1 @Transactional Properties

Property	Type
propagation	enum
isolation	enum
readOnly	boolean
timeout	int
rollbackFor	an array of class Objects
rollbackForClassname	an array of class names
noRollbackFor	an array of class Objects
noRollbackForClassname	an array of String class names

Source: [7] http://static.springframework.org/spring/docs/2.0.x/reference/transaction.html#transaction-declarative

Above transactional properties can be applied as in Figure 7.

In the simple scenario of wine search by wine id, client sends a web service call to the server; server wraps the call with transactional properties and displays the results as shown in the Figure 9. Client window is shown in the Figure 8.

```
@Transactional (readOnly = true)

public class DefaultWineServices implements
WineService {


      public Wine findbyID(Short wineID) {
         …
     }

      @Transactional(readOnly = false,
propagation = Propagatoin.REQUIRES_NEW)
      public void updateWine(Short wineID) {
        …
      }
…
}
```

Figure 7 Usage of @Transactional Properties

Figure 8 Client Browser

IV. RELATED WORK

This section examines different frameworks for transactional model with Web Services.

Spring Framework [7] is an open source framework for managing complex enterprise application development which uses Inversion of Control (IOC) and Aspect Oriented Programming concepts. Its architecture is based on the Dependency Injection (DI) design pattern. The key benefit of this approach is that it enables the developers to build loosely coupled applications. It is much easier to use and more elegant. It has been getting more popular and adapted in wide range of projects by many industries. Service integration code is exposed as part of the programming interface, which gives application developers the flexibility to assemble services as needed. The flexible service assembly makes virtual objects, instead of real service objects into the application, which is better for unit testing. It uses declarative services using XML configuration files which are complex. However, it has its own restrictions. Since Spring Framework sits on top of the application servers and service libraries, it is more difficult to optimize the interaction between the framework and the services. There is no simple way to leverage clustering services in Spring application.

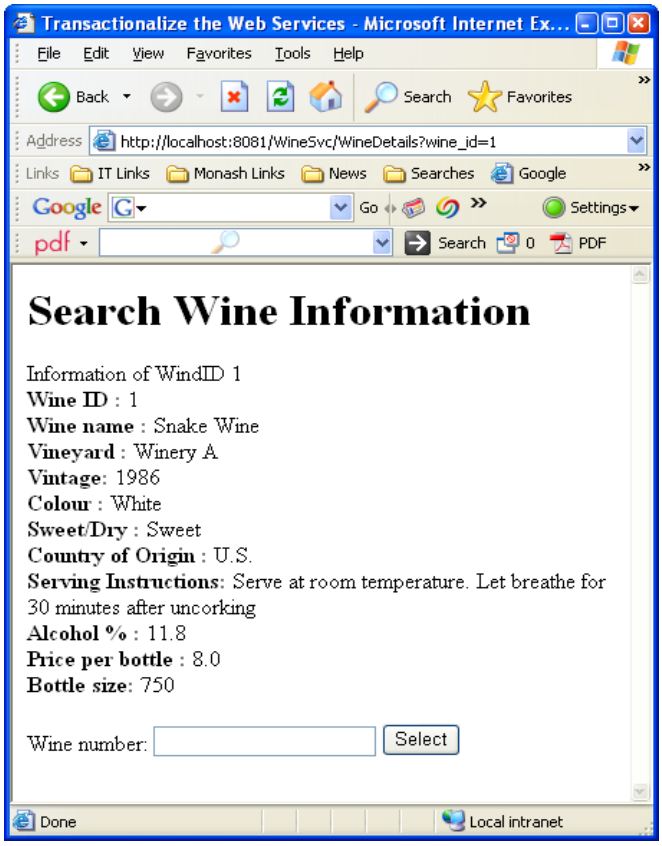

Figure 9 Wine details

Struts Framework [8] is an open source framework for building Servlet/JavaServer Pages (JSP) web based application which is based on Model View Controller (MVC) approach. It provides utility classes to handle many of the most common tasks in Web application development. It uses centralized file-based configuration which is better than hard coding information into Java programs. This loose coupling means changes can be made without modifying or recompiling Java code. This approach helps in implementing business logic and focus on specific tasks without overall system overview. It provides custom tags to populate JavaBean component which can be associated to Hypertext Mark-up Language (HTML) forms. It provides a declarative validation framework for both client-side and server-side. Struts Framework has its own disadvantages due to its complexity. Using Struts Framework is bigger learning curve since developer needs to understand JSP and Servlet API and the availability of the resources are poor. Apache documentation is poorly organized. Since Struts applications are less transparent, it is harder to understand and harder to benchmark and optimize. Since it uses MVC approach, the framework makes it difficult to uses other approach [9] and [10].

WebTransact is a framework to build reliable, maintainable, and composite Web Services which consist of multilayered architecture, XML language and a transactional model. It uses Web Services Transaction Language (WSTL) for describing the transactional support and content of the Web Services rules. Mediator services play important role in creating homogenous interfaces for transaction interactions. The architecture provides the functionality of composition of Web Services and explicit definition of transaction semantics. When there is an usage of different transaction protocols in Web Services, some of the services like compensation services do not work [9].

There has been extensive research in transaction support in distributed computing environment, transactional process coordination and workflow management systems. Some of the related technologies are Web Services Business Process Execution Langauge (WSBPEL), Web Services Choreography Description Language (WS-CDL), WS-AtomicTransaction, WS-BusinessActivity and Web Services Transaction Management (WS-TXM). The problems with these are composition and coordination of services in heterogeneous environment. Web Services use service oriented architecture to connect between heterogeneous platforms. However, transactions in Web Services are problematic and different from transactions in distributed environment. Some of the works done in composition of Web Services are Web Services Flow Language (WSFL) [11], eXtensible Language (XLANG) [12] and Web Services Conversation Language (WSCL). These are XML based Web Services technology. The problem with this technology is that dissimilar transaction support cannot be handled [9]. There are some transaction protocols [13] for the Web Services that have been discussed, such as Tentative Hold Protocol (THP) [14], Business Transaction Protocol (BTP) [15], Business Process Execution Language (BPEL) [1] and

Web Services Coordination and Web Services Transaction (WS-C/WS-Tx) [16] and [17].

V. Conclusions and Future work

In this paper, the open source application framework – Spring Framework is plugged into Smart Proxy (sProxy) Transactional model to support transactional properties. Spring Framework in sProxy transaction model increases the transactional interoperability in Web Services context. The paper explores adoption of open source application framework in sProxy Transaction model for transaction support. Future work includes compatibility of other frameworks, further development of other components of the sProxy, analysis of safety, serializability and validation of the system. Besides that, other requirements to be checked are correctness, robustness, and reliability and delay factors in transactions.

The published research is supported by ARC (under ARC Linkage grant no. LP 0455234)

References

[1] "Business Process Execution Language for Web Services version 1.1," [Online] Available:http://www.ibm.com/developerworks/library/ws-bpel/, 2007.

[2] "Extensible Markup Language (XML)," [Online] Available: http://www.w3.org/XML, 2007.

[3] "Simple Object Access Protocol (SOAP)," [Online] Available: http://www.w3.org/TR/soap, 2007.

[4] "Web Services Description Language (WSDL) Version 2.0 Part: 1 Core Language," [Online] Available: http://www.w3.org/TR/wsdl20/, 2007.

[5] S. Pradhan and A. Zaslavsky, "A Smart Proxy for a Next Generation Web Services Transaction," Proc. 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS), IEEE Press, July 2007, pp.646-651, Melbourne, Australia, doi: 10.1109/ICIS.2007.44.

[6] K. P. Birman, *Reliable Distributed Systems*. New York: Springer, 2005.

[7] "Spring Source," [Online] Available: http://www.springsource.com/, 2009.

[8] "Struts Framework," [Online] Available: http://struts.apache.org/, 2007.

[9] P. d. F. Pires, "A Framework for Specifying and Coordinating Reliable Web Services Compositions," Federal Univesity of Rio De Janeiro, Coppe 2002.

[10] M. Hall, L. Brown, and Y. Chaikin, *Core Servlets and Java Server Pages*, vol. 2, 2nd ed: Prentice Hall, 2007.

[11] "Web Services Flow Language (WSFL)," [Online] Available: http://xml.coverpages.org/wsfl.html, 2002.

[12] "XLANG," [Online] Available: http://xml.coverpages.org/xlang.html, 2001.

[13] M. Little, "Transactions and Web Serivces," *Communications of the ACM*, vol. 46, pp. 49-54, 2003, doi: 10.1145/944217.944237

[14] J. Roberts and K. Srinivasan, "Tentative Hold Protocol part 1: white paper," W3C 2001.

[15] "OASIS Technical Committee Advances Business Transaction Protocol (BTP) Specification," [Online] Available: http://xml.coverpages.org/ni2004-12-24-a.html, 2004.

[16] "Web Services Transaction (WS-Transaction)," [Online] Available: http://dev2dev.bea.com/pub/a/2004/01/ws-transaction.html, 2004.

[17] M. Little and T. Freud, "A comparison of Web Services transaction protocols: A comparative analysis of WS-C/WS-Tx and OASIS BTP," [Online] Available:http://www.ibm.com/developerworks/webservices/library/ws-comproto/, 2003.