A Knowledge Driven Approach in Leveraging Web Services Delivery

Wei Dai  
School of Information Systems  
Victoria University  
Melbourne, Australia  
Email: wei.dai@vu.edu.au

Ning Shen  
School of Information Systems  
Victoria University  
Melbourne, Australia  
Email: ning.shen@students.vu.edu.au

Paul Hawking  
School of Information Systems  
Victoria University  
Melbourne, Australia  
E-mail: paul.hawking@vu.edu.au

Abstract

The paper investigates a new approach of leveraging Web Services technologies to support systems integrations. The paper provides a literature review of the current practice of Web Services from both technical and business perspectives to highlight their strengths and weaknesses. A solution framework aiming at improving the current limitations of Web Services particularly on capability for integration and transactions is subsequently proposed to demonstrate how techniques from different computing disciplines may be deployed to answer the questions that have been raised. The solutions focus on a knowledge driven approach through a knowledge management component. The links between the various framework components and their responsibility in delivering Web Services are also discussed.

Keywords: knowledge management, knowledge-based systems, Web Services.

1. Introduction

Web Services are emerging technologies driven by the will to securely expose business logic beyond the firewall. IBM (2000) defines Web Services as:

“a new breed of Web application, which is self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes.”

Once a Web Service is deployed, other applications (and other Web Services) can discover and invoke the deployed service. The Gartner Group (2001) held a similar view as they defined a Web Service as:

“A software component that represents a business function (or a business service) and can be accessed by another application (a client, a server or another Web Service) over public networks using generally available ubiquitous protocols and transports (i.e. SOAP over HTTP”).

Microsoft’s (2001) description is more succinct, “a Web Service is programmable application logic, accessible using standard Internet protocols”.

From the perspective of a 3-tier architecture, a Web Service is the “veneer for programmatic access to a service which is then implemented by other kinds of middleware” (Gartner Group 2000). An overview of Web Services development tools and products can be found in (Jones, 2001).

2. Background Technologies

Web Services technology is the compilation of technologies of XML, Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Discovery, Description, and Inventory (UDDI) that allow users to develop, catalogue, and publish business services for delivery and use on the Web (Venugopal & Kupper 2000). Through Web Services one can encapsulate existing business processes, publish them as services, search for and subscribe to other services, and exchange information throughout and beyond the enterprise as Web Services are designed to enable application to application e-marketplace interactions with
the aim of removing the inefficiencies of human intervention (Jones, 2001). Warzecha (2001) argued that due to the growing use of Web Services technology for business transaction services, it was reported that “more than 85% of G2000 organizations believe they will deploy these products by 2004”. Others believe that the application of open standards to middleware will result in a 50 fold increase in the number of Web-based transactions during the next four years (Scannell & Sullivan, 2001).

The greatest impact of Web Services, according to Venugopal & Kupper (2000), will be in the business-to-business (B2B) market space. They put forward the view that whereas companies had to work closely with other business partners in the past to open communication channels and develop functionality, companies may now develop Web Services patterned on core competency, and publish those services for any partner to use. Moreover, organizations will be able to combine Web Services published by various partners to provide clients services both en masse and on a customized basis.

In terms of integration, Web Services fulfill external integration needs for organizations in the sense of allowing partners or customers secure access to information residing on multiple internal systems. At the moment, organizations can do this easily by supplying a WSDL file describing the logic being exposed for external use. Furthermore, Web Services eases the issue of data communication between partners moving integration from proprietary, point-to-point transformation to transformation at the communication layer (XML, SOAP XML Schemas) Venugopal & Kupper (2000). An example of Web Services technical architecture can be found in Figure I.

Figure 1: Example of Web Services Configuration and Implementation – Source: ‘A web services primer’, 2001

2.1 XML

Web Services can communicate because they use the same language: XML to describe their interfaces and to encode their message (Systinet 2002). In Vasudevan’s (2001) view, XML provides a meta-language in which you can write specialized languages to express complex interactions between clients and services or between components of a composite service. Behind the facade of a web server, the XML message gets converted to a middleware request and the results converted back to XML. Another standard of Web Services is ebXML (Electronic Business XML) – The ebXML standard was developed to help make XML the worldwide language for electronic data transactions, much as English has become the standard vernacular for international business transactions. The ebXML registry lists a company’s capabilities in a standard profile, allowing businesses to find one another through the registry, define agreements, and exchange XML messages that facilitate business transactions. The aim is to allow all the operations to be performed automatically without human intervention over the Internet.

2.2 Simple Object Access Protocol (SOAP)

SOAP, based on XML and remote procedure calls (PRCs), provides the backbone of Web Services technology for remote invocation as it defines a standard communications protocol for the Web Service. Linthicum (2000) defined SOAP as a protocol used for exchanging structured data in a decentralized, distributed environment. Systinet (2002) argued that SOAP provides a simple and consistent mechanism that allows one application to send a XML message to another application. It contains three parts: the envelope, the header (optional), and the body. The envelope marks the beginning and ending of a SOAP message. It can also specify encoding rules for serializing or marshalling the data over the wire. A SOAP header might contain a mail to address or addresses, a payment code, or information about a RPC style interaction. There may be several headers in a SOAP envelope or none at all. The SOAP message body carries the data formatted as either a self-describing structure or as an RPC-style interface.
2.3 Universal Description, Discovery, and Integration (UDDI)

UDDI is a consortium formed by Microsoft, IBM and Ariba for creating an Internet standard for the description, registration and discovery of Web Services. Systinet (2002) even argued that UDDI is itself a Web Service with which users communicate by using SOAP messages. The UDDI registry serves as Internet business yellow pages, providing access to published Web Services potentially anyone with a browser (Venugopal & Kupper, 2000). The resultant UDDI framework is “a set of databases where

3. Commercial Perspective: a SAP Scenario

In the case of large organizations, the Enterprise Resource Planning systems (ERP) and Customer Relationship Management systems (CRM) will be the most likely users of Web Services to improve integration (Martin 2002). Many of the world’s leading companies consider ERP systems as an essential information systems infrastructure to survive and prosper in today’s economy. Deloitte Consulting (1999) defined an ERP system as a packaged business solution that is designed to automate and integrate business processes, share common data and practices across the enterprise and provide access to information in a real time environment. Even though there are many benefits associated with the use of an ERP system, initially for many companies the implementation of this type of system was a technological solution to the integration of disparate systems (Deloitte, 1999; Davenport et al 2002).

The integration of disparate systems was achieved by “simply” incorporating these systems’ functionality in the ERP system. Often companies when describing their ERP system implementation would measure its success by indicating the number of “legacy systems” that had been replaced. However rarely does an ERP replace all the existing systems (Sandoe et al 2001). This necessitates interfaces to be created between the ERP system and the existing systems to enable the flow of data between these systems. The complexity of these interfaces often has a serious impact on the implementation (Trepper 1999). For many companies, the lack of understanding of the complexity of these types of implementations resulted in a failure to realize potential benefits of systems integration (Deloitte, 1999; Davenport et al 2002).

This lack of benefit realization has resulted in companies revisiting their ERP implementation in an attempt to leverage their investment by attaining the purported benefits. A CSC study (2001), which surveyed 1009 IS managers from around the world, identified “optimising enterprise wide systems” as their main priority. In many cases this optimization is achieved by increased integration between the ERP system and other systems both within and outside the company. One of the purported benefits of an ERP system is related to the greater visibility of information to facilitate effective decision making (Deloitte 1999; Davenport 2002). However as the ERP system is usually the core system that interfaces to many other systems the potential for this visibility is limited. For many companies, in an attempt to integrate the information from the various systems they implemented a data warehouse solution (Stein and Hawking, 2002). However even though data warehouse solutions facilitated decision making the types of decisions are more long term as data warehouse solutions do not operate on real time data.

Many of the other purported benefits of ERP systems rely heavily on the effective integration with other systems both within an outside the company. Accordingly ERP vendors have included many technologies and methodologies to assist with this integration (Hawking and Stein 2003). SAP who is the major ERP vendor has recently launched their NetWeaver solution which encapsulates many of their existing integration technologies (SAP 2003). NetWeaver acknowledges that enterprise application integration is a complex issue and accordingly is subdivided into technologies that support people, information or process integration. An important component of this strategy is the application of Web Services technology. New releases of SAP’s ERP systems incorporate Web Services technology to facilitate their integration. SAP connects to the outside world through Web Services cab with its endorsement of Java, Java 2 Enterprise Edition - J2EE (SAP Web Application Server 6.2 as an example for multi-tier web application), and the Java Connector Architecture (a standard method of communication to legacy systems including ERP vendors; SAP, PeopleSoft, Oracle and CRM vendors such as Siebel) (Martin 2002).

Figure 2 shows such a configuration scenario adopted from an existing source.
Web Service interoperability goals are to provide seamless and automatic connections from one software application to another (Cohen, 2002). SOAP, WSDL, and UDDI protocols define a self-describing way to discover and call a method in a software application -- regardless of location or platform. Data is marshaled into XML request and response documents and moved between software packages using HTTP or message-based protocols. Interoperability problems creep in at the discovery, definition, and request/response mechanisms. The current weakness and limitations of the technologies are also well recognized.

4. Limitations and Weakness of Web Services

In this section we discuss some of issues that are attributable to the weaknesses and limitations of Web Services technologies and focus on the integration solutions and the enabling technologies to support real-time transactions across the Web.

4.1. Integration

Web Services are ideal when light internal integration needs exist within an organization. Light integration is the transfer of data between two or more systems (Jenz, 2001). A typical scenario is when an employee’s information needs to be passed into various downstream applications. However, in a scenario where more complex integration technologies are needed like transaction processing, business process automation, and so on, Web Services’ performance is not as satisfactory as they excel at communicating data. When composition of business services is required in a single atomic operation with complex workflow, Web Services do not yet provide such mechanisms (Venugopal & Kupper, 2000). Zemic (2002) argued that Web Services currently provide only basic technology for communication and lack integration capabilities including reliable messaging, content-based receiver determination, transformation, accessing legacy systems, etc.

4.2. Transactions

Transactions are one of the basic concepts that directly affect the success of e-business. Without doubt, transactional Web Services would solve a number of interoperability problems. Yet, the overall goal of Web Services, i.e. to enable application integration over the Internet regardless of programming language or operating environment, is not achievable without support for transactions. So far, each software platform, such as Microsoft’s COM+ or Sun’s J2EE/EJB platforms implements its specific transaction model (Jenz, 2001). Since WSDL does not provide information on non-functional characteristics of the service, there is no way to declare Web Services as transactional. The underlying implementation directly (by calling transaction services explicitly) or indirectly (in the case of container-managed transactions) determines transaction boundaries. The transaction issue is very hard to solve as existing component models, such as J2EE/EJB push developers towards utilizing container-managed transactions (Jenz, 2001).

5. The Visions and Challenges Ahead

At the current stage, Web Service technology excels at enabling machine-to-machine communication, but not at operational and workflow processing though many companies are looking to automate and streamline their business processes. Web Services promises the ability to combine individual services into more complex, orchestrated services that will provide sophisticated business process and workflow automation capabilities to the enterprise. However, such composition and orchestration is still on the drawing board (Darwin Partners & ZapThink, 2002). This bottleneck is largely caused by the integration problems between the standards of enterprises’ existing information systems or legacy systems and that of the Web Services. As argued by TIBCO (2002), most of today's legacy systems do not
have Web Services capability, but these systems hold most of the information and functionality that businesses want to expose as Web Services. Without solving this problem, Web Services would not deliver the optimal results as most companies expect.

We believe the developing trend of Web Services solution should address the issue of integration to enhance the transaction capability as the top priority. Support for SOAP and other Web Services’ standards is a key element in changing the way to effectively carry out the integration tasks within the existing standards. Hence, Web Services will herald a shift in distributed computing toward loosely coupled, standards-based, Service-oriented architectures (ZapThink, 2000). Based on this, we developed a Web Service solution framework aiming to meet the needs of enterprises. It is more than just the support for the description of services using WSDL, the SOAP transmission protocol, and service discovery via UDDI. Rather, it is a truly comprehensive Web Service solution that is transport-agnostic so it can meet different performance and scalability requirements. It has a robust XML-based data representation and provides tools for schema management and transformation to eliminate the need for manual coding into and out of XML. Finally, it enables the orchestration of processes that span both Web Services and non-Web Services resources. All these features will enable companies to connect their disparate systems in a flexible, responsive IT infrastructure.

6. The Solution Framework

Our solution framework takes into account Enterprise Information Systems (EIS) and legacy systems and their integration with problem-solving process through Web Services. The framework as described in Figure 3 is based on the existing Web Services infrastructure such as the communication networks and protocols including XML, SOAP and WSDL etc.

The followings are the key components of the framework:

- Knowledge Management - directing the information/task flow and problem solving process.
- Data Management - coping with the framework internal and external data retrieval and storage requests.
- Component Integrator – software packaging glue facilitating systems integration and systems interoperability.
- Communication Management – responsible for all the external communication needs of the framework such as the facilities for Internet communications.
The knowledge Management component offers the framework the capability of interacting with the client applications/request intelligently as well as directing client tasks accurately to the responsible targets across the network. The Data Management component will ensure that the framework is capable to deal with a wide range of external database systems regardless of their locations or formats. The integration of knowledge management and data management delivers the framework the power of processing large volume of information in a real-time manner and make the Web Services a useful and practical solution. The strength of having powerful business logic processing capability supported with all the conventional computing services such as data management services opens up a new solution perspective on real-time transactions delivery via the Web.

The Component Integrator and Communication Management are the enabling techniques to resolve data interoperability issues among internal and external components to allow the framework to best deploy its potential based on the available techniques and tools from external world. Through our Web Services solution architecture, legacy systems and Enterprise Information Systems (EIS) can be dynamically linked with SOAP/XML protocols and adaptors to form an integrated solution in response to a client’s request, therefore effectively tackling the integration issues. The key to building truly interoperable systems is using consistent data formats for application integration (Thelin and Murray 2002). The deployment of component integrator offers the leverage to the existing Message-Oriented Middleware (MOM) infrastructure. MOM decouples application components by placing a logical queue between them. The linking process is triggered by a messaging sender placing a message onto the queue. At some point in the future, a messaging receiver reads that message from the queue and processes it accordingly. This very simple concept allows the application components to be decoupled. In addition, it allows for the possibility that both processes may not be active at a given time. Following up the solution framework diagram, a more detailed description of how standard Web Services facilities are incorporated with the Web-based Knowledge Management component is given in Figure 4.

Figure 4: Communication Process from Web Services Perspectives

When a service provider wants to make the services available to service consumers, the provider describes the service using WSDL and registers the service in the UDDI registry. The UDDI registry will then maintain pointers to the WSDL description and to the service. When a service consumer wants to use a service, the consumer queries the UDDI registry to find a service that matches his/her needs and obtains the WSDL description of the service, as well as the access point of the service. The service consumer uses the WSDL description to construct a SOAP message with which to communicate with the service. Presently, the Knowledge and Data Management components are deployed into the computer laboratories at Victoria University (http://www.vu.edu.au) in Melbourne Australia. The implementation plan for the full deployment of the framework is currently underway.

7. Summary

Web Services technology has the promise to provide a new level of interoperability between software
applications. The reality of Web Services is that dozens of platform providers, independent software vendors, and utility software developers have implemented Web Services’ protocols (SOAP, WSDL, and UDDI) in their products. The strength and the benefits that Web Services can offer to commercial companies can be summarized as the following. Web Services support heterogeneous systems by allowing a link between different applications regardless of the platform, operating system, programming language they are written or place where they are located. By publishing services on-line through Web Services, enterprises are able to be available 24 x 7 days for everyone who needs get connected to their systems.

In this paper, the existing work on Web Services has been carefully reviewed from an industrial perspective, which reveals some limitations and scope for improvement on the current state of the technologies. The proposed knowledge driven solution framework focuses on two key technology issues, i.e. transactions and integrations that are closely associated with the quality of Web Services. The knowledge driven solution framework creates added value by integrating information that was unable to be linked previously. This new technology facilitates customers, suppliers, business partners, employees and everyone else involved in the business environment who needs to work together to share the knowledge. The framework is at an early prototype stage and is moving towards full implementation.

References:


Darwin Partners and ZapThink (2002), ‘Using Web Services For Integration’.
www.xml.org/xml/wsi.pdf


http://www3.gartner.com/Init.


byname/B803C4598F0F4A6986256A86004B3E75).


http://www.webservices.org/article.php?sid=444

http://searchsap.techtarget.com/tip/1,289483,sid21_gci817246,00.html

http://msdn.microsoft.com/library/techart/websvcs_platform.htm


Sandoe, K., Corbitt, G., Boykin, R., (2001), Enterprise Integration, John Wiley and Sons, USA.


http://iwsun4.infoworld.com/articles/hn/xml/01/05/14/010514hn
ibm.xml

Systinet Corp. (2002). ‘Introduction to Web Services’

http://www.capeclear.com/messaging/WebServicesMessaging.p df

Tibco (2002), ‘Integration Delivers the Promise of Web Services’,

http://itmanagement.earthweb.com/entdev/article.php/614681

http://www.webservices.org/article.php?sid=421

http://www.metagroup.com/cgi-bin/inetcgi/search/displayArticle.jsp?oid=20362,
http://www.metagroup.com/cgi-bin/inetcgi/search/displayArticle.jsp?oid=22261
Zapthink (2000), ‘WEB SERVICES’,
http://www.zapthink.com/reports/ZTRWS105.html?PHPSESSID=D=eb3986d1a5767442115c5f2d21419b5
Zemic (2002), ‘S---SAP and web services’,