A Flexible Services Architecture Based Translator Web Services

Phill Miller, Sushil K. Sharma, Fred L. Kitchens

[1]Made2Manage Inc., Indianapolis, Indiana, USA
pmiller@Made2Manage.Com
[2]Department of Management, Ball State University, Muncie, IN 47306, USA
ssharma@bsu.edu
[3]Department of Management, Ball State University, Muncie, IN 47306, USA
fkitchens@bsu.edu

Abstract: Despite skepticism concerning the latest buzz phrases about Web services, leading-edge companies have already begun deploying Web services that are giving them competitive advantages. Major platform vendors are now rolling out revolutionary tools aimed at simplifying the creation and deployment of Web services. Promising cheap, reliable, flexible, and scalable computing as well as quality web services is a real challenge to developers. The authors have created a language-translator web service based on the flexible services architecture model. The language translator web service is both technically powerful and could provide real business value for a provider. This web service could be utilized to reduce the communication problems that occur between languages, particularly when one travels or conducts business internationally. This paper presents the details of a translator web service and flexible services architecture model.

Key Words: Web Services, Distributed enterprise computing, flexible architecture, translator web services

1 Introduction

Web services are the next wave of distributed enterprise computing. They provide a layer of interoperability that allows applications to be described, published, located, and invoked irrespective of underlying architectures. The evolution of computing has witnessed three phases; mainframe to client server, client server to web-based applications, and now web-based applications to web services.

Web services are a new development that will help organizations conduct business in a much more open and flexible manner. They are a new class of application that can talk and work with one another over the Internet. As more and more companies seek to conduct significant business over the Internet, they face the problem of making their applications work with those of their customers and suppliers [1]. Web services are applications that use a universal language to send data and instructions to one another, with translation required. They use the Internet, so most of the connection problems are eliminated. Web services are primed to be the next big development for Internet-based applications and transactions. The aim is to seamlessly integrate systems and applications that communicate over a network [2].

The first phase of computing began with mainframe computing services, where a powerful central computer handled all transactions and users accessed it through terminals directly connected to the mainframe. The second phase started with the advent of the personal computer revolution. Here, the approach of building centralized solutions was shifted towards distributed processing. This is called the client-server application phase, in which the processing is distributed between a server, generally a database, and a client. With the advent of the Internet, the client server phase helped integrate applications across the globe and made it possible for systems to exchange data in the form of e-commerce [3]. As more and more applications were developed on client-server architecture, developers started experiencing problems with it. The initial problem with client-server applications was that the applications were not scalable when there were a large number of users. To resolve this problem, system architects created three-tiered applications such as; data layer, a business layer, and a user interface layer to distribute processing and make applications scalable. These layers were then separated further to create n-tier applications, with as many machines as needed to handle the traffic. The proliferation of the Internet aided this type of development because the user-interface layer could still be a server-hosted web application, which eliminated deployment problems for developers. N-tier applications architecture resolved the problems to a greater extent, but it still had several limitations. The first limitation is that these layers need to communicate with each other through Remote Procedure Calls (RPCs), and there is no standard for this communication between objects. Several proprietary communication protocols have been developed including Microsoft's Component Object Model (COM) and the Common Request Broker Architecture (CORBA), but they were device and operating system dependent. There was a great need to standardize the communication between components of software. The second problem with n-tiered solutions is that while they work well over networks, they do not communicate over the Internet. RPCs can be distributed across computers on a network where they can maintain their state, but the Internet is a stateless environment. Because they do not scale over the Internet, applications that are built in this way are not able to take advantage of new technologies like Personal Digital Assistants, cellular phones, and other Internet-enabled embedded systems. End-users today are becoming accustomed to being able to access applications from anywhere, at anytime, and on any device.

These problems were solved with the advent of the eXtensible Markup Language (XML). XML provides a data format that is device and platform independent, and that can be transported over a common protocol, the Hypertext Transfer Protocol (HTTP). XML is replacing all of the proprietary tools for communicating across networks and the Internet. Using XML, devices can communicate information over the Internet to any other device quickly and easily.

This is how the evolution of web services began. Web services are self-describing applications that reside online and that, using standards such as Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery and Integration (UDDI); can be accessed and used by any client. Web services are designed to bring disparate information sources together. It enables developers to build extranets that link organizations with the rest of their value chains by integrating data from their disparate applications. Web services provide interactive functionality such as report listing, viewing, refresh and drill-down as Web services through a portal interface [4].

Web services can be designed to provide integrated applications that can automatically conduct business without human intervention. Application logic to perform a variety of business processes can be aggregated from servers in various locations. Application code can be reused on disparate platforms. Web services offer many benefits in systems design such as:

- Encourage modular system architecture
- Change underlying program logic without greatly affecting - interfaces
- Hide underlying system complexity via standard interfaces
- Extend and enhance legacy systems without changing underlying code
- Offer platform- and vendor-neutral applications

Largely due to their many benefits, web services are gaining momentum [5].

Figure 1 shows the basic flow of a web service, from the consumers of web services, applications, web sites, and other devices, all the way to the service code and the database that supports it.

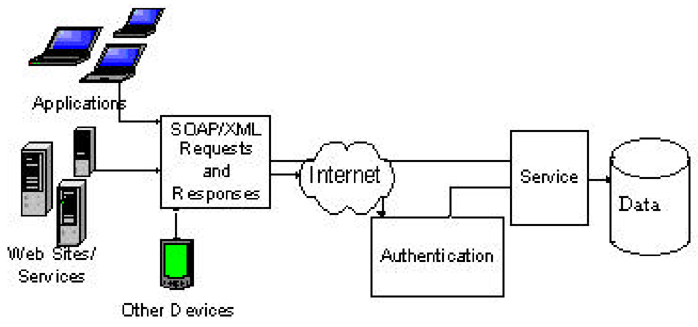

Figure 1: Web Service Processing Model

Jupiter Media Metrix conducted a survey recently to determine the number of companies that are responding to new technology of web services. The survey found that 60% of the responding companies have started using Web services technology for enterprise application integration (EAI), and 53% desire to use it to integrate their applications with those of their partners and customers [6]. With web services proliferating quickly through the software industry, there is a real need for powerful, flexible web services to drive the trend. The key to making web services the new software standard is not only to build technically powerful web services, but also to build services that have strong business models behind them. This paper presents an approach to building a web service based on a flexible services architecture (FSA), and outlines the details of a language-translation web services that the authors have developed using the flexible services architecture. The translator web services model seems to be promising and may be quite helpful for resolving communication barriers across different languages.

2 Web Services Architecture - Anatomy Of A Powerful Web Service

Web services architecture is a set of emerging protocols and standards it offers a different approach to enterprise integration and development, as shown in Figure 2. Architecturally, Web services are typically made available by use of a common transport mechanism, namely SOAP, through which agreements and binding can be universally facilitated. The directory, or repository, is accomplished through UDDI. The interface is described in WSDL, and the transport is managed seamlessly using SOAP, allowing companies to communicate with the outside application regardless of what platform, system, or standards are being used behind the scenes at either company.

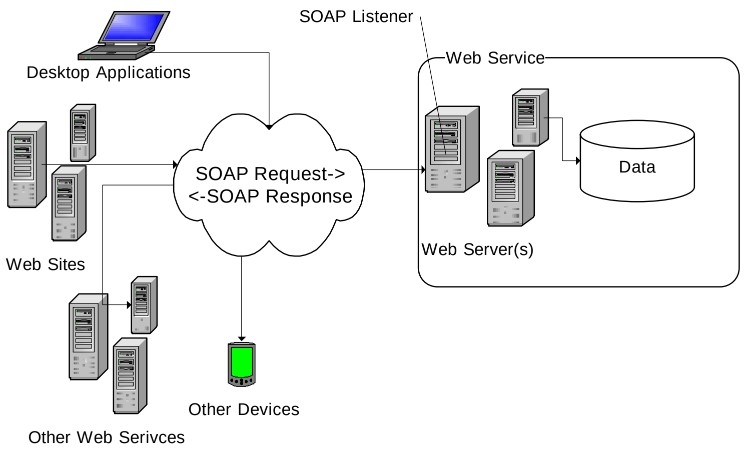

Figure 2: Web Service Architecture

Developers can build standard interfaces to existing and new systems using the SOAP and describe how to access the data using the WSDL. These let each department access data in other departmental databases without custom programming on an application-by-application basis. The re-use of Web services functionality cuts development time. A developer only has to identify the departmental data to access and link to the corresponding WSDL interface, using existing development tools. Presently, web service models are using four standards: Soap, WSDL, XML, and the UDDI protocol. These comprise the basic capabilities necessary to build the discrete elements of a services-oriented architecture.

The architecture of a web service is not any more complicated than n-tier development and is logically partitioned into a service layer and a data layer. The key to any web service is the SOAP listener on the web server machine that accepts and invokes web service calls from consumers. Most web servers have integrated SOAP listeners into the standard web service package. The listener is the first layer of security for a web service because it can filter out calls based on the IP address of the consumer. The middle logical layer of a web service is the web service code. Code for web services can be written on many platforms, but the most common are Microsoft's .NET, Java, and IBM's WebSphere. Nearly all powerful web services are going to depend on a data source. A powerful web service needs to have a data source that is optimized for speed, yet secure enough to expose to web servers. The most important factors when looking at the architecture of web services are speed and scalability. It must be able to respond quickly and it must be scalable to handle the large quantities of users that could be accessing it simultaneously.

In the early stages, developers used to achieve services-oriented architecture by developing applications with modular, loosely coupled interfaces that hide the complexity of the underlying systems. The Web-services standards extend a method of designing systems with modularity, flexibility, and platform independence. The primary components in a services-oriented architecture are all services. A service is a collection of business or application logic that can be accessed via a messaging protocol such as SOAP, which provides a standard way to decouple software components. It is useful when system changes or upgrades are needed. SOAP also lets services collaborate with each other over a network. Database access becomes a Web service rather than a piece of program logic hard-coded into each database application, because it can be accessed via SOAP. The database client application needs to know only how to send and receive SOAP messages and parse the XML-encoded data within each message. XML, a language for encoding data, plays multiple roles for Web services. It's used to define protocols such as SOAP and WSDL, and to encode the data stream that applications use to communicate. Because the database client developer needs to decode only the XML data stream and deal with SOAP messages, all other aspects of the database deployment are hidden. The same database client can be modified to connect to other departmental databases by modifying the XML data stream and SOAP message handlers. When all departments have deployed Web services front-ends using SOAP and WSDL, such modifications become relatively easy, compared with the old methods of developing application-specific interfaces for each department. Web services provide a common implementation, as opposed to possibly hundreds of custom-interface implementations that may have been necessary in the past.

A web service is only valuable if people are using it, so any good web service must be well published and documented. The first aspect of this is developing a description of the services that are provided. The standard format for achieving this is the WSDL. A WSDL file gives a description of how the web service functions and provides all necessary information for consuming the web service. Every service should have a WSDL file that describes its methods and properties, and this file should be published and made available to potential consumers. Many vendors have created tools that make creation of WSDL files simple. Microsoft, Ariba, and IBM have combined to produce a common place for publishing web services called UDDI. The purpose of UDDI is to provide a type of library for web services, where consumers and users can search for web services that meet their requirements.

3 Building Powerful And Flexible Web Services

The first key to building a flexible web service is to make it completely independent of its consumers. There are so many devices that are able to consume a service that limiting the devices that can consume a service limits the reach of its effectiveness and of its revenue base. A consumer may want to access the service from a cell phone, a web browser, a Personal Digital Assistant, or from an embedded car navigation system. The service itself should not make a distinction between these consumers.

The most important principle in building a consumer-independent service is to use well defined, published standards. The first standard that should be followed is the current eXtensible Markup Language (XML) standard, published by the World Wide Web Consortium (W3C). Since its introduction, XML has been recognized as vital in interoperability between systems. XML, by its very nature, is device independent, allowing data to mover freely between any devices by providing the data and its description in a common format. The second standard that should be closely followed is the Simple Object Access Protocol (SOAP), widely accepted and approved by system developers. While services can be accessed by the hyper-text transport protocol (HTTP), POST, and GET protocols, SOAP allows greater flexibility and power, and allows developers to build more robust and secure consumers. To access a service using SOAP, an application packages information about the object and method to invoke, along with parameters to that method, into a SOAP request. This request is then sent via HTTP to the service provider that invokes the object and method with parameters provided. The service then packages its response into a SOAP response and sends it back to the consumer [7].

Another important issue in developing flexible services is trying to maintain compatibility for older devices and consumers. While many IT professionals love to use the latest and greatest technologies, services can often gain a large number of users by being accessible from older consumers and devices. Many users and businesses do not stay on the cutting edge in their technologies and devices. Architects and developers should be mindful of these users when creating a service. The easiest way to do this is to tailor a service to the oldest and most widely accepted set of standards that fulfills the requirements. This will allow all clients that are current with these standards to have access to the service, while still allowing the older applications to access it.

Services must not only be based on standards, they must also be rich in the data that they supply. They must provide all possible data that a consumer might need. The consumer can then decide how to use this data based on the user's requirements and the device's capabilities. To provide a rich service, it must provide information that utilizes the most advanced consumer with adequate data services. The goal of any service should be to provide the best possible experience to its consumers and users. It should provide a wide selection of content and allow the consumer to select what will be used. The service must provide the best possible experience for the users, based on the consumer's capabilities.

4 Flexible Services Architecture Based Translator Web Services

To demonstrate the principles of flexible services based architecture, the au thors have created a web service that that is both technically powerful and could provide real business value for a provider. This web service attempts to help reduce the communication problems that occur between spoken languages. Language can be a big barrier for anyone that travels or conducts business internationally. A web service could be a powerful solution to this problem. A traveler may need to translate a word or phrase while traveling or while talking on the phone. The traditional method of translation would be to use a book to translate the word and then try to pronounce it properly. A web service could do provide an audio translation and an image representation of a word or a phrase, making it much more powerful than a standard dictionary. For example, a traveler could stop at a restaurant in France and not know what to order or how to order it. He could call the web service on his mobile phone and type in what he wants. Then, let the cell phone play the audio file that was returned so that the restaurant's waiter can listen to it. Another application might be a language teacher who might use the web service to help students learn how to pronounce foreign words properly.

Based on the proposed model, a data store containing words in different languages would be located on the provider's servers. Next, the provider would have to create objects that search the database and return the word or words that match. These objects must have at least one method that is exposed publicly to be invo ked from the SOAP listener. The request XML for a typical word lookup might look like this:

```
<translator>
      <class>translator</class>
      <method>search</method>
      <baselanguage>English</baselanguage>
      <foriegnlanguage>Spanish</foriegnlanguage>
      <word>bread</word>
</translator>
```

This XML includes the service class to invoke and the parameters necessary to perform the search. This request XML would be packaged inside a SOAP message to allow the SOAP listener to interpret and authenticate the message. When the listener received this message it would look for the translator class and invoke the search method with the parameters specified. When the method finished executing, it would return a response XML in the following format:

```
<word>
      <foriegnword>pan</foriegnword>
      <part>noun</part>
<image_url>http://translator.net/pan.jpg</image_url>
<audio_url>http://translator.net/pan.mp3</audio_url>
</word>
```

This XML has the translation of the word, its part of speech, and the URLS to both an image of the word and an audio clip of the word being properly pronounced. The client now has all of the information that it needs and can process the results accordingly.

Our translator web service meets the criteria of a powerful web service because it is based on standards, it is rich, and it is consumer independent. First of all, the service is built on the XML standard using the SOAP protocol. Next, the service is rich because it provides enough information to support a client with audio and video capabilities, but a client that does not support audio and video will still be able to consume it. If a user's cell phone does not support images, it can still use the service while disregarding the image. These two qualities of this web service lead to the third quality of consumer independence. A web service such as this is valuable in a business model because it provides critical information to many users and can provide a steady revenue stream to its providers. Frequent business travelers may subscribe to the site for their travels. Language instructors may subscribe to the service to utilize it in a classroom setting. Tourists could use a transaction-based model to handle several translations during a vacation. The potential revenue from these customers makes it a valuable tool for a provider.

5 Business Model For Translator Web Services

Any web service must have a strong business model behind the service and a user authentication system that supports the business model. A flexible web service must provide real value to the users, and the authentication in the service must allow users to be billed for the service, regardless of the device consuming it.

5.1 Business Models

There are several methods of putting a business model in place for a web service [7]. For our translator web service model, we recommend the two most popular business models, the subscription model and the transaction model. In the subscription model, a user would pay a set fee per month or per year to use the service. Frequent users of the service would find this subscription method very useful, and the provider of the web service would derive a constant stream of revenue based on the number of subscribers. Another model that could be used is the transaction model. This model would derive revenue from the service on a per use basis. A user may only need the service a few times and could pay a set fee per use. The transaction model would not bring the steady revenue flow to the provider like the subscription model. However, because of its flexibility to the users, it may still be a strong revenue producer. Powerful services can provide layers of functionality to different types of users, and can derive revenue based on the depth and breadth of service that they provi de. Users can be separated into groups based on the functionality that they need, and the provider can set up pricing models to meet each group's needs. This layering of a service depends greatly on the richness of the web service.

5.2 Authentication

A web service should have a robust user directory that can handle all aspects of the business model and be extensible if the business model should grow or change. The authentication needs to support the business model process to derive revenue for its provider. If the service is subscription based, it will have to check the status of the user's subscription. If it is transactional based, it will need to bill the user directly for the transaction. If a service is layered, the authentication can filter the information based on the level of service the user requires. Whether the authentication is done internally or through an authentication service like Microsoft .NET Passport, the authentication must be fast and secure.

6 Conclusion

Web services provide a means of integrating applications via the Internet. By using XML messaging to exchange data, Web services allow companies to link applications and conduct e-business regardless of the computing platforms and programming languages involved. Web services are quickly becoming the way to develop systems, for obvious reasons. They eliminate the major problems associated with network and distributed software, and they provide a new source of revenue for companies that provide the service. There is a real nee d for powerful and flexible web services in the marketplace because users can now access applications from many different devices, and web services help to support this diversification. The Web browser brought businesses closer to their customers, Web ser vices will help simplify business-to-business (b-to-b) transactions, driving down costs and smoothing the way to collaborative relationships. This translator web service tool is capable of helping travelers and other business organizations collaborate without language barriers. It could be a very useful web service for people worldwide.

References

1. Meehan and Copeland, *Understanding the value of web services.* eWeek, 2002. 19(3): p. 39.
2. Andress, M., *The road to secure web services.* InfoWorld, 2002. 24(2): p. 52.
3. Borck, J.R., *InfoWorld Technology of the year: Web services.* InfoWorld, 2002. 24(4): p. 48.
4. Johnston, S.J., *State of Web services.* InfoWorld, 2002. 24(5): p. 17.
5. Dostan, D., *Solving the web services puzzle.* InfoWorld, 2002. 23(38): p. 44.
6. Fox, P., *Web services can help put squeeze on costs.* Computer World, 2001. 35(9): p. 36.
7. Schultz, B., *Assembling a top of the line Web services model.* Network World, 2002. 19(7): p. 56-58.