

Web Services – The New Hope for EAI

Mehran Nikoo, Dunstan Thomas Consulting
<http://consulting.dthomas.co.uk>



Overview of Web Services

Reports from Gartner and IDC say that Web Services will dominate IT market over the next 10 years. But what are 'Web Services' and do they offer the solution we had all hoped for from Enterprise Application Integration (EAI)?

First, the definition. When a user is browsing a site on the Internet, he or she uses what is called a 'User Interface' to interact with the web site. Growing demand for process automation means that applications increasingly need to communicate with each other. When a user is in fact another application a programmatic interface is required and these interfaces are referred to as Web Services.

Are Web Services the new EAI?

One of the primary success factors for any EAI solution is how well it is agreed upon (and used) among various organisations and software vendors. Web Services are based on well established and widely used standards and specifications like XML and HTTP. Almost all Internet users are using HTTP as their transport protocol, while XML has become very popular because of its flexibility.

An additional benefit of XML and HTTP is that they are platform-independent. This means you can have a Web Service implemented using Microsoft .NET while the consumer is based on J2EE. Although the publisher and consumer of the service are targeted at a specific platform, the interface (which is XML over HTTP) is platform independent.

Barriers to the adoption of Web Services

Although Web Services are based on XML, the specification for Web Services does not outline what XML documents should look like. This is good as it provides flexibility. But problems arise when the publisher and consumer of the service have different understandings of the contents of a message. Standards are emerging from bodies such as OASIS, who defines and publishes schemas for different types of messages (such as Auto Repair, Legal XML and Tax XML). But more needs to be done in order for the true benefit of Web Services to be realised.

The second barrier is created by legacy applications. Almost every company has existing applications in place and most have not been designed with a service-oriented mindset. Software designers did not generally consider the fact that this application may be used by other applications as well as human users. So another challenge is to build 'wrappers' around existing applications and services so that they can be consumed by Web Services clients.

This can get more complicated when building the Web Service interface for a mainframe application that is running batch schedules and cannot be used interactively. This introduces the idea of using asynchronous clients for Web Services so that the client does not have to wait for the response and can carry on with other tasks in the meantime.

This can be taken even further by looking at Web Services as a way of message passing, and not making remote calls to objects. This means that you send a message to a service but do not expect an instant response. The only thing you get back is a receipt for your message. In the future you will receive the response to your message and you can use a tracking mechanism (like using a unique identifier) to correlate those messages together.

The third issue relates to managing aspects like security and transactions, which is usually done using proprietary mechanisms. For example if you are using Windows operating system, you use the

Distributed Transaction Coordinator (DTC) service to manage distributed transactions. When it comes to managing security you rely on built-in authentication and authorisation mechanism that is built into the operating system. The question is what to do when systems are disconnected, asynchronous and talking to each other using protocols like HTTP and XML?

The best solution so far to this problem is the enhancements that have been made to the specifications of Web Services like WS-Transactions and WS-Security. The positive aspect of Web Services is that it is based on Simple Object Access Protocol (SOAP) and SOAP was designed with extensibility requirement in mind. Basically none of these enhancements make any modifications to original protocols, and initial specifications are now published as standards.

Technical Issues

SOAP provides a way of encapsulating calls to remote objects in XML format. If you look at the contents of a SOAP message, you can see the name of the remote operation to be executed as well as the data being passed in and out of that remote operation.

Since the adoption of SOAP, new concepts like XML Schemas have been introduced. XML Schemas replaced the built-in encoding mechanism for the data, leaving developers with two ways of representing the data in SOAP messages: Encoded (which uses the built-in mechanism) and Literal (which uses XML Schemas to define data elements).

The trend towards Service-Oriented Architectures forced industry participants to look at those interactions as message passing and not making remote calls to objects. As a result, today we see two different styles for Web Services (Document-Oriented and Remote Procedure Call).

From an implementation perspective these various styles and encoding mechanisms have much in common and the differences are minimal, but the difference is enough to break the compatibility between the consumer and publisher of the Web Service. Today software vendors and standard bodies are making recommendations for organisations to use the Document-Oriented style, (which uses a message passing concept rather than making remote calls to objects) with a Literal encoding type (which uses XML Schemas to define data).

Another technical challenge is availability of good development environments for developing and consuming Web Services. Currently there are various products from software vendors such as Sun and Microsoft to support developers, but because of the fast developments in Web Services specifications, they have to make frequent updates to their products. Microsoft has implemented and released some of these enhancements as add-on tools to .NET framework and Visual Studio.NET, and developers will see major changes when these improvements are baked into the environment in "Indigo", code-name for the next generation of Web Services tools and concepts from Microsoft.

In summary, Web Services seem to be the best solution to EAI so far. Developments have been fast but the technology is not mature yet. It needs more time as well as improved support from software vendors to build and consume Web Services and to provide the solution that will make everyone's lives easier.