



Web Services - The discovery of paradise



The Xpragmatic View #49
July 2002
by [Marc Buyens](#), Xpragma
marc.buyens@xpragma.com
[url: http://www.xpragma.com/view49.php](http://www.xpragma.com/view49.php)

Over the past months, the economic slowdown continued to have its impact on the financial results of most technology vendors, including our group of EAI vendors. The "sky is the limit" feeling that we knew two years ago has completely vanished. Still, in this negative climate, marketing has to do its job. One of the latest darlings of the hype factories are web services, the next promise of paradise?

It has been a while, but holiday and some other (less interesting) business activities blocked my agenda. Fortunately, these past few months were also rather calm. Hardly any important business event took place in our little world of EAI and BPM. Most of the vendors continued facing disappointing financial results and an economic revival cannot be expected for the next few quarters.

However, even in this dead season, technology vendors and analysts have to try to get renewed attention by bringing some interesting and especially more positive messages. For such marketing purposes, emerging technologies always provide a good basis to build upon and today, web services seem to be the latest and the greatest of these technologies.

This said, there is nothing wrong with web services. It is a technology that has its value and that will conquer its place in the future IT architectures. However, we have to keep everything into perspective. Web services will not solve every problem. Therefore, our first step must be to get a correct understanding of the value web services will bring, but also of the limitations of this approach.

In this View, we will not go into the technical details of web services technology. We assume that our readers are familiar with the concepts of XML, SOAP, WSDL and UDDI. If not, there are sufficient places on the web where you can find such information. Instead, we want to concentrate on the business value of web services.

The basics

In essence, web services are a group of technologies that allow for the existence of software components (on the web or on your intranet) that can invoke each others "services" making use of platform and application independent standard protocols.

This seems a rather straightforward definition, yet the interpretation of how this is achieved is quite diverse. Currently, I do have a collection of almost 50 articles and white papers that discuss the subject of web services. While reading all of this content, it is difficult to believe that all of this talks about the same concept. Consequently, for the average observer who will only get in touch with a minor portion of this content, it all depends upon what sources you will read. If you are lucky, you might get the opinions that are applicable for your specific business situation. If not, you are very likely to base your positive or negative assessment upon

incomplete or biased information.

Therefore, let us have a look at the basics of web services. Doing so, a good starting point might be to ask the question why web services were "invented" in the first place.

Essentially, web services find their roots in the fact that, with the growing success of the Internet, developers wanted to be able to do RPC (Remote Procedure Calls) calls over the Internet.

There are all sorts of good business reasons for doing this but unfortunately, the HTTP Internet protocol was not designed for invoking RPC calls. Therefore, something new was needed and in essence, this is what the SOAP component of web services now offers. The other components of web services (WSDL and UDDI) both have their specific role and value proposition, but are less important for the discussion in this View.

This capability of doing RPC calls over the Internet, using platform independent standard protocols, creates enormous technical and business opportunities. Having a single, standards-based way of communicating between heterogeneous applications is bound to speed up the development of distributed applications, while reducing the overall development cost.

In addition, it facilitates the creation of "composite" applications that make use of independently written software components, whose functionality, interfacing methods and presence are described via protocols such as WSDL and UDDI. Especially in B2B environments, this can be the basis for real supply chain integration.

All of this seems to solve forever a number of issues of traditional IT development. Unfortunately, these basic characteristics also determine the basic limitations of the approach.

What you ask is what you get

The most important one of these limitations results from the basic ambition of SOAP: doing RPC calls over the Internet. When you talk about RPC, you talk about a request/reply paradigm, which implies that both services (the requester and the invoked service) have to be "present". In addition, the calling application (or service) is blocked while awaiting the reply.

For numerous types of applications, this is an acceptable approach. However, this also introduces severe limitations that make it an unacceptable approach for numerous other applications. In the past, this has been part of the reason for the emergence of things such as message queuing, publish/subscribe technology, message brokering and in essence, EAI as we know it today.

Therefore, the often-raised question whether web services will replace EAI, is a non-issue since both technologies offer different approaches to solve different business issues and therefore rather complement each other. This said, it is clear that there is much appeal in a completely standards-based approach such as web services, compared to the current world of EAI that is still largely dominated by proprietary technologies.

As a result, it can be expected that EAI solutions will extend their reach into the world of web services. This is a very natural evolution since for most EAI solutions web services only represent an additional protocol and application environment to support. In addition, it can be expected that the type of standards-based approach of web services will also find its way into the newer EAI solutions. Compared to this, extending the web services request/reply approach to also support other forms of interaction is a more fundamental evolution.

Simplicity versus complexity

A second major limitation of web services is, again, the result of its initial ambition. Compared to the existing EAI suites, the ambition of the initial approach was much more limited. EAI was created and designed for an environment wherefore it was assumed that it would be completely heterogeneous and without any consensus regarding protocols. The fact that the initial EAI technologies were not Internet focused is thereby less important.

Compared to this, the initial web services approach was essentially consensus-based, building upon the promise (or hope) that everybody would adhere to the same protocols and standards. This was a viable approach given the fact that the underlying basis was essentially HTTP, which was already the de-facto standard, complemented with an XML-based approach, which was also generally accepted, be it that the various flavours still are inhibitors for a real XML standardisation.

As a result, the web services approach does address much less "complexity" compared to the EAI solutions. In the EAI approach, things such as availability, transactional integrity, security and management are all key requirements given the reality and the complexity of the supported environment. None of these are really part of the initial design of the web services concept. Nevertheless, there is nothing that will inhibit a further extension of the web services solution with such aspects of security, manageability, availability, etc.

So, will web services technology evolve to incorporate missing EAI capabilities or will EAI become web service-enabled? We expect to see both evolutions. However, we do think that it is better to look at this dilemma in a different way.

Components

People tend to forget that what we discuss today as "web services" are essentially two distinct value propositions. First, there is the idea of isolating or componentising business logic into "services". This is completely identical to what our future view was for the further evolution of EAI technology. Moreover, we already described this in our initial [white paper on EAI](#) where we defined a component layer that would have characteristics such as:

In our messaging based EAI approach, business components are logical objects that represent the execution of one or more business functions. Being a logical object, they are instantiated by one or more (physical) executables that perform the programming logic. These instances receive requests for processing and send replies via well defined but not necessarily uniform (messaging) interfaces.

Business components are fully logical elements, without an understanding of the underlying technical infrastructure. Components manipulate "logical data" (meta-data). Therefore, any business component is free to interact directly with any other business component.

This is the same value proposition we find in the web services approach. The only difference is the fact that we now have a number of standards (XML, WSDL and UDDI) that facilitate the definition and interaction of components.

Interaction

The second element of the current web services paradigm is the way the transport of requests and replies is done and how services are invoked and interact. Here, web services and EAI seem to be in direct competition, web services offering the lowest cost approach whereas EAI is often overkill.

In reality, it is not a matter of choosing either approach. Both approaches are valid but typically not in the same business context. As a result, we do expect that both web services technology and EAI will complement each other to offer a portfolio of approaches to build composite applications upon. Most likely, this will lead sooner or later to the integration or consolidation of both approaches.

Conclusion

In summary, the question is not whether web services are a better approach than an EAI message broker or vice-versa. The real question is what approach to take for a specific business

problem. Web services have the advantage of lower cost, smaller footprint and more standardisation, whereas EAI is the better approach for the more complex environment.

Both approaches do support the concept of business components that can be "linked" to create composite applications. So from an architectural point of view there already is a lot of consensus, although none of the approaches already offers a perfect solution.

In addition, neither of the approaches is the correct one for a real enterprise-wide deployment. Web services simply do not offer a valid solution for such extended environment, whereas EAI can do the job, but is overkill for a number of environments.

Therefore, the likely approach is a combination of both approaches, making use of an EAI backbone that is web services enabled, complemented by web services deployments for tactical or less critical business environments. In many cases, this will mean that the deployment of web services will be limited to intra-company applications. The latter might seem strange given the initial ambition of the approach, but as most enterprises now deploy their applications over an intranet, this is not too different from the initial web ambition.


Have fun...

About the author




Marc Buyens is analyst, management consultant and owner of Xpragma. Marc started Xpragma in 1999 after a 20+ years career in the IT sector. Today, he provides advice, training and mentoring services focusing on the intersection of technological evolution, organisational change and business strategy: a messy world of unfulfilled promises.

 LinkedIn profile: <http://www.linkedin.com/in/marcbuyens>

 Follow on Twitter: <http://www.twitter.com/mbuyens>

© 1999-2009, Xpragma bvba. All Rights Reserved.

Xpragma bvba - Mechelsesteenweg 254 - 2820 Bonheiden - Belgium
Tel. +32-(0)15-340 845 - info@xpragma.com - www.xpragma.com

 RSS feed: <http://www.xpragma.com/english/rss/xpven.xml>