



Service Oriented Architecture and Web 2.0

■ **The IT industry has been busy over the past few years** building up infrastructures based on Software Oriented Architecture (SOA). It is a form of distributed system where software components are linked to each other remotely using standardised data exchange formats, description of service as well as discovery. Use of technologies like WSDL, SOAP, Service registry, BPEL and ESB are very common in SOA solutions. However, much of these assets sits in the backend of most enterprises and are never directly exposed outside an enterprise.

On the other hand, we see a new wave of changes based on Web 2.0 (a term coined by Tim O'Reilly) arriving. We have begun to see richer user interfaces on web applications and developers are adopting simpler and quicker web application development process. We see use of mashups, content publishing and aggregation based on RSS/ATOM, data assessing via REST (representational state transfer) etc. Many of these Web 2.0 sites involve high level of participation from the end users who have contributed large amount of contents. As oppose to SOA, most of the Web 2.0 implementations are expose to the world wide audience outside of enterprises.■

Mr Law Chee Yong
Lecturer, School of Information Technology
Nanyang Polytechnic

1 WEB SERVICES AND SERVICE ORIENTED ARCHITECTURE (SOA)

Prior to the advent of Web Services, there is no lack of technologies that allow software components or services to be executed over the network. OMG CORBA, Microsoft DCOM and Java RMI are some of the prominent examples. These technologies are no doubt powerful and used widely within their respective domains, however, none is able to claim to be universally accepted and certainly does not interoperate with each other easily.

The arrival of extendable markup language (XML) ushered in various technologies that bridge the gap. Web Services Description Language (WSDL), based on XML, allows easy interpretation of the interfaces between software components running on different platforms. It specifies all the necessary details such as the parameters of the operations, data type definition as well as service locations. This allows any client software to access the services offered by the remote software components once the WSDL document is available. Similarly, Simple Object Access Protocol (SOAP), again based on XML, provides the standardised messaging protocol to allow data to be transported between the client and the targeted software services.

Other technology soon followed that allowed software components to invoke each other in a seamless and standard manner. For example:

- XML Schema for type definition.
- Universal Discovery Description and Integration (UDDI) for registration and discovery of services.
- Business Process Execution Language (BPEL) for service choreography.
- Enterprise Service Bus (ESB) for message translation and routing.

The key issue here is that all these technologies are based on standards and no one single party owns them outright. This contributed to the wide acceptance of web services and later SOA.

For once we have universally accepted technologies for building loosely-coupled, remotely accessible software components. Realising the potential, many organisations started setting up task force or committee to look into implementing SOA. Large investments were made in training and upgrading of skills, purchasing tools for SOA development and revamping software architectures. All these occurred at about the same time that applications on the Internet are getting interesting with the emergence of Web 2.0.

2 WEB 2.0

Web 2.0 came about quickly and seemingly out of nowhere, a steady growing stream of successful web applications emerged. These web-based applications shared certain characteristics that made them very successful. This is actually pretty amazing considering that this follows the burst of the dot com bubble in 2001.

As cited in many articles, Tim O'Reilly noted the common characteristic of the web applications and used the term Web 2.0 to characterise them, but what actually does it mean? It was fuzzy at first and led him to clarify it again in 2005. He suggested Web 2.0 to be a platform, "a set of principles and practices".

The Network Effect

The network effect refers to the effect where the usefulness of the system increases proportionally with the number of users (an example is the telephone network). A Web 2.0 application skillfully harnesses this effect. eBay, del.icio.us, Flickr are prime examples of this.

User Contributed Contents

A common term usually associated with Web 2.0 is social networking. This relies on users instead of the host or application providers to provide the content. Wikipedia, the online encyclopedia relies on users to submit entries. Facebook or MySpace has huge followings because of all the content contributed by individual users.

Change in Software Development Cycle

In traditional software development, software are designed, developed and put through several rounds of testing, from alpha to beta to final release. The cycle repeats and new version is released after each cycle. Web 2.0 applications

seem to break from this trend; some services are seemingly in its beta phase perpetually, with constant improvements and bug fixes. End users do not seem to mind as long as these applications work.

Use of Lightweight Programming Model

Perhaps due to the nature of the Internet, where dynamics are constantly changing, Web 2.0 applications need to constantly evolve as well. It is easy to see why Lightweight programming model is favoured by many software developers. Developers opted for the simplicity of using RSS, REST and scripting languages. Such pragmatism allows them to quickly develop applications and dramatically shorten the software development cycle.

Assembly and Reuse

Similar to SOA Web Services, Web 2.0 application routinely mix and match services to produce highly innovative solutions. Such reuse is so common that a new word "mashup" is coined. Mashups refer to web applications that combine data from more than one source into a single integrated tool. An example will be a web application that embeds a map provided by Google or Yahoo and provide real-estate labels from a separate data source.

Rich User Interface

Perhaps the most obvious characteristic of Web 2.0 web application is the rich user interface. Prior to Web 2.0 applications, most users are used to simple interactions like filling in a form in text boxes and clicking buttons. Web 2.0 brings with it highly interactive and user controls that rival desktop applications. Drag-and-drop is becoming commonplace and first time user will surely be impressed by the fisheye widget provided by DOJO. It is even possible to mimic the whole windows desktop that runs the web (see Ext JS).

3 WEB 2.0 VERSUS SOA

It would seem that SOA and Web 2.0 are built on very different philosophy. One is built on the strength of standardisation where almost every aspect is carefully deliberated and agreed upon by the industry through a standardisation process. On the other hand we have Web 2.0 applications being developed with the main goal of getting it out of the door as soon as possible.

On one hand, SOA applications are usually associated with enterprise backend processes with the goal of integrating the various loosely-coupled services into business processes. On the other hand, Web 2.0 is more associated with serving consumer with highly interactive and impactful visual coupled with appealing user generated contents. Of course there are certain traits that the two seems to share, for example they are very much open (not tied to a particular vendor) and support cross-platform service integration, either by design or by nature. Figure 1 summarises the differences as well as the similarity between the two.

Interestingly, the argument that the two can work together and complement each other has merit. While each of the set of technologies is used for different purposes and targets different segments, it is precisely this difference that makes the two well suited for each other. It is a matter of time that the potential of combining Web 2.0 and SOA is realised.

To see that value, we can envision how Web 2.0 will be able to extend the values of SOA solutions that have already been implemented in many enterprises.

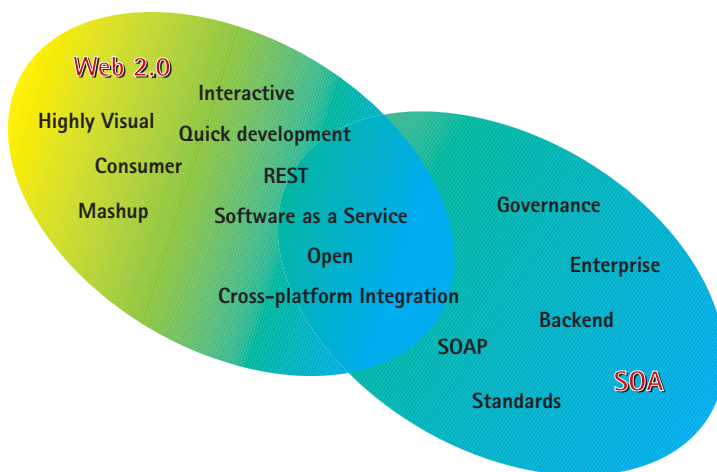


Figure 1: Differences and Similarities between Web 2.0 and SOA

Improved Accessibility

Many enterprises hold significant amount of data and SOA services that are developed over the years. Such services and data are currently very much limited to consumption within the enterprise or with business partners. A globalised SOA in which web services are exposed globally were envisioned but has not been, or may never be, realised. Enterprises can expose such services using tried-and-true methods as seen in Web 2.0 applications, providing a simpler interface via REST and RSS or wrap services in widgets. In short, such data or services should be easily and quickly accessible. Simplicity is the key here.

Remix and Reuse

Remixing and reuse of services was an important goal of web services but Web 2.0 seems to be having a lead here. While enterprise agonised over the granularity of each web services in the SOA solutions, Web 2.0 applications went straight ahead to develop coarse-grain solutions that are pretty much self-contained and worked well. Such approach seems to be working, the prime example being Google map. We can see how simple and easy it is to embed a full-featured and highly customisable mapping solution within a web application.

SOA web services should follow Web 2.0 approach and provide the same level of ease of use. Fortunately, due to the loosely-couple nature of the services in SOA, building such solutions should not be too costly in terms of effort. Web 2.0 brings with it many technologies that are geared toward reuse of software components, sometimes even by

non-programmers. An example is the use of widget that wraps around services so that they can be easily reused. It is not a far stretch to be able to wrap widgets around services provided by SOA, this will improve ease of use and boost reuse in SOA services.

Rapid Development

The industry has been moving relentlessly to simplify and improve the speed of software development. We see developers favouring Plain Old Java Objects (POJO) instead of Enterprise Java Bean (EJB). IBM Websphere sMash released last year to the public provides quick and much simplified development environment. The new environment allows developer to use scripting languages like groovy and PHP. Common tasks like database access, generation of contents using XML or JSON are drastically simplified. For example, refer to the following lines of codes as shown in Figure 2.

```
def onList()  
{  
    request.view = "JSON";  
    request.json.output = Manager.create("books").queryList("SELECT * FROM book", Book.class);  
    render();  
}
```

Figure 2: Codes to retrieve information

The above shows some groovy codes to retrieve information from a MySQL database table, saved it to Java Bean objects and send the object details as an JSON stream back to the client.

Due to these changes, it is a matter of time before we see more SOA systems accessing services using REST as much as using SOAP. In fact, this is probably already happening, AXIS2, the popular Apache Web Service engine already supports exposing web services using REST.

Innovation

Maybe because they are relatively new, but when we look at the new crop of Web 2.0 applications, one thing is certain, they are challenging our perception of web applications and business models. We see many innovative applications being offered, for example, we are being offered gigabytes of storage spaces without paying a single cent, instant searches of the contents of scanned books from the library, viewing maps of any place in the world, sharing bookmarks, networking with friends and strangers and so on. At the same time, we see many innovative development framework being developed to provide high visual impact application. An example is the Google Web Toolkit that provides a nice and clean alternative of developing AJAX application using Java instead of the sometime messy Javascript. Innovative use of SOA services currently locked behind many enterprises should allow enterprise to reap greater rewards.

4 CONCLUSION

Web 2.0 is taking shape and looks like it is here to stay. It brings with it changes and a new way of looking at developing solutions within an enterprise. SOA solutions should be able to integrate well with Web 2.0 technologies due to the fact that both are highly reusable. The two complement each other very well, one provides the secured and highly reliable backend services and processes while the other provides a highly interactive and visually appealing front-end. Together they form a complete solution which may be the way solutions are developed for years to come.