

# The SOA Magazine

## Feature Article



### Services as Web Services: "Are We There Yet?"

### How Web Service Technology Stacks Alone Cannot Fulfill the Goals of SOA

by David Webber

Published: December 10, 2007

(SOA Magazine Issue XIII: December 2007, Copyright © 2007)

[Download this article as a PDF document.](#)

*Abstract: All too often developers and architects focus on the technology of Web services when exploring options for how to realize SOA. Various technology stacks have emerged, listing and combining recommended Web services standards and extensions into frameworks that promise to provide everything needed to realize true service-orientation. While a fundamental consideration, these frameworks can be misleading in their scope and in their role as part of a "true" SOA initiative. This article challenges the importance of Web services technology stacks and highlights what they tend to miss and what you should take into consideration when planning your SOA project.*

#### Introduction

A common (and reasonable) question that we've all encountered has been "Can't I just use plain Web services to do SOA?" This is often followed by something along the lines of: "How does my legacy stuff fit into a SOA? I don't want to lose all that investment."

Practitioners often turn to the Web service technology stacks established by vendors or industry professionals in various publications as a guiding beacon illuminating their path toward a true SOA implementation.

The problem with these pre-defined frameworks is that they focus too much on the "nice to haves" and not enough on what you really need to pay attention to when establishing a service-oriented architecture. The omissions from these proposed frameworks are also instructive. For example, standards relating to business process definition (like WS-CDL, BPMS, XPD, ebBP) are often missing, along with anything pertaining to e-Business technologies and legacy systems and emerging open source deliverables.

So where do all these other pieces fit in? What we really need is not a technology standard laundry list, but an understanding of real-life functional capabilities and how they relate to business service delivery.

Google usually offers instant salvation – but when searching on "SOA Specifications" and "SOA standards" you begin to understand that while technology component standards are important to implementing SOA solution components, they are not by themselves an instant answer. Precisely because SOA is offering a technology neutral approach that is not dependent on any particular syntax it requires certain capabilities with functional characteristics and operational behaviors that can be expressed as templates and patterns.

So the question may not be "Are we there yet?", but "How do we know where we are now, and where we should go next?" The roadmap is more important than the brand and location of certain gas stations on the way. How does this all relate to defining the business process definitions involved? It is the guiding service-orientation principles that can help us solve the puzzle in a systematic and robust way that will persist as a business solution well beyond the latest versions of vendor preferred syntax and angle bracket technology.

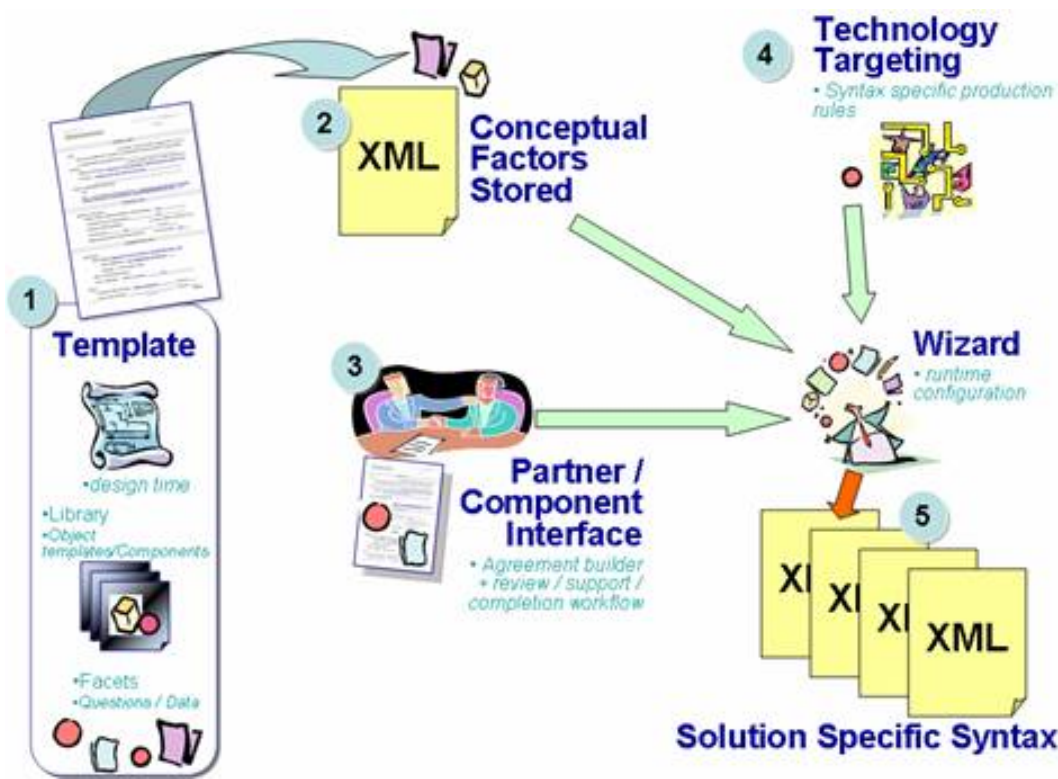
Without applying sound principles we see only too often technology-driven point components that do not fit well into a

SOA and indeed often cause integration issues and costs of use because of the "that's the syntax they are using and we can't fix that right now" syndrome.

### Using a Template-Driven Abstraction Approach

What is immediately clear when studying popular Web services frameworks is that technology is not only a moving target but also that the criteria used to assemble these frameworks is inevitably going to reflect the biases and backgrounds of those creating them.

Fortunately one of the true strengths of XML is the adaptability it provides for the content stored in it. What this means is that once you understand the concepts and driving factors underlying a particular technology aspect, then you can create an abstraction template that suits your own requirements (and biases). From such a template you can then generate target XML syntaxes that meet the needs of specific service implementations while providing a buffer against technology changes and a better way to manage the related technology stack. Figure 1 shows how such a template method works, going from a business view of the functionality into a machine runtime set of coordinates and parameters.



*The common pieces of a template-driven abstraction approach.*

Let's consider the techniques and considerations to use this approach and define corresponding business solution needs.

### Applying SOA Principles and Concepts

When designing reusable services the first need is to define the business processes and business goals associated with the eventual service implementations. At this point the actual implementation technology is a distant speck on the horizon. The only real concern from the business perspective is that technology capable of supporting the business requirements is available and at a cost point that makes the implementation viable. Indeed from the SOA stance the design needed is deliberately technology neutral, and one that can support interchangeable methods for implementation.

Too often this is not the case and technologies are selected first before other requirements are taken into consideration. Such technologies may exclude important business options; for example, it may simply be stated that it is too expensive to support more than the existing solution stack and hence no further technology analysis is required.

This stance of course fades rapidly when presented with the reality of business environments. For example, a key customer may demand a different technology approach or a partner community may be largely unsophisticated and therefore require less complex and freely deployable tools. Even internally, different divisions may already have selected and deployed different versions and flavors of base Web services. Or, they have decided upon different migration schedules necessitating a pluralistic solution.

Therefore one should plan from the outset to have a solution stack that is adaptable and that presents the correct amount of supporting functional capabilities to match the business scenarios rather than attempting to brow-beat all business needs into a one-size-fits-all technology base.

This applies to legacy system support as well. While typical Web service connections use WSDL and SOAP for synchronous query-and-response exchanges, many existing systems use batch-oriented, asynchronous messaging. Therefore, there may be more suitable solutions (such as ebXML perhaps) that are also SOAP based, but provide for the type of formal delivery options that introduce less risk and security exposure.

Again this is not a radical concept; messaging gateways pre-date SOA and provide for flexible interchange configuration and the secured, scheduled delivery of services between participants. What these old-style messaging gateways lacked, though, was transaction handling flexibility, business process awareness, and especially contextual behaviors that the promise of modern SOA platforms offers. Achieving such aspects, however, requires an understanding of the solution stack that enables the corresponding features.

### **Standardized Web services and SOA**

The original formal W3C stance had Web services defined as SOAP messaging based on WSDL and XML Schema definitions for the transactions and service actions [REF-1]. The W3C is primarily focused as an organization on the promotion of the World Wide Web; their vision is decidedly Web sites and content exchanges and not specifically business system orchestrations or enterprise information systems nor service-oriented architectures.

Therefore it comes as no surprise that the original W3C Web service model is missing much in terms of SOA solution delivery (although searching on "SOA" at the W3C site does show how much work their members are doing on related items). The W3C really does not see itself as fulfilling a primary role in defining such business-centric methods and hence other standards defining organizations (such as OASIS and OMG) have augmented original W3C deliverables for this purpose.

The notion of a Web service is thus not static but continually evolving and embracing new transport methods and techniques. This can be seen from the recent trend of experimenting with REST-based interfacing and development of REST interchange profiles.

Where does this leave implementers of service-oriented solutions in terms of being able to prepare robust technology components to deliver on their business goals and enterprise needs? Let's find out by making some comparisons.

### **Understanding the Web Service Technology Stacks**

The following two figures show the comparison between a basic Web service WSDL and a typical SOA suite.

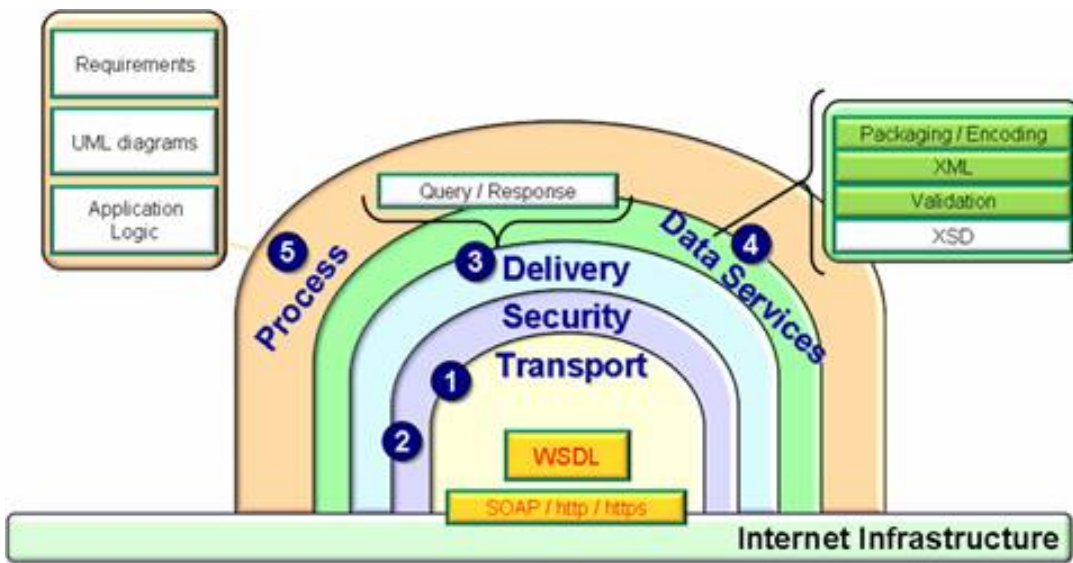


Figure 2: A technology stack for WSDL-based Web services.

In this approach the five core aspects: transport, security, delivery, data and process are all limited in functionality and leverage familiar existing Web technology developed primarily by the W3C. The business process model is limited to simple query-and-response interactions with little state management beyond succeed-or-fail interactions.

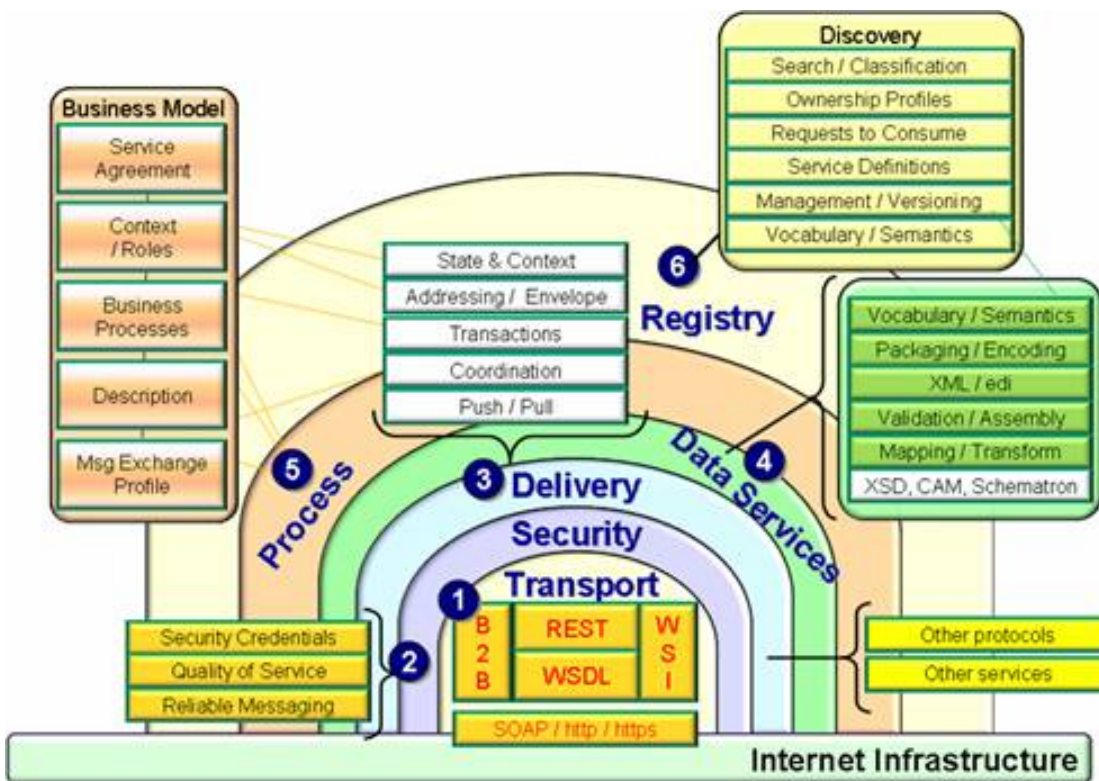


Figure 3: A technology stack for a Web services-based SOA.

In an SOA stack, the basic WSDL model has been expanded to provide for broad and complete-use architecture. A variety of transport modes are supported and security includes more than just physical access control, while delivery methods encompass a full set of options.

Data services are not limited to just WSDL and XML Schema and the process and business modeling capabilities are likewise expanded. The new capability of registry services has been added to allow for the management of all extended functions in support of multi-step information exchanges.

The W3C WSDL approach in Figure 1 relies simply on what XML Schemas can provide, whereas a registry is able to manage semantics and meanings along with service definitions. This all provides for much richer business process

support of extended interactions between both internal enterprise systems and external partners.

The question of "Services as Web services are we there yet?" becomes more of one relating to the level and sophistication of the implementers and their business goals and how they choose to leverage the existing capabilities in the such a technology stack.

### **Leveraging Business-Centric and Technology-Agnostic Methods**

When building any automated solution, there are two general approaches you can take. One is technology-centric where configuration, control, and operation are driven purely by engineering factors such as network type, security control codes, and time-to-retry factors. These requirements, all of which engineers comprehend, are closely tied to a particular technology. As a result, business staff will have little knowledge thereof.

Conversely a business-centric approach looks at factors that relate to the operational effectiveness of the solution and are neutral to the actual underlying technology. Hence, a business user may choose from profiles that select the type of customer from "occasional buyer," "frequent buyer," or "main partner," and then the associated connect type would be "infrequent," "daily," or "always on." Each such profile would result in different delivery profile and connection configurations at the technology level. This business-centric approach allows for the technology to be tailored to whatever the business needs dictate.

Storing business configuration partner profiles as the default format allows the technology control parameters to be generated via scripts that allow new technology (and new versions of technology) to be swappable and pluggable.

Familiar technology (such as MS Word or Excel documents) can be used, allowing business users to complete and confirm their requirements directly with their business partners. This is an important part of setting expectations for business service levels, roles, and responsibilities between participants. Those same documents can then, via their respective scripting tools, actually generate the technology level configurations to direct the physical systems accordingly. Many of these design options are currently being ignored by the engineers who see it as their responsibility to set up new connections without the need to involve or even communicate with the business staff.

When we now revisit the notion of pre-defined Web services frameworks and stacks, we can see how removed these representations are from the need to support actual business operational solutions in the real world.

### **Anticipating Future Needs**

Just like the changing transport method landscape that Web services have helped evolve, the business nature of information exchanges themselves are also adapting and evolving.

These information exchanges are tied into the business processes both internally and externally and return us to the notion that the definition and understanding of the business processes first and foremost provide the foundation for an SOA adoption.

Again, however, defining business processes needs to occur in a technology-neutral manner that the business staff can confirm and validate. For example, one such approach is to use the visual icon based BPMN notation supported with simple spreadsheet tabulations of steps and systems. (Often this is a huge business challenge in itself because business processes can contain vague and unclear steps.)

Business processes also can include external and internal aspects. As with the transport methods chosen, this may require different technology selections to achieve the right business operational results (for example sharing one set with external partners, while using tighter coupling for internal systems within the enterprise).

A practical study of the successful marriage of these factors is the Helena Chemicals project [REF-2] implemented using Oracles' Web services technology stack. Here, the internal and external interfaces are presented quite differently, but use the flexibilities inherent in XML to bridge across the two. Again, the ability to create partner profile templates that configure across technology components proved crucial to successfully managing this implementation.

### **Conclusion**

So can we ever really build and catalogue an industry-level Web services technology stack that is definitive in

meaningful ways? Perhaps if we approach this from the stance of business-centric templates related to capability aspects needed to deliver the business solution. But then people are most likely to find that the technology stacks they want are in the templates themselves and the associated business process definitions. The technology selection details are then widely flexible depending on the answers those templates provide.

## References

[REF-1] W3C Web Services Architecture ([www.w3.org](http://www.w3.org))

[REF-2] Helena Chemicals project ([dotnet.sys-con.com/read/318418.htm](http://dotnet.sys-con.com/read/318418.htm))

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[Home](#) [Past Issues](#) [Contributors](#) [What is SOA?](#) [SOA Glossary](#) [SOA Books](#) [Legal](#) [RSS](#) Copyright © 2006-2007 SOA Systems Inc.

All Rights Reserved