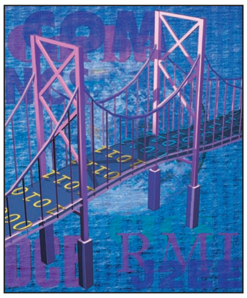

WS-Nonexistent Standards

Steve Vinoski • *IONA Technologies*

Beginning with SOAP, Web Services (WS) have now been around for about five years. They have reached the point where mainstream customers are beginning to use them or at least seriously consider doing so. Based on my own experience, most successful WS applications reside inside the enterprise, where they help integrate disparate systems. Typically, these integrations are based on fairly simple uses of SOAP and Web Services Description Language (WSDL), with homegrown security also present in some cases.

What hasn't occurred in that same timeframe is significant WS standardization. The numerous WS specifications introduced by various parties over the past few years, which I'll discuss in more detail later, show that there's a clear desire to fill out the WS architecture stack. Unfortunately, converting those specifications into actual industry standards is elusive. Although polls and surveys seem to indicate that companies are poised to spend more on WS development in 2005, the lack of actual WS standardization could hurt the industry.

WS-This and WS-That

The specifications (collectively known as WS-*) are numerous and daunting. A coalition of developers and architects from BEA Systems, IBM, and Microsoft authored most of them, though different specifications also include contributions from several other smaller companies. Because the same author companies didn't write all the specifications, at least two different lists exist. (You can find complete lists of the WS-* specifications at http://msdn.microsoft.com/Webservices/understanding/specs/ and www106.ibm.com/developerworks/views/Webservices/standards.jsp.) If you look at those lists, you'll see that, collectively, the specifications cover just about every WS-related topic imaginable:

- WS-Addressing
- WS-Attachments
- WS-BusinessActivity
- WS-Coordination
- WS-Discovery
- WS-Enumeration
- WS-Eventing
- WS-Federation
- WS-Inspection
- WS-Manageability
- WS-MetadataExchange
- WS-Notification
- WS-PolicyFramework
- WS-Provisioning
- WS-ReliableMessaging
- WS-Resource
- WS-Security
- WS-Topics
- WS-Transactions
- WS-Transfer

This is not a full list because some areas have two, three, or more specifications devoted to them. The extent of the topics on this list is impressive, but the list's completeness also raises a few questions. How do these specifications fit together? Are they all really necessary? If they are necessary, how and when will they become actual standards?

Anyone for WS-Architecture?

In his doctoral thesis, Roy Fielding, cofounder of the Apache Software Foundation and chief scientist of Day Software, stated that "a software architecture is defined by a configuration of architectural elements — components, connectors, and data — constrained in their relationships in order to achieve a desired set of architectural properties."[1] Presumably, all the WS-* specifications fit into someone's idea of an overall architecture. In that case, and also presumably, an architecture document — perhaps titled

"WS-Architecture" — should have been one of the first WS-* specifications published. As you can see, however, it's missing from the list.

Naturally, Microsoft, IBM, BEA Systems, and other companies publishing WS specifications all have ideas of what the WS architecture stack should look like. Not surprisingly, their ideas differ slightly. The problem with company-specific WS architectures is just that: they're company-specific. Without an architecture that enjoys broad industry agreement, companies try to push their own agendas. They agree on some areas, but diverge in others. The result is that the WS-* specification list represents a union of the competing companies' architectures, complete with duplications, overlaps, and dead ends.

I was a charter member of the World Wide Web Consortium's (W3C) Web Services Architecture working group (WSAWG), formed in 2002. Many members had high hopes for the group because it was clear even then that a WS architecture with broad industry consensus would help guide further development of needed WS standards. Unfortunately, by the time its charter ended in early 2004, the WSAWG had produced only a note describing the few architectural areas in which it had made some progress — not a WS architecture specification.[2] I knew early on that the group was doomed when it took two iterations, each approximately three months long, to reach agreement on a basic definition for a Web service.

Part of the WSAWG's inability to make progress was due to the arguments between those who focused on the "Web" part of Web services and those who focused on the "services" part. To the Web proponents, the Web's architecture, known as Representational State Transfer, or REST,[1] was already sufficient to support Web services; the WSAWG simply needed to fit WS into it. Others disagreed, citing that the Web's primary orientation was toward serving hypertext documents to interactive browser sessions; thus, it was insufficient for supporting application-to-application conversations and integration. This unsettled argument remains a nearly daily discussion topic on several weblogs (blogs) around the world.

Overall, both sides' intransigence continued unabated throughout the WSAWG's lifetime, making it difficult for the group to find consensus on any issue. Unfortunately, the WSAWG's ineffectiveness appears to have soured peoples' tastes for further attempts at defining an industry-standard WS architecture, given that no standards body has chartered a new open WS architecture working group.

In September 2004, Microsoft published a useful document that explains how its portion of the WS-* landscape fits together.[3] It provides a decent introduction to the overall WS stack and clearly describes how various stack aspects relate, at least at a high level. The document could be improved, however, because it doesn't explain all the WS-* specifications — specifically (and not surprisingly) leaving out those that Microsoft didn't help develop. Also, it stays at a relatively high level, so it won't help if you really want to understand the details of any particular WS-* specification.

How do these specifications fit together? Are they all really necessary?

Do We Need All These Specs?

With no agreed-upon standard WS architecture specification, we know that not all of the WS-* specifications fit together. Thus, not all of them are necessary, given the likely overlap and inconsistencies among competing specifications that address the same functional areas.

Early in 2004, for example, a group that included Microsoft published WS-Eventing, and a group that included IBM published WS-Notification.[4,5] As both aim to support event-based Web services, they significantly overlap conceptually, but naturally differ at the detailed level. Fortunately, in this particular case, the competing groups seem to be moving toward agreement, given that IBM subsequently signed on as an author of the WS-Eventing specification. It's unclear, however, whether consensus is also building in other overlapping areas.

In a wide-ranging discussion across several blogs in September 2004, some posts pointed out that numerous specifications define other areas, such as the Internet and the World Wide Web, without those areas suffering from standards overload or confusion. For example, Chris Ferris of IBM noted that the Internet Engineering Task Force (www.ietf.org) has issued nearly 4,000 requests for comment. However, comparing IETF RFCs to WS specifications is flawed for several reasons. For one, the IETF RFCs date back to 1969, and newer ones have replaced older ones in several cases. Also, not all RFCs are actual standards — some are simply informational documentation. Most important, though, is that the IETF is open to anyone who wants to participate. Conversely, closed groups — primarily a few employees from large companies such as IBM and Microsoft — assembled most of the WS specifications.

Others in the blog discussion pointed out that only vendors and suppliers need to worry about all the WS-* specifications; they argued that users should simply rely on development tools to shield them from each specification's details. Unfortunately, for reasons too numerous to fit into this installment, this is poor advice. Vendors and users must understand these specifications in detail if there is any hope of building viable platforms and applications that use those platforms. Some also argued that people need not use all of these specifications on every project, but should use only what they need. In the end, that is certainly true, but you still need to understand the specifications before you can understand what you need and what you don't.

Moving Toward Standards

There are several ways to create a standard, but standardization is difficult and time-consuming work, regardless of the approach. Traditionally, if you wanted to form a new standard, you approached a standards body to convince it to create a new working group. This group would work to build consensus around a viable approach suitable for standardization.

Today, some, including the vendors who authored the WS-* specifications, view the traditional standards-development approach with great disdain. They say it takes too long and is too risky, given that one disgruntled party can derail an entire standardization effort. Some even cite the failed W3C WSAWG as proof that the traditional approach doesn't work.

To get around these problems, WS-* authors appear to be taking a different approach toward standardization:

1. Write a specification and make it publicly available.
2. Invite interested parties to one or more private workshops where they can learn more details about the specification and provide feedback.
3. Iterate steps 1 and 2 until chosen feedback from the workshop participants has been incorporated, and the specification is considered finished.
4. Submit the specification to an official standards body with the hope of fast tracking it to actual standardization with minimal changes.

Overall, this approach reduces the number of participants involved, which can be a good thing because it reduces the overall volume of communication required to create the specification and resulting standard. However, it can also reduce the resulting standard's effectiveness, even rendering it useless, because it circumvents at least some of the process of building consensus by not being a truly open process. A standard that is not generally agreed on is a standard on paper only.

A standard that is not generally agreed on is a standard on paper only.

Ideally, we should base standards on implementation experiences, and users — not vendors — should write them. When a vendor writes a standard, the product that the vendor eventually wants to build and sell is often what becomes standardized. Sometimes, they even wait to build such products until the standardization is nearly complete, to avoid wasting effort building something that will get changed during standardization. The end result can be standards that are full of holes and ambiguities due to the lack of actual implementation experience.

Significant user participation in the process is the key to developing good standards. Many users know precisely where the pain points are, what standards they need to address them, and why. Their desire to avoid vendor lock-in means they often see the big picture better than vendors, who typically want only to standardize their own platforms. Often, users can devise abstractions that better allow for multiple implementation possibilities than vendors can, because vendors are tied too tightly to their own preferred approaches.

The short-circuited standardization process described earlier seems to place users at a disadvantage, favoring only those vendors who control them in the first place. I am doubtful that viable WS standards will result from the current approach. Returning to the more traditional standardization approach described earlier and ensuring that these standards are driven at least as much by users as they are by vendors are the keys to achieving the broad consensus necessary to turn the WS-* specifications into practical industry standards.

References

1. R.T. Fielding, *Architectural Styles and the Design of Network-Based Software Architectures*, doctoral dissertation, Dept. of Computer Science, Univ. of California, Irvine, 2000; www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.
2. D. Booth et al., "Web Services Architecture," W3C Working Group Note, 2004; www.w3.org/TR/ws-arch/.
3. L.F. Cabrera, K. Christopher, and D. Box, "An Introduction to the Web Services Architecture and Its Specifications, version 1.0," Microsoft, 2004; http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnWebsrv/html/introwsa.asp.
4. S. Vinoski, "Web Service Notifications," *IEEE Internet Computing*, vol. 8, no. 2, 2004, pp. 86–90.
5. S. Vinoski, "More Web Service Notifications," *IEEE Internet Computing*, vol. 8, no. 3, 2004, pp. 90–93.

Steve Vinoski is chief engineer of product innovation for IONA Technologies. He's been involved in middleware for 16 years. Vinoski is the coauthor of *Advanced Corba Programming with C++* (Addison Wesley Longman, 1999), and he has helped develop middleware standards for the Object Management Group (OMG) and World Wide Web Consortium (W3C). Contact him at vinoski@ieee.org.