Workflow and Web Service Standards

Keith D. Swenson
Fujitsu Software Corporation
3055 Orchard Dr.
San Jose, CA, 95134, USA
kswenson@fsc.fujitsu.com

Workflow and Business Process Management (BPM) standards would greatly help workplaces by allowing multiple systems to be easily integrated. Such standards have been under development for a long time, resulting in what seem to be very few real results. There are number of reasons why these standards seem like they can be developed quickly, but in practice they are rather difficult to agree upon. It reminds me of the Dartmouth Summer Research Project which was hoped to solve the few remaining unsolved barriers to artificial intelligence – in the summer of 1955 [6]. Almost half a century later true AI still eludes us. This is surprising only because intelligence (the natural kind) seems to come so easily to us. Similarly, business processes (the natural kind) seem so easy to us that the lack of standards seems unreasonable. There are a number of reasons for this, but before discussing the inherent difficulties, let's start with a quick overview of the history and the current status.

The Workflow Management Coalition (WfMC), which was established in 1993 by a collection of workflow users and vendors, was the place where the first concerted efforts were made to standardize interactions of long running transactions. By 1995 the WfMC had produced a standard set of terms and definitions, and a standard reference architecture [11]. The reference architecture includes the famous interfaces 1 through 5 which describe 5 different ways that external systems might wish to interact with a BPM system. By 1996 interface 4 had been defined using SMTP messages. Interface 1 had been realized as a standard process definition language called WPDL [10]. WfMC members had worked with the Object Management Group (OMG) to create a standard workflow interface for CORBA environments [7].

This was right around the time that XML was invented. A number of WfMC members (IBM, Netscape, FileNet, Oracle, HP, and a number of others) formed a working group to create a standard architecture for systems that support 'long running interactions'. Even today a BPM system is the best example of formalized long running interactions. The result of this working group was the Simple Workflow Access

Protocol (SWAP) which was submitted to the Internet Engineering Task Force (IETF) in August of 1998 but was never formally adopted [2, 8].

The basic concepts of SWAP are remarkably simple: every different type of ‘long running interaction’ that can be invoked remotely is given its own address in form of a URL. In order to start such a transaction, an HTTP request is made to that address passing a standard XML message that provides the context data with which to start the process. This starts a process instance, which has its own URL, and returns the URL of the process instance to the initiator. From this point on, that URL can be used to get the current status of that process instance, including whether it has completed or not. If the initiator provides an observer URL at start time, then immediately upon completion of the remote process the remote system will make a standard request to the observer URL with the results of the process.

While SWAP was available and implemented by a small number of vendors, it languished like many other workflow standards short of full adoption. The internet at that time was not mature, and the idea of linking processes across the internet was not seen as important. SWAP was invented before the Simple Object Access Protocol (SOAP), so naturally it was incompatible with SOAP when that was finally proposed. Nobody could be sure how the standards would play out in the long run.

WfMC took up the cause, simplified the protocol and defined all aspects of it in a more rigorous fashion. The result of this effort was called Wf-XML. Version 1.0 was released in 1999, and version 1.1 followed shortly thereafter, which is still the current version [5, 9, 12]. There has been an effort to produce a Wf-XML version 2.0 based on SOAP, incorporating XML Schema and other recent standards. Initial results of this effort can be found in a document named Asynchronous Web Services Protocol (AWSP) at www.awsp.info. Work is still ongoing in this area. WfMC also advanced their WPDL into a new XML based revision of the language named XPDL [13].

While all of this was going on, another group was developing SOAP, which focuses on the encoding and packaging of XML messages for the use in Internet message exchange. Standardizing the basic envelope, header, and body structure allows these common features to be reused in all the protocols built on SOAP. The development of SOAP has followed a direction very similar to CORBA: Interface definitions were needed so the Web Service Definition Language (WSDL) was developed. Strong typed structures were needed so XML Schema was introduced. A naming service was needed, so UDDI was created. Reliable messages were needed, so WS-Reliability was invented. And so were a dozen other proposals to fill in for the way that large scale distributed systems had traditionally been developed. Even the subject of ‘long running transactions’ was attempted in form of the OASIS BTP specification. Web Services became an accepted concept.

Then, in the summer of 2002, the term “web services choreography” came into popular use. Choreography refers to the idea that multiple requests may be made to a web service in order to, for example, start and monitor the status of something. Choreography is the specification of the calls that can be made, in what order, at what times. If you remember that a business process definition is a specification of what jobs can be done in what order at what time, you can easily see that a ‘choreography’ is very similar to a business process definition.

In August 2002 IBM and Microsoft announced their Business Process Execution Language for Web Services (BPEL4WS) [3]. Backed by both dominant players in the web services domain, and designed as the best parts of IBM’s and Microsoft’s internal projects, it seemed likely that BPEL would become the language of choice of process definitions. But, this standard was not even submitted to a standards body. Microsoft and IBM reserved the right to change it at any time, without open review. Furthermore, there were strong implications that BPEL embodied intellectual property which would have to be licensed from the companies under unspecified terms.

Sun, BEA, Oracle and a few others knew that an open standard would have to appear before there would be widespread adoption. They got together with others from the BPMI and proposed the Web Services Choreography Interface (WSCI, pronounced “whiskey”) [1]. This coalition promised to submit a specification to a standards body for an open review process. The World Wide Web Consortium (W3C) was the preferred choice, and after a long, somewhat political struggle, a working group on Web Services Choreography (WS-Chor) was formed. This groups started with a big bang at a face-to-face meeting, where Microsoft showed up on the first day, and then publicly announced withdrawal two days later. Since then the group seemed to be occasionally mired in discussions of language theory and other esotery.

Then IBM and Microsoft started a working group at the Organization for the Advancement of Structured Information Standards (OASIS) in order to standardize BPEL. The first few meetings covered little more than licensing issues (which are still unresolved at the time of this writing).

These groups claim to be about choreography: are they the same as SWAP, AWSP, or Wf-XML? Not exactly, but there are some overlaps. It seems that the web services world is sharply split into two worlds: the REST-ers and the SOAP-ers.

REST (which stands for REpresentational State Transfer) is a design philosophy for information systems. The term was first used by Roy Fielding in his doctoral thesis from UC Irvine on highly scalable system design [4].

SOAP, which started as a simple message format, has become much more than that, to the point that now it has become the very symbol of what the REST community does not like about the way that web services are evolving. Here is the reason that it is important for us to understand this: the REST-ers believe that if you structure the information correctly, there is no need for a choreography; while the SOAP-ers believe that if you have choreography, you do not need to worry about the structure of the information.

The debate between REST and SOAP is surprisingly similar to the debate about object orientation that raged 20 years ago. (Has it really been that long?) REST says that everything is a resource, and has a distinct web address. In its purist form, it only supports HTTP, and the only supported operations are GET, PUT, POST and DELETE. Getting a resource gives you an XML structure. Web locations are objects, and you send the operation to the resource. SOAP, on the other hand, uses an address to fix the location of an API call. Each operation has an address, and if you want to talk about an object, you pass a reference (or a copy) of the object to the operation. So there we go: it is object-oriented vs. procedure-oriented all over again.

A second aspect of the debate has to do with tight-coupling vs. loose coupling. The SOAP/choreography people tend to move toward tighter and tighter coupling. Moving past strongly typed interfaces, they have advanced to strongly choreographed sequences of calls. REST-ers strive for loose coupling which can be as easy to set up as typing in the URL, and this makes run-time linking practical. Strong coupling is a benefit because you can catch more mistakes at design time before the system is deployed. The problem with strong coupling is that you need a design tool to do the linking. For example: if a partner has defined a strict interface/choreography for interacting with them, you can build a system to link to their system, but if another partner comes along with a slightly different (or maybe wildly different) interface, you are forced to go back to the design tool in order to link up with this new partner. The strong coupling is more complex, which is OK when you have one or two partners, but what if you have 172 partners? What about 10,000? The complexity scales geometrically with the number of points of connection. Early hypertext systems had this problem, but the WWW was successful because its loose coupling kept the complexity from increasing to unmanageable levels.

It is my observation that tight coupling is good for internal systems over which a single entity has control, while loose coupling is necessary for bridging between organizational boundaries. This leads me to believe that tight coupling is probably a good idea in EAI style internal integration, while B2B is best served by a completely different approach.

It should be noted that SWAP, Wf-XML, and AWSP are very REST oriented. The proponents of REST claim that a RESTful system can scale to size of the internet; indeed the World Wide Web became successful precisely because it is based on this design principle. A company can have a web site; if it gets too busy, half of it can be moved to a new server, and once the links are patched up nobody even notices the difference. This is not so with SOAP. Once you have the declaration of where the API is located this becomes the single place where all calls must be made. Of course, people are working on how to make clustering or other forms of distribution work to get scalability, but my point is that this *is* work, and it does not come inherently from the system design. Wf-XML can scale to internet scale workflow without problem.

Most of us, however, are somewhere between the REST/SOAP extremes. We like SOAP because it gives us a common structure to hang the pieces on, and allows us to hook into other work, such as security, transactions, etc. We like WSDL because it defines a common structure to define our messages and operations in. But WSDL requires that you list the locations of the web service addresses, a critical stumbling block. The trend in the choreography groups is toward exposing the complexity of the internal interactions, when you would rather expose a simplification. It is my impression that this trend is being powered by vendors who make tools that attempt to automatically tame this complexity. They will show how their tools can read in a WSDL (or BPEL) file, digest it, and automatically generate the glue code to link things together. Because they can automatically handle this complexity, there is no need to try to simplify things. But notice that to make a new link you are forced to go back to those same design time tools.

If a process instance is exposed as an address that is accessible from the internet, it can be manipulated with a simple set of commands, and there is little need for a custom protocol for manipulating it. If an activity (such as "Approve Document") is a point on the web, then again, you can complete an activity using a generic "Complete" command, instead of a custom "DocumentIsApproved" command. With the location approach, all this information is available at run time for browsing or otherwise determining what is happening. This makes it very easy to link things at run-time without a design tool.

The net result: those people pushing for WSDL/BPEL or BPML do not see any need for a Wf-XML/SWAP because WSDL can describe any interface and any sequence of messages. Wf-XML or SWAP represent just one of those possible sequences. At the same time, those who like loose coupling, see that it can all be done with a generic interface like Wf-XML/SWAP, and there is little or no need for BPML/BPEL. Clearly, these camps will not agree on a single approach.

The most reasonable strategy is to recognize that there is a place for both the tightly-coupled and the loosly coupled. Both are assets in different situations. Tightly coupled WSDL/SOAP system, along with the plethora of associated support standards will begin to dominate the intranet in order to bridge between systems with an organizational boundary. But look to the loosely coupled REST based systems to cross organizational boundaries, and to scale up without limits. The most entertaining spot to watch, in the coming months, is going to be in the interaction of these these two domains, at the boundary, where the internal systems meet the external systems. Finally, be patient. Don’t expect a “Summer Session” to resolve the few remaining unresolved problems in the field of BPM standards.

References

[1] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy*et al.*, *Web Services Choreography Interface (WSCI) 1.0*, W3C Note, W3C, 2002-08-08 2002.

[2] G.A. Bolcer and G. Kaiser, "Leveraging the Web to Manage Workflow," *IEEE Internet Computing*, vol. 3, no. 1, 1999, pp. 2-5.

[3] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana, *Business Process Execution Language for Web Services, Version 1.0*, BEA, IBM, Microsoft, 2002-07-31 2002.

[4] R.T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, Doctoral Dissertation, University of California, Irvine, CA, 2000.

[5] J.G. Hayes, E. Perovian, S. Sarin, M.-T. Schmidt, K. Swenson, and R. Weber, "Workflow Interoperability Standards for the Internet," *IEEE Internet Computing*, vol. 4, no. 3, 2000, pp. 37-45.

[6] J. McCarthy, M.L. Minsky, N. Rochester, and C.E. Shannon, *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*, Research Proposal, published August 31, 1955.

[7] Object Management Group, *Workflow Management Facility Specification Version 1.2*, Document Number bom/00-05-02, OMG, 2000.

[8] K. Swenson, *Simple Workflow Access Protocol (SWAP)*, IETF Internet-Draft, Internet Engineering Task Force, August 7, 1998, expires February, 1999. 1998.

[9] R. Weber, "Workflow-Interoperabilität über das Internet mit dem Standard Wf-XML," *Wirtschaftsinformatik*, vol. 45, no. 3, 2003, pp. 345-348.

[10] WfMC, *Interface 1: Process Definition Interchange Process Model*, Document Number WfMC-TC-1016-P Version 1.1 Final, Workflow Management Coalition, 1999.

[11] WfMC, *Terminology and Glossary, 3rd Edition*, Document Number WFMC-TC-1011, Workflow Management Coalition, 1999.

[12] WfMC, *Workflow Standard - Interoperability, Wf-XML Binding, Version 1.1*, Document Number WFMC-TC-1023, Workflow Management Coalition, 2001-11-14 2001.

[13] WfMC, *Workflow Process Definition Interface - XML Process Definition Language. Document Status - 1.0 Final Draft.*, Document Number WFMC-TC-1025, Workflow Management Coalition, 2002-10-25 2002.