Trends & Controversies
Jan/Feb 2003 issue of IEEE Intelligent Systems
Web Services - Been there done that?

Don't go with the flow: Web services composition standards exposed

W.M.P. van der Aalst
Dept. of Technology Management, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, w.m.p.v.d.aalst@tm.tue.nl[1]

The recently released Business Process Execution Language for Web Services (BPEL4WS) is said to combine the best of other standards for web services composition such as WSFL from IBM and XLANG of Microsoft. BPEL4WS allows for a mixture of block structured and graph structured process models thus making the language expressive at the price of being complex. Although BPEL4WS is not such a bad proposal by itself, it is remarkable how much attention this standard receives while the more fundamental issues and problems such as semantics, expressiveness, and adequacy do not get the attention they deserve. Having a standard is a very good idea. However, there are too many of them and most of them die before becoming mature. A simple indicator of this development is the increasing length of acronyms: PDL, XPDL, BPSS, EDOC, BPML, WSDL, WSCI, ebXML, and BPEL4WS are just some of the acronyms referring to various standards in the domain. Another problem is that these languages typically have no clearly defined semantics. The only way to overcome these problems is to critically evaluate the so-called standards for web services composition, i.e., Don't go with the flow!

Web services composition

There are two trends coming together in the world of E-business that are creating both opportunities and pressures to automate business processes across organizational boundaries. One is the technology push created by enabling technologies taking XML-based standards and the Internet as a starting point. The other trend is the need to improve the efficiency of processes from a business perspective. After the dotcom crash there is a pressing need to truly utilize the potential of Internet technology by automating business processes across enterprise boundaries. The goal of web services is to exploit XML technology and the Internet to integrate applications than can be published, located, and invoked over the Web. A typical example of a web services application is the Galileo system that connects more that 42,000 travel agency locations to 37 car rental companies, 47,000 hotels, and 350 tour operators.

To truly integrate business processes across enterprise boundaries it is not sufficient to merely support simple interaction using standard messages and protocols. Business interactions require long-running interactions that are driven by an explicit process model. This raises the need for web services composition languages such as

[1] The author would like to thank Arthur ter Hofstede, Bartek Kiepuzewski, Marlon Dumas, and Petia Wohed for contributing to the results mentioned in this paper.

BPEL4WS, WSFL, XLANG, WSCI, and BPML. These languages are also known as web services flow languages, web services execution languages, web services orchestration languages, and web-enabled workflow languages. Before discussing BPEL4WS and the likes, we focus on the typical technology they are building on.

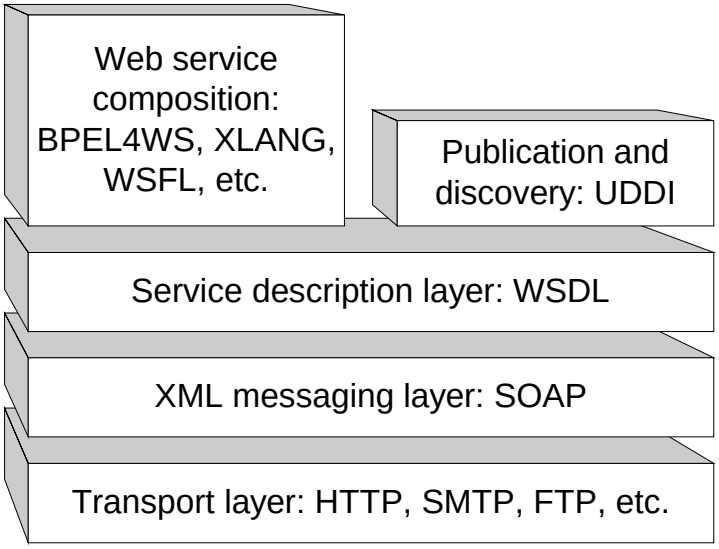

Figure 1: Overview of web services technology.

Figure 1 shows the relation between web services composition languages and other standards such as SOAP, WSDL, and UDDI. SOAP (Simple Object Access Protocol) is a protocol for exchange of information in a decentralized, distributed environment using typed message exchange and remote invocation. It is an XML-based protocol that consists of three parts: (1) an envelope that defines a framework for describing what is in a message and how to process it, (2) a set of encoding rules for expressing instances of application-defined datatypes, and (3) a convention for representing remote procedure calls and responses. SOAP can potentially be build on top of any transport layer, e.g., an HTTP-based infrastructure. WSDL (Web Services Description Language) is an XML format for describing network services based on a standard messaging layer like SOAP. A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows for the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding, and a collection of ports defines a service. UDDI (Universal Description Discovery & Integration) is the definition of a set of services supporting the description and discovery of: (1) businesses, organizations, and other web services providers, (2) the web services they make available, and (3) the technical interfaces which may be used to access those services. Simply put: UDDI can be used to build “yellow pages” for web services. At this point in time, there seems to be consensus on the use of SOAP, UDDI, and WSDL. Therefore, we assume these standards to be in place in the remainder.

Web services composition languages build directly on top of WSDL. A language like BPEL4WS both provides and/or uses one or more WSDL services. Note that a WSDL

service is composed of ports that provide operations. Each operation receives a message (one-way), receives and sends a message (request-response), sends and receives a message (solicit-response), or sends a message (notification). WSDL services and the corresponding operations are glued together to provide composed services. To glue such services together a process model is needed to specify the order in which the operations are executed. A web services composition language provides the means to specify such a process model. An important difference between WSDL and a language like BPEL4WS is revealed when considering the states. WSDL is in essence stateless because the language is not aware of states in-between operations. The only state notion supported is the state in-between sending and receiving a message in a request-response or solicit-response operation. Any technology supporting a web services composition language will have to record states for processes that are more complex than a simple request-response. Only by recording the state it is possible to determine what should/can be done, thus enabling long-lived business transactions. This triggered the development of languages like BPEL4WS, WSFL, XLANG, WSCI, and BPML.

Overview of so-called standards

The recently released BPEL4WS specification builds on IBM's WSFL (Web Services Flow Language) and Microsoft's XLANG (Web Services for Business Process Design). XLANG is a block-structured language with basic control flow structures such as sequence, switch (for conditional routing), while (for looping), all (for parallel routing), and pick (for race conditions based on timing or external triggers). In contrast to XLANG, WSFL is not limited to block structures and allows for directed graphs. The graphs can be nested but need to be acyclic. Iteration is only supported through exit conditions, i.e., an activity/subprocess is iterated until its exit condition is met. The control flow part of WSFL is almost identical to the workflow language used by IBM's MQ Series Workflow. This may be surprising given the fact that this workflow language is very different from most languages. For example, the so-called "Death-Path Elimination" allows for the so-called "Synchronizing merge pattern". This way routing is not restricted to explicit AND-joins and XOR-joins as in most workflow products. Although this is a nice feature it is quite "exotic" and not supported by most systems. Although the correspondence between the WSFL standard and IBM's workflow product may be surprising for people not involved in the standardization process, the correspondence between WSFL and MQ Series Workflow can easily be explained by the fact that both languages are defined by the same set of people (most notably Frank Leymann). Similar comments can be made for XLANG and Microsoft's BizTalk Orchestrator. XLANG is completely based on the current middleware solution of Microsoft and therefore hardly qualifies as a "standard".

Unfortunately, BPEL4WS, WSFL, and XLANG are not the only standards that have been proposed in recent years. Sun, BEA, SAP, and Intalio have introduced another candidate for web services composition: WSCI (Web Service Choreography Interface). Intalio also initiated the Business Process Management Initiative (BPMI.org) which developed the BPML (Business Process Markup Language). OASIS and UN/CEFACT support ebXML (Electronic Business using eXtensible Markup Language). Part of ebXML is BPSS (Business Process Schema Specification), yet another standard having a similar scope as BPEL4WS, WSFL,

XLANG, WSCI, and BPML. The abundance of overlapping standards for web services composition is overwhelming. Some authors refer to these competing standards without clear added value as the *Web Services Acronym Hell* (WSAH).

Outside the web services domain there have been other initiatives to standardize the specification of executable business processes. Most notable is the initiative of the Workflow Management Coalition (WfMC). Since 1993, the WfMC has been active to standardize both a workflow process definition language and the interfaces between various workflow components. In August 2002 the WfMC released XPDL (XML Process Definion Language, Version 1.0 Beta) to support the exchange of workflow specifications between different workflow products. According to Jon Pyke, WfMC Chair and CTO Staffware, XPDL is consistent with BPEL4WS, but goes far beyond the standards for web services composition. Clearly, many people working on standards for web services composition did not benefit from the experiences in the workflow domain. Therefore, comments like "Been there done that" are justified. However, it is also clear that the standards of the WfMC have not been adopted by the workflow vendors. Some of the systems can export to XPDL, but none of them can import XPDL from *another* system and still produce *meaningful* results. One of the reasons is that after working on workflow standards for more than a decade, there is still no consensus on the workflow constructs that need to be supported and their semantics. It is remarkable how many different interpretations of a join construct exist in contemporary workflow languages: "Wait for all (AND-join)", "Wait for first and reset (XOR-join)", "Wait for first and block until all have arrived", "Wait for all to come", etc.

Comparing BPEL4WS, XLANG, WSFL, XPDL, and WFM products

Development with respect to web services composition languages have been mainly driven by software vendors like IBM, Microsoft, Sun, BEA, SAP, and Intalio. This has resulted in an abundance of standards having overlapping functionality. When looking at the standards in more detail, it is clear these are often based on existing products. A good example is WSFL, which is almost a copy of IBM's Flowmark/MQ Series Workflow language. Standards, which involve multiple software vendors, are often a compromise between competing viewpoints. As a result such standards tend to be imprecise or unnecessarily complex. WfMC's XPDL is an example of a standard which is imprecise thereby allowing vendors to have their own interpretation of the standard (thus making the standard useless). BPEL4WS joins viewpoints from both WSFL and XLANG thus making the language very complex.

Given these observations it is useful to look for objective measures for comparing web services composition languages. For the control-flow aspect of such languages, one can use some of the results from workflow research. One way to compare standards like BPEL4WS, XLANG and WSFL is to use the set of workflow patterns available from http://www.tm.tue.nl/it/research/patterns. Each of these patterns corresponds to a routing construct often required when designing a workflow. The whole set of patterns has been used to evaluate and compare about 20 workflow management systems.

Table 1: Comparison of BPEL4WS, XLANG, WSFL, XPDL, and four workflow products.

	BPEL4WS	XLANG	WSFL	XPDL	Staffware	MQ Series Workflow	Panagon eProcess	FLOWer
Pattern 1 (Sequence)	+	+	+	+	+	+	+	+
Pattern 2 (Parallel Split)	+	+	+	+	+	+	+	+
Pattern 3 (Synchronization)	+	+	+	+	+	+	+	+
Pattern 4 (Exclusive Choice)	+	+	+	+	+	+	+	+
Pattern 5 (Simple Merge)	+	+	+	+	+	+	+	+
Pattern 6 (Multi-choice)	+	-	+	+	-	+	+	-
Pattern 7 (Synchronizing Merge)	+	-	+	-	-	+	+	-
Pattern 8 (Multi-merge)	-	-	-	-	-	-	-	+/-
Pattern 9 (Discriminator)	-	-	-	-	-	-	-	+/-
Pattern 10 (Arbitrary Cycles)	-	-	-	+	+	-	+/-	-
Pattern 11 (Implicit Termination)	+	-	+	+	+	+	+	-
Pattern 12 (Multiple Instances Without Synchronization)	+	+	+	-	-	-	+	+
Pattern 13 (Multiple Instances With a Priori Design Time Knowledge)	+	+	+	+	+	+	+	+
Pattern 14 (Multiple Instances With a Priori Runtime Knowledge)	-	-	-	-	-	-	-	+
Pattern 15 (Multiple Instances Without a Priori Runtime Knowledge)	-	-	-	-	-	-	-	+
Pattern 16 (Deferred Choice)	+	+	-	-	-	-	-	+/-
Pattern 17 (Interleaved Parallel Routing)	+/-	-	-	-	-	-	-	+/-
Pattern 18 (Milestone)	-	-	-	-	-	-	-	+/-
Pattern 19 (Cancel Activity)	+	+	+	-	+	-	-	+/-
Pattern 20 (Cancel Case)	+	+	+	-	-	-	+	+/-

Table 1 shows a comparison of some of the web services composition languages, XPDL and four concrete workflow management systems. The first five patterns correspond to the basic routing constructs (e.g., sequence) one can find in any language. The other patterns refer to more advanced constructs not supported by most standards and products. Every "+" refers to direct support, i.e., there is a construct in the language that directly supports the pattern. A "-" in the table refers to no direct support. Note that this does not mean that it is not possible to realize the pattern through some workaround. (E.g. any of the constructs can be realized using a standard programming language. This does not imply that there is direct support for all workflow patterns.) Sometimes there is a feature that only partially supports a pattern, e.g., a construct that imposes certain restrictions on the structure of the process, and the support is rated "+/-". Without going into details, several observations can be made. First of all, BPEL4WS is indeed the combination of XLANG and WSFL when it comes to supporting the patterns. Second, WSFL and MQ Series Workflow are indeed identical when it comes to process specification. Third, it appears that XPDL is less expressive than BPEL4WS. (In a way XPDL can been seen as the Greatest Common Denominator of existing workflow languages rather than the Least Common Multiple.) Finally, there are relevant differences between web services compositions languages and workflow management systems when it comes to supporting routing constructs. Note that of the four workflow management systems listed only FLOWer is block structured like XLANG. The other three systems (Staffware, MQ Series, and eProcess) are graph based like WSFL and XPDL.

Lessons learned

The site of BPMI.org, one of the organizations proposing a web services composition standard, states that "BPMI.org defines open specifications such as the Business Process Modeling Language (BPML) and the Business Process Query Language (BPQL) that will enable the standards-based management of e-Business processes with forthcoming Business Process Management Systems (BPMS), in much the same way SQL enabled the standards-based management of business data with off-the-shelf Database Management Systems (DBMS)." The goal to obtain standards similar to SQL for web services is ambitious. As history shows such standards do not originate from vendors pushing their own products. Recall that the Entity-Relationship model by Chen and the Relational Model by Codd enabled languages like SQL. Although there are well-established process modeling techniques combining expressiveness, simplicity and formal semantics (cf. Petri nets and process algebras), the software industry has chosen to ignore these techniques. As a result, the world is confronted with too many standards which are mainly driven by concrete products and/or commercial interests. The only way to stop this is to ignore standardization proposals that are not using well-established process modeling techniques. This will force vendors to address the real problems rather than creating new ones.

References

1. W.M.P. van der Aalst, K.M. van Hee. Workflow Management: Models, Methods, and Systems. MIT press, Cambridge, MA, 2002.
2. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. QUT Technical report, FIT-TR-2002-02, Queensland University of Technology, Brisbane, 2002. (To appear in

Distributed and Parallel Databases, also see http://www.tm.tue.nl/it/research/patterns.)
3. W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and P. Wohed. Pattern-Based Analysis of BPML (and WSCI). QUT Technical report, FIT-TR-2002-05, Queensland University of Technology, Brisbane, 2002.
4. F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, S. Weerawarana, S. Thatte. Business Process Execution language for Web Services (Version 1.0). IBM, July 2002.
5. F. Leymann. Web Services Flow Language (WSFL 1.0). IBM, May 2001.
6. S. Thatte. XLANG: Web Services for Business Process Design. Microsoft, 2001.
7. Workflow Managment Coalition. Workflow Process Definition Interface - XML Process Definition Language (XPDL), WFMC-TC-1025, Version 1.0 Beta, 2002.
8. P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Pattern-Based Analysis of BPEL4WS. QUT Technical report, FIT-TR-2002-04, Queensland University of Technology, Brisbane, 2002.