ebPML.org

Carnets de bord

Last June, Intalio, SUN, BEA and SAP have released a new specification WSCI, the Web Service Choreography Interface. Here is the analysis I published on the ebXML lists. I also published more recently an analysis of BPML 1.0 which is now dependent on WSCI.

Summary

As there has been some comments made with respect to the relationship between WSCI and ebXML (BPSS in particular). I review WSCI with the perspective of an ebXML implementer and try to identify what does it mean for me. In particular, I am not trying to identify the merits of WSCI with respect to WSFL or XLANG. If you want a more general review of the spec look at the CoverPages or at Nick Patience analysis.

At the first level, WSCI provides a way to describe the behavior of a Web Service as an interface or an API (from a flow, transaction, correlation perspective...). At the second level, with the concept of global model, they provide the ability to "link" operations in a "collaboration" between the interfaces they defined. The main focus of the specification is application-to-application integration, not in the EAI sense where loose coupling is the primary goal, but in a relatively tightly coupled fashion: namely they address the problem "how two APIs which have been web service enabled cooperate with each other?". The typical example they give is a travel agent / airline integration to deliver a common experience to a customer. The spec does not offer any specific semantics relative to "Business Transactions", they are simply treated as regular system-to-system transaction over the internet. In particular, there is no attempt to guarantee state synchronization between the two parties (or the two applications) using a specific protocol like in BPSS.

I feel that this spec has very limited applications, typically like the one provided in the example. It cannot be used effectively to model general B2B and EAI scenarios. In terms of choreography it does not really bring anything new. Since it is coming directly from BPML, it bears a lot of similarities with XLang (block structured). So far, I still think that WSFL is the best thought out choreography model. The novelty, if that can be considered a novelty is the BPML correlation, transaction and exception model which are used in the context of web services choreography.

There is little substance in WSCI to reach the level of a specification. This is merely a non-refereed paper which look at some deficiencies of the web services specification framework, managed by the W3C, and propose some solution. By contrast, ebXML is a stable set of specifications complementing each other and driven by a global architecture specification. Nobody in ebXML develops "rogue" spec, which main purpose it to make a marketing "coup".

Disclaimer

all the thoughts expressed here only represent my opinion and not the one of my company.

Analysis

General comments

This is the 4th specification dealing one way or another with Web Services Choreography after XLANG, WSFL, WSCL. The introduction tells us: "WSCI provides an important foundation for ... application-application collaboration." So this as an alternative to EAI.. uhm where is B2B here? .. Ah yes they talk about it later ... "These business processes may reside entirely within the confines of a single company, or span across multiple organizations or companies." Yes this is a constant amongst web services people: lump sum EAI and B2B. B2B is just application-to-application integration beyond the firewall, so what's the big deal about B2B? Well, you should try to convince Ford which has 2500 enterprise systems world-wide, or Boeing which has a mere 84 procurement systems that it has to do application-to-application integration over the internet with its tens of thousands of suppliers.

In the overview section of the spec, I notice that they continue to emphasize the same theme: "...interactions between services – either in a business context or not" that statement suggest to me that they have no Business semantics but let's verify that. And they cannot emphasize this point enough: "WSCI does not assume that Web Services are from different companies, as in business-to-business; it can be used equally well to describe interfaces of components that represent internal organizational units or other applications within the enterprise." The problem is that in traditional EAI the business semantics are an unnecessary burden, while no business semantics makes the spec unusable in B2B scenarios. This is where layering seems to be the right thing to do.

The key semantic of EAI is publish/subscribe. This has been established for at least a decade with the developments of MOM and yet, the authors of WSCI try to convince us that key application-to-application integration semantics are: "Most Web Services, however participate in longer conversations, spanning beyond the boundaries of a single operation. In these conversations, the ability to perform certain operations depends upon the previous exchange of messages as well as the way to interpret the content of each message may be influenced by the previous exchange." There are very few scenarios in EAI that can be supported by this kind of approach. You are back to point-to-point integration. I am actually surprised of this approach as Assaf had found a very interesting concept in BPML (Consume and Produce). However, the spec breaks the concept with the global model, it is no longer a pub/sub semantic but rather a binding semantic.

"Business processes are increasingly relying upon collaboration", why didn't I think of that before? I actually would say "almost exclusively". How would you do business otherwise if your processes did not have collaboration points with suppliers and customers alike? This is by all means a very important milestone in the way the authors of BPML look at business processes !

"WSCI is not a "workflow description language"": strange, because all the semantics are coming directly from BPML 0.4. who's wrong here? BPML or WSCI?

In my opinion, collaborations are mostly a B2B thing. Yes you can model some API interaction in point-to-point manner. Yes that looks a bit attractive, but it is rather a step backwards. On the other hand, you don't publish your orders or your invoices hoping someone will subscribe to it, you send them to your suppliers or customers! You also don't care that your supplier "received" your order, what you care about is that your supplier was able to "process" the order. This is the essence of the basis of a commitment. The problem again is that WSCI does not support any business semantics nor any traditional EAI semantics such as pub/sub. It is truly a point-to-point integration specification. We all know the limits of this approach.

Detailed comments on each aspect of the spec

I like the distinction between WSDL (Static interface definition) as opposed to WSCI (dynamic behavior of the static definition) because and API has also a behavior it is not just a series of unrelated calls. As such WSCI brings a lot in giving you the ability to "model" your existing APIs and help you communicate this behavior.

1) Message choreography

"The Action, is the basic construct of WSCI, and it is bound to some WSDL operation. WSCI specifies how multiple actions are choreographed in a simple sequence." An action can be viewed as the usage of a particular operations within a particular collaboration. In particular this allows WSCI to specify that an operation can be called multiple times in a given sequence within the same collaboration. The operation will be executed in the context of different actions.

They continue by saying that "WSCI describes the behavior of a Web service in terms of choreographed activities.", but fortunately they add "the most common atomic activity is action." This is a poor choice of word as activity and action have precise semantics in this context.

We can see directly from the example that WSCI is "asymmetric" in other words, it is designed from the perspective of one side of the collaboration. In real scenarios you need to map the actions of one side to the one of the other. Microsoft has struggled (euphemism) a lot with this asymmetry in XLang, WSFL came out with a more elegant approach and the concept of global model. WSCI "borrow" from WSFL, the concept of global models. Where is the reference to WSFL?

In comparison, ebXML is designed in a totally symmetric manner, such that the business service interfaces of each partner can be configured from the same collaboration definition. That's important because two parties have something to agree on. It is harder to look at the interface of a business partner and say "I agree with the way it works, do you agree with mine?". I think that this is a fundamental flaw of the web services concept when used in collaborative applications (not in Stock quotes), because you deal with twice as much xml, plus you need to glue the pieces together. That is ugly. With that respect ebXML BPSS is far more elegant and efficient. I have shown earlier how you can use ebXML BPSS to model application-to-application collaboration. Furthermore, I have shown in the past that you can take a BPSS specification and automatically generate the skeleton of a business process definition (BPML or other) that will implement the given collaboration. The algorithm is relatively simple to write. Overall, I think that when you describe a "peer-to-peer collaboration" you cannot take the point of view of one side, you have to describe the collaboration without any point of view as a pure message flow. The interface is secondary, almost insignificant, and can be calculated from the message flow.

The control flow is a "block-structured" control flow and is specified with the semantics: Complex processes Sequential execution Parallel execution Looping Conditional execution (all this is coming directly from BPML 0.4)

I think that block structured control flow is not the most appropriate choreography model for collaboration. You have definitely negotiation pattern that would be harder to model with a block structured approach. However, block structured control flow makes it easier to deal with transactions.

> *On a side note, could you explain me something? Just like ebXML you specify the choreography of a collaboration, why is it within a <process> element? Would <collaboration> be more appropriate.*

2) Transaction boundaries and compensation

In BPSS, we did not find the notion of Roll-back appropriate (roll-back is more app-to-app kind of thing). So yes WSCI describes also compensating transaction, which is more the way a business operates. BPSS does not have that, but we did not find it a hurdle to model real world collaborations, as they can be modeled anyways, the only difference is that these business transactions are not "tagged" as compensating transactions. Since collaborations are usually simple, this is not a big issue. This has more value if the collaboration had a lot of steps and anything could happen at any time. In B2B scenarios the "undo" is often in a relatively limited scope. Anyways ebXML is also working at the level of patterns which can be viewed as pre-packaged choreography for doing stuff like negotiation/commitment/fulfillment. When this effort is complete (it is well under way), business analyst will not even have to deal with the choreography, they will choose a pattern and just specify the business transactions of the respective patterns. Personally, I think this is the way to go, you really don't want to transform business analyst in Java programmers specialists of the UML state diagram. Also the agreement will be easier to form as patterns will have a well defined behavior that everybody recognizes.

3) Exception handling

The exception behavior is yet coming from BPML. ebXML BPSS is able to identify exceptions (technical or business) but let the designer of the collaboration specify the course of action. If it looks more appropriate to use the WSCI approach when you are java programmer), from a practical perspective I don't think you gain much there. There is a key semantic that is missing in WSCI, which I think cannot be added: BPSS:acceptanceAcknowledgement which signals that a message was correctly processed by the receiving application. Unless, the web service is "THE" application itself, in which case we go back to a poor integration model. In B2B this is essential to synchronize the state of two companies. In our world, PLM, people exchange large CAD files (we have customer which ships a whole car !). As part of the exchange, you really want to know not only that your message got there (Via a web service), but that it correctly loaded in the receiving application whether it is a CAD system, or even if it has to be reviewed by a user !! If there is any error detected at this stage, the data exchange should be considered null, and the state of both companies could not be synchronized. By that time a web service operation would have long succeeded while a BPSS infrastructure will patiently return the appropriate signals that the state could not be synchronized. These concepts are essential to B2B data exchange but totally absent from WSCI.

4) Thread management

WSCI (like BPML...) also uses the concept of "correlation" to identify the collaboration instance of a given message. I personally dislike this approach and rather carry a cookie around to maintain context. I think this is more a matter or preference. I feel that the cookie is cleaner and more general. You don't get stock if the correlation is ambiguous or need to span several elements of a document. The major advantage of WSCI approach is in multiparty collaboration as shown in their example. But I don't think they fully solve the multiparty collaboration problem just yet. BPSS does not either.

5) Properties and Selectors

The property concept is good, as it decouples the choreography specification from the document formats of the message exchange. BPSS should use that concept.

6) Connectors

Did not find much about this

7) Operational context

Did not find much about this

8) Dynamic participation

Did not find much about this though it can be found in BPML 0.4

Looking at the TravelAgent-N-Airline example

Figure 5-3 is pretty clear about the accomplishments and goal of WSCI. It basically tells us that the choreography of a collaboration can be built by specifying the choreography of the individual interfaces (hence modeling interfaces) and then connecting operations

two-by-two assuming that the choreography of any-two interfaces will always be compatible.

This is actually a pretty strange way of looking at the problem. By contrast BPSS collaboration definition are specified without any particular point of view. It simply look at the message flow between parties. A business partner may decide to support a particular collaboration definition. Its BSI can be pre-set with the collaboration definition alone. Internally it will calculate the interface that it needs to implement (something close to WSCI but not exposed as a standard since you don't need to agree on this). When two parties have decided on a CPA, the technical parameters are loaded into the BSI and MSH to specifically do business with this other business partner.

I don't believe that interfaces should be designed in isolation of each other and later on "connected". The likelihood of this working remains pretty small. Now even if they were designed jointly or in parallel, you would still run the risk (in more complex cases) that that two interface choreography is incompatible. I would rather design a collaboration a la ebXML BPPS, agreed upon by everyone and then have interfaces (APIs) calculated to implement this particular collaboration. An API can often be uses in more than one ways. "Modeling" the interface is good in the sense that you can verify (simulate) that a given collaboration is going to work based on the two interface models. However, it is the collaboration definition which governs the interface and not the other way around. So I still think that it is more logical to have an independent collaboration definition, rather than two interfaces with a behavior glued together with a global model.

I have proposed an alternative web services choreography model (not a specification) based on these principles in the following document. I also show in the document how web services and ebXML business transactions can be used simultaneously in multi-party collaborations.

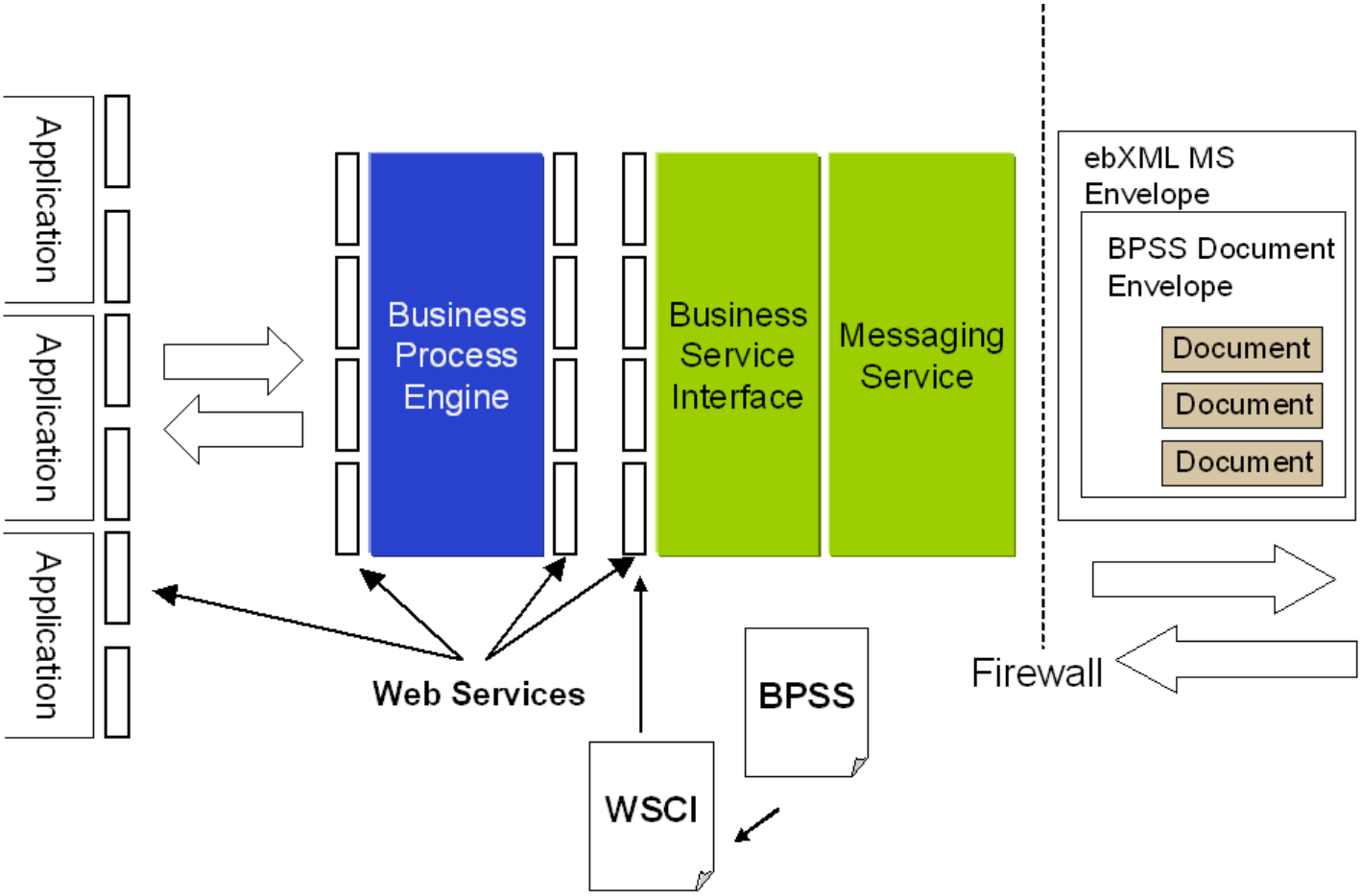
Relationship to BPSS

WSCI provides a possible missing piece for linking ebXML to Business Process Management Systems, as explained by the authors in the WSCI FAQs. WSCI allows me to take a BPSS specification and create an "interface" on one side of the collaboration which can then be used to align a business process supporting this collaboration and defined with BPML. This means that the B2B semantics of business transactions managed by the BSI do not need to be known by the Business Process Engine, and this is a good thing.

In particular, the business process engine only sees the messages and not the signals, and gets notified only when an exception occurs. It would be very hard for instance to define business process definition capable of dealing with all signals of a binary collaboration. This allows to establish the semantics of the business process definition (namely BPML) solely based on web services which is what you want.

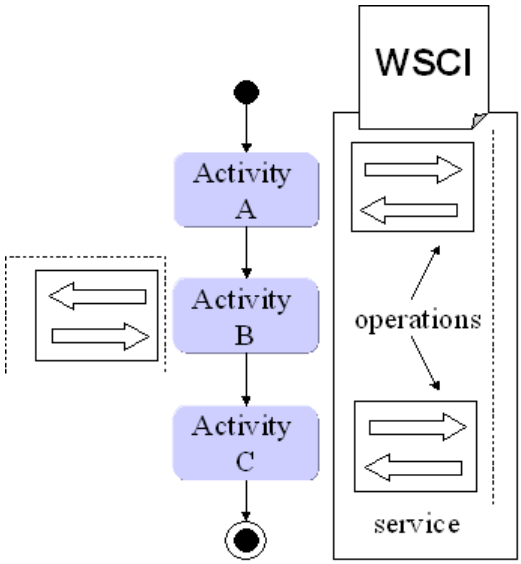
As I explained in chapter 6 of the book, Professional ebXML foundation, you need to wrap the ebXML business service interface into web services in order to let your enterprise systems use the B2b/Ebxml infrastructure. These web services are merely as a proxy representing the business partner services to the business process engine.

A BPSS specification would be used to generate the correspond WSCI interface for the role of this system. The business process definition can then rely on web service based interface like any other which carry out message exchanges when executing operations.



(adapted from "Professional ebXML Foundation", Chapter 6 Wrox 2001).

The process definition would then look something like that:



Conclusion

WSCI introduces a new level of choreography. The "interface (i.e. API)" choreography. We already had the choreography of the business process definition (BPML, XLang, WSFL) and the choreography of the business collaboration (ebXML BPSS). I don't believe that this third level of choreography really brings anything. An interface is an old-view of the world, the client-server view. The server exposes his interface, which has a behavior described by WSCI such that a client knows how to use it. As such this is

complementary to WSDL. However, this does not fit the new view of the world which is peer-to-peer. In a peer-to-peer mode, the interface disappears behind a "collaboration definition" which specifies how the two peers collaborate to achieve a particular purpose. Furthermore, I have shown that this is relatively easy to bind BPML with BPSS, you really don't need a choreographed intermediary level, WSDL is enough.

So yes this is clearly an attempt to compete with the collaboration definition of BPSS using a global model definition between two choreographed API. So far, I believe that this is much easier to form an "agreement" on a real collaboration definition rather than on two interface definitions which may eventually not be compatible. The global model does not guarantee that the two interfaces are interoperable, it is merely a one-to-one mapping between ports types.

I think they fall short of understanding the B2B semantics required for B2B collaborations that ebXML implements (I actually don't think they even attempted to define these semantics, though the other pretend these concepts apply to B2B message exchange). I personally cannot find a customer that told me, no "I don't need the business semantics provided by ebXML (and BPSS in particular)". The context in which this specification is useful is the second example provided in the spec which that 2 distant applications running in two different companies can "cooperate" to give the user a uniform experience. I guess this can be called a form of B2B. This is a case were the relationship involves only a few operations, is highly trusted and changes rarely. Sincerely this could be done with ebXML as well, with the advantage of having real business semantics between the travel agent site and the airline, which are often needed even in a trusted, hard-wired relationship.

"Collaboration" is mostly a B2B concept, and therefore should have B2B semantics such as non-repudiation, legally binding, confidentiality, security, tamperproof, collaboration agreement ... I talk with real customers every day which need these semantics.

When it comes to the relationship between WSCI and BPSS, they are somewhat complementary. In particular WSCI enables a good level of isolation between internal process definition and ebXML BPSS collaboration definition. I don't see anytime soon the collaboration model of WSCI replacing the one of ebXML.

Jean-Jacques Dubray

References

http://wwws.sun.com/software/xml/developers/wsci/faq.html#1q1

http://titan.intalio.com:8080/intalio/wsci/

http://wwws.sun.com/software/xml/developers/wsci/

6311