What are we managing?

Igor Sedukhin (igor.sedukhin@ca.com)

12/6/2003

This write-up represents an attempt to analyze the choices presented to the OASIS WSDM TC to answer the nagging question: “What are we managing in the subject domain of Web Services”. Attached to that is the question: “What is the identification of what we’re managing?”

Summary of choices

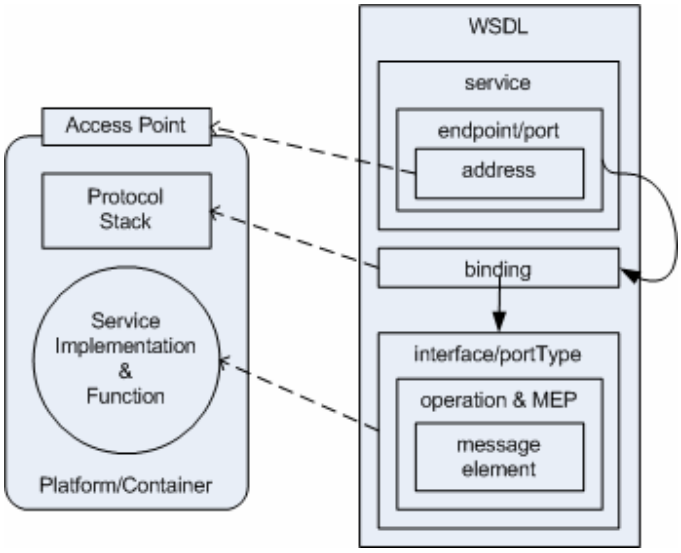
1. Use 80%/20% rule and manage endpoints as described by WSDL 1.1 port components. This is how WSs published with existing tools and platforms are and this is reinforced by changes in WSDL 2.0.

2. Cover a few more cases (e.g. 10% more) by managing AccessPoint/ProtocolStack/ServiceImplementation «aggregate»s. Define such thing, name it, and identify by address (URI) + WSDL binding component pair. Note that this does NOT cover 100% cases and therefore does not solve the problem.

3. Cover the rest of the cases by managing the «aggregate» as in choice 2, but identify it as a “messaging component” by an address (URI) + XML message elements + exchange patterns. This does cover 100% cases and does solve the problem.

Normal behavior

What could be considered the “normal behavior” is developing Web Services and their clients using existing Web Services development tools and running Web Services on existing Web Services platforms. For example, MS VS.Net and Windows, IBM WebSpehere, BEA WebLogic, Eclipse/Netbeans and Axis/Tomcat, etc. Analyzing the “normal behavior” one could deduce the following logical diagram.



Access Point is listening for messages at a certain address (URI). Protocol Stack is capable of processing messages according to the specifics of the protocol (XML/SOAP). Service Implementation is a piece of code that implements the desired Function in response to requests interpreted by the Protocol Stack and received by the Access Point. Service Implementation is run/executed/contained by the Platform. Usually, the Access Point and the Protocol Stack are provided by the Platform.

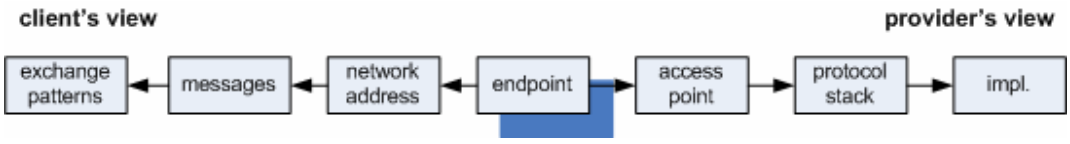
The client of a service experiences its Function via Access Point, Protocol Stack and Service Implementation altogether. If we consider the clients' experience the most important aspect of the service, then it would make sense to find something that could represent it at the service side for applying manageability aspects to.

When described in WSDL (components!, not necessarily WSDL document elements), a service's Function is represented in the interface/portType which is a collection of operations. Each operation is a message exchange pattern in which every message is described by an XML element which actually appears in the messages. Binding describes how a Protocol Stack expects the messages to be formed and sent to the Access Point. Binding is always referring to the messages of a particular interface. Service itself is described as a composition of its endpoints/ports. Each endpoint contains an address of the Access Point and also refers to a particular binding.

In case of the "normal behavior", the clients' experience of a service comes via an endpoint as described in WSDL. The Web service developed with existing tools and run by existing platforms would not be otherwise. The WSDL endpoint always describes the «aggregate» of Access Point, Protocol Stack and Service Implementation which the client is sending messages to and there is no ambiguity here. It will require intentional tweaking of WSDL document to violate the "normal behavior" case.

The lowest subjective assessment is that 80% of the WSs conform to "normal behavior" case, and if we were to abide by 80/20 rule, it would be most natural to manage the endpoints as described in WSDL. Moreover there is a trend among the Web Service platform/tools vendors to make the 80% case as crisp as possible and eliminate the possibility of misinterpretation and misuse (e.g. single interface per service in WSDL 2.0).

Web Services Architecture and WSDL define endpoint as an association between a binding and a network address specified by a URI. It is this association that makes it the nexus point between the client's view and the provider's view. Any deviation from the nexus point makes it unfair to the other side as shown on the following picture.



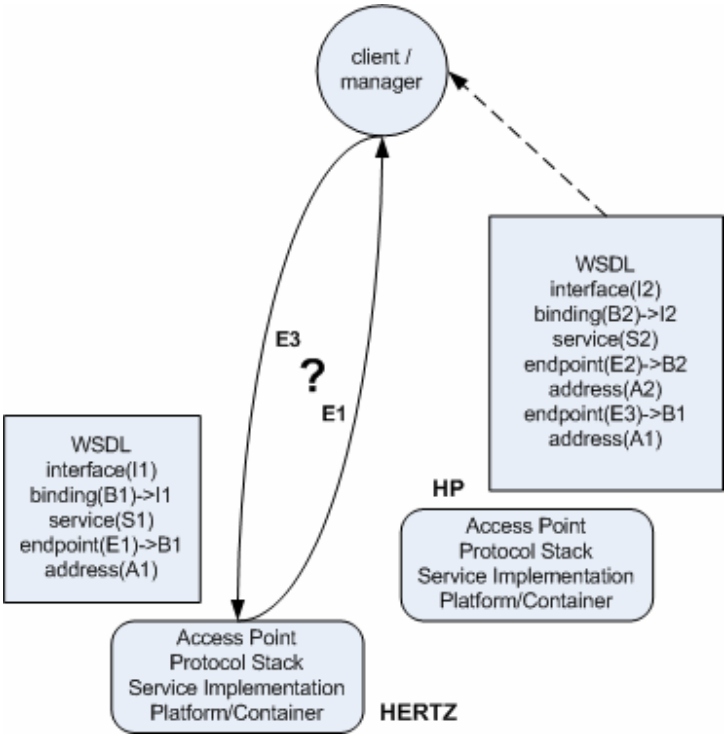
That is not to say that certain other cases are not possible.

Other things that could happen

First of all there would be no homogeneity among all the “non-normal behavior” cases. Consider two variants that present real technical challenges to the “normal behavior”:

1. there could be many different WSDL endpoints (of same of different services) that share same address and same binding definition
2. there could be many different WSDL endpoints (of same or different services) that share same address and same message exchanges, but not the binding definitions

The use case presented to the WSDM TC at the recent F2F by William Vambenepe was to justify variant 1 above. The following is possible with WSDL 1.1, however WSDL 2.0 makes it impossible without both HP and Hertz implementing/binding the same interfaces. And then the question remains why would anyone do something like that instead of merely pointing to or including the Hertz’s service definition…



As one may observe, in these cases, the clients’ experience of a service’s Function may come via many endpoints described in WSDL, however it is the same Access Point, Protocol Stack and Service Implementation «aggregate» that responds to the clients’ messages. And therefore the provider of manageability needs to identify such «aggregate». There are the following possible solutions:

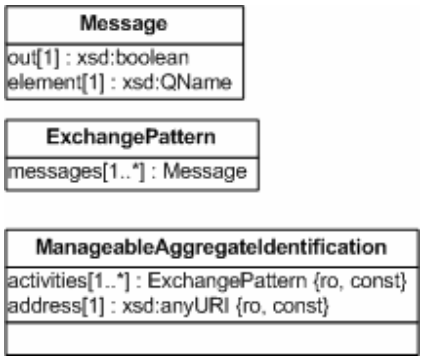
1. If the WS was developed using existing Web services development tools and run by one of the existing Web services platforms, then always identify the WSDL endpoint that describes the WS that was deployed. In the example above, Hertz’s provider of manageability would always return E1 identifier.

2. One could imagine identifying the Hertz’s side «aggregate» by the binding (B1) and the address (A1). In this case the provider of manageability does not need to

worry about existence of E1 and E3, however, arguably, the Hetz's side perspective is that (B1+A1)=E1.

Solution 2 presents another problem: it is incomplete. Applying the same argument of rewriting a WSDL, one could describe the Hertz's side «aggregate» with an endpoint having address A1, but the binding and even the interface could be different (WSDL components). The messages and the exchange patterns, however, will be the same. Therefore (binding + address) is not a sufficient identification of an AccessPoint/ProtocolStack/ServiceImplementation «aggregate». The only way to identify the «aggregate» would be the following.

3. Identify the «aggregate» by an address + message elements + exchange patterns. In this case message element is the XML element that represents the message "payload" and exchange patterns are essentially ordered groups of message elements with directionality attribute. It could be represented by the following UML diagram of an «aggregate» identification manageability capability.



Solution 3 does solve the problem, however the identification of the «aggregate» becomes rather complicated and whether this justifies breaking the 80/20 rule remains to be discussed.