W3C Workshop on Constraints and Capabilities for Web Services
Position Paper[1]

Ken Laskey, MITRE Corporation

There are ample examples, several included in the workshop announcement, of the need for a Web service to express the constraints it will impose in order to allow a user to make use of the service. While the needs in the examples are valid, there are other considerations that must also be supported in order to create an environment where services are discoverable and can be dynamically composed. Consider the following questions:

1. Can a common mechanism be developed to express general constraints of a service as compared to specific constraints related to components of its WSDL interface?

2. Is there a difference between (a) constraints defined by a service that must be satisfied in order to allow access and (b) constraints advertised by a service and which the user would evaluate to decide whether to invoke the service at all?

3. How does providing input to a constraint engine differ from providing input to any other application with a Web service interface?

4. How does the degree of expressivity of the constraint language impact the choice of evaluation engines?

In the following discussion, the metadata associated with a service is assumed to describe many of its properties including (but not limited to) its name, descriptive text and keywords, the entity responsible for maintaining the service, the WSDL interface descriptions, and constraints sets describing different restrictive aspects of the service. It is assumed that a single service may (in fact, often will) have more than one invocation[2] sequence, where these may be expressed in terms of separate WSDL binding and/or service components. For example, there may be separate invocations for different qualities of service (QoS). Multiple constraint sets may also be present, some associated with the service as a whole and some associated with the specific invocation directly. For example, the constraint set for a service may indicate that a secure connection must be established to invoke a service. Once the secure connection is established, the user can choose the WSDL service component for a given quality of service but the request must be checked against the user's ID to see if the user qualifies for that QoS.

It is not yet clear to what extent the partitioning of capability, such as different QoS, should be among separate services with individual WSDL descriptions, separate interface operations, or

---

[1] This paper represents the technical opinion of the author and does not constitute an official position of the MITRE Corporation.

[2] In the following, the term *invocation* will be used when referring to the mechanism to begin service processing so as not to confuse this with the term *interface* which will only be used when referring to the interface portion of the WSDL syntax.

separate service ports/endpoints. While architecturally supported, there are also few examples of using partial WSDL descriptions across multiple services. What does it mean to apply constraints to each of these partitioning approaches?

In considering the way constraints may be used, let's begin by looking at the typical example where a Web service has a constraint set that needs to be satisfied by the prospective service user. First, the description of the Web service must indicate that such a constraint set exists. As mentioned above, the constraint may apply to the service in general or may be specific to the service invocation, and Question 1 asks whether a common mechanism can be applied across all these cases. If the constraint is generally applicable to the service, for example, have certain licensing terms been satisfied, then the constraint mechanism must be expressible outside the WSDL structure and without the user accessing the WSDL. This indicates an XML syntax that is easily located by simple XSLT processing, assumed preferable to requiring a sequence where constraint tag sets must appear first in relation to other XML tags.

If, as in the QoS example, the constraint applies to the particular service invocation, there are a number of possible means of expressing these. One could collect in one place in the metadata all constraints related to the service and all its invocation sequences and could evaluate the applicable constraints at one time prior to trying to invoke the service. This would likely be unwieldy, both in logically expressing the constraint conditions and in managing constraints that are separated from their associated invocation points. If collected constraints appear unattractive, this implies that the constraint should be part of the WSDL component to which it applies. In this case, an ordering of the XML tags may be necessary so that the constraint is the first thing the processor sees upon entering the invocation sequence. This does not preclude using the same mechanism for expressing the constraints incorporated in the WSDL as those applicable to the service as a whole, only a difference in where the constraint set is located. If the means to express the constraints are the same, the same suite of evaluation engines should serve all levels of constraint descriptions.

In looking at the case where the constraints are on the user's authority to invoke the service, any processing agent that would attempt to access the WSDL would first have to investigate whether general service level constraints exist. Once service-level constraints were satisfied, the WSDL would be accessed and sent to a WSDL processor which would evaluate the next level of local constraints before invoking the service. If, as asked in Question 2, the constraints were not just on the conditions surrounding the user access but rather constraints also expressed the underlying assumptions on which the service was based, then the user would have to separately initiate the search for and evaluation of constraints to be sure the service was compatible with current needs. For example, a user looking for a solution for supersonic flow over an airfoil would not use a service whose solver assumed incompressible flow. The user would want to know and, if so, evaluate underlying constraints that would have implications on the applicability of the service.

This indicates that there are at laest two types of constraints which are used differently, and for efficiency, the mechanism for expressing constraints should be capable of indicating whether the constraints are primarily intended for evaluation on behalf of the service or on behalf of the service user. This should only be an indication and not itself a constraint because a service user may be interested in constraints primarily intended for the service as, for example, the user may not want to interact with a service that checks for licensing terms. Thus, both the user and the

service may have use for rules intended for the other party. Although the party initiating the constraint evaluation may change, it appears at this point that the same evaluation tools would still be invoked (possibly as services). Therefore, unless other intrinsic differences are found, the way of expressing the constraints should be common for both service and service user constraints, but the constraint language should provide information on the perceived primary intent. In addition, the constraint language should be able to identify the class of constraint being expressed (for example, licensing) along with a link to the namespace that describes the constraint vocabulary where the term is defined. A standard constraint type vocabulary may be part of a constraint language effort, but the language should include the ability to specify alternate vocabularies.

We see that different parties may require constraint processing and so having the processing itself be available as a service is attractive. Question 3 asks whether a constraint evaluation service is different from any other service. The primary difference would seem to be that the constraints associated with a constraint service must be carefully managed to ensure we do not get into an recursive loop of the constraint service needing to evaluate its own constraints before invoking itself. This is akin to some of the security scenarios where it appears one could forever be calling the security service to ensure the previous call is authorized. This appears to require other safeguards beyond those required for expressing and otherwise evaluating the constraints.

Finally, Question 4 asks about the expressivity of the constraint syntax and to what extent this requires matching with a constraint evaluation engine. This implies several possibilities:

(1) The constraint language should contain well defined levels of expressivity so that the same specification can address multiple levels of sophistication.

(2) The constraint language should support the most general aspects needed for specifying constraints and the language should provide for a rich set of extensions. The general syntax would support accessing a possibly external description of the detail, much in the way that one can point to a specific XML schema.

(3) The mechanisms for expressing and evaluating constraints in the logic and decision support community are diverse enough that for all but the simplest rules there will be a need for using specialized rule languages with specialized processors. In this case, the XML language would identify the rule set of interest, the rule markup being used, and the set of engines recommended for evaluating such rules.

As we move from possibility 1 to 3, the generality and ability to seamlessly interoperate is decreased. However, it may be preferable to sacrifice immediate interoperability rather than select one technical approach whose internal assumptions may eventually limit the constraint language's long-term usefulness across the whole of the Web community. This will require careful consultation between the community seen to be using constraint descriptions for Web-based access and those in the technical community who are developing the evaluation techniques that will be applied.

The following use case encompasses the range of situations noted.

A number of language translation services are made available by several independent service providers. Each provider offers services to handle different language pairs, such as French to English or Dutch to Mandarin, and the language pair is specified in the service metadata. The services are commercial offerings and a condition (present at the top service metadata level) for invoking the service is to have an established account or to provide a credit card. General information about translation capabilities and how to establish an account are readily available at the service Web site but the account constraint must be satisfied before processing (accessing the service WSDL) will continue. Recently, a number of free services have become available and a service user may want to query a number of services to find free ones (services that do not have the account constraint). Thus, in the initial stages of service access, we have the need to specify a constraint fully enough (1) for it to be evaluated and (2) for a user to simply know if a constraint of this type exists.

There are a number of variations of the translation service, each with different capabilities, such as translation of documents written in the formal form of the source language or translation of audio files, or whether the result is a detailed translation or gisting of the source content. From a service user perspective, the requirements and underlying assumptions of each variation are constraints they need to access and evaluate to decide which variation suits their needs. It will be a design issue whether these constraints are maintained together at the service metadata level or whether these will reside closer to the invocation of the variation. These different variations could be implemented in the service component of a WSDL description with multiple ports/endpoints to the different variations. Each service point would have associated metadata (possibly a WSDL extension referencing a translation description namespace) that could describe the service and would convey processing constraints such as whether the appropriate source type (MIME) has been supplied. From the service perspective, the processing constraints could be evaluated before attempting to invoke the service or the needed information could be passed to the service and it would be responsible for its own constraint evaluation.

Finally, as translation services have expanded, a variety of premium services have appeared. One of service providers offers to evaluate the user source material and respond with how its translation engines have performed on similar challenge problems conducted periodically within the translation research community. Another has offered immediate access to human translators if automated processing is not sufficient (i.e. does not satisfy some constraint). A third offers guaranteed secure communications for the exchange of both the source material and the results. Thus, the range of available services and the associated constraints that must be expressed may grow rapidly. With a growing number of communities using the translation services, the underlying infrastructure will need to deal with the challenges of a set of user communities with (1) different vocabularies relevant to their own domains and (2) limited experience with formally expressing or understanding constraints and the vocabulary of the technical community developing constraint evaluation techniques. Semantic negotiation again becomes an issue and the ultimate constraint for using a service may be whether there is sufficient information for the service to understand the user request.

The net effect is that constraints will be used by a wide audience in need of straightforward mechanisms for expressing and evaluating constraints of increasing complexity, and the mechanisms must be flexible enough to support advances in the underlying technology without requiring the vast number of users to be experts in those advances.