# Web Service Capabilities and Constraints in WSMO *

Sinuhé Arroyo Christoph Bussler Jacek Kopecký Rubén Lara
Axel Polleres Michał Zaremba

<firstname.lastname>@deri.org
Digital Enterprise Research Institute (DERI)
Galway, Ireland and Innsbruck, Austria

August 31, 2004

**Abstract**

This paper summarizes the necessities on the semantic modeling of Web Service constraints and capabilities from the viewpoint of the Web Service Modeling Ontology (WSMO) working group. We will give a short motivating use case from the travelling domain which we shall use to classify possible capabilities and constraints to be described on the service and requester side.

## 1 Introduction

Web Services enable communication between distributed systems and integration of applications over the network by supporting program-to-program interactions. Existing Web Services cornerstone technologies such as UDDI [1], WSDL [2] and SOAP [5] provide the basic functionality for discovering (UDDI), describing interfaces (WSDL) and exchanging messages (SOAP) in heterogeneous, autonomous and distributed systems.

In B2C and B2B interactions, additionally to requirements of providing richer description of Web Services functionality, we recognise that Web Services and their clients have constraints and capabilities, which must be addressed to successfully carry business transactions. The motivation of our work is to promote the abstraction of capabilities and constraints from the existing efforts in Web Services community, which still remain focused on syntactical aspects of Web Services execution.

Web Services standards do not provide any mechanism to specify how to include additional information describing capabilities and constraints, which matters in some more sophisticated use cases beyond simple request/response interactions. In this paper we present the viewpoint of the Web Services Modeling Ontology (WSMO) working group on constraints and capabilities.

Inspired by a simple use case for WSMO from [7], which we will introduce in Section 2, we discuss selected aspects which are adequate to address issues of defining capabilities and constraints on both client and service sides. Based on this use case, Section 3 presents a classification of capabilities and constraints from the viewpoints of

*This work is supported by the SFI (Science Fundation Ireland) under the DERI-Lion project and by the European Commission under the projects DIP, Knowledge Web, SEKT, SWWS, and Esperonto, and by the Vienna city government under the CoOperate programme. The authors thank all members of the WSMO working group (cf. http://www.wsmo.org/) for fruitful discussions on this document.

Web Service providers and requesters. We introduce WSMO in Section 4 and present its main conceptual elements. In the same section we briefly identify how the constraints identified in Section 3 can be realized in and influence future extensions of WSMO. We close the paper with some open points for future work not restricted to WSMO in Section 5

# 2 Use case

The use case, taken from WSMO Use Case Modeling and Testing Working Draft [7], presents a scenario for booking international train tickets on-line. The goal is to search and buy a train ticket for traveling from Innsbruck (Austria) to Frankfurt (Germany).

In a nutshell we would like to get a two-way ticket, traveling to Frankfurt on November 15, 2004 and coming back to Innsbruck on December 1, 2004. We would like to depart from Innsbruck in the afternoon and arrive back before 1:00 am. Also, for both of the itineraries we do not want to change trains more than one time. Preferably, we would like to travel in first class in both ways, and, in case there are no trains available to arrive in Innsbruck before 1:00 am the next day, we would make use of the facilities of a sleeping train, thus traveling during the night. Regarding the payment information, the service has to accept MasterCard and we will only disclose our credit card details to services having a MasterCard certificate. Finally, we'd like any information disclosed to the service to be confidential and the data exchanged to be encrypted. The services themselves require payment using a credit card, can guarantee confidentiality, and require encryption from the consumer side.

# 3 Classification of Constraints

In the context of semantic description and discovery of Web Services, we identify capabilities and constraints on both the provider side and the consumer side. To find a service that can satisfy the needs of a client, the process of automatic discovery matches the capabilities of the service against the goal and the constraints of the client and then checks that the client has the capabilities to satisfy all the constraints of the service. In the following we give a (not necessarily complete) classification of the most common such capabilities and constraints.

## 3.1 Capabilities, Constraints, and Preferences

The distinction between a capability and a constraint is not always clear and unambiguous. For example the fact that a service operates only in business hours could be viewed both as a capability and as a constraint. Our definitions of the terms are as follows:

**Capability** expresses a provision by either the provider or the requester. On the provider side it expresses what the service can offer e.g. selling train tickets for Austria. Another example of provider-side capability is the support for different encryption algorithms and confidentiality of the data provided by the consumer. We also identify capabilities on the consumer side, which express what the client can offer but does not really insist on, for example a client can provide his credit card details for billing, but if a service offering free tickets is found, the credit card number will not need to be exchanged; or a client supports a number of different encryption algorithms and will likely only use one of them when communicating with the selected service.

**Constraint** (also known as *hard constraint*) expresses a certain requirement on the partner in a collaboration. On the provider side it is what a service requires to be able to fulfill its capability, e.g. a valid credit card number for billing or a given security protocol to be used. On the consumer side a constraint is a requirement on what the service must be able to provide in order to suit the client needs, for instance providing a train ticket from Innsbruck to Frankfurt on November 15th, 2004, perform encryption, guarantee confidentiality or accept MasterCard. The satisfaction of constraints might not be directly achievable but include a process of *negotiation* exchanging constraints and capabilities in turn, for instance concerning the disclosure of information. For example, a consumer will only disclose his credit card details to services providing a MasterCard certificate.

**Preference** (also known as *soft constraint*) describes a preferred condition affecting the operation. Preferences are usually modeled on the consumer side, where they are used for selecting from among multiple discovered services the one that will best suit the client needs. For example, we might prefer a train service that offers a connection arriving before 1:00 am in first class. While these conditions can be viewed as a "soft" variant of constraints, preferences can also model optimization criteria, for instance keeping the number of train minimal. In case of several independent preferences, measures for prioritizing different preferences might be useful. Preferences are also useful on provider side, for instance for indicating to the client that the use of a secure a connection is preferred although not required.

## 3.2 Functional vs. Non-functional

Another common distinction for capabilities, constraints, and preferences is the division between functional and non-functional properties of services and requests. A functional capability on the provider side describes the purpose of the service (*what* the service does), and a functional constraint is a requirement mandated by that purpose. On the consumer side, a functional constraint describes the goal of the client (*what* the client wants). Conversely, non-functional capabilities or constraints are incidental details specific to the implementation or running environment, irrelevant to the actual purpose of the service or goal of the client but necessary for successful and interoperable communication (i.e., *how* the service can achieve a goal).

In our use case, the services have the functional capability to sell train tickets and the non-functional constraint that the client has to perform encryption; the client has the functional constraint – *goal* – to buy a ticket from Innsbruck to Frankfurt and back for the specific days, non-functional constraint that communication has to be encrypted and the data disclosed be kept confidential, and finally the non-functional capability of providing valid credit card information.

In modelling a concrete scenario, it is often not clear whether a given capability or constraint is functional or non-functional, and in fact it depends on the architect's view of the purpose of the service or the goal of the client. For instance, authentication could be a non-functional constraint of an on-line ticket service, but it would be a functional constraint (maybe, in fact, a functional capability) of a security door-guard system providing certain kinds of authentications.

During automatic discovery, it may be useful to drop the distinction between functional and non-functional because a client requiring something as a functional constraint (its goal) might miss a service that can provide that but views it as a non-functional capability.

# 4 Realization in WSMO

The Web Service Modeling Ontology (WSMO) [4] is a formal ontology for describing the various aspects related to Semantic Web Services, representing the backbone for the development of Web Service Modeling Language (WSML) and Web Service Modeling Execution Environment (WSMX). The objective of WSMO and its surrounding efforts is to define a coherent technology for Semantic Web Services.

WSMO defines the modeling elements for describing several aspects of Semantic Web Services. The conceptual grounding of WSMO is based on the Web Service Modeling Framework (WSMF) [3], adhering to the principles of loose coupling and strong mediation services. The four main components of WSMO are:

**Ontologies** provide the terminology and formal semantics to the information used by all other components.

**Goals** specify objectives that a client may have when consulting a Web Service. They provide the means to express high level description of a concrete task.

**Web Services** represent the functional part which must be semantically described in order to allow their semi-automated use. In the WSMO specification the properties of Web Services are described by means of a (functional) capability and other, non-functional, properties.

**Mediators** are used as connectors in order to provide interoperability among the other components.

WSMO goals are described in terms of the desired information and world state that must result from the execution of a given service. They are used to model the functional constraints of the requester. Web Service capabilities, which are part of the description of Web Services, capture the functionality of a given service. Such functional capabilities are modelled in terms of the preconditions and assumptions for the correct execution of the service and the postconditions and effects resulting from it.

Non-functional properties are also modelled on both sides. However, non-functional capabilities and constraints are not explicitly differentiated. Such a differentiation is expected to be part of future extensions of WSMO.

When analyzing WSMO with respect to our previous classification we see that, currently, WSMO does not explicitly model capabilities on the requester side, since goals only model requester side functional constraints. The modelling of requester capabilities will be considered in future versions of WSMO. Moreover, we expect to add support for preferences on both sides. While currently only functional constraints are considered, also the modelling of information disclosure constraints might be added to WSMO in future versions. Some ideas on how to support such constraints have been discussed in [6].

# 5 Open Discussion Points

Firstly, an interesting point is the handling of temporal and geographical (location) limits, like the above-mentioned service operating only during business hours, or a service provided only in Europe. The modeling of these limits is not yet settled, nor is the view on whether these limits should form a part of the capability of a service or whether they should be constraints of the service. Similar questions arise on the consumer side.

Secondly, it is not yet clear what scope capabilities and constraints should have. In WSDL [8] terms, capabilities and constraints can be applied on the fine granularity of operations or on the endpoints and perhaps even on the whole service.

Third, the conversational interface of a service, if viewed as a grouping of operations, could also be the target for applying capabilities and constraints, or on the other hand this interface can be could be viewed as a constraint on how a service can be invoked.

Finally, in this paper we have discussed the declarative specification of capabilities, constraints, and preferences, both on the requester and provider side on a semantic level. However, how the declared constraints are ensured and the preferences considered at execution time must also be addressed. For that purpose, constraints and preferences must be grounded to existing technologies. We envision the following steps to achieve this: (a) extend WSDL/BPEL4WS with constrains and preferences and introduce an explicit model for requesters and providers (WSDL and BPEL4WS do not distinguish between requesters and providers), (b) control process execution based on the described capabilities and preferences. Even though constraints might be checked and preferences considered before execution, it can happen that they are violated at actual execution. Therefore, the control of the execution must also specify what happens if the constraints are violated and foresee methods for compensation. This is specially relevant for composite processes, where such violations by a given process might require compensation for the processes of the composition executed before.

# References


[1] T. Bellwood, L. Clment, and C. von Riegen. *UDDI Specification Version 3.0.1.* UDDI Spec Technical Committee, October 2003. `http://uddi.org/pubs/uddi_v3.htm`

[2] R. Chinnici, M. Gudgin, J. Moreaum, and S. Weerawarana. *Web Services Description Language (WSDL) Version 1.2.* World Wide Web Consortium, March 2003. `http://www.w3.org/TR/wsdl20/`

[3] D. Fensel and C. Bussler. *Web service modeling framework (WSMF).* Electronic Commerce Research and Applications, 1(2), 2002.

[4] H. Lausen, D. Roman, and U. Keller (*editors*). *Web service modeling ontology - standard (WSMO-Standard).* Working draft, Digital Enterprise Research Institute (DERI), March 2004. `http://www.wsmo.org/2004/d2/v1.0/`

[5] N. Mitra. *SOAP Version 1.2 Part 0: Primer.* World Wide Web Consortium, June 2003. `http://www.w3.org/TR/soap12-part0/`

[6] D. Olmedilla, R. Lara, A. Polleres, and H. Lausen. *Trust negotiation for semantic web services.* First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), July 2004.

[7] M. Stollberg, H. Lausen, A. Polleres, and R. Lara (*editors*). *Wsmo use case modeling and testing.* Working draft, Digital Enterprise Research Institute (DERI), July 2004. `http://www.wsmo.org/2004/d3/d3.2/v0.1/`

[8] World Wide Web Consortium. *Web Services Description Language, Version 2.0*, August 2004. Last Call Working Draft, `http://www.w3.org/TR/wsdl20/`