Web Service Offerings Language (WSOL) and Web Service Composition Management (WSCM)

Vladimir Tosic, Bernard Pagurek, Babak Esfandiari, Kruti Patel, Wei Ma
Network Management and Artificial Intelligence Lab
Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario, Canada
{vladimir, bernie, babak, kpatel, weima}@sce.carleton.ca

ABSTRACT
Our research on Web Services is organized into two related projects. First, we develop Web Service Offerings Language (WSOL) for formal specification of various constraints and classes of service for Web Services. A service offering in WSOL is a formal representation of one class of service of one Web Service. It contains formal representation of various constraints: functional (pre-, post-, and future-conditions), non-functional (a.k.a. Quality of Service - QoS), and access rights. It also contains statements about price/penalty and management responsibility. The Web Service, its consumer, or one or more designated third parties (usually SOAP message intermediaries) can evaluate WSOL constraints. For easier specification of similar service offerings, WSOL enables specification of constraint groups (CGs) and constraint group templates (CGTs). We have also developed a format for representation of dynamic relationships between service offerings. WSOL service offerings are simple contracts and SLAs (Service Level Agreements) between Web Services.

Our second project is on Web Service Composition Management (WSCM). WSCM is management of Web Service compositions. This is a different task from Web Service Management (WSM). WSM is management of a particular Web Service or a group of Web Services within the same domain of management responsibility. While WSCM is related to network and system management (NSM) and business process management (BPM), there are important differences. We research various aspects of WSCM and develop a comprehensive WSCM architecture. Monitoring and manipulation of WSOL service offerings can be important part of both WSM and WSCM.

Keywords
Web Service, class of service, service offering, SLA, WSOL, Web Service management, Web Service composition management

1 INTRODUCTION

There are many similar, yet different, definitions of the term 'Web Service'. In this paper, by a **Web Service** we mean a unit of business, application, or system functionality that can be accessed over a network using XML messaging. While Web Services can be used for providing services to human end users, the true power of the W3C's Web Services framework [6] is leveraged through compositions (a.k.a. orchestrations, choreographies, flows, networks) of Web Services. Hereafter, by a **consumer** (a.k.a. requester) of a Web Service *A* we assume another Web Service that is composed with *A* and collaborates with it, not an end user (human) using *A*. On the other hand, we refer to *A* as the **supplier** (a.k.a. provider) Web Service. The composed Web Services can be distributed over the network, running on different platforms, implemented in different programming languages, and provided by different vendors.

We work on two projects related to management of Web Services and Web Service compositions. First, we develop Web Service Offerings Language (WSOL) for formal specification of various constraints and classes of service for Web Services. WSOL service offerings are simple contracts and SLAs (Service Level Agreements) between Web Services. Second, we work on a comprehensive architecture for Web Service Composition Management (WSCM). In this paper we present expressive power and language features of WSOL. We also discuss characteristics of WSCM and relate it to Web Service management (WSM), network and system management (NSM), and business process management (BPM). This paper builds upon our previous publications. In [7] we have discussed the need for WSOL, dynamic adaptation using manipulation of classes of service, and the need for WSCM. In [8] we have provided an illustrative example of WSOL syntax and discussed some of WSOL features and applications. Since the latter publication, WSOL was improved in several ways. Some of the new language features will be discussed in this paper. Due to the space limits, we will leave examples of the new WSOL syntax for future publications. This paper will also not go into details of our comprehensive WSCM architecture, which is still under development.

2 WEB SERVICE OFFERINGS LANGUAGE (WSOL)

We have advocated and illustrated the need for multiple classes of service for Web Services in [7] and [8]. By a **class of service** we mean a discrete variation of the complete service and quality of service (QoS) provided by one Web Service. We have concluded that it can be useful to enable a Web Service to offer several different classes of service to consumers. Classes of service can differ in usage

privileges, response times guaranteed to consumers, verbosity of response information, payment models, price, etc. Providing classes of service is a lightweight approach to Web Service customization, with limited complexity of required management.

A class of service of a Web Service is determined by a combination of various constraints. We define a **service offering** as a formal representation of one class of service of one Web Service. Consequently, a service offering is a combination of formal representations of various constraints that determine the corresponding class of service.

Web Service Offerings Language (WSOL) is our language for formal specification of constraints and classes of service for Web Services. It is XML-based (Extensible Markup Language) and compatible with WSDL (Web Services Description Language) version 1.1, which is a de-facto standard for functional description of Web Services. WSOL service offerings are specified separately from the WSDL description of the Web Service. All service offerings of one Web Service relate to the same characteristics described in the Web Service's WSDL file, but differ in constraints that define classes of service. The syntax of WSOL is defined using XML Schema.

The following **WSOL constructs** are used for formal specification of constraints and service offerings: constraint, statement, constraint group (CG), constraint group template (CGT), and service offering.

In WSOL, every **constraint** is a Boolean expression that states some condition that should be checked (i.e., evaluated) before execution of some Web Service operation starts and/or after this execution ends. WSOL enables formal specification of functional and QoS (a.k.a. non-functional) constraints and access rights. Functional constraints (pre-, post-, and future-conditions) define conditions which a functionally correct operation invocation must satisfy. They usually check some characteristics of message parts of the invoked operation. QoS constraints describe properties like performance, reliability, availability, etc. They check whether the monitored QoS metrics are within specified limits. An access right specifies conditions under which any consumer using the current service offering has the right to invoke a particular operation. If access is not explicitly allowed, it is forbidden. In other words, access rights are used in WSOL for service differentiation. On the other hand, specification of conditions under which a particular consumer (or a class of consumer) may use a service offering and other security issues are outside the scope of WSOL. WSOL also contains a general <constraint> element for definition of additional constraint types using XML Schema mechanisms.

Boolean expressions in constraints can contain standard Boolean operators (AND, OR, NOT, IMPLIES, EQUIVALENT), references to operation's message parts of type Boolean, as well as comparisons of arithmetic, string, date/time, or duration expressions. Arithmetic expressions can contain standard arithmetic operators (+, -, unary -, *, /, **), arithmetic constants, and references to operation's message parts of numeric data types. WSOL provides only basic built-in support for string and date/time/duration expressions. However, there is a possibility to perform external operation calls in any expression. Here, 'external' means outside the Web Service for which the constraint is specified. These external operations can be implemented by other Web Services or they can be implemented by the management entities evaluating the given constraint. In the latter case, although these external operations are described with WSDL, they are invoked using internal mechanisms, without any SOAP call. Note that WSOL does not support operation calls upon the same Web Service because there is no way to guarantee they are side-effect free (i.e., not changing the state of the Web Service). Evaluation of constraints must be side effect free. WSOL also supports checking operation's message parts that are arrays (of any data type) using quantifiers ForAll and Exists.

Let us discuss briefly the concept of a future-condition that we have introduced into WSOL. A future-condition is a Boolean expression evaluated some time after the supplier finishes execution of the requested operation and sends results to the consumer. This is different from a postcondition, which is evaluated when the supplier sends results to the consumer. In WSOL, one can specify that a future-condition should be evaluated: a) on a particular date/time; or b) after a specified duration elapses from the completion of the invoked operation. If a future-condition is not satisfied, operation invocation is considered invalid and the supplier has to pay some penalty. The concept of a future-condition enables specification of operation effects that cannot be easily expressed with postconditions. This includes some effects that a Web Service operation can have in the physical world. An example is delivery confirmation for goods bought using Web Services.

For specification of QoS constraints, WSOL needs external ontologies of QoS metrics and measurement units. We have summarized requirements for such ontologies in [9]. In our current implementation of WSOL, we have simply assumed that ontologies of QoS metrics are collections of names with information about appropriate data types and measurement units. Similarly, ontologies of measurement units are simple collections of names without any additional information. A more appropriate definition of ontologies of QoS metrics and measurement units is planned for a future version of WSOL.

A WSOL **statement** is any construct that states some important information about the represented class of service, but it is not a constraint. WSOL enables formal specification of price/penalty statements and management responsibility statements. Price statements specify price that a consumer using the particular service offering has to pay for successful use of the Web Service. Penalty statements specify monetary amount that the Web Service has to pay to a consumer if the consumer invokes some operation, but the

Web Service does not fulfil all constraints in the service offering. WSOL price/penalty statement support subscription and pay-per-use payment models, as well as their combination. Management responsibility statements specify what entity has management responsibility for checking a particular constraint, a price/penalty statement, a constraint group, or the complete service offering. A management entity can be the supplier Web Service, the consumer, or an independent third party trusted by both the supplier and the consumer. When the consumer submits a request for executing a supplier's operation, the management third parties are organized as SOAP intermediaries for the request, as well as the eventual response message. Some QoS metrics (e.g., availability) can be measured using probing instead of message interception. WSOL supports this by modeling probing entities as separate Web Services that provide results of their measurements through operations of some agreed-upon interfaces. These operations can be invoked in appropriate QoS constraints in WSOL service offerings, using the WSOL external operation call mechanism.

A **constraint group (CG)** is a named set of constraints and/or statements. A CG can also contain other CGs (including instantiations of CGTs). Nesting of contained CGs may be recursive. Definition of CGs has several benefits. First, it can be used for solving the problem of separation and integration of concerns related to constraints for Web Services, discussed in [8]. Second, a CG can be re-used across service offerings as a unit. Third, it is possible to specify that all constraints from a CG are evaluated by the same management entity. Fourth, constraints in different CGs can have the same constraint name, so using CGs enables name re-use. Fifth, one can use CGs to define aspects of service offerings. For example, one can group all functional constraints for one port type into one CG, QoS constraints for the same port type into another CG, and access rights for this port type into a third CG.

When a new CG is defined, if some of the contained constraints and CGs were already defined elsewhere, there is no need to define them again. They can be simply included into the new containing CG. On the other hand, new constraints and CGs can also be defined inside a containing CG. A new CG can be defined as an extension of an existing CG, inheriting all constraints and defining some additional ones. Extension is, in fact, single inheritance of CGs. We have also studied multiple inheritance, but it is not part or the current version of WSOL. Benefits similar to multiple inheritance can be achieved in WSOL by including several existing CGs inside the new CG. If inside one CG two or more constraints of the same type (e.g., two preconditions) are defined for the same operation, they all have to be satisfied. This means that the Boolean AND operation is performed between such constraints.

A **constraint group template (CGT)** is a parameterized CG. At the beginning of a CGT, one defines one or more abstract CGT parameters, each of which has a name and a type. CGT parameters often have type 'numberWithUnit', which requires additional information about the used measurement unit. Definition of parameters is followed by definition of constraints and nested CGs, in the same way as for CGs. Constraints inside a CGT can contain expressions with CGT parameters.

A CGT is instantiated when appropriate constants are supplied as values for all CGT parameters. The result of such instantiation is a new CG. A CGT can be instantiated inside a CG, definition of another CGT, or a service offering. One CGT can be instantiated many times with different parameter values. For example, one can define a CGT with one parameter 'maxRT' and one constraint that the measured response time must be less than 'maxRT'. Then, this CGT can be instantiated with 'maxRT' parameter values 20 milliseconds, 50 milliseconds, 2 seconds, etc.

The concept of a CGT in WSOL is a very powerful specification mechanism. Many classes of service (and SLAs) contain constraints with the same structure, but with different constant values. In our opinion, it is even more important specification concept than single inheritance (i.e., extension) of CGs, CGTs, and service offerings. However, the WSOL concept of a CTG also has some limitations. First, CGTs must not be nested. In other words, one must not define a CGT inside another CGT. Next, since constraints inside a CGT may contain expressions with CGT parameters, these constraints must not be included inside other CGTs, CGs, or CLSOs. Further, WSOL supports single inheritance (i.e., extension) of CGTs, similarly to extension of CGs. However, a CGT extending some other CGT must not define additional CGT parameters. Only addition of new contained constraints, statements, and CGs is allowed.

A **service offering** is a set of constraints, statements, and CGs (including instantiations of CGTs) that all refer to the same Web Service. We also use the term **component-level service offering (CLSO)** with the same meaning. (Note that in WSOL there is no need for a special concept of a port-level service offering—PLSO—because this can be represented as a CG.) The concept of a service offering is the central concept in our work on WSOL. One service offering is a formal specification of one class of service for a Web Service. It can also be viewed as one contract or one SLA between the supplier Web Service, the consumer, and eventual management third parties. A Web Service can offer multiple service offerings to a consumer, but a consumer can use only one of them at a time.

Syntactically, a WSOL service offering is just a big CG. The rules discussed for CGs also apply for service offerings. An important exception is that service offerings must not be nested. Similarly to CGs, WSOL supports single inheritance (extension) of service offerings. We make CLSO a separate concept in WSOL to emphasize its special run-time characteristics. Most importantly, consumers can choose and use service offerings, not CGs. This is because CGs are usually not complete and consistent from the usability view. For the same reason, dynamic relationships

can be specified only for service offerings, not for CGs.

Relationships useful in the process of selection of service offerings and in dynamic adaptation of consumer-supplier liaisons are dynamic. In other words, they can change during run time, e.g., after dynamic creation of a new service offering. Such relationships can be represented as triples <SO1, S, SO2> where SO1 is a service offering; S is a set of constraints, statements, and CGs from SO1 that are not satisfied; and SO2 is the appropriate replacement service offering. These triplets are specified outside WSOL files (in a special XML format) to make their evolution independent from the evolution of other characteristics of a service offering.

As argued in more detail in [7] and [8], WSOL can be used in several ways. It supports selection of appropriate Web Services and service offerings. It also helps reduce unexpected interactions between the composed Web Services. We are particularly interested in management applications of WSOL. WSOL service offerings can be used for Web Service monitoring, metering, control, and billing. They are precise and complete enough to serve as contracts or SLAs between Web Services. In addition, dynamic (i.e., run-time) manipulation of service offerings is a useful tool for WSM and WSCM.

Our work on WSOL draws from the considerable previous work on differentiated classes of service and formal representation of various constraints in other areas. At the beginning of our research, there was no relevant work of this kind in the area of Web Services. In the meantime, several related works emerged. First, the notion of WSEL (Web Services Endpoint Language) was mentioned in the literature [4], but with no detailed publication to date. One of the goals of WSEL is specification of some constraints, including QoS, for Web Services. Further, the DAML-S (DAML-Services) language initiative [3] works on semantic description of Web Services, including specification of functional and some QoS constraints. Next, the work on WSLA (Web Service Level Agreements) [2] is an XML specification of SLAs (primarily QoS constraints) for Web Services. The work on the OGSA (Open Grid Services Architecture) [4] also encounters the need for contracts and SLAs. A couple of other works with similar goals are in early stages. A detailed comparison of these various approaches requires a separate publication. In short, WSOL has a distinct set of advantages (but also disadvantages) compared with any of these related works. Some general strengths of WSOL are formal specification of various constraints and multiple service offerings per Web Service, orientation towards management applications, expressive power with relatively little overhead, full compatibility with WSDL, etc.

3 WEB SERVICE COMPOSITION MANAGEMENT (WSCM)

In [7] we have argued extensively that to further increase flexibility and adaptability of Web Service compositions, they have to be managed. In one project in our research group, we attempt to develop a comprehensive architecture for WSCM. It should build upon and encompass our past and ongoing work on dynamic service composition, hot-swapping of software components, peer-to-peer (P2P) service advertisement, trust acquisition and propagation based on reputation, as well as WSOL and manipulation of service offerings. It will be based on proven network and system management (NSM) concepts, but it will also relate to business issues. We find development of such comprehensive WSCM architecture a major challenge. The architecture is still under development and will be presented in a forthcoming publication.

In this paper, we want to point out several premises in our work on this architecture. First, there is difference between Web Service Composition Management (WSCM) and Web Service management (WSM). **WSM** is management of a particular Web Service or a group of Web Services within the same domain of management responsibility. For example, Web Services provided with one application server might be managed as a group. Similarly, Web Services provided by the same business entity might be managed in a unified manner as a group, either completely or only in some respects. A large number of companies already claim that their products perform some kind of WSM, predominantly in performance management. Many WSM products are platform-specific application management products, often based on JMX (Java Management Extensions). Another often WSM approach is to use Web Service gateways, hubs, or proxies that serve as a single point of control, metering, and management. Contrary to WSM, **WSCM** is management of Web Service compositions. In a general case, the composed Web Services are distributed over the Internet and provided by different business entities. Some of these business entities might not want to relinquish or outsource control over their Web Services. They often have mutually incompatible and even conflicting management goals. In addition, management of the Internet infrastructure is a very challenging task. Consequently, WSCM will usually not be able to involve full WSM management of the composed Web Services and management of the Internet communication infrastructure. Therefore, the emphasis in WSCM must be on decisions related to which Web Services are composed and how they interact. Contrary to WSM, there are relatively few results on WSCM.

Second, WSCM is positioned between traditional network and system management (NSM), including service management, and business process management (BPM) tools. Significant analogies can be made between WSCM and both NMS and BPM tools. WSCM can learn from significant body of knowledge. However, there are also important differences. One of the differences between WSCM and traditional NSM is that the composed Web Services are under full control of their vendors, with different management goals. An important difference between WSCM and BPM is that a Web Service composition is not only a representation of a business process, but also a business service

and a distributed system component. Neither NSM nor BPM solutions completely address the complexity of multi-party, multi-goal, multi-level, multi-aspect management.

The third point that Web Service compositions can have very different characteristics. In some cases, there will be an explicit description of the composition. Several languages—including the new Business Process Execution Language for Web Services (BPEL4WS)—have appeared in this area. However, in some cases there will be no such description. A Web Service composition can emerge spontaneously, from a series of (primarily) bilateral contracts. There is no 'master plan' and no common goal; all participants have their own interests, mutually conflicting. In such cases, WSCM becomes even harder, but it is possible.

Fourth, we strongly believe that appropriate specification of management information is the key for successful management activities. WSOL describes for Web Services what QoS metrics to monitor, what constraints to evaluate, as well as when (and to some extent: how) to perform particular management activities. Consequently, WSOL has important place in our work on WSCM.

4 CONCLUSIONS AND FUTURE WORK

Providing multiple classes of service and formal specification of various constraints for Web Services have numerous practical benefits. WSOL service offerings are simple contracts and SLAs between Web Services, so WSOL and its manipulation are useful in both Web Service Management (WSM) and Web Service Composition Management (WSCM). WSOL enables formal specification of various constraints and management-related statements and their grouping into constraint groups (CGs), constraint group templates (CGTs), and service offerings. It supports single inheritance (i.e., extension) of CGs, CGTs, and service offerings. CGs and CGTs are a powerful mechanism for specification of similar service offerings. Dynamic relationships between service offerings are specified outside WSOL files, in a simple XML format.

It is important to note that the current version of WSOL is based on WSDL 1.1. When WSDL 1.2 becomes stable, we will make WSOL compatible with it. The emphasis of our current efforts on WSOL is improvement of WSOL syntax and development of a proof-of-concept WSOL parser with syntax checks and some semantic checks. We are also studying automatic generation of constraint-checking code from WSDL and WSOL files. In this respect, we would like to re-use results from the composition filters [1] and similar aspect-oriented approaches. This is because a constraint-checking SOAP intermediary in our approach can be related to a composition filter. We also plan a Java API for easier generation of WSOL files, but its development is still in an early stage. Other issues for our future work on WSOL include more appropriate definition of ontologies of QoS metrics and measurement units, formal specification of some other constraints (e.g., roles played in design patterns and coordination protocols), etc.

We are also working on a comprehensive architecture for WSCM, building on our expertise in network and system management. Some of the premises in this research project are distinction between WSM and WSCM; re-use of expertise in both network and system management and business process management tools; management with or without an explicit description of a Web Service composition; and importance of specification of management information.

REFERENCES

1. Bergmans, L., Aksit, M. Composing Crosscutting Concerns Using Composition Filters. *Comm. of the ACM*, Vol. 44, No. 10. (Oct. 2001), pp. 51-57, ACM.
2. Dan, A., Franck, R., Keller, A., King, R., Ludwig, H. Web Service Level Agreement (WSLA) Language Specification. In *Documentation for Web Services Toolkit, version 3.2.1* (August 9, 2002), International Business Machines Corporation (IBM).
3. The DAML Services Coalition. *DAML-S: Semantic Markup for Web Services*. WWW page. (December 12, 2001) On-line at: http://www.daml.org/services/daml-s/2001/10/daml-s.html
4. Ferguson, D. F. Web Services Architecture: Direction and Position Paper. In *Proc. of the W3C Workshop on Web Services – WSWS'01* (San Jose, USA, Apr. 2001), W3C. On-line at: http://www.w3c.org/2001/03/WSWS-popa/paper44
5. Foster, I., Keselman, C., Nick, J. M., Tuecke, S. Grid Services for Distributed Systems Integration. *Computer*, Vol. 35, No. 6 (June 2002), pp. 37-46, IEEE -CS
6. International Business Machines Corporation (IBM), Microsoft Corporation. Web Services Framework. In *Proc. of the W3C Workshop on Web Services – WSWS'01* (San Jose, USA, Apr. 2001), W3C. On-line at: http://www.w3.org/2001/03/WSWS-popa/paper51
7. Tosic, V., Pagurek, B., Esfandiari, B., Patel, K. On the Management of Compositions of Web Services. In *Proc. of the OOWS'01 (Object-Oriented Web Services 2001) workshop at OOPSLA 2001* (Tampa, Florida, USA, Oct. 2001), ACM. On-line at: http://www.research.ibm.com/people/b/bth/OOWS2001/tosic.pdf
8. Tosic, V., Patel, K., Pagurek, B. WSOL - Web Service Offerings Language. In *Proc. of the Workshop on Web Services, e-Business, and the Semantic Web (WES) at CaiSE'02* (Toronto, Canada, May 2002). To be publ., Springer-Verlag, Lecture Notes in Computer Science.
9. Tosic, V., Esfandiari, B., Pagurek, B., Patel, K. On Requirements for Ontologies in Management of Web Services. In *Proc. of the Workshop on Web Services, e-Business, and the Semantic Web (WES) at CaiSE'02* (Toronto, Canada, May 2002). To be publ., Springer-Verlag, Lecture Notes in Computer Science (LNCS).