On Comprehensive Contractual Descriptions of Web Services

Vladimir Tosic*
The University of Western Ontario, Canada
vladat@computer.org

Bernard Pagurek
Carleton University, Canada
bernie@sce.carleton.ca

Abstract

Comprehensive contractual description of Web Services and Web Service compositions is needed for selection of appropriate Web Services and their service and quality of service (QoS) levels, for monitoring of operation of Web Services, and for management of Web Services and Web Service compositions. We systematically examined what types of technical contracts are useful for Web Services and Web Service compositions and classified them into three broad categories (functional, quality, and infrastructure contracts), each containing several contract types. Our study of how prominent Web Service languages can or cannot be used for specification of these contract types shows that they enable specification of only particular types of contracts, sometimes even in incompatible ways. Consequently, we advocate a unified framework for comprehensive contractual description of Web Services and Web Service compositions. At the end, we outline one possible approach to comprehensive contractual description, based on extending existing Web Service technologies.

1. Introduction and motivation

The primary goal of XML (Extensible Markup Language) Web Service technologies [5] is to address the problem of dynamic (run-time) application-to-application (A2A) integration. They can be used for business-to-business (B2B) integration and/or for Enterprise Application Integration (EAI) within companies. Consequently, the true power of Web Service technologies is achieved through compositions of Web Services, which can take the form of orchestrations or choreographies [7].

To achieve selection of appropriate Web Services and their service and quality of service (QoS) levels, monitoring of operation of Web Services, and management of Web Services and their compositions, it is important to explicitly and formally describe Web Services and interactions between them [12]. For example, specification of QoS guarantees such as response time and availability helps in comparing Web Services implementing the same interfaces, determines which Web Service operations to monitor and when, and can guide internal activities (e.g., resource management) of the Web Service to meet these QoS guarantees. Such descriptions can be provided in various contracts [13]. In a broad sense, a **contract** is any formal agreement between collaborating entities (e.g., composed Web Services) and, potentially, supporting parties (e.g., performing contract monitoring). For example, a contract between Web Services can contain descriptions of provided operations, guarantees of maximum response time, prices, and/or legal responsibilities.

In business-to-business Web Service compositions, a party usually has no direct insight into or control over the internal operation of other parties. This means that all aspects of collaboration have to be explicitly and formally captured in contracts and that contract management becomes the primary means of managing Web Service compositions and, to some extent, individual Web Services [13, 11, 12, 4]. Since specification of management information critically influences management activities, a study of the specification of contracts is an important step towards more powerful and easier contract selection/negotiation and contract management. For example, it is important for contract management to understand and formally describe various relationships, such as those between functional constraints, QoS guarantees, and prices.

Web Service technologies are often classified into a conceptual stack [5]. This classification identifies that several description mechanisms, such as Service Level Agreements (SLAs) and Business Level Agreements (BLAs), are useful for Web Services. However, it does not explore in detail the comprehensive contractual description of Web Services. As the number of developed Web Service technologies rapidly grows, it is important

*This research is partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) post-doctoral fellowship awarded to Vladimir Tosic.

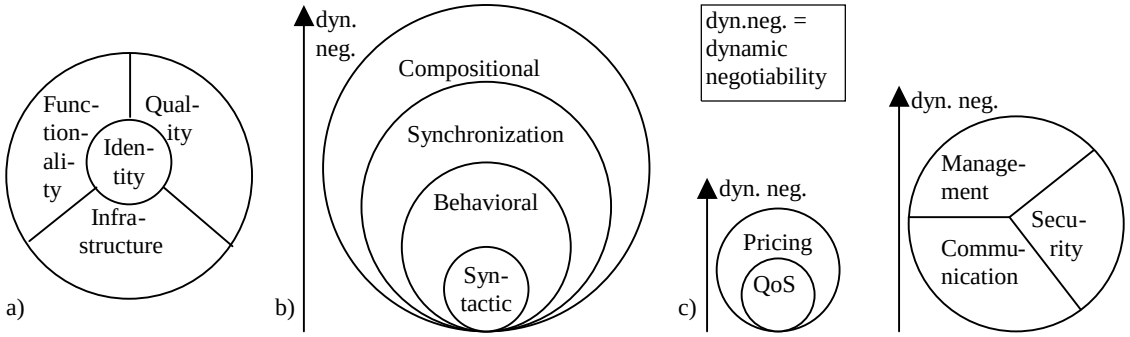

Figure 1. Categories and Types of Contracts for Web Services: a) Contract Categories, b) Types of Functionality Contracts, c) Types of Quality Contracts, d) Types of Infrastructure Contracts

to systematically and critically assess the progress towards the needed comprehensive description of Web Services and suggest further steps. In this respect, the World Wide Web Consortium (W3C) recently started a discussion of how to proceed towards more comprehensive description of constraints and capabilities of Web Services [14]. This paper is one academic contribution in this area and examines it from a previously unexplored viewpoint, based on our research experience [11, 12]. Note that the emphasis of this paper is on technical contracts. While we also discuss business and legal contents of contracts, we make no claim of comprehensiveness of our study in these domains.

2. A classification of contracts for Web Services

While there is no prior systematic study of contracts for Web Services, [1] discussed this topic for software components. They identified and discussed four types of contracts for software components: syntactic, behavioral, synchronization, and QoS contracts. These four contract types are ordered by increase of dynamic changeability and negotiability. Syntactic contracts are non-negotiable, while QoS contracts are usually dynamically negotiable.

Having [1] as our inspiration, we critically examined Web Service technologies to determine what types of technical contracts are useful for comprehensive description of Web Services. We studied the similarities and differences between Web Services and software components, the Web Service languages that are already developed or are under development, and a number of scientific and industrial papers discussing the need for additional Web Service technologies.

Our conclusion is that comprehensive description of Web Services requires several different types of contracts. We find that these contract types can be classified into three broad categories of contracts for Web Services: functionality, quality, and infrastructure contracts. Consequently, we call this classification "**FQI**". These three contract categories are shown in Figure 1.a as a comprehensive shell around the Web Service identity. This identity is, by the W3C's definition of a Web Service [8], a set of Uniform Resource Identifiers (URIs) for Web Service's endpoints (ports). A different set of URIs means a different Web Service. Every contract category contains several contract types, shown in Figures 1.b, 1.c, and 1.d and discussed in the following subsections. Our technical report with further discussion and examples is available to interested readers.

1) A **functionality contract** describes WHAT a Web Service does. We identified the following types of functionality contracts: syntactic, behavioral, synchronization, and compositional contracts. First, apart from the information about identity, the essential information for invocation of a Web Service (as a software module) includes names of provided interfaces, names of operations, names and data types of input and output parameters, possible exceptions, and attributes. This information determines a **syntactic contract**. Second, specification of syntactic contracts is not enough for dynamic application-to-application collaboration. Specification of semantics is required to unambiguously convey the meaning of operations. Semantics is a complex topic and it crosscuts multiple types of contracts in our classification. Some aspects of semantics are captured in **behavioral contracts**. They describe requirements for and effects of correct execution of Web Service's operations. They can contain operation pre-conditions, post-conditions, future-conditions [12], and invariants, collectively known as functional constraints. Specification of references to ontological definitions relevant for the invocation also belongs to behavioral contracts. Third, oftentimes there are dependencies between invocations of operations of a Web Service. For example, a client might have to log in before using other operations. Information about such dependencies between operation invocations is specified

in **synchronization contacts**. Synchronization contracts can also describe concurrent invocations by multiple clients. Common synchronization dependencies include sequence, arbitrary ordering, potential parallelism, and mutual exclusion. Fourth, **compositional contracts** describe correct Web Service compositions (both orchestrations and choreographies), which Web Services participate in a composition, and flow of messages between them. This information complements the other functionality contracts, which describe restrictions on using one Web Service. Figure 1.b shows relationships between the types of functionality contracts for Web Services. The contract types in the figure are represented as onion layers, analogously to the figure given in [1]. The higher (i.e., external) layers can be negotiated dynamically more easily than the lower (i.e., internal) layers. Further, one contract on a lower layer can map to many contracts on a higher layer. In addition, when some changes are introduced into a lower layer contract, the corresponding higher-level contracts must be checked for consistency and updated if needed. This notation does not mean that a contract on a higher level is a superset of a contract on the lower level.

2) The global market of Web Services will contain many Web Services providing the same functionality. To differentiate between such Web Services, it is important to explicitly and formally specify extra-functional (non-functional) properties, such as performance, availability, reliability and price. In some cases, extra functional properties are as crucial as functionality. First, **quality contracts** describe HOW WELL a Web Service performs its functions. They enable monitoring and management of extra-functional properties. We identified two types of quality contracts: quality of service (QoS) contracts and pricing contracts. **QoS contracts** describe quantifiable extra-functional properties of Web Service's operation, from both the computing viewpoint (quality of Web Service – QoWS) and the business viewpoint (quality of business service – QoBS). For example, response time is a QoWS property, while delivery time of ordered goods is a QoBS property. QoS contracts contain QoS constraints (requirements and guarantees), such as service level objectives (SLOs). One of the issues with providing QoS contracts for Web Services is that QoS properties can depend upon execution of the underlying infrastructure (e.g., the Internet infrastructure) and parties (e.g., other Web Services) that the Web Service cannot control. Second, for B2B applications of Web Services, it is essential to specify prices for using a Web Service or some of it operations, as well as monetary penalties for not keeping guarantees from other contracts. For example, prices can be subscription-based, pay-per-use, or pay-per-volume. This is captured in **pricing contracts**. Figure 1.c shows the two types of quality contracts onion layers, where pricing contracts are an external layer to QoS contracts. Pricing contracts are more dynamically negotiable than QoS contracts because there are few restrictions on setting prices and monetary penalties, while QoS contracts depend on capabilities of the execution environment.

3) While technologies (e.g., programming languages, operating systems, computer platforms) used to implement a Web Service itself are irrelevant, technologies (e.g., protocols, security mechanisms) used to implement Web Service's communication with other Web Services must be described explicitly. An **infrastructure contract** specifies what underlying infrastructure technologies, services, and/or entities a Web Service uses and/or requires its client and provider Web Services to use. In other words, it describes BY WHAT MEANS a Web Service collaborates with the others. We identified three types of infrastructure contracts: communication, security, and management contracts. First, Web Services can communicate using several XML messaging protocols, most notably SOAP. Further, SOAP can run over a number of transport protocols, e.g., the HyperText Transport Protocol (HTTP). **Communication contracts** describe what protocols are used for packaging and transporting messages between the communicating Web Services. Second, security and privacy are important, sometimes crucial, topics for the use of Web Services. **Security contracts** describe security and privacy aspects of using a Web Service. They can describe what security technologies are used for accessibility, authentication, authorization, confidentiality, integrity, and/or non-repudiation. They can also specify which privacy technologies are used. In addition, they can contain security and privacy policies, e.g., authorization policies. Third, it is not enough to only specify contracts. The conformance to the contracts of all previously identified types has to be checked during the run-time and enforced. For example, functional constraints in behavioral contracts have to be evaluated, QoS metrics used in QoS contracts have to be measured and/or calculated operation executionand corresponding QoS constraints evaluated, the use of the Web Service has to be accounted, appropriate prices or monetary penalties have to be calculated and billed. **Management contracts** describe what entities perform the actual monitoring, metering, accounting, and control of constraints specified in other contracts. Examples of possible management entities are the Web Service itself, its clients, trusted third party Web Services, specialized management infrastructure, and human administrators. Management contracts also

specify what technical and/or legal actions will be performed in certain conditions, such as when requirements and guarantees specified in other contracts are not met. Figure 1.d shows the infrastructure contract types for Web Services. Contrary to Figures 1.b and 1.c, infrastructure contracts are not shown as onion layers, because dependencies between them are not as strong and clear. There is however, a difference in dynamic negotiability because the choice of available technologies, third-party entities, and policies enables easiest negotiation of management contracts, then security contracts, and then communication contracts.

Contrary to Figures 1.b, 1.c, and 1.d, Figure 1.a does not compare dynamic negotiability and changeability of contract categories. This is because our attempts to order contract types from different contract categories by dynamic negotiability did not produce an unambiguous result. Dependencies between contracts from different categories are possible. For example, a change in a behavioral contract can cause a change in a pricing contract. However, these dependencies are not as strong as those between contract types from the same category. For example, a change in a synchronization contract will likely affect corresponding compositional contracts, but will probably not affect QoS or management contracts.

3. Languages for contractual description of Web Services

We examined a number of existing Web Service languages to check what types of contracts can be specified with them. A partial summary of this study is shown in Table 1. This table shows the Web Services Description Language (WSDL) [2], the Business Process Execution Language for Web Services (BPEL4WS) [10], the Web Services Choreography Description Language (WS-CDL) [3], the Web Services Policy Framework (WS-Policy) [9], the Web Service Level Agreement (WSLA) [4], the Web Service Offerings Language (WSOL) [11, 12], and the Web Ontology Language – Services (OWL-S) [6]. Apart from these languages, there are many other languages [12], mostly of experimental nature and without wide acceptance.

This study of existing Web Service languages shows that although every language enables specification of only particular types of contracts, solutions for specification of all contract types are relatively well developed in different languages. WSDL is the de-facto standard for specification of Web Service identity, syntactical, and communication contracts. The other languages from Table 1 (except OWL-S syntactical contracts) are compatible with WSDL. By 'compatibility'

Table 1. Some existing languages for the specification of contracts for Web Services

Language	Contract Category	Contract Type	Identity	Functionality				Quality		Infrastructure		
				Syntactic	Behavioral	Synchronization	Compositional	QoS	Pricing	Communication	Security	Management
WSDL	+	+								+		
BPEL4WS						+	+					
WS-CDL						+	+					
WS-Policy					+						+	
WSLA								+	+			+
WSOL					+			+	+			+
OWL-S				+	+	+	+					

we mean that different languages predominantly address different contract types and that for same or similar concepts they use the same (or at least analogous) language constructs. Unfortunately, the studied languages are not always compatible. For example, WS-Policy and WSLA are not fully compatible because they define similar concepts (e.g., policy assertion in WS-Policy and SLOs in WSLA) in different ways. Such incompatibilities significantly reduce ease of use of these languages. Similarly, BPEL4WS and WS-CDL are somewhat competing and not fully compatible standardization efforts for compositional and synchronization contracts. Another important fact is that there is still no widely accepted standardization initiative for quality contracts, in spite of a large number of diverse academic and industrial works in this area [12].

4. Principles for a unified framework

As discussed in Section 2, to fully specify contractual obligations and guarantees of a Web Service, contracts of several different types have to be specified. On the other hand, the comparison from Section 3 shows that there is no single language or a set of mutually compatible languages that enables specification of all identified contract types. This significantly affects the interoperability of Web Services and limits the potential for dynamic Internet-wide application-to-application collaboration. For example, when there are several competing languages for quality contracts, it becomes very hard or impossible to qualitatively com-

pare Web Services that use different languages. Further, performance monitoring and management requires code dealing with descriptions in different languages.

Therefore, we argue that specification languages for all identified types of contracts have to be standardized. To fully achieve the promise of Web Service technologies, companies should compete with the content, not specification formats, of contracts. Further, we suggest development of a unified framework that would coordinate standardization of languages for different contract types. We want to achieve its wide acceptance, ease of use, and expressive power. Based on the previous discussion, survey of the literature in the field, and our research experience, we suggest the following principles for the work on this unified framework:

1) **Modularity.** One aspect of modularity is that a Web Service can specify only some, not all, contracts and support only the necessary contract languages. Web Service technologies are modular in this sense. However, modules that can be reused for contracts of the same or even different contract types are also needed. For example, definitions of used QoS metrics, measurement units, and currency units can be moved from QoS and pricing contracts to specialized external, re-usable and extensible, ontologies. Modules reusable across different contract types are discussed next.

2) **Unification and standardization of common contract elements.** Contracts of different types have some similarities. Most importantly, specification of expressions is essential for both behavioral and QoS contracts and can be used in synchronization, compositional, pricing, security, and management contracts. The unification and standardization of such common contract elements enables easier reasoning about contracts and significantly reduces the run-time overhead [12]. This makes easier both selection of Web Services and their operational characteristics and enforcement and management of contracts. While this somewhat reduces flexibility of contract forming, it significantly improves contract usability and compatibility.

3) **Extensibility.** It should be possible to modularly extend the contract languages in the framework to more precisely describe supported contracts types. Such language extensions should be done without modifications of the language core and with minimal impact on the existing language tools. If a need for an additional, previously unforeseen, contract type is determined, it must be possible to extend the existing contract languages or, at least, add new ones.

4) **Use of only few contract languages.** We argue that the number of used contract languages should be kept small [12]. This is because there is less run-time overhead in supporting one language than a group of languages, even if they are compatible and modular. Further, this enables better expression of dependencies between contracts of different types and reduces redundancies and potential incompatibilities. Note that this requirement does not conflict with the requirement for modularity if the used languages and corresponding tools are modular and extensible.

5) **Reuse and extension of the widely accepted Web Service languages.** Development of new languages or popularization of less-known languages will probably not be as effective as reuse and extension of languages in which companies made investments. WSDL is the only Web Service language that is widely accepted, so it has to be used for the specification of identity, as well as syntactic and communication contracts. Among several languages for compositional and/or synchronization contracts, the most widely accepted and used is BPEL4WS. The newer WS-CDL has some advantages for synchronization contracts and somewhat different application domain (Web Service choreographies, instead of orchestrations), but it is not yet widely accepted. Regarding the other contract types, we find that WSLA, WSOL, and OWL-S have technical advantages over WS-Policy, but that the latter has much bigger support from the industry. The general WS-Policy Framework has constructs (in WS-PolicyAssertions) that can be used for behavioral contracts and an extension (WS-SecurityPolicy) for security contracts. Some of the issues with WS-Policy are [12]: a) there are no concepts of a contract, an SLA, and a class of service, in spite of their similarity to policies; b) the format for expressions in WS-PolicyAssertions is not standardized; c) there are no detailed WS-Policy extensions for QoS, pricing, and management contracts. Since WS-Policy and, to a lesser degree, BPEL4WS are not yet as ubiquitously accepted and used as WSDL, it is still feasible to extend them towards comprehensive contractual description of Web Services.

6) **Specification of relationships between contracts.** It is important to capture relationships between contracts, both within the same type and across different types. These relationships influence contract negotiation/selection and management. For example, specifications that one contract extends another one, that two contracts instantiate a common template, or that one contract includes a part of another one can significantly ease comparison of contracts. Another important set of relationships between contracts of the same type states what is a suitable replacement contract if the current contract becomes inappropriate for some reason. Relationships between contracts of different types capture their dependencies. For example, such a relation-

ship can state that after a client decides to change used QoS contract from *HighQuality* to *LowQuality*, the change in used pricing contract from *HighPrice* to *LowPrice* should be performed automatically. Solutions for specification of various relationships between contracts are built into WSOL [11, 12].

5. Conclusions and future work

Having these principles in mind, we suggest building a comprehensive contractual description of Web Services based on: 1) WSDL, 2) one language that integrates concepts from BPEL4WS and WS-CDL, and 3) WS-Policy Framework significantly extended with relevant concepts from other languages (most notably, WSLA and WSOL). We call this framework proposal "**DBP+**" ("WSDL, BPEL4WS, WS-Policy extended").

We argue that an integration of BPEL4WS and WS-CDL would best address compositional contracts for both orchestrations and choreographies, as well as synchronization contracts. Probably, the best way to proceed in this direction would be to extend BPEL4WS, since it is more widely accepted. We also argue for a significantly extended WS-Policy Framework, in which contracts are modeled as groups of policies. The most important characteristic needed to make WS-Policy usable for different types of contracts is standardization of the used expression mechanism. In our opinion, the most suitable existing expression format is the one built into WSOL, but several others, such as the one built into WSLA, can be the basis for this standardization. The second crucial item for future work on WS-Policy is standardization of extensions for QoS, pricing, and management contracts. We suggest using WSLA as the basis for this standardization. In particular, WSLA SLOs can be adapted into QoS policy assertions, price information can be used for pricing policies, while information about parties that participate in the contract and their obligations can become management policies in WS-Policy extensions. Good concepts and characteristics from other languages in this domain can also be integrated into such extended WS-Policy Framework. Particularly useful are WSOL solutions for specification of dynamic relationships between contracts, reusability constructs modeling static relationships, classes of service and QoS, and integration of different types of contracts into one language [11, 12]. Further, specification of pricing contracts is more developed in WSOL than in WSLA, while WSOL behavioral contracts are more detailed than in WS-Policy.

One of the items for future research is a comprehensive study of business and legal contracts associated with Web Services. We have an impression that business and legal topics crosscut different contract types and that the presented classification of contracts can accommodate the major business and legal topics, such as quality of business service, pricing, and actions taken if guarantees are not met. However, a more thorough study of these areas could result in extensions of our classification and the suggested framework.

References

[1] Beugnard, A., Jezequel, J.-M., Plouzeau, N., Watkins, D. Making Components Contract Aware. *Computer*, Vol. 32, No. 7 (July 1999), IEEE-CS, pp. 38-45.

[2] Chinnici, R., Gudgin, M., Moreau, J.-J., Schlimmer, J., Weerawarana, S. (eds.). *Web Services Description Language (WSDL), Ver. 2.0, Part 1: Core Language*. World Wide Web Consortium (W3C) working draft (Nov. 10, 2003).

[3] Kavantzas, N., Burdett, D., Ritzinger, G., Lafon, Y. (eds.). *Web Services Choreography Description Language. Ver. 1.0*. W3C working draft (Oct. 12, 2004).

[4] Keller, A., Ludwig, H. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Jounal of Network and Systems Management*, Vol. 11, No 1 (Mar. 2003), Plenum Publishing, pp. 57-81.

[5] Kreger, H. Fulfilling the Web Services Promise. *Comm. of the ACM*, Vol. 46, No. 6 (June 2003), ACM, pp. 29-34.

[6] The OWL Services Coalition. *OWL-S: Semantic Markup for Web Services*. Ver. 1.0. WWW page (Nov. 2003) at: http://www.daml.org/services/owl-s/1.0/owl-s.html

[7] Peltz, C. Web Services Orchestration and Choreography. *Computer*, Vol. 36, No. 10 (Oct. 2003), IEEE-CS, pp. 46-52.

[8] Schlimmer, J.C. (ed.). *Web Services Description Requirements*. W3C working draft (Oct. 28, 2002).

[9] Schlimmer, J. (ed.). *Web Services Policy Framework (WS-Policy).* WWW document (Sep. 2004) at: ftp://www6.software.ibm.com/software/developer/library/ws-policy.pdf

[10] Thatte, T. *Business Process Execution Language for Web Services. Ver. 1.1*. WWW document (May 5, 2003) at: http://www-128.ibm.com/developerworks/library/ws-bpel/

[11] Tosic, V., Pagurek, B., Patel, B. Esfandiari, B., Ma, W. Management Applications of the Web Service Offerings Language (WSOL). In Eder, J., Missikoff, M. (eds.). *Proc. of CAiSE'03* (Velden, Austria, June 2003). Lecture Notes in Computer Science (LNCS), No. 2681. Springer-Verlag (2003) 468-484.

[12] Tosic, V. *Service Offerings for XML Web Services and their Management Applications*. Ph.D. Dissertation. Carleton University, Ottawa, Canada. August 2004.

[13] van Moorsel, A. Ten-Step Survival Guide for the Emerging Business Web. In Bussler, C. et al. (eds.) *Proc. of the Workshop. on Web Services, e-Business, and the Semantic Web at CAiSE'02* (Toronto, Canada, May 2002). LNCS, No. 2512, Springer-Verlag, pp. 1-11.

[14] World Wide Web Consortium (W3C). *W3C Workshop on Constraints and Capabilities for Web Services*. On-line proceedings (Redwood Shores, USA, Oct. 2004) at: http://www.w3.org/2004/06/ws-cc-cfp.html