On Comprehensive Contractual Descriptions of Web Services

Vladimir Tosic
Department of Computer Science
The University of Western Ontario
Middlesex College, London, ON, N6A 5B7, Canada
vladat@computer.org

Bernard Pagurek
Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Dr., Ottawa, ON, K1V 6L8, 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 categories: functional, quality, and infrastructure contracts. Functional contracts include syntactic, behavioral, synchronization, and compositional contracts; quality contracts include QoS and pricing contracts; while infrastructure contracts include communication, security, and management contracts. 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. We suggest that principles for such framework should be: modularity, unification and standardization of common contract elements, extensibility, use of only few contract languages, reuse and extension of the widely accepted Web Service languages, specification of relationships between contracts, and standardization of quality contracts. At the end, we outline one possible approach to comprehensive contractual description, based on extending existing Web Service technologies.

1. INTRODUCTION

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 Web Service 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 relationships such as those between functional constraints (pre- and post-conditions, invariants), QoS guarantees (e.g., average response time), 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 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].

In this paper, we first discuss the need for a comprehensive contractual description of Web Services and Web Service compositions and the motivation for our work in this area (Section 1). Then, we identify types of technical contracts that are useful for Web Services and Web Service compositions and propose a new way of classifying them (Section 2). Next, we examine which of the identified contract types can or cannot be specified with which of several prominent Web Service languages (Section 3). Further, we suggest principles for a unified framework for comprehensive contractual description of Web Services and Web Service compositions and outline one possible approach towards such a framework, based on extending several prominent Web Service languages (Section 4). At the end, we summarize conclusions and note items for future work (Section 5).

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: 1) syntactic, 2) behavioral, 3) synchronization, and 4) 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

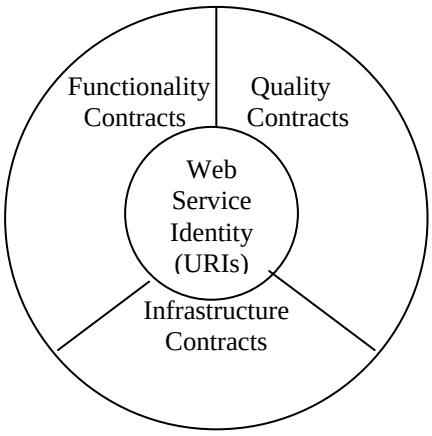

Figure 1. Categories of Contracts for Web Services

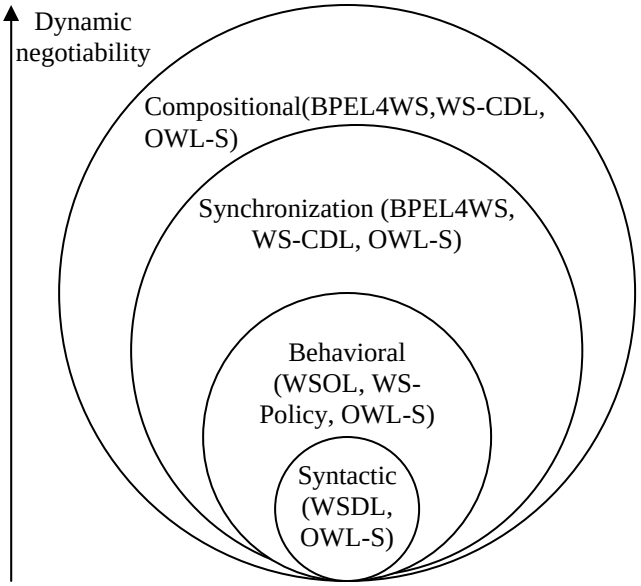

Figure 2. Functionality Contracts for Web Services

Services: functionality, quality, and infrastructure contracts. These three contract categories are shown in **Figure 1** as a comprehensive shell around the Web Service identity. The identity of a Web Service is, by the W3C's definition of a Web Service [8], a set of Uniform Resource Identifiers (URIs) for the 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 2 to 4** and discussed in the following subsections. The latter figures also show (in parentheses) some existing languages for these types of contracts.

2.1 Functionality Contracts

To use a Web Service, it is essential to understand WHAT it does, i.e., its functionality. A **functionality contract** describes functional characteristics of the operation of a Web Service. We identified the following types of functionality contracts: syntactic, behavioral, synchronization, and compositional contracts.

Apart from the information about the identity, the basic information for invocation of a software module, including a Web Service, contains names of provided interfaces, names of operations, names and data types of input and output parameters, exceptions that can occur, and attributes. This information determines a **syntactic contract**. Syntactic contracts cannot be negotiated and cannot be changed dynamically, unless the implementation of the Web Service is changed.

While essential, specification of syntactic contracts is not enough for dynamic application-to-application collaboration. In addition, 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 correct execution of Web Service's operations, as well as effects of correct operation invocations. They contain operation pre-conditions, post-conditions, and invariants, collectively known as functional constraints. To differentiate between post-conditions evaluated immediately after the execution of an operation, and those evaluated in a more distant future, [12] introduced the concept of a 'future-condition' for the latter. Most future-conditions are related to enacted business processes, while the other post-conditions are often related to execution of Web Service's code. For example, a future-condition can be used to specify that an item bought using a Web Service has to be delivered within 7 days form the day of purchase. Further, the specification of references to ontological definitions relevant for the Web Service, implemented interfaces (port types), operations, their input or output parameters, exceptions, and attributes also belongs to behavioral contracts. While it can be argued that such ontological information represents a special type of contracts, in essence it describes requirements for and effects of the execution. Behavioral contracts usually cannot be negotiated dynamically. However, in some cases a Web Service may offer to its clients several behavioral contracts to accommodate the need for guarantees of different strength.

Oftentimes there are dependencies between invocations of operations of a Web Service. For example, when a client uses an on-line banking Web Service, it has to invoke the operation '*login()*' before invoking the operation '*checkBalance()*'. Information about such dependencies between operation invocations is specified in **synchronization contacts**. Common synchronization dependencies include sequence, arbitrary ordering, potential parallelism, and mutual exclusion. Apart from describing ordering of invocations by one client, synchronization contracts can be used for describing concurrent invocations by multiple clients. While there is some potential for dynamically negotiating synchronization contracts provided by a Web Service, this is often a choice between some alternatives predetermined by the underlying implementation.

While synchronization contracts describe restrictions on using a Web Service, **compositional contracts** describe correct Web Service compositions. They describe how a Web Service participates in one or more Web Service compositions and/or the flow of messages between Web Services in a composition. For example, let us assume that Web Service *FinancialAnalysis* implements two asynchronous operations: the input-only operation '*recieveStockInfo()*' and the output-only operation '*sendRecommendation()*'. A synchronization contract can state that an invocation of '*recieveStockInfo()*' is followed by an invocation of '*sendRecommendation()*'. It describes what Web Service *FinancialAnalysis* can do. However, a compositional contract can specify that if Web Service *StockInfo* invokes '*recieveStockInfo()*', then *FinancialAnalysis* will send '*sendRecommendation()*' to the Web Service *DecisionSupport*. This compositional contract describes one Web Service composition in which *FinancialAnalysis* participates. One synchronization contract can be implemented with many different compositional contracts. On the other hand, no com-

positional contract should violate the underlying synchronization because the Web Service might not behave correctly. Compositional contracts can be negotiated dynamically.

Many authors (e.g., [7]) differentiate between two types of Web Service compositions: orchestrations and choreographies. In an orchestration, one party controls the order of the execution of the participating Web Services. Contrary, a choreography is a collaborative, peer-to-peer (P2P), exchange of messages between the participating Web Services. In our opinion, the main difference between orchestrations and choreographies is not in the nature of the contract, but in the control of contract execution. Consequently, we categorized contracts for both orchestrations and choreographies into the same category of compositional contracts.

Figure 2 shows relationships between the types of functionality contracts for Web Services. The contract types in the figure are represented as onion layers, similarly 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.

2.2 Quality Contracts

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. They describe HOW WELL a Web Service performs its functions. In some cases, extra functional properties are as crucial as functionality. Without their formal description, they cannot be monitored and managed. Therefore, **quality contracts** are needed to describe extra-functional characteristics of the operation of a Web Service. 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. Examples of such properties are maximum response time, average response time, throughput, and availability. QoS contracts contain QoS constraints (requirements and guarantees), such as service level objectives (SLOs). Some QoS constraints are evaluated before and/or after an execution of an operation of a Web Service. An example of such QoS constraint is a maximum response time guarantee. Other QoS constraints are evaluated periodically, at particular times and/or dates. For example, guaranteed availability can be checked periodically. 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.

QoS can be classified into quality of Web Service (QoWS) and quality of business service (QoBS). For example, if a Web Service is used to order goods, a constraint on the time for on-line processing of the order describes QoWS, while a constraint on the time for delivery of the ordered goods describes QoBS. This is analogous to the distinction between a service level agreement (SLA) and a business level agreement

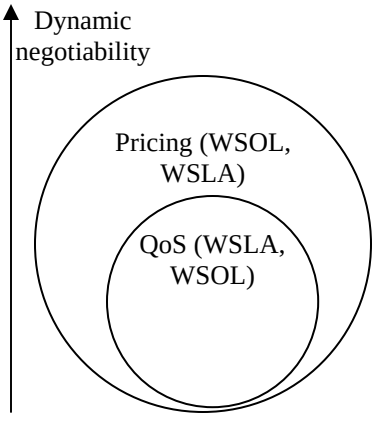

Figure 3. Quality Contracts for Web Services

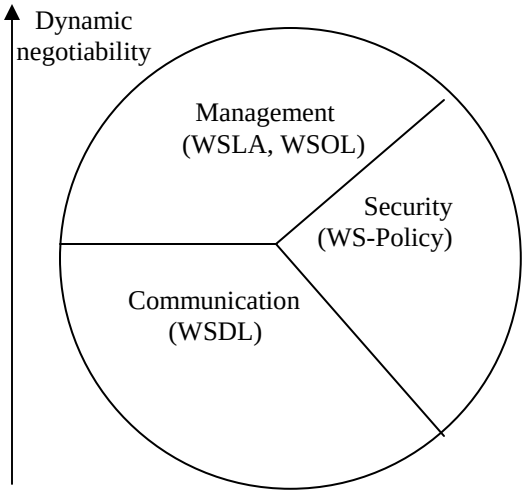

Figure 4. Infrastructure Contracts for Web Services

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		+	+	+	+					

(BLA) made by some authors (e.g., [5]). However, we decided not do separate QoS contracts into two groups in our classification, because for some Web Services QoBS is identical to QoWS. The differentiation between QoWS and QoBS can be made inside QoS contracts by defining whether a particular used QoS metric refers to QoWS or QoBS or both.

For business-to-business (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**. They are classified as quality contracts because they quantitatively describe goodness (namely: cheapness) of a Web Service.

Figure 3 shows the two types of quality contracts onion layers, where pricing contracts are an external layer to QoS contracts. Both QoS and pricing contracts can be changed and negotiated dynamically relatively easily. 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 the realities of the execution environment. Further, a QoS contract can map into many pricing contracts. Of course, it is possible to dynamically negotiate and change (hopefully: upgrade) QoS without changing price, but this is not as frequent case as changing price without changing QoS.

2.3 Infrastructure Contracts

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.

Web Services can communicate using several XML messaging protocols, most notably SOAP. Further, SOAP can run over HTTP (HyperText Transport Protocol), SMTP (Simple Mail Transfer Protocol), and other transport protocols. **Communication contracts** describe what protocols are used for packaging and transporting messages between the communicating Web Services. A Web Service can support several communication protocols and provide several endpoints (ports) implementing the same interface (port type), but with different communication protocols. In such cases, limited dynamic negotiation and change of communication contracts is possible by choosing from one of these endpoints.

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. In addition, they can contain security policies (e.g., authorization policies). Since privacy is related to security, we also group specification of privacy information and privacy policies into this contract category. There is some room for negotiating and changing these contracts during the run-time.

In practice, 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 and corresponding QoS constraints evalu-

ated, the use of the Web Service has to be accounted, and appropriated prices or proprietary penalties to be paid have to be calculated an billed, etc. **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. Consequently, a management contract references other contracts. In an ideal case, monitoring and enforcement of contracts and execution of management operations is performed automatically by management software, without external intervention. In practice, human involvement is necessary in some situations, e.g., because management activities are too complex or they contain legal material. Management contracts can be negotiated during run-time, but with some limitations. For example, while a Web Service can accept a previously unknown consumer, it might not accept a previously unknown management third party due to trust issues.

Figure 4 shows the infrastructure contract types for Web Services. Contrary to Figures 2 and 3, 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. Negotiation of communication contracts is limited mostly to choosing from some predefined options. Since there are many security and privacy technologies for Web Services and security and privacy policies can be customized, there is more room for the negotiation of security contracts. Due to multiple possibilities for dynamic customization of management contracts, they have the highest dynamic negotiability of all three types of infrastructure contracts.

2.4 Relationships between Contracts from Different Categories

Note that, contrary to Figures 2 to 4, **Figure 1** 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. One possible ordering with increasing dynamic negotiability is: 1) syntactic, 2) behavioral, 3) communication, 4) synchronization, 5) security, 6) QoS, 7) management, 8) compositional, 9) pricing contracts. However, exceptions are possible. It is also important to understand that there can be dependencies between contracts from different categories. For example, two behavioral contracts of a Web Service can be linked with two different pricing contracts. However, these dependencies are not as strong as those between contract types from the same category.

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. Apart from the shown languages, there are many other languages, mostly of experimental nature and without wide acceptance.

The **Web Services Description Language (WSDL)** [2] is the de-facto standard language for describing Web Services. It enables specification of Web Service identity, as well as syntactical and communication contracts. Hereafter, we will say that a language "builds upon WSDL" if it uses WSDL to describe this information. Since WSDL does not enable specification of the other contracts for Web Services, it has to be complemented with other Web Service languages.

The **Business Process Language for Web Services (BPEL4WS)** [10] is used for description of Web Service orchestrations. It enables specification of compositional and, to some extent, synchronization contracts. It builds upon WSDL.

The **Web Services Choreography Description Language (WS-CDL)** [3] is intended for Web Service choreographies. It supports synchronization and, to some extent, behavioral contracts. It builds upon WSDL.

The goal of the **Web Services Policy Framework (WS-Policy)** [9] is specification of policies for Web Services. It is only a general framework, while the details of the specification of particular categories of policies will be defined in specialized extensions. The only such extension currently developed is WS-SecurityPolicy. In addition, WS-PolicyAssertions can be used for the formal specification of functional constraints. Specialized WS-Policy extensions for QoS, pricing, and management contracts can be developed, but no detailed extension was published in these areas. WS-Policy builds upon WSDL.

The **Web Service Level Agreement (WSLA)** language [4] enables specification of QoS, pricing, and management contracts. It can build upon WSDL.

The **Web Service Offerings Language (WSOL)** [11, 12] enables specifications of behavioral, QoS, pricing, and management contracts. It also has support for specification of access rights. It builds upon WSDL.

OWL-S [6] enables specification of syntactic, behavioral, synchronization, and some compositional contracts. It contains placeholders for specification of other service properties, such as QoS, price, and, potentially, security. However, these placeholders are very general and cannot be considered as contract specifications. OWL-S uses WSOL for specification of service location and communication contracts.

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. Unfortunately, different languages are not always compatible. By 'compatibility' we mean that for the same or similar concepts these languages reuse same constructs or at least define them in the same way. For example, WS-Policy and WSLA are not fully compatible because they define similar concepts (e.g., policy assertion and SLO) in different ways. This is a significant problem.

4. TOWARDS 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 compare 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. This is because the existing conceptual stack of Web Service technologies [5] does not fully explore comprehensive contractual description. Due to the recent interest in, and possible standardization of, the specification of constraints and capabilities for Web Services [14], our suggestion is important and timely. 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. This makes easier both selection of Web Services and their operational characteristics and enforcement and management of contracts. While this somewhat reduces flexibility of the contract specification, it significantly improves their 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.** There are already many languages for Web Services, which more-or-less cover all identified types of contract. 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. Regarding the other contract types, we find that WSOL, OWL-S, and WSLA have technical advantages over WS-Policy, but that the latter has much bigger support from the industry. Some of the issues with WS-Policy are [12]: 1) there are no concepts of a contract, SLA, and class of service, in spite of their similarity to policies; 2) the format for expressions in WS-PolicyAssertions is not standardized; 3) 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 relationship 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].

7) **Standardization of specification formats for quality contracts.** A large number of academic works and some industrial products are related to QoS contracts for Web Services. However, they are very diverse. While the example from the beginning of this section illustrates the need for standardization of specification formats for quality contracts, there is still no widely accepted initiative in this direction.

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 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. Although this solution has technical advantages (consequences of the minimization of the number of needed contract languages), it might not be accepted due to other issues. 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 WS-Policy extensions for QoS, pricing, and management contracts. We suggest using WSLA as the basis for this standardization. In particular, WSLA service level objectives (SLOs) can be viewed as 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 contracts (modeling static relationships), classes of service and QoS, and integration of different types of contracts into one language [11, 12]. Specification of pricing contracts is also more developed in WSOL than in WSLA, while

WSOL behavioral contracts are more detailed than WS-PolicyAssertions.

5. CONCLUSIONS AND FUTURE WORK

Due to the importance of comprehensive contractual descriptions of Web Services and Web Service compositions for selection/negotiation and management of Web Services and Web Service compositions, Web Service standards must provide their explicit, formal, and consistent specification. The research results presented in this paper can be used as guidelines for future work on contract specification, selection/negotiation, and management.

We identified that several types of technical contracts are useful for Web Services and Web Service compositions. We suggested their classification into three categories: functional, quality, and infrastructure contracts. Functional contracts include syntactic, behavioral, synchronization, and compositional contracts; quality contracts include QoS and pricing contracts; while infrastructure contracts include communication, security, and management contracts. The presented classification also answers some questions about relationships between contracts from different categories.

Our study of prominent 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. Consequently, these languages are a good basis for comprehensive contractual description of Web Services. However, one of the problems is that there is no standardization effort on behavioral, quality of service (QoS), pricing, and management contracts. Another problem is that there are incompatibilities between different languages, particularly those related to the above mentioned contract types. In addition, more work is needed on integration of contracts of different types.

Therefore, we advocate a unified framework for comprehensive contractual description of Web Services and Web Service compositions. This framework would coordinate standardization of Web Service technologies to ensure that future standards for all contract types are developed, appropriate, compatible, and usable in various combinations. We suggest that principles for such framework should be: modularity, unification and standardization of common contract elements, extensibility, use of only few contract languages, reuse and extension of the widely accepted Web Service languages, specification of relationships between contracts, and standardization of quality contracts. We suggest that such a framework should be based on:

1) the Web Service Description Language (WSDL),

2) one language that integrates concepts from the Business Process Execution Language for Web Services (BPEL4WS) and the Web Services Choreography Description Language (WS-CDL), and

3) the Web Services Policy Framework (WS-Policy) significantly extended with relevant concepts from other languages such as the Web Service Level Agreement (WSLA) language and the Web Service Offerings Language (WSOL).

One of the items for our future research is a comprehensive study of the contents 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 (QoBS), 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.

ACKNOWLEDGEMENT

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

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). On-line at: http://www.w3.org/TR/2003/WD-wsdl20-20031110

[3] Kavantzas, N., Burdett, D., Ritzinger, G., Lafon, Y. (eds.) Web Services Choreography Description Language. Ver. 1.0. World Wide Web Consortium (W3C) working draft (Oct. 12, 2004). On-line at: http://www.w3.org/TR/2004/WD-ws-cdl-10-20041012/

[4] Keller, A., Ludwig, H. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Jour. 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) On-line 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*. World Wide Web Consortium (W3C) working draft (Oct. 28, 2002). On-line at: http://www.w3.org/TR/2002/WD-ws-desc-reqs-20021028/

[9] Schlimmer, J. (ed.) *Web Services Policy Framework (WS-Policy)*. (Sep. 2004). WWW document. On-line 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. (May 5, 2003). WWW document. On-line 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). To be publ. in *Information Systems*, Elsevier. Early version in: *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 *Proc. of the Wsh. on Web Services, e-Business, and the Semantic Web at CAiSE'02* (Toronto, Canada, May 2002). Lecture Notes in Computer Science (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). On-line at: http://www.w3.org/2004/06/ws-cc-cfp.html