# Towards an Ontology-enabled Service-Oriented Architecture

Maksym Korotkiy

Vrije Universiteit Amsterdam, Department of Computer Science
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
maksym@cs.vu.nl

**Abstract.** In the field of Semantic Web Services (SWS), the OWL-S and WSMF approaches provide us with ontology-based frameworks for WSDL web services. These description frameworks enable automation of high-level tasks such as discovery, invocation and composition of web services. In our work we focus on the *software architectural aspects of ontology-enabled services* to define an Ontology-enabled Service-Oriented Architectural style (Onto-SOA). The proposed style provides for a general integration mechanism for ontologies and SOA in a manner consistent with the SOA constraints. Onto-SOA is independent from an ontology language and a particular web service technology. Therefore, it can be easily combined with other styles to devise software architectures for existing SWS frameworks as well as novel approaches to integration of ontologies and SOA.

## 1 Introduction

The Service-Oriented Architectural style (SOA) is gaining momentum. Gartner predicts that SOA will become a prevailing software engineering practice in coming years[1]. SOA attracts industry attention by providing for a flexible and cost effective way to re-use functionality captured in loosely coupled, business-aligned services. WSDL [1] web services is the most well-known, however, *not the only possible*, implementation of SOA.

The Semantic Web [2] is a vision of the next generation of the Web that makes resources published on the Web "understandable" by machines, thus improving usability of these resources. One of the main means in achieving the vision is ontologies, the responsibility of which is to be a carrier of semantics shared among agents.

The significant potential of the combination of web services and the Semantic Web has been recognized by the Semantic Web Services research area, with OWL-S [3] and WSMF [4] approaches being the two most well-known representatives. Both OWL-S and WSMF are aimed at providing an extensive ontology-based description framework for WSDL web services. In our work we analyze the combination of ontologies and web services from the viewpoint of *software architectures* [5] and one of our goals is *to define an Ontology-enabled Service-Oriented Architectural style (Onto-SOA) to provide for a general, technology- and ontology (language) independent framework for integration of ontologies and service-oriented architectures*.

[1] http://www.gartner.com/resources/114300/114358/114358.pdf

On the more practical side, our work is motivated by the problem of applying ontologies in design of applications for the e-Science domain. In the e-Science domain a large variety of knowledge intensive computational resources exists: experiment design and execution environments, model simulation and statistical analysis services. Ontologies have a great potential to improve (re-)usability of these resources. In order to realize this potential, end-user applications have to be developed. However, at present, an application developer faces both complexity of the state-of-the-art approaches and very limited guidance available on how these approaches can be applied to development of ontology-enabled software. We address this problem by *further operationalizing the proposed Onto-SOA style into a pragmatic framework ready to be applied in ontology-enabled applications*.

The rest of the paper is devoted to a brief introduction to SOA (Sec. 2) followed by the analysis of relations between SOA, web services (Sec. 3) and Semantic Web Services (Sec. 3). During this analysis we outline a gap between SOA and current approaches to ontology-enabled web services. We address the gap by defining Onto-SOA (Sec. 4). Finally, we conclude with a summary and directions for future work.

## 2 Service-Oriented Architectural Style (SOA)

An *architectural style* is “a coordinated set of constraints on architectural elements and relationships among those elements within any architecture that conforms to that style” [6]. The general architectural elements are *processing components*, *connectors* and *data*. Processing components can transform data elements. Connectors provide for an abstract mechanism that mediates communication, coordination or cooperation between components [7]. From the processing component perspective, connectors transfer data without modifying it. However, internally a connector can contain a complex subsystem that can subject the data to a number of intermediate transformations.

SOA is a style that is constrained to induce a number of desirable characteristics on a compliant architecture. In the literature, SOA is often characterized as a style that *supports loosely coupled, business-aligned, networked services to enable flexibility and interoperability, in a technology-independent manner*.

Services in SOA represent coarsely-grained expertise from an application (business) domain. A service makes this problem-solving expertise available to a consumer. SOA achieves loose coupling between its processing components (a service consumer and a service) by requiring that *connectors are simple, generic and application independent*, and that *descriptive messages constrained by an extensible schema are delivered through these connectors*. A connector is application independent and generic if it does not introduce *real dependency* and minimizes *artificial dependency* between a service and consumer.

Since the connectors are generic, all application-specific semantics must be expressed in descriptive messages that communicate a problem description from a service consumer to a service provider. The messages specify what is to be solved but not how it should be solved. The reason for this is that a service provider possesses problem-solving expertise that is missing in the service consumer.

In order to enable “understanding” between communicating components, messages *must have unified syntax and structure* and *must be expressed in a vocabulary shared by communicating components*. The vocabulary is defined in a schema, extensibility of which is very important due to an open range of possible application domains.

In SOA, loose coupling is achieved by enforcing application independent connector elements. All application-specific semantics is to be contained in data elements (messages). A service can be uniquely characterized by a set of messages it can properly interpret. These messages are constrained by a certain schema that can be seen as an abstract definition of a service. From a service consumer point of view, a service is known via this abstract definition only. Since a service represents problem-solving expertise in an application domain, the schema effectively plays a role of a specification of this expertise. Such view on a schema in SOA allows us to make a natural transition from a schema to an ontology and treat an ontology as an integral component of SOA.

## 3 Web Services

In the literature the term *web services* most often refers to WSDL/SOAP-based services. However, in our work we employ it in more general sense to refer to *a derivation of SOA that employs standard Web transport protocols (HTTP, for example) and XML language to express messages*. The main reason for this is that WSDL/SOAP is not the only possible way to apply SOA on the Web.

**WSDL/SOAP Web Services** presently are the most popular application of SOA on the Web. WSDL provides a description framework for web services and is aimed at service invocation primarily. SOAP [1] provides for a standard way to structure messages that can be carried over a variety of transport protocols with HTTP being most frequently used one.

In the very beginning SOAP supported the RPC (Remote Procedure Call) communication style only[2] and later on the document-based flavor of SOAP was introduced. The document-based SOAP messages are capable of carrying any XML-based content and are not restricted to the RPC communication style only.

From the SOA perspective, RPC introduces an additional *artificial dependency* between a consumer and a service and, therefore, hinders loose coupling between these components. Furthermore, the RPC style messages tend to be prescriptive rather than descriptive: with the RPC-message a consumer prescribes a service *how to solve a problem* rather than describing *what is to be solved*. In RPC web services, message semantics often becomes workflow-dependent and that hinders integration of services. With SOAP 1.2 the RPC style has become optional, thus, enabling us to avoid it in WSDL/SOAP web services. However, RPC still attracts most of the attention in the research community.

Document-based WSDL/SOAP web services are fully compatible with SOA. Normally they rely on the XML Schema language to describe a structure of an XML document. However, XML schemas do not allow us to capture semantics of message ele-

[2] http://www.xml.com/pub/a/ws/2001/04/04/soap.html

ments. WSDL extension elements allow us to use different schema languages capable of capturing semantics, however, there is little guidance available on that.

WSDL/SOAP web services introduce a significant amount of conceptual and architectural elements [8]. A significant part of it is devoted to the RPC communication style that is to be avoided in SOA. Moreover, in many cases a document-based invocation that is performed via the well-established communication interface (HTTP protocol and the request-response message exchange pattern) is sufficient to define an operational SOA. Such a SOA can be both simpler and sufficient for many practical applications.

**Representational State Transfer (REST) Web Services** are based on the REST architectural style [6] that is designed around concepts such as Resource and URI. The interface among agents in REST is limited to the HTTP protocol that provides for both the transport layer and actions applicable to resources. REST web services normally employ XML to express messages and XML Schema as a vocabulary definition mechanism. However, since the infrastructure underlying REST web services is simple, it becomes rather straightforward to employ an ontology-based schema definition mechanism in REST web services. Another important feature of REST is that this style has been designed to fulfill requirements of the Web [9]. REST web services, therefore, gain many architectural properties of the Web which have already proven to be successful.

WSDL/SOAP web services require a significant infrastructure and have a high adoption barrier. On the other hand, REST web services require little infrastructure in addition to what is already provided by the Web, are fully compatible with the SOA constraints, can be easily made to employ an ontology-based schema definition languages and sufficient for many applications.

Despite the fact that REST web services are not the standard web services and less widely publicized, they are known in some cases to be preferred over WSDL/SOAP web services. For example, Amazon provides interfaces for both WSDL/SOAP and REST web services, and 85% of the usage is on the REST interface[3].

**Semantic Web Services** The state-of-the-art approaches to Semantic Web Services [10] such as OWL-S and WSMF employ ontologies to provide semantically rich descriptions of WSDL/SOAP services. Such descriptions provide us with meta-data about a service and can be applied to automate web service-related tasks such as discovery, invocation, composition etc. The fact that a number of related but independent tasks are addressed within a single framework makes its application cumbersome if only some of these tasks are important for a given SOA.

From the SOA perspective, SWS define a relatively complex semantic connector element responsible for establishing a connection between a service and its consumer in a way that insures semantical compatibility of messages. The connector element can perform complex operations, however, the service still operates on semantics-free messages (SOAP RPC in most cases). This makes us to believe that *the state-of-the-art approaches to SWS do not address direct exchange of semantically rich messages between processing components in SOA*. For example, if there is a web service designed

[3] http://www.oreillynet.com/pub/wlg/3005

to operate over RDF [11] messages then the-state-of-the art SWS approaches are of little use here, and if the service is not a WSDL/SOAP-based one then there is even less possibilities to apply the existing SWS approaches.

## 4 Onto-SOA: an Ontology-enabled SOA

We propose Onto-SOA – SOA that assumes a direct exchange of semantically rich messages between processing components. Onto-SOA is general meaning that it can be combined with any SOA-compatible architecture (WSDL/SOAP or REST) and with any ontology language. The style can be gradually refined into derivative styles that will support additional service-related activities.

We derive the Onto-SOA style from SOA by introducing an additional constraint on architectural data elements (messages): *An ontology language must be used to express a schema underlying messages*. This constraint explicitly addresses the semantical interoperability of messages. In Onto-SOA we assume that both a service and its consumer are aware about an ontology underlying messages.

**Implementing an Onto-SOA with an Ontology Language** To check whether the newly-derived Onto-SOA is compatible with an underlying SOA, we have to validate the introduced constraint against the general SOA requirement on schema extensibility.

In most of the modern applications of SOA, XML and XML Schema provide a unified syntax and vocabulary definition mechanism to messages. However, the XML Schema language addresses structural aspects only, leaving semantics of a defined vocabulary implicit. This leaves a problem of semantical interoperability among processing components unsolved but, on the other hand, does not restrict the flexibility in defining domain-specific vocabularies.

An ontology can provide a domain vocabulary, semantics of which is precisely defined in terms of ontological primitives of the underlying language. The ontology languages such as RDFS [11] and OWL [11] have fixed semantics and do not allow a user to extend it. The user is limited to model an application domain in terms provided by an ontology language. This means that the more restrictive an ontology language is – the less flexible an Onto-SOA based on this language becomes and, therefore, the more it diverges from the general requirements to SOA.

This allows us to conclude that in order to achieve the level of flexibility required by SOA, semantics of an ontology language must be extensible. However, at present, such languages are not available and we cannot foresee whether they will appear in future. If an extensible ontology language is unavailable then the least restrictive one (such as RDFS, for example) might provide a fair approximation.

In order to operationalize the introduced Onto-SOA, we have devised *MoRe* – a derivation of Onto-SOA (and a corresponding implementation) that combines the RDF/S languages with the elements of REST web services. *MoRe* has been applied to provide a number of services for the domain of units of measure captured in the UnitDim[4] ontology. The services are employed within the Unit Converter tool [12].

[4] Rijgersberg, H., Top, J.: UnitDim: an ontology of physical units and quantities. http://www.atoapps.nl/foodinformatics. Sec. News (2004)

## 5 Conclusions and Future Work

We propose Onto-SOA – an Ontology-enabled Service-Oriented Architectural style that addresses a direct exchange of semantically-rich (ontology-based) messages between a service and its consumer. Onto-SOA is independent from a particular web service technology (WSDL/SOAP, for example) and, therefore, can be employed within any SOA-compatible architecture. Onto-SOA makes no assumptions about an ontology language employed as a schema definition mechanism. This enables us to combine Onto-SOA with any ontology languages.

In our future work we intend to further analyze the applicability of the modern ontology languages such as RDF/S and OWL within the proposed Onto-SOA style. Further applications the *MoRe* framework – an operational RDFS/HTTP derivation of Onto-SOA – within the e-Science domain will allow us to validate the main ideas behind Onto-SOA and to provide additional guidance to software and ontology developers in designing an ontology- and service-enabled architectures.

## References

1. W3C: Web Services: Recommendations, Specifications and Documents (WSDL, SOAP, etc). (http://www.w3.org/2002/ws)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific american **279(5)** (2001) 35–43
3. W3C: OWL-S: Semantic Markup for Web Services. (http://www.w3.org)
4. Fensel, D., Bussler, C.: The Web Service Modeling Framework WSMF. Electronic Commerce: Research and Applications **1** (2002) 113–137
5. Perry, D.E., Wolf, A.L.: Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes **17** (1992) 40–52
6. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, UNIVERSITY OF CALIFORNIA (2005)
7. Shaw, M., Clements, P.C.: A field guide to boxology: Preliminary classification of architectural styles for software systems. In: COMPSAC '97: Proceedings of the 21st International Computer Software and Applications Conference, Washington, DC, USA, IEEE Computer Society (1997) 6–13
8. W3C: Web Services Architecture. http://www.w3.org/TR/ws-arch (2004)
9. Fielding, R.T.: Principled design of the modern web architecture. In: Proceedings of the 2000 International Conference on Software Engineering. (2000)
10. Cabral, L., Domingue, J., et al: Approaches to semantic web services: an overview and comparisons. In Bussler, C., Davies, J., Fensel, D., Studer, R., eds.: ESWS. Volume 3053 of Lecture Notes in Computer Science., Springer (2004) 225–239
11. W3C: Semantic Web Recommendations and Specifications: RDF/S, OWL, etc. (http://www.w3.org/sw)
12. Korotkiy, M., Top, J.: MoRe Semantic Web Applications. In: Proceedings of the ESWC'05 workshop on User Aspects of the Semantic Web. (2005)