Unified Lightweight Semantic Descriptions of Web APIs and Web Services

Unified Lightweight Semantic Descriptions of Web APIs and Web Services

Carlos Pedrinaci, Jacek Kopecký, Maria Maleshkova,
Dong Liu, Ning Li, John Domingue
Knowledge Media Institute, The Open University, UK
{c.pedrinaci, j.kopecky, m.maleshkova, d.liu, n.li, j.b.domingue}@open.ac.uk

Abstract

Recently, Linked Data and Web APIs have emerged as the preferred means of exposing data and Web application functionality. In this paper we argue that service systems should be adapted in the light of both trends. In particular we believe that i) common means for discovering and interacting with Web services and Web APIs are necessary, and that ii) we should bridge the gap between services and linked data both by supporting the publication of services as linked data and by enabling the processing of linked data by services. We show a set of technologies we have devised towards this goal.

1 Introduction

Service-orientation prescribes the development of software applications by reusing (possibly remote) services, that is software components offered via programming-language independent interfaces. Standards such as WSDL, SOAP and further WS–∗specifications were devised in order to provide the necessary technologies to support this. Although highly appealing from an engineering perspective and despite the progress in the area, the core technologies originally devised for this purpose, e.g., WSDL, still require substantial manual effort related to locating, interpreting and integrating existing services. The main reason for this is that the semantics of these services and of the data they manipulate remain implicit.

Semantic Web Services (SWS) have long tried to overcome these limitations by enriching Web service descriptions and data models with semantic annotations. The landscape of SWS is characterised by a number of approaches that, despite a few common characteristics, remain essentially incompatible due to the use of different representation languages as well as because of certain conceptual differences. Major proposals include OWL-S, WSMO, and SAWSDL.[1] Furthermore and regardless of the differences at the semantic level, the vast majority of the SWS initiatives are predicated upon the semantic enrichment of WSDL Web services, which have turned out not to be prevalent on the Web.

[1] http://www.w3.org/Submission/OWL-S/, http://www.w3.org/Submission/WSMO/, http://www.w3.org/TR/sawsdl/

The world of services on the Web has recently been marked by the proliferation of Web APIs, also called RESTful services when they conform to REST principles [3]. Major Web sites such as Facebook, Flickr, Salesforce or Amazon provide access to their data and functionality through Web APIs. This trend is largely driven by the simplicity of the technology stack as well as by the rapidity with which third parties are combining diverse APIs into mashups that provide added-value solutions. Yet, Web APIs are most often described solely through HTML Web pages that are intended for humans and provide no means for supporting their automated discovery, invocation, and composition [6].

In parallel, the publication of data on the Web is also experiencing a significant evolution that has given birth to the Web of Data, "a Web of things in the world, described by data on the Web" [1]. Underpinning this evolution is a set of best practices for publishing and connecting structured data known as linked data. Nowadays, there are about 30 billion statements about diverse domains such as media, government, life sciences, and geography, captured in this manner, which better supports applications in processing data and discovering related distributed data automatically.

In the light of the recent evolution of the Web we believe that it is necessary to rethink the principles and technologies underlying services and the development of Web applications. Our position consists of the following tenets:

- Service systems should provide an homogeneous view over the heterogeneous service technologies in use nowadays, i.e., WSDL and Web APIs;
- Semantics are essential to reach a minimum level of automation during the life-cycle of services and service-oriented applications;
- Service technologies should be aligned with linked data to promote services integration and discoverability.

In the remainder of this paper, we briefly introduce a number of technologies we have developed which implement these tenets in real solutions.

2 Service Description

Effectively supporting the development of service-oriented applications in heterogeneous environments like the Web requires models that capture in sufficient detail the characteristics of the services, such as the operations offered and the data manipulated, as well as the semantics of both the services and the data that drives them. In the light of the state of the art in service technologies but also guided by lessons learnt from research on the Semantic Web and Semantic Web Services, we propose the Minimal Service Model (MSM), a simple model for capturing service descriptions, which

- covers WSDLs and Web APIs homogeneously;
- captures the core semantics of services and data employed by the main semantic Web service models;
- minimizes modelling and processing overhead;
- closer aligns services with linked data.

In a nutshell, MSM is a simple RDFS ontology based on the principle of minimal ontological commitment. It captures the common structures of existing conceptual models for services. Thus, it does not aim to be yet another service model to bring further heterogeneity to the SWS landscape; it is instead an integration model at the intersection of existing formalisms. This model is able to capture the core semantics of both Web services and Web APIs in a way that enables the homogeneous publication and discovery of both kinds of services.

MSM characterises **Services** as being composed of a number of **Operations**, which in turn have **input**, **output** and **fault MessageContent** descriptions. **MessageContent** may be composed of mandatory or optional **MessageParts**. The model is complemented by the WSMO-Lite vocabulary [2], which defines classes for describing the four core aspects of service semantics identified by previous research on service semantics,[2] namely, functional semantics, nonfunctional semantics, behavioural semantics, and an information model. These types of service semantics are relevant for advanced discovery, selection and composition, among other tasks. The main classes of WSMO-Lite are **Condition**, **Effect**, and **FunctionalClassificationRoot**, used for capturing functional and behavioral semantics, and **NonfunctionalParameter** for nonfunctional semantics.

To attach the semantics to the service model, we use the RDF mapping of SAWSDL, which defines three properties, namely **modelReference**, **liftingSchemaMapping** and **loweringSchemaMapping**. The former links service elements to semantic models; WSMO-Lite clarifies the semantics of model reference annotations. Schema mapping properties indicate data transformations between Web service messages and their semantic representations, providing a grounding from the service's Information Model to the concrete on-the-wire messages.

2.1 Syntactic WS–∗ and Web API Descriptions

MSM is largely a simplification of WSDL; Web service description in WSDL, annotated with SAWSDL, can thus be mapped to MSM in a straightforward manner. However, the situation is more complicated for Web APIs.

Even though there have been proposals like WADL,[3] there is currently no established interface description language for capturing Web APIs. Instead developers most often provide a plain HTML web page documenting the API, which needs to be manually interpreted on a per-API basis. Additionally, current Web APIs are highly heterogeneous both in terms of the formats used for representing data (XML, JSON, and others), as well as with respect to the flavor of the interface (RESTful, RPC-oriented or hybrid) [6].

Given the fact that usually the only public element on the Web indicating the existence of a Web API is an HTML document, we have designed a simple poshformat hRESTS [4] that allows one to structure and semantically annotate HTML documents in order to adequately characterise the corresponding Web API. hRESTS defines the HTML classes **service**, **operation**, **input**, **output**, and **parameter** which we believe are self-explanatory. Additionally, in order to appropriately support the invocation of Web APIs, hRESTS can capture resource URI templates, HTTP methods, and the mapping of input message parameters to the HTTP request (URI

[2] http://lsdis.cs.uga.edu/ lib/presentations/WWW2003-ESSW-invitedTalk-Sheth.pdf
[3] http://www.w3.org/Submission/wadl/

template parameters, HTTP headers and body). With the hRESTS poshformat, we can make the HTML documentation of a Web API machine-processable, with information about the structure of the API akin to what WSDL can express about WS–∗ services. Finally, we will note that since hRESTS ultimately expresses a service description in the RDF-based MSM, RDFa can be employed as a straightforward alternative to the poshformat syntax.

3 Service Discovery and Publication

Thanks to its simplicity, MSM captures the essence of services in a way that can support scalable service matchmaking using state of the art algorithms. We have developed iServe [7], an open registry for publishing and discovering services which uses MSM and WSMO-Lite as its core conceptual model. The registry transparently supports the discovery of heterogeneous services, mainly WSDL services (described in WSDL and SAWSDL) and Web APIs (documented with hRESTS). Additional support is also provided for OWL-S services.

The essence of the approach followed by iServe lies in exposing the registered service descriptions as linked data, which we call linked services, better supporting their discoverability and explicitly capturing the essential relationship existing between the services and the data they manipulate. On the basis of this core conceptual model iServe provides a range of advanced service analysis and discovery techniques that can transparently be applied across different types of services and description formalisms. For instance, iServe currently supports input/output discovery using RDFS and SKOS reasoning, functional classification-based discovery with RDFS reasoning, and similarity-based discovery given services' textual descriptions.

4 Service Invocation

OmniVoke [5] is an invocation engine which provides a single interface for invoking linked services. The engine takes RDF data as input and returns RDF data as a response, enabling a seamless integration of linked services within applications as linked data producers and/or consumers. For services that do not handle RDF natively, the engine uses lowering and lifting schema mappings as declared on the service description in order to transform, respectively, the RDF input into the suitable data format the underlying endpoint accepts and viceversa. Currently, OmniVoke embeds an XSPARQL[4] engine to this end.

A distinctive feature of OmniVoke is its generic support for transparently invoking most Web APIs that can be found on the Web thanks to the use of semantic annotations as proposed. OmniVoke uses the grounding information supported by the MSM, which covers the vast majority of APIs one can encounter on the Web.

[4]http://www.w3.org/Submission/xsparql-language-specification/

5 Discussion

In this paper we have argued that service systems should be adapted in the light of the recent emergence of Linked Data and Web APIs as the preferred means of exposing data and Web application functionality. In essence we suggest that i) service systems should transparently support heterogeneous service technologies, esp. WSDL and Web APIs; ii) semantics are essential to provide sufficient automation in service-based applications, and iii) we should bridge the gap between services and linked data both by supporting the publication of services as linked data and by enabling the processing of linked data by services.

We have presented a set of technologies we have developed towards this goal. An important aspect of our approach is that it takes a "Remote Procedure Call" view over services. The main reasons for adopting this view are that i) a survey we carried out [6] showed that currently the majority of Web APIs take the RPC view; and ii) our analysis of the interactions between Web APIs and their programmatic clients shows that the RPC model is a good fit [4]. Additionally, maintaining an RPC view enables the reuse of most technologies produced thus far for supporting the development of service-oriented technologies.

References

[1] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data – The Story So Far. *Intl Journal on Semantic Web and Information Systems (IJSWIS), Special Issue on Linked Data*, 2009.

[2] Dieter Fensel, Florian Fischer, Jacek Kopecký, Reto Krummenacher, Dave Lambert, and Tomas Vitvar. WSMO-Lite: Lightweight Semantic Descriptions for Services on the Web, August 2010. W3C member submission, available at `http://www.w3.org/Submission/WSMO-Lite/`.

[3] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures.* PhD thesis, University of California, Irvine, 2000.

[4] Jacek Kopecký, Karthik Gomadam, and Tomas Vitvar. hRESTS: an HTML Microformat for Describing RESTful Web Services. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence (WI-08)*, Sydney, Australia, December 2008.

[5] Ning Li, Carlos Pedrinaci, Maria Maleshkova, Jacek Kopecky, and John Domingue. OmniVoke: a framework for automating the invocation of Web APIs. In *ICSC 2011 Fifth IEEE International Conference on Semantic Computing*, 2011.

[6] Maria Maleshkova, Carlos Pedrinaci, and John Domingue. Investigating Web APIs on the World Wide Web. In *Proceedings of he 8th IEEE European Conference on Web Services (ECOWS 2010)*, 2010.

[7] Carlos Pedrinaci, Dong Liu, Maria Maleshkova, Dave Lambert, Jacek Kopecký, and John Domingue. iServe: a Linked Services Publishing Platform. In *Proceedings of 1st International Workshop on Ontology Repositories and Editors for the Semantic Web, ORES 2010, colocated with* 7^{th} *ESWC*, 2010.