Towards Uniform Access to Web Data and Services

Andreas Harth[1], Barry Norton[2], Axel Polleres[3], Brahmananda Sapkota[4], Sebastian Speiser[1], Steffen Stadtmüller[1], and Osma Suominen[5]

[1]Institute AIFB, Karlsruhe Institute of Technology (KIT), Germany
[2]Ontotext, UK
[3]Siemens AG, Austria
[4]Information Systems Group, University of Twente
[5]Semantic Computing Research Group, Aalto University, Finland

Abstract

A sizable amount of data on the Web is currently available via Web APIs that expose data in formats such as JSON or XML. Combining data from different APIs and data sources requires glue code which is typically not shared and hence not reused. We derive requirements for a mechanism that brings data and functionality currently available via ad-hoc APIs into a coherent framework. Such standardised access to content and functionality would reduce the effort for data integration and the combination of service functionality, leading to reduced effort in composing data and services from multiple providers.

1 Introduction

The trend towards publishing data on the Web is gaining momentum, particularly spurred by the Linking Open Data (LOD) project[1] and several government initiatives aimed at publishing public sector data. Data publishers often use Linked Data principles[2] , which leverage established Web standards such as Uniform Resource Identifiers (URIs), the Hypertext Transfer Protocol (HTTP) and the Resource Description Framework (RDF[3]) Data providers can easily link to data from third parties via reuse of URIs. However, there is still a lot of data residing in silos that could be beneficially linked with other data, but will not be published as a fully materialised knowledge base. Access to such data is commonly provided via Web APIs, which are often based on Representational State Transfer (REST) principles [4]. Currently deployed Web APIs return data as JSON or XML, which requires glue code to combine data from different APIs.

With regard to Web services a lot of effort was put in the extension of the existing WS-* stack, grounded in SOAP and WSDL, with semantic capabilities. This approach is commonly referred to as *Semantic Web Services* (SWS).

Linked Services (LS) on the other hand follow the idea of combining RESTful services with Linked Data principles. LS aim to provide uniform access to data behind Web APIs as well as the provision of complex functionalities.

In this paper we present use cases, derive requirements based on the use cases, point out existing work and opportunities for standardisation.

2 Use Cases and Requirements

2.1 Use Case 1: Medicine Reminder and Dispenser Service

A "medicine-reminder-and-dispenser" (MRD) service offers complex functionalities to help people with memory problems to take the right medicine at the right time. A dispenser service enables the release of medicines at the right time and also monitors whether the medicines have actually been taken. A reminder service sends a reminder message to a subscribed user at the predefined times as long as the medicine was not taken. At a later point in time, the requirement changes in such a way that if the recipient of the reminder message does

[1]http://linkeddata.org/
[2]http://www.w3.org/DesignIssues/LinkedData
[3]http://www.w3.org/TR/rdf-concepts/

not take the medicine even after N repetitions of the reminder, an alarm should be raised to seek external help for those recipients who are having life threatening health conditions.

Since both basic services (reminder and dispenser) relate to each other, a MRD service needs to compose them in a complex way. Additionally the functionality of the MRD is subject to conditions derived from existing datasets about the recipients, government regulations, and the medical protocols.

2.2 Use Case 2: Geospatial Integration

The Helsinki Region Service Map[4] allows residents to look for local government service points such as hospitals, parks and kindergartens. The service map also provides a REST service API which exposes the data in XML and JSON formats. However, using the API together with other data requires custom glue code.

A Linked Data Service wrapper has been created which provides the same data in RDF format. The RDF format is based on standard schemas such as FOAF, vCard and the Organization Ontology. The instance data is linked to other data sources such as the Finnish Municipalities Ontology. The wrapper is implemented as a Java servlet that performs an XSLT transform from the original XML into RDF/XML.

2.3 Requirements

Based on use cases and experience with previous projects, we derive the following requirements:

- **Uniform data representation:** to facilitate data access and exchange, we aim at a uniform data model.
- **Uniform access protocol:** to facilitate data access and exchange, we aim at a uniform data access and exchange protocol.
- **Referencing:** to enable data providers to interlink data from different sources, we require means to reference data items.
- **Self-describing data:** to facilitate understanding of data, we require self-describing data.
- **Self-describing services:** to facilitate understanding of inputs and outputs of services (to e.g., allow for querying service descriptions), and what the service does, we require the ability of a service to describe itself.
- **Access control:** to allow for guarded access to data and update functionality (for security reasons or to charge money), access to services must be controlled.
- **Streaming access:** to allow for realtime access to content, we require the ability to access streams of data.
- **Long-running jobs:** to allow for large inputs, we require that the service is able to run for a duration of hours or days.
- **Scale:** to enable scale (both on the social level and the technical level), the mechanism should be lightweight and close to widely-deployed solutions. In addition, a layered and distributed architecture is preferable to a monolithic one, to enable separation of concern and distribution of work.

3 Evolution: from a Syntactic Monolith to Distributed Semantics

The following section gives an overview of the existing approaches related to data and service integration. Figure 1 illustrates the classification of these approaches. Additionally we provide a mapping of the formalised requirements.

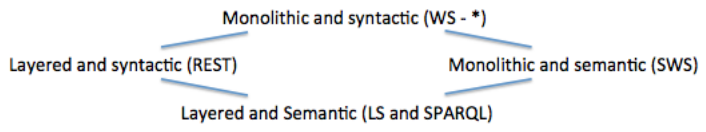
Monolithic and syntactic (WS - *)
Layered and syntactic (REST)
Monolithic and semantic (SWS)
Layered and Semantic (LS and SPARQL)

Figure 1: Classification of described approaches

3.1 WS-*

The mainstream XML standards for interoperability of Web services, such as WSDL, specify only syntactic interoperability, not the semantic meaning of messages. Data modelling is typically performed using XML Schema. However, XML Schema is not really concerned with modeling data, but is rather a language for defining hierarchical document models, especially their structure and syntax but not their semantics. Thus, it enforces a hierarchical representation on the data which in most cases is unnatural [2]. While messages are

[4]`http://www.hel.fi/palvelukartta/`

exchanged in the standardised SOAP format, there is support for various transportation channels including HTTP, but also email protocols.

3.2 Semantic Web Services

General approaches to semantic web services include WSMO [3] and OWL-S [5], both follow the approach of annotating messages with ontological classes. This does not demark the data that will be exchanged at the semantic level (for instance an output annotated foaf:Person does not make clear whether the identifier for a user account will be returned, or the metadata that may apply as properties in the FOAF schema). Furthermore, both rely on extensions in non-standard rules languages to express the relation between service input and output, but equally leave this ambiguous as the provided examples use the same properties to express changes in the state of the world (i.e., do we return the foaf:firstName already associated with a person, or assert a new one?). SAWSDL (Semantic Annotations for WSDL and XML Schema)[1], the only standard in the SWS area, extends WSDL merely with class-based annotations and does not address the second question at all.

3.3 JSON over HTTP

JSON, standing for JavaScript Object Notation, is the serialisation format for simple data structures, originally parsed out of and into JavaScript objects but now available via libraries in many programming languages, based on associative arrays. These extend key-value pairs with hierarchical structure without the overhead of XML, in terms of structural complexity, physical size and the use of explicit and pre-defined structural schemas. Unfortunately intentional schemas are also missing in this approach, so the key 'address' in one serialisation cannot be compared with the use of the same string in another serialisation except when these directly result from interaction with the same resource, or at least API. The JSON Schema effort[5] attempts to address at least the structural definition of JSON data models.

To a greater or lesser degree APIs, often advertised as RESTful, allow the exchange of JSON using HTTP against a family of resources (for instance albums, photos and comments in the Flickr API) with a common interpretation for keys. These semantics are not, however, shared between APIs.

3.4 SPARQL 1.1

We note that the goals of presented requirements are fulfilled to some extent already by the recent extensions of SPARQL towards SPARQL1.1[6]:

- The SPARQL1.1 Query Language will add various expressive query language features including aggregates and subqueries. The query language is useful for expressive descriptions of the relations between service inputs and outputs, and furthermore is suitable for processing self-describing RDF data.
- The SPARQL1.1 Protocol and the SPARQL 1.1 Graph Store HTTP Protocol provide standard interfaces to SPARQL services, i.e., services providing and together with the SPARQL1.1 Update Language also consuming data.

Still, SPARQL1.1 misses some bits which still await action from within W3C towards standardisation, such as:

- **Linking between RDF, XML, and also JSON**; while there are proposals such as the XSPARQL W3C member submission [8] towards combined XML-RDF query languages for facilitating RDF-XML transformations, there is no standard means to integrate these two formats as of yet, except the detour via RDF/XML or the SPARQL XML result format. Note that, along similar lines, for JSON, a SPARQL JSON result format is currently underway, along with discussions about standard RDF serialisations of JSON in the recently re-chartered RDF1.1 working group [7]
- Despite **Templating** is supported in many practical extensions of SPARQL, there is no standardised mechanism yet to describe SPARQL query and update templates and, respectively, means to invoke such templates via a standard protocol for SPARQL queries.
- Further, the current service description vocabulary is still very rudimentary; agreement on several extensions of service descriptions towards standard annotations with ontologies/vocabularies of capabilities is still missing, e.g. advertisement of said templates in the previous bullet point.

3.5 Linked Services

By combining LOD technologies with RESTful services [4], *Linked Services* (LS) [9] [6] try to overcome limitations arising in the development of Linked Data, that result from the static aspect of data. LS offer complex

[5] `http://json-schema.org`
[6] `http://www.w3.org/standards/techs/sparql`
[7] `http://www.w3.org/2011/rdf-wg/wiki/TF-JSON`

Web service functionalities as RDF prosumers and aim to establish efficient service functionality against the background of the Web of Data. Additionally LS provide uniform access to legacy data by automatic interlinkage with existing data sets.

LS base their service descriptions on the notion that Linked Data provides a better description for service input and output requirements: the graph patterns provided by the SPARQL query language. These provide the advantage of familiarity to Linked Data producers and consumers, but also of a more thorough description of what should be communicated and the possibility for increased tool support. This also allows for an easy discovery by introducing the notion of *service templates* [7].

As a principle LS demand to make the relation between input and output explicit, which happens by reusing variables from the input in the output graph pattern. It is possible to differentiate between safe and unsafe variables: variables in the output graph pattern, that are bound by the service functionality, do not appear in the input graph pattern and are called unsafe.

3.6 Feature Overview

As shown in Table 1, the traditional WS-Stack as well as the communication of JSON data via HTTP are lacking semantic capabilities. This hinders automatic data and service integration, which is enabled by the characteristic to be self-describing and the possibility to easily reference data from different sources.

	WS-*	SWS	JSON o. HTTP	SPARQL 1.1	LS
Uniform data representation	-	x	-	x	x
Uniform access protocol	-	-	x	x	x
Referencing	-	o	-	x	x
Self-describing data	-	x	-	x	x
Self-describing services	-	x	-	o	x
Access control	x	x	x	o	o
Streaming access	o	o	-	o	-
Long-running jobs	x	x	-	-	-
Scale	-	-	x	x	x

Table 1: Feature matrix for described approaches (x = Standard or known approach, o = available to some extend, − = N/A)

The more on semantics focused approaches of SPARQL1.1 and Linked Services are at the moment failing to address directly the more traditional issues of service provisioning. This can partially be explained by the fact, that SPARQL 1.1 and LS relinquish this to other technologies, thus enabling a layered architecture (e.g. in the case of access control). But other problems, like long running jobs, still have to be solved in that area.

By annotating WSDL with semantic descriptions, approaches like SAWSDL try to combine these two worlds. This however, results in a loss of some semantic capabilities (e.g., relating service input and output) and suffers from a lack of scalability on the social level.

4 Standardisation Opportunities

To enable standardised access to content and functionality provided by Web APIs, we see the following areas which would benfit from standardisation:

- Data representation and access: provide mappings between XML and JSON to RDF as unifying data format; introduce the notion of URIs into content from APIs. HTTP is the preferred access protocol for Web APIs using RESTful interfaces.
- Self-describing data and services: current Web APIs lack machine-readable descriptions of their interface. RDF and SPARQL could serve as representation mechanism for describing input/output, functionality and other characteristics of services.
- Access control: current offerings use OpenID or OAuth for authenticating users.
- Streaming access: RDF over HTTP could be one candidate for offering access to data streams. Efficiency has to be considered as streams can consume large amounts of bandwidth.
- Long-running jobs: as data volumes rise, a standardised mechanism for the integration of heterogeneous data and services requires means to cater for long-running jobs (i.e., jobs with a duration of hours or days).

5 Conclusion

It is our belief, that for data and service integration lightweight and layered mechanisms are preferable over monolithic, potentially complex approaches. REST over HTTP provides an already widely-deployed architecture to achieve this goal. Furthermore, the capabilities of semantics, introduced by Linked Data and SPARQL, can be leveraged to achieve an easy data integration and service interoperability. It should be noted, that SPARQL 1.1 and LS do not compete, but rather share a common ideology: SPARQL 1.1 follows the idea of a lightweight approach for data manipulation with the Graph Store HTTP Protocol. LS on the other hand, sharing some common functionalities with SPARQL 1.1, allow for a simple LOD-style data access. Additionally LS can make use of SPARQL to provide functionalities, that go beyond the intention of SPARQL endpoint services, namely the provision of complex services with possible real world side-effects.

References

[1] SAWSDL Working Group (W3C Web Services Activity). Semantic annotations for wsdl and xml schema: W3c recommendations, 2007.

[2] Michael Blaha. Data modeling is important for soa. In Juan Trujillo, Gillian Dobbie, Hannu Kangassalo, Sven Hartmann, Markus Kirchberg, Matti Rossi, Iris Reinhartz-Berger, Esteban Zimányi, and Flavius Frasincar, editors, *Advances in Conceptual Modeling – Applications and Challenges*, volume 6413 of *Lecture Notes in Computer Science*, pages 255–264. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-16385-2_32.

[3] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services.* Springer, 2006.

[4] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, Irvine, 2000.

[5] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara. OWL-S: Semantic markup for web services. Available at `http://www.daml.org/services/owl-s/1.1/overview/`, November 2004.

[6] Barry Norton, Reto Krummenacher, Adrian Marte, and Dieter Fensel. Dynamic linked data via linked open services. *Workshop on Linked Data in the Future Internet at the Future Internet Assembly*, pages 1–10, 2010.

[7] Barry Norton and Steffen Stadtmüller. Scalable discovery of linked services. In *Proceedings of the Fourth International Workshop on REsource Discovery*, volume 737, Heraklion, Greece, Mai 2011. RED Workshop, CEUR-WS.

[8] Axel Polleres, Thomas Krennwallner, Nuno Lopes, Jacek Kopecký, and Stefan Decker. XSPARQL Language Specification, January 2009. `http://www.w3.org/Submission/xsparql-language-specification/`.

[9] Sebastian Speiser and Andreas Harth. Integrating linked data and services with linked data services. In *Proceedings of 8th Extended Semantic Web Conference, ESWC 2011*, pages 170–184, 2011.