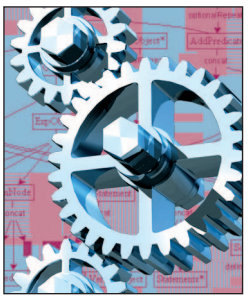

SA-REST: Semantically Interoperable and Easier-to-Use Services and Mashups

Amit P. Sheth and Karthik Gomadam • *Wright State University*
Jon Lathem • *University of Georgia*

Services based on the representational state transfer (REST) paradigm, a lightweight implementation of a service-oriented architecture, have found even greater success than their heavyweight siblings, which are based on the Web Services Description Language (WSDL) and SOAP (see www.w3.org/2005/Talks/1115-hh-k-ecows/ for a comparison). By using XML-based messaging, RESTful services can bring together discrete data from different services to create meaningful data sets; *mashups* such as these are extremely popular today.[1]

Mashups

Although mashups fully embrace the idea of customization on the Web, read–write is another story. Without technical training, it's difficult for average users to create a mashup — they need to understand not only how to write the code but also the APIs and descriptions of data elements for all the services to be included. To solve this problem, several companies are developing tools for mashup creation that require little or no programming knowledge. These tools, exemplified by Yahoo's pipes, IBM's QEDwiki, and Google's Mashup Editor, facilitate the selection of some number of RESTful Web services or other Web resources and chain them together by piping one service's output into the next service's input while filtering content and making slight format changes.

One drawback of these tools is that they're limited in the number of services with which they can interact — typically, they deal with services internal to the company that created them or to services that have standard types of outputs such as RSS or Atom. Anyone with experience in data interoperability and integration also knows that it's hard to integrate data through purely syntactic and structural means — semantic techniques are required. If a company developing a mashup tool wanted to add a new service that didn't have a standard output or that wasn't internal to their tool, it could modify its existing tooling in order to incorporate the new service's interface. However, this solution isn't scalable because of the rate at which new services are coming online. The need to change the tool itself also negates the idea of a customizable Web.

How, then, to address these limitations and find a less complex yet scalable approach? Reuse and data mediation have led to several proposals for Semantic Web services, leading to the W3C recommendation for the Semantic Annotation of WSDL and XML Schemas (SAWSDL; www.w3.org/TR/sawsdl/). But adding semantics to REST is more challenging than adding semantics to WSDL. Unlike WSDL, REST-based services are often embedded in Web pages written largely in XHTML. Although WSDL was specifically created to capture service descriptions and has a supporting schema for doing so, XHTML is a more general-purpose language that adds semantic annotations only to those page elements that wrap a service or a service description.

For an open, flexible, and standards-based approach to adding semantics to RESTful services, we built the SA-REST description by borrowing the idea of grounding service descriptions to semantic metamodels via model reference annotations from SAWSDL. The key difference between SAWSDL and SA-REST is that although SAWSDL annotations were added to formal service descriptions in WSDL, SA-REST annotations will have to be added to the services that are usually described in Web pages composed in HTML. Consequently, SA-REST uses RDFa (www.w3.org/TR/xhtml-rdfa-primer/) and Gleaning Resource Descriptions from Dialects of Languages (GRDDL; www.w3.

 1089-7801/07/$25.00

org/TR/grddl/) to add and capture annotations.

SA-REST: Foundations and Annotations

Following several efforts to add formal semantics to traditional Web services, the W3C recommendation for SAWSDL has became the baseline for developing a Semantic Web service based on WSDL (http://knoesis.wright.edu/library/resource.html?id=00068).[2] SA-REST borrows many ideas first presented in our work on WSDL-S (www.w3.org/Submission/WSDL-S/) and then adapted in SAWSDL, specifically the model reference attribute, which links and maps a service element to the ontological concepts that describe it.

The basic annotations that SAWSDL adds are inputs, outputs, operation, interfaces, and faults; in SAWSDL, semantic annotations are simply bits of XML embedded as properties in WSDL (basically, URIs of ontology objects). In other words, the annotation of a concept in SAWSDL or SA-REST ties that concept to an ontology or a conceptual model. As with SAWSDL, SA-REST doesn't enforce the choice of language for representing an ontology or a conceptual model, but it does allow the use of OWL or RDF, which are now accepted as preferred and standards-based approaches to represent ontologies. Because SAWSDL and SA-REST are more concerned with data structure than with the relationships between objects and reasoning, developers will likely use RDF more frequently in the near future because of its simplicity.

Annotation Techniques and Languages

In SAWSDL, semantic annotations that describe a service appear in that service's WSDL, which is logical because of the one-to-one correlation between WSDL and a traditional SOAP-based Web service. Most RESTful Web services don't have WSDL because the main objective of REST is simplicity (WSDL facilitates significant tooling support).

Most RESTful Web services have HTML pages that describe to users what the service does and how to invoke it — in one sense, this HTML is somewhat equivalent to WSDL for RESTful Web services, making it an ideal place to add semantic annotations. The problem, however, with treating HTML like WSDL is that the former is meant to be human readable whereas the latter was designed to be machine readable. Microformats might offer a solution: they offer a way to add semantic metadata to human-readable text in such a way that machines can glean semantics. Recently the W3C has worked on standardizing two different microformat technologies, GRDDL and RDFa. GRDDL, which recently became a W3C recommendation, offers a way for the human-readable text's author to choose any microformat and specify a translation into machine-readable text; RDFa offers a way to embed RDF triples into an XML, HTML, or XHTML document. For SA-REST, we recommend using RDFa because it's a subset of RDF, extends XHTML to annotate with markups or annotations, has built-in support URIs and namespaces, and is recognized by the W3C.

In SA-REST descriptions, we embed semantic annotations in RDFa into the HTML page that describes the service, thus making the page both human and machine readable and creating a single place to make an update if the service changes. SA-REST leaves it up to the user as to how and where to embed triples — they can be intermingled with the HTML or clustered together and not rendered by the Web browser. The triple's subject should be the URL at which the service is invoked; the predicate of the triple should be `sarest:input`, `sarest:output`, `sarest:operation`, `sarest:lifting`, `sarest:lowering`, or `sarest:fault`, where `sarest` is the alias to the SA-REST namespace. The triple's object should be either a URI or a URL to a resource, depending on the predicate. Figure 1 gives a detailed example of an SA-REST document for a Web service to search for houses on craigslist.com.

```
<html xmlns:sarest="http://lsdis.cs.uga.edu/SAREST#">
…

 <p about=" http://craigslist.org/search/">
  The logical input of this service is an
  <span property="sarest:input">
   http://lsdis.cs.uga.edu/ont.owl#Location_Query
  </span>
  object. The logical output of this service is a list of
  <span property="sarest:output">
   http://lsdis.cs.uga.edu/ont.owl#Location
  </span>
   objects. This service should be invoked using an
  <span property="sarest:action">
   HTTP GET
  </span>
  <meta property="sarest:lifting" content=
   "http://craigslist.org/api/lifting.xsl"/>
  <meta property="sarest:lowering" content=
   "http://craigslist.org/api/lowering.xsl"/>
  <meta property="sarest:operation" content=
   "http://lsdis.cs.uga.edu/
  ont.owl#Location_Search"/>
 </p>
```

Figure 1. SA-REST document. For a craigslist.com service, this document describes semantic annotations both inside the <meta> tag as well as inside formatting tags such as . Annotations in the <meta> tags aren't visible to the user, but he or she can see annotations in the .

Using GRDDL

To build in more flexibility and lower the barriers to entry, SA-REST also allows the use of GRDDL for attaching

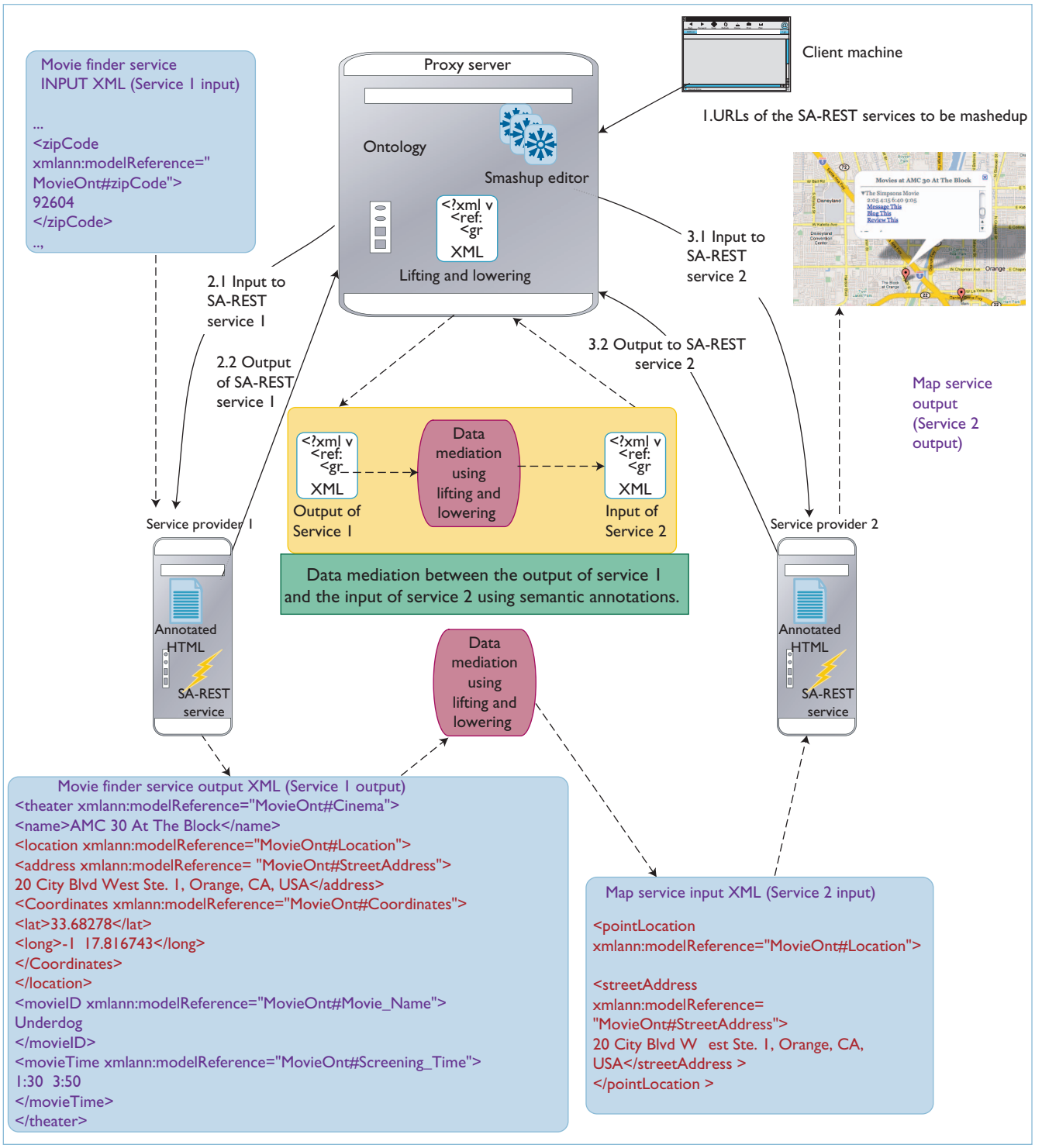

Figure 2. Mashup architecture. User query (1) results in invocation of service1 (2.1 and 2.2); data mediation (DM1 and DM2) of output XML from service 1(2.2) maps the input of service2; and the proxy server is the container for the smashup editor, ontologies, and data mediation rules (as XSLT).

annotations. To annotate an HTML page with GRDDL, the author must first embed the annotations in a microformat and add a profile attribute to the `<head>` tag in the HTML document. This attribute is the GRDDL profile's URL, which tells the agents coming to the HTML page that it was annotated with GRDDL. The final step

is to add a link tag inside the head element that contains the translation document's URL. Although you can use any format to add annotations to this page, the data extracted after translation must result in RDF triples identical to those that would be generated via RDFa embedding. In other words, a page annotated with GRDDL must still produce triples whose subject is the URL used to invoke the service, whose predicate is the type of SA-REST annotation applied, and whose object is the URI or URL of the resource to which the predicate refers.

GRDDL's advantage is that it's less intrusive than RDFa, and it lets the user embed annotations in whatever way is most convenient to them. RDFa's advantage is that annotations are self-contained in the HTML page, so the user only needs to create and maintain a single document (GRDDL forces the user to create two documents, the HTML page and the translation document). RDFa also has the advantage of being a standardized microformat, which makes it simpler for a developer to maintain and understand a page created by someone else.

Creating Mashups with SA-REST

A mashup uses RESTful Web services to query providers and get content, usually in XML format. However, the different data definitions and representations used by various providers necessitate a semantic approach for seamless data integration. Using semantics to integrate and coordinate mashups thus gives us *smashups* (semantic mashups).[3]

Annotations give smashups the ability to know more about a service's inputs and outputs and what the service does, which facilitates data mediation. Figure 2 shows a typical smashup architecture that uses two RESTful Web services, (a movie finder service and a mapping service).

The key component of this architecture is the proxy server, which hosts the smashup editor; the ontologies that capture the semantics are also deployed here, which lets a smashup developer achieve various tasks including semantic reasoning and data mediation. Users can also specify data mediation using the principles of lifting and lowering with the XSLTs that capture these mediation rules. In Figure 2, we used two services to create a smashup, and the user submits the URLs of the annotated HTML pages to the proxy server. The proxy server applies an XSLT to the annotated pages and extracts (or gleans, in the case of GRDDL) the RDF triples captured in the annotations and thus creates the service descriptions. The user then uses these descriptions to create the smashup.

At this time, the user can go through and specify from where all the inputs should be gathered and whether each input should be an input to the service or an output from a service higher up the chain. In other words, if the first service returns a location object as an output, the input to the second service can either be obtained as an input to the smashup or be the location object from the first service. In the example in Figure 2, the user sends a zip code object to the movie finder service. The proxy server extracts the location information from the output of the movie finder service and sends it as an input to the mapping service. Once the mapping service returns the map canvas, the rest of the information about the movie title and the timings are displayed in the map.

We discuss SA-REST in more detail elsewhere;[4] it's still in an early form, but just as WSDL-S matured into SAWSDL with community participation in the W3C working group, we hope additional effort will make SA-REST more mature and useful. The W3C's Semantic Web services testbed incubator, SWS-XG (www.w3.org/2005/Incubator/swsc), is also expected to provide a forum for community discussion. Additional discussions on the role of semantics in the broader context of services science that encompasses Web services and technical issues, as well as human and organizational issues, appear elsewhere,[4] as does a view of the number of experts in the Semantic Web services area.[5]

References

1. B. Worthen, "Mashups Sew Data Together: Software Tools Can Cut Costs, Time for Linking Information Sources," *The Wall Street J.*, 31 July 2007, p. B4.
2. K. Verma and A. Sheth, "Semantically Annotating a Web Service," *IEEE Internet Computing*, vol. 11, no. 2, 2007, pp. 83–85.
3. A. Sheth, K. Verma, and K. Gomadam, "Semantics to Energize the Full Services Spectrum: Ontological Approach to Better Exploit Services at Technical and Business Levels," *Comm. ACM*, vol., 49, no. 7, 2006, pp. 55–61.
4. J. Lathem, K. Gomadam, and A. Sheth, "SA-REST and (S)mashups: Adding Semantics to RESTful Services," *Proc. IEEE Int'l Conf. Semantic Computing*, IEEE CS Press, 2007, pp. 469–476.
5. D. Martin et al., "Semantic Web Services: Part 2," Part 1," *IEEE Intelligent Systems*, vol. 22, no. 5, pp. 12-17. Also, D. Martin et al., "Semantic Web Services: Part 2," to appear in *IEEE Intelligent Systems*, vol. 22, no. 6.

Amit P. Sheth is the LexisNexis Ohio Eminent Scholar and director of the Knowledge-enabled Information and Services Science (Kno.e.sis) Center (http://knoesis.wright.edu) at Wright State University. He is an IEEE fellow. Contact him via http://knoesis.wright.edu/amit.

Karthik Gomadam is a PhD student and a researcher at the Kno.e.sis Center in the areas of semantic middleware and Semantic Web services. Contact him at gomadam-rajagopal.2@wright.edu .

Jon Lathem is a software engineer at National Crash Registry (NCR). He has an MS in computer science from the University of Georgia. Contact him at lathem@cs.uga.edu.