From Moby to SADI - Modeling Semantic Web Services with the Semantic Automated Discovery and Integration Framework

Mark Wilkinson, Benjamin Vandervalk, Luke McCarthy, Edward Kawas, David Withers
Heart + Lung Institute at St. Paul's Hospital, University of British Columbia, Vancouver, BC, Canada
Email: markw@illuminae.com **Project**: http://sadiframework.org **Code**: http://sadiframework.org/content/links-and-docs/

In 2001 the BioMoby project was established. It's goal was to define a framework for modeling, annotating, and discovering Web Services to improve interoperability between bioinformatics resources. A key aspect of the Moby solution was a centralized ontology describing bioinformatics data structures; all data within the BioMoby system had to be represented as instances of these ontological classes. The Object Ontology, while being the key to Moby's success, was also the most mis-understood, mis-used, and widely disliked feature of the Moby solution. This monolithic structure conflated semantics and syntax in a manner that was confusing for newcomers to the project. In addition, Moby data, while being highly predictable in structure, could not be represented in XML Schema, thus making existing Web Service toolkits largely unusable. For these and other reasons BioMoby reached a peak of ~1500 Services in 2008, but has not seen any appreciable adoption since then.

In 2004, the W3C announced their recommendation of RDF (Resource Description Framework) and OWL (Web Ontology Language) for syntax and semantics , respectively, on the Semantic Web. Using our experiences with BioMoby as a requirements-guide, and utilizing these new Semantic Web technologies, we have designed and implemented a novel Semantic Web Service Framework – SADI (Semantic Automated Discovery and Integration). SADI adheres closely to existing standards, eliminates troublesome Web Service technologies such as SOAP, embraces the openness and distributed nature of the Web, while offering greater discovery power and interoperability than we achieved in BioMoby.

The core of the SADI "technology" are three simple, lightweight best-practices for modeling Web Services on the Semantic Web:

1. Web Services should consume and produce data in RDF syntax, consistent with how data is represented on the Semantic Web.
2. Web Services should consume OWL Individuals (“instances”) of one ontological Class, and should generate OWL Individuals of another ontological Class. These two owl Classes, therefore, define the interface for that Web Service in much the same way as the "message" and "types" portion of a WSDL document define traditional Web Service interfaces. This OWL document is published openly on the Web for indexing by any mechanism.
3. The URI (Uniform Resource Identifier – usually a URL) of the OWL Individual passed as input to a service must be the same as the URI of the OWL Individual that is returned as output. **This best-practice is key to all of the semantic service discovery behaviours of SADI.**

The SADI project makes following open-source codebases available from its project homepage: http://sadiframework.org under the New BSD License. Perl modules are available from CPAN.

- SADI::SADI - Perl support for creating SADI-compliant services in Perl. The SADI libraries use our PLUTO module (also on CPAN) to automatically create Perl objects corresponding to OWL Class definitions. Consuming input and creating output data for a SADI service simply requires getting/setting values on these objects, with no direct manipulation of RDF.
- SADI for Java: tools for building SADI clients and services in Java, including Maven plugins for service generation and testing and a rich client API for discovering and invoking SADI services based on the Jena Semantic Web framework.
- SHARE for Java: a Java library for dynamic resolution of SPARQL queries using publicly-available SADI services.
- SADI Taverna Plugin: enables semantically-aided discovery of SADI services to add into Taverna workflows (LGPL 2.1 license; bundled with Taverna)
- SADI Protegé Plugin: Currently only in prototype, this integrates creation, testing, and deployment of SADI services in both Java and Perl into the Protegé ontology editor.