ABSTRACT

Service descriptions allow designers to document, understand, and use services, creating new useful and complex services with aggregated business value. Unlike RPC-based services, REST characteristics require a different approach to service description. We present the Resource Linking Language (ReLL) that introduces the concepts of media types, resource types, and link types as first-class citizens for a service description. A proof of concept, a crawler called RESTler that crawls RESTful services based on ReLL descriptions, is also presented.

Categories and Subject Descriptors
H.3.5 [Information Storage and Retrieval]: Online Information Services—Web-based services, Data sharing

General Terms
Design, Documentation, Languages

Keywords
REST, Web Services, SOA, Crawling

1. INTRODUCTION

RESTful [1] Web services are getting interest in the industry due to properties such as high scalability achieved as result of a loosely coupled design [3] and facility for deployment since it builds up on Web infrastructure and standards. REST’s main tenets include the primacy of resources, identified using URIs, and a uniform interface generally implemented by the HTTP protocol. Resources are manipulated through representations portrayed according to a media type (e.g. HTML, Atom, etc.) and some metadata. A representation represents the state of the client’s interaction within the application and contains links that are required to change the client’s state (e.g., a submit form).

Service descriptions are useful since they allow to document and publish the available functionality, requirements and restrictions, so that consumers can make assumptions, understand and invoke services safely. RPC-oriented Web services are described by the Web Service Description Language (WSDL) in terms of an endpoint that exposes functions and their input and output parameters, and some descriptions for REST services have been also proposed.

For instance, in Web Application Description Language (WADL) [2], resources are first-class objects in a description. Resources have URIs; requests have a method, input parameters and HTTP header information; and responses have HTTP response codes and media types including fault information. URI patterns for composing query parameters are also supported. As for limitations, WADL supports only the HTTP protocol, requires fixed URIs for resources that follow a structure which makes them very sensitive to URI changes, and links are second-class properties modeled as sub-elements of parameters in resource representations. Popular programmatic approaches such as ADO.Net, Restlet, JAX-RS and Open Data Protocol support also fixed URIs, and depend on the HTTP protocol. All these approaches do not properly support the hypermedia constraint of REST.

A RESTful service description thus is an description of how to interact with a set of interconnected resources. Following this approach, we propose a model that serves as the basis for a language, the Resource Linking Language (ReLL), for describing REST services. As a proof of concept, we implemented a crawler that uses ReLL descriptions to traverse real-life RESTful services. We envision that ReLL may serve for guiding automated clients in the process of composing new services (e.g. mashups), describing complex contracts involving quality of service properties, and guide programming in REST-oriented frameworks. The graph-oriented nature of REST can be also exploited to harvest Semantic Web data from REST services.
2. RELL

Figure 1 shows the metamodel of REST service descriptions. A service provides one or more resources that have optionally a URI pattern describing the constraints for resource unique identifiers instead of a fixed structure for a URI. Each resource may have representations, which are the serialization of the resource in some syntax. Each representation can contain links relating one resource to another target resource. A link has a link type with a name and a description. Links can be retrieved from representations through selectors that can be specified for example through XML Path Language (XPath), but the actual specification of the selectors depend on the representation format. Links follow the rules specified by a protocol, including the method to be used for the request, plus additional information.

3. RESTLER

RESTler is a crawler that uses ReLL descriptions as instructions for traversing a RESTful service and produces a typed graph of the crawled resources and the links connecting them. RESTler was tested by traversing the website of the School of Information (iSchool) at UC Berkeley and Twitter. The structural relations among the crawled resources where visualized using NodeXL.1

Figure 2 depicts the results found for the iSchool, it is a graph of (some of) the iSchool Web pages and their typed connections (as described by ReLL). We considered only resources such as people classified into faculty (i), students (e), staff (d) and visitors (c). People may have their own websites. The course list collection (f) (e.g., Fall 2009, Spring 2009, etc) and each course (g) page are represented by diamonds shapes. The publication collection is shown as a circle (a), each small circle in the cluster represent an individual publication. Pages are linked to each other, they form a ring near the center of the circle (b).

Cross linked relationships such as people who teach a course, and courses taught by a person are shown in clusters (g) and (k). Complex relationships also arise, for instance faculty members that teach various courses, teach the same course together or at various points in time (l), teach courses together with a student (m), or courses that are no longer taught (k).

Figure 3 shows the results of crawling a subset of the Twitter REST API. It considers the public timeline (a) shown as the figure vertex, that is the list of the 20 most recent tweets published. We then obtained the users information (c) and the users’ timeline (b), that is the list of the 20 most recent tweets published by each user. We also retrieved the 20 next pages of tweets for each user (e). For each status, we retrieve the other users mentioned in the status by computing a link based on the in_reply_to elements of the response (f). Such features are not present in the Twitter REST API, but can be expressed in ReLL. The circular shape represents the central starting point of the public timeline and how information is connected to it.

4. CONCLUSIONS AND FUTURE WORK

ReLL focuses on providing a typed linkage between typed resources, and we expect it to evolve when considering more complex services and scenarios. For instance we are using ReLL for generating a service’s Semantic Web representation, transforming individual resources and their links into sets of RDF triples. Our emphasis on interlinking may also facilitate the description of composed REST services and/or mashups, by naturally allowing to merge various resources. Challenging questions arise regarding how to handle differences in identification and authentication methods, how to represent state in a consistent way across services, how to deal with data flow across composed services, and how to express the set of business rules that describe the intended usage of a service.

5. REFERENCES


1http://www.codeplex.com/NodeXL