From RESTful Services to RDF: Connecting the Web and the Semantic Web

Rosa Alarcon [1] and Erik Wilde [2]
[1] Departamento de Ciencia de la Computacion, Pontificia Universidad Catolica de Chile
[2] School of Information, UC Berkeley

UC Berkeley School of Information Report 2010-041
June 2010

Available at http://escholarship.org/uc/item/3425p9s7

Abstract

RESTful services on the Web expose information through retrievable resource representations that represent self-describing descriptions of resources, and through the way how these resources are interlinked through the hyperlinks that can be found in those representations. This basic design of RESTful services means that for extracting the most useful information from a service, it is necessary to understand a service's representations, which means both the semantics in terms of describing a resource, and also its semantics in terms of describing its linkage with other resources. Based on the *Resource Linking Language (ReLL)*, this paper describes a framework for how RESTful services can be described, and how these descriptions can then be used to harvest information from these services. Building on this framework, a layered model of RESTful service semantics allows to represent a service's information in RDF/OWL. Because REST is based on the linkage between resources, the same model can be used for aggregating and interlinking multiple services for extracting RDF data from sets of RESTful services.

Contents

1 Introduction

The core model of the *Semantic Web* [6] is centered around resources that are identified by URIs, and by descriptions that make assertions about these resources based on properties (which also are identified by URIs) and values assigned to these properties (which can be URIs or literal values). This interconnected network of URI-described resources is defined by the *Resource Description Framework (RDF)* [19], and more sophisticated languages then build additional layers of semantics on this foundation. The underlying assumption of this approach is that the Web exposes a large number of resources, and that RDF can therefore be used to describe this large set of resources. In this paper, we describe how resources can be discovered using the Web's basic architectural principle (REST), and how this discovery process can be used to expose resources and their relationships as RDF data.

Large amounts of RDF interlinked data are required in order to provide a critical mass of information for developers, and reaching this critical mass is one of the initial problems of the Semantic Web vision. Therefore, there is a lot of activity in research projects that create large collections of RDF data by transforming structured data sources into RDF using specialized mappings, exposing the generated RDF dataset in RDF triple stores, often exposing query interfaces in a RDF-oriented query language such as *SPARQL* [26]. Resolvable semantic resource URIs are provided and triples provide connectivity in this graph of RDF data for steering semantic crawlers and Semantic Web browsers. Although this approach creates large collections of RDF data, they result in centralistic approaches where access is typically mediated through a single "endpoint" and require sophisticated mechanisms to retrieve, process, and publish the information [8].

On the other hand, there is an increasing interest in the relationship of *Representational State Transfer (REST)* [16], the architectural principle underlying the Web, and the Semantic Web. Approaches in this area vary from the semantic annotation of resources (e.g., hREST, SA-REST), to middleware that mediates resources handling, following in general the same approaches of more traditional SOAP/WSDL semantic services (e.g., WSMO). But REST requires a different approach because services are based on the principles of resources identified with unique and opaque URIs, that are resolved to clients in various "representations" with a media type, and are handled through a uniform interface. REST resources are interlinked through hyperlinks that are found in these representations and guide clients in their interactions with a service.

This paper presents a metamodel for describing RESTful services and a language for creating descriptions of these services that is based on REST's central principle, the hyperlinking of resources. This approach provides a natural mapping from the graph-oriented world of RESTful services (resources interlinked by links found in resource representations) to the graph-based model of RDF. It is possible to directly expose structured data in Web services that expose an interface in line with *Representational State Transfer (REST)* [16], thereby supporting lightweight approaches for structured data [30]. Based on this starting point of using plain Web technologies, it is possible to go one step further and expose the same data based on Semantic Web technologies. Individual resource representations as well as the hyperlinks connecting them can be mapped to RDF in a variety of ways. *Gleaning Resource Descriptions from Dialects of Languages (GRDDL)* [10] is a framework for doing this, but as long as there are well-defined mappings, any RESTful service can be transformed into an RDF graph. This even includes representations such as images or PDF documents, which might contain embedded metadata, which can be extracted with an appropriate toolset.

2 Related Work

Related work in Web services and Semantic Web can be broadly categorized into two areas. *Semantic Web Services* (Section 2.1) deal with how to either annotate service descriptions with semantic annotations, or how to design and describe Web services that directly provide support for Semantic Web technologies. *Harvesting* (Section 2.2) deals with the question of how to use existing Web services (often those which do not expose any Semantic Web data) in a way so that they can serve as providers for Semantic Web data.

2.1 Semantic Web Services

Semantic Web Services (SWS) address mainly SOAP/WSDL services which, without additional annotations, are only focused on the syntax required for describing exposed functionality (operations, input and output types). SWS approaches extend WSDL services with semantic models. For instance, OWL-S [22] proposes a meta-ontology describing services in terms of operations, inputs, outputs, preconditions, and effects; WSMO [28] follows a similar approach though domain ontologies and goals are allowed for service discovery and composition, requiring a highly specialized reasoning platform. A lightweight approach is SAWSDL [14], which allows annotations in WSDL descriptions referring to elements in a semantic model.

A similar approach has been proposed for RESTful services. Since REST services lack a service description, SA-REST [21] and hREST/MicroWSMO [20] propose a service description as an annotated resource (e.g., an HTML page) containing the list of input and output parameters, methods, and URIs exposed by a service by means of property value pairs or RDFa [1] annotations. The description is transformed to RDF using a GRDDL-based [10] strategy for generating a domain ontology in RDF, but no information about the REST resources themselves (instances) are retrieved. Battle and Benson [5] provide similar annotations to WADL [18] documents describing REST services, and also propose extensions to SPARQL in order to support an HTTP REST uniform interface. Extensions to the payload of the HTTP REST methods (e.g., `PUT`, `DELETE` and `GET`) are also proposed for keeping consistency between a REST resource and its semantic equivalence (a triple) in some triple store.

The main problem of these approaches is that they are based on the assumption of a "service endpoint" (which is then described semantically), so they basically reflect the RPC-style service model of WSDL/SOAP. By using a RPC-style for the description of the service, though, they do not align well with the principles of RESTful service design, since they disregard fundamental properties such as the hypermedia nature of REST, and the possibility of multiple representations for resources. They also introduce coupling in their design by adhering to URI templates for describing the URIs of resources, input, and output parameters [23], or in the case of Battle and Benson, they introduce new semantics to the standard REST interface.

EXPRESS [2] is a SWS model that explicitly avoids the RPC-orientation of the approaches mentioned so far. It starts from HTTP's uniform interface, and then describes the available resources in an OWL ontology. However, the model of EXPRESS is a centralized one as well, because it is assumed that there is a complete description of a Web Service's available resources, and then this description is used to generate URIs for classes, instances, and properties.

2.2 Harvesting RDF data from Web resources

As a rough classification of how to extract RDF from existing Web sources, there are approaches built specifically around using one particular dataset/service, such as DBpedia [4] and the growing set of other datasets exposed as linked data, and there are generic approaches. The generic approaches can be further categorized into those that extract information based on the explicit structures found in data sources; and those that utilize additional rules for information extraction, often based on *Natural Language Processing (NLP)* or other information retrieval methods.

In the approach described by Futrelle [17], RDF is used as the "integration layer" in a scenario of heterogeneous data sources, and the main focus is on harvesting well-known and cooperating data sources. This approach thus falls in the category where it can be applied to a variety of data sources, but they have to be cooperating in the sense that they expose RDF themselves. The harvester's main role is to be notified of new and updated data, and to pull it in from these sources. While this scenario uses RDF's power to unify heterogeneous data sources on the metamodel level, it is only applicable in closed and cooperating settings. In our approach, data sources are not required to publish RDF themselves. As long as access to data is provided through RESTful services, they can be harvested and used as RDF. A weakness of the current implementation is that updating is not supported in a way that allows efficient incremental updates, but we

plan to address this issue in our future work mentioned in Section 9, where we describe extensions to our language that represent update services (and thus the ability to use those for incremental updates) on the language level.

SOFIE [29] focuses on information extraction from Web resources, and ANGIE [25] on using both extracted information and Web services endpoints, for building a more interactive system that does not require an exhaustive crawl of data, but retrieves information on demand. SOFIE thus falls into the category of approaches that start from resource representations, and use information retrieval methods to extract RDF from them. The current implementation of ANGIE focus on the dynamics of query processing in the RDF data managed by the system, and uses a hardwired set of Web services as the back-end. Similar to SA-REST, it uses a set of lowering/lifting transformations to translate the results of function calls from and to RDF. ANGIE focuses on SPARQL processing (the framework is able to use Web services while processing SPARQL queries), and less on the ability to easily accommodate a large variety of RESTful services.

Deimos [3] is another system that starts with information found on Web pages or through Web forms, and then uses semantic analysis to map the syntax of these representations to semantically richer information. Instead of relying on the richness of links discovered in known resources, though, the approach taken in Deimos uses tagging services to discover new resources.

3 REST Semantics

Unlike WSDL services that expose URIs identifying endpoints where service functionality can be invoked (based on the underlying RPC model), REST services expose URIs for a set of resources and clients encounter URIs by following hyperlinks. In order to avoid coupling between clients and servers, resource URIs must be opaque, that is, no assumptions must be made about the structure of the URI (a popular approach that violates this principle is that of URI templates, where URIs are composed based on a template and instantiation rules). Instead, resource URIs must be discovered by following the hyperlinks embedded in a resource representation, which ensures that clients are not tightly coupled to any particular URI structure [24].

REST requires a uniform interface that depends on the scheme used for a URI, in case of HTTP, the standard methods are `GET`, `PUT`, `POST`, `DELETE`, and `OPTIONS`. Methods are external to the resources, and are invoked by sending standard messages to the server indicating the URI of the requested resource, the method, the payload of the message and possibly standard metadata (HTTP header fields). These invocations may force changes in the state of the resources according to the semantics of the method. REST resources are conceptual entities that belong to the application logic, and are rendered to the user as representations that convey a standardized format or media type (e.g., `text/html`,`application/xml`, etc.). The content of representations depends on the application scenario and requested resource, but in RESTful designs they must contain the necessary links that allow clients to discover other resources (or affect the state of a resource) by following such links.

These properties imply that there are no "endpoints" for the case of REST services, instead there is a collection of resource URIs and a set of standard operations. A resource can have multiple representations (e.g., plain text, HTML, or PDF) that can be negotiated. Resource URIs are discovered by following the links contained in representations, that is, representations contain links to related representations. The sources for semantic information differ greatly from the WSDL model: In REST graphs, resources, links and representations are fundamental, whereas in WSDL, the complexity of an exposed service is often mostly exposed as a set of available functions that must be known in advance, and then can be invoked by the client.

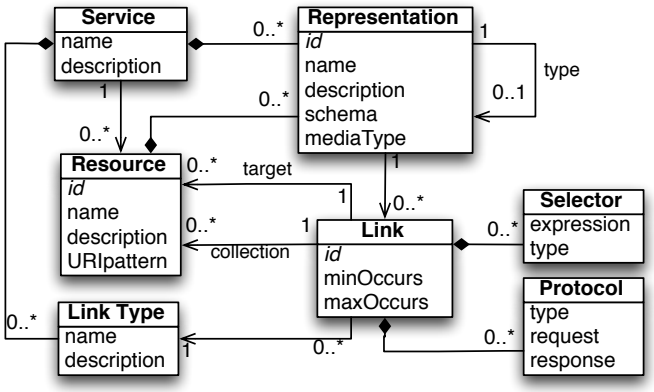

Figure 1: ReLL Metamodel for the description of REST services

4 REST Semantics in ReLL

Figure 1 shows a metamodel for REST service descriptions. This metamodel is the basis for the *Resource Linking Language (ReLL)* which is a language describing interlinked REST resources, and thus the service that can be accessed by interacting with those resources. A *service* provides one or more *resources*, with human readable names, descriptions and a *URI*. URIs are opaque and are only described as patterns (e.g., regular expressions), but even those are optional. Resources may have *representations*, that is, the serialization of the resource in some syntax (e.g., HTML, Atom, etc.) and can be associated with *schemas* for possible validation (of retrieved resources).

Representations may contain *links*, relating the represented resource to a *target* resource. A link may have a *link type* defining the semantics of the link, a name, and a description. Links can be extracted from resources by using a *selector*, which in case of XML-based representations is an *XML Path Language (XPath)* expression that allows structured selections within XML document trees. Links can also provide additional information on how to use a specific *protocol* when following the link. ReLL does not restrict the use of protocols to *Hypertext Transfer Protocol (HTTP)* [15], but for HTTP, it supports additional information about the method to be used in the request, and optionally the request payload if additional information such as query parameters or a request entity is required. Methods depend on the protocol, and thus additional methods provided by HTTP extensions such as *WebDAV* [12], *CalDAV* [11] or `PATCH` [13] can also be used.

Links that do specify a URI pattern might use URIs that do not match the pattern, indicating that it leads to a resource that is outside the scope of the service description. This means, that an application such as the crawler described in Section 8 shall not dereference that link. This highlights the fact that it is possible to have more than one description of a service, depending on the specific interests when using that service. Thus, a ReLL description often serves a specific purpose when using a service, and thus different users of a service might want to use different ReLL descriptions of that service, excluding or constraining the service with regard to representations or links they are not interested in.

Well-known linking patterns such as collections (e.g., "paged" representations) are modeled as *collection* links. These links represent sets of representations that correspond to a collection of resources. Collection links are not allowed to specify a *target*, since the target of a collection link is the same resource that contains the collection link.

```
<resource xml:id="person">
  <name>Person Page</name>
  <desc>The home page for a person.</desc>
  <uri match="http://.*?/people/(faculty|students|staff|visitors)/[a-zA-Z]+" type="regex"/>
  <representation xml:id="person-html" type="http://www.iana.org/assignments/media-types/text/html">
    <name>Person HTML Page</name>
    <link xml:id="person-website" type="website">
        <selector select="//div[@class = 'field-field-person-website']//a/@href" type="xpath"/>
    </link>
    <link xml:id="person-course" type="personcourse" target="course">
        <selector select="//span[@class = 'views-field-title']/a/@href" type="xpath"/>
        <protocol type="http">
            <request method="get"/>
            <response media="http://www.iana.org/assignments/media-types/text/html"/>
        </protocol>
    </link>
  </representation>
</resource>
```

Figure 2: A snippet of a ReLL Description for the School of Information Web Site

5 Describing Services with ReLL

ReLL descriptions are XML documents created according to the ReLL schema. Figure 3 shows part of the description of the service that publishes interlinked HTML pages on the Web site of the School of Information at UC Berkeley. This service is a Web *Content Management System (CMS)* publishing interlinked Web pages. Even though we have no control over that CMS, we can describe the set of paged made available my it as a set of interlinked resources. Figure 2 shows only one resource, a person's page that links to the courses taught by that person and the person's personal home page, and in general, we did not strive for completeness in that demonstration scenario.

Rectangles in Figure 3, which is a graphic representation of the ReLL model, represent resources, and rounded rectangles their *representations*. Arrows between representations and resources are links. Collections are sets of resources of the same type (e.g., *courseList*). The collection itself is a conceptual resource with no representation or URI. The rectangle labeled as *"Website"* corresponds to a resource that is out of the scope of the service, it thus has no representation. The link that leads to this resource from a *"person-html"* representation does not include the *target* as show in Figure 2.

6 Harvesting RDF from Resources

Figure 4 shows the ReLL metamodel mapped to a RDF/OWL semantic model. We consider four layers in the semantic model. Layer 1 corresponds to the *upper ontology* that describes general REST semantics, layer 2 corresponds to a domain ontology describing a specific RESTful service (e.g., the Web site of the School of Information). Layer 3 contains the data discovered when crawling the service, that is, the REST resources. Layer 4 contains information about concrete representations that have been used for establishing the resource relationships of Layer 3.

The elements with dashed outlines in Figure 4 correspond to the following strategy: The *resource*, *representation* and *collection* elements in the ReLL metamodel correspond to classes of the upper ontology in layer 1. *Selector* is not considered since it is a medium for selecting elements embedded in the Web representation content. We have made a difference between how links *work* in the realm of the Web, and

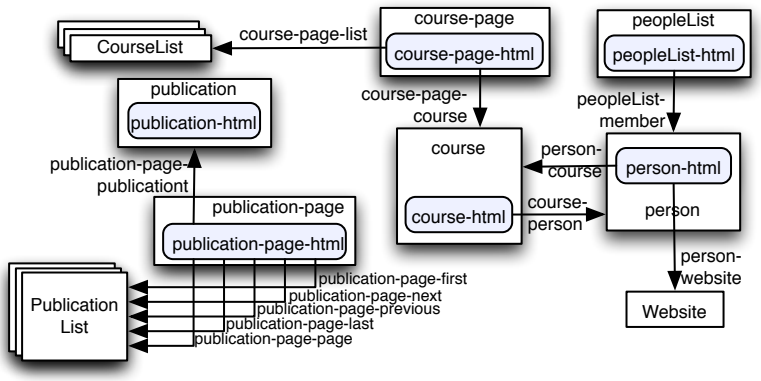

Figure 3: A ReLL based model for the School of Information Web site

what links *assert* in the realm of the Semantic Web. In the former case, links are contained in representations and thus relate representations to resources; in the latter case, representations are considered as provenance, merely indicating the source of the crawled information. Hence, in our model, representations *represent* resources, and *links* relate resources to resources. Figure 5a presents the upper ontology in N3 format.

Layer 2 corresponds to the domain ontology describing a REST service. REST resources do not have types, only their representations have media types. In the Semantic web, resource types allow a more expressive model that facilitates tasks such as querying and inferencing. ReLL *resources* and *representations* are translated into subclasses of the upper ontology[1] and *link types* become subproperties of the `rell:link` property. Figure 5 presents a subset of the generated classes (c) and properties (d). Labels and comments are omitted and the subset is limited to the example shown in Figure 4. Collections are classes (e.g., courselist) and their members are explicitly linked by means of the *link type*.

Layer 3 contains REST resources modeled as individuals of the domain classes (`rdf:type`). Since REST resources have unique identifiers, we maintain them for identifying the individuals. We create subproperties of the domain relationships for relating actual individuals and use the *link identifiers* for naming them. Figure 5e describes the case for `person-course` and `course-page-course` properties. We also restrict their domain and range. Figure 5f presents the triples asserted for the individuals shown in Figure 4 as well as their relationships. The REST resources themselves are transformed to RDF following a GRDDL approach. Figure 6 shows the attributes obtained for individuals of type `person`. Notice that it is possible to annotate the relationships between the REST resource (`erikwilde`) and its attributes. In the figure these relationships are annotated with *vCard*, but other information models can be used.

Layer 4 represents provenance. Triples obtained in layer 3 are produced when crawling a REST service, that is, a REST resource URI was dereferenced and its representation obtained, and the links to related resources were retrieved from the representation by evaluating the XPath expressions indicated by ReLL selectors. This process generates a number of triples that are tied together as a named graph [9], where the name is a generated ID (e.g., `school:r391588838`). The ID identifies a representation which is also asserted as an instance of a media type (and hence an instance of a `rell:representation`). A relationship between the representation and the resource (`rell:represents`) is also asserted (Figure 5g). We have modeled concrete representations as classes that derive from abstract representations, which correspond to *media types*. Figure 5b presents an example of the media type classes describing `text/html`, `image/jpeg`, `application/atom+xml`, and `application/xml` media types.

[1] For representations, the upper ontology contains all standardized media types from the IANA registry as classes.

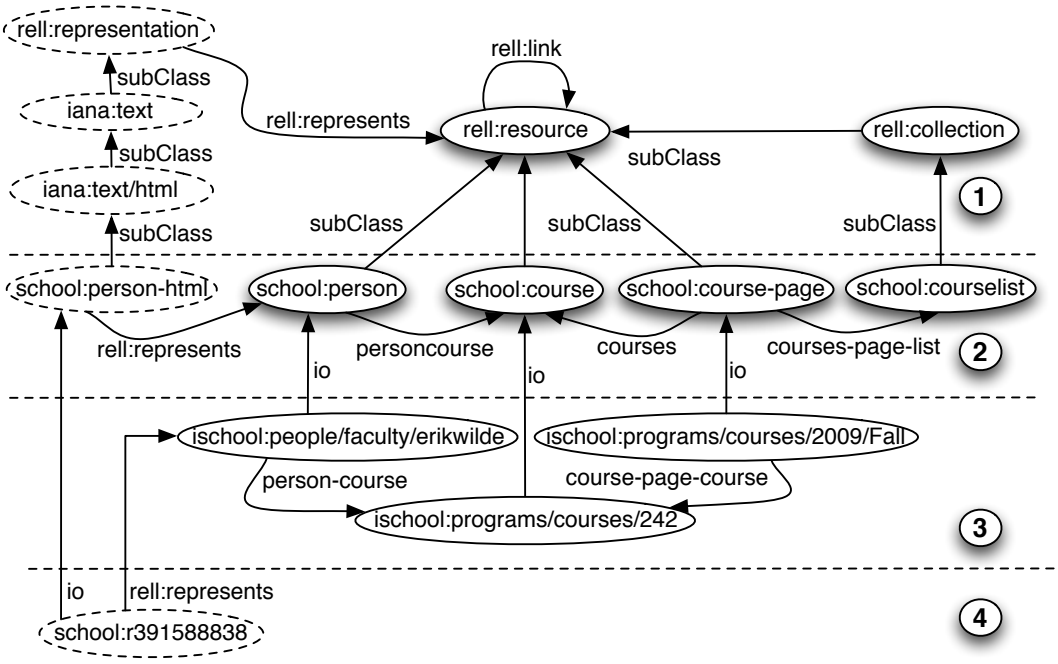

Figure 4: Semantic model for the REST Services Metamodel

Media types are annotated in the ReLL descriptions (Figure 2). Abstract representations are supported as classes that serve as the basis for other abstract or concrete representations. In Figure 4, the `text` media type is an abstract representation that serves as the basis for the `text/html` media type, which is also an abstract representation and serves as the basis for a concrete representation, that is an HTML page describing a person.

7 Composition as a Service

Three other REST services besides the School of Information Web site have been described using ReLL: a service that publishes Atom feeds through a REST API corresponding to Twitter, a service that publishes interlinked HTML pages corresponding to the Web site of Flickr, and a service that publishes an XML document listing the identities of the users of these applications (this service provides the "glue" for associating different user identities in the other services). Figure 7 presents ReLL models for each of these applications labeled as 1 (School), 2 (Twitter), 3 (Flickr) and 4 (UserMap) respectively.

The UserMap service publishes an XML document that groups together user identities for all three services (School, Twitter, and Flickr), and thus the composition accomplished by the UserMap service is simply another service, one that provides the "glue" that is required to connect the aggregated services into a connected composition of resources. For the RDF mapping, an XSLT transforms the XML document into triples that establish the equivalences between these resources through `owl:sameAs` assertions. Figure 8a presents the triples for one user. Notice that for the Twitter case, two URIs are considered, the URI of the resource as obtained from the REST API, and the URI of the resource as obtained from the Twitter Web site. The proposed approach allowed us to perform SPARQL queries that cover many REST services, as shown in Figure 8b. The query retrieves the list of distinct `flickr:cameras` for all Flickr photos of a

`person`. Figure 8c presents a snippet of the results for the user `erikwilde`.

Figure 8c can serve as a demonstration of the overall value of our approach. The fact that a particular URI identifies a person is established by using the information available on the School of Information Web site. The fact that this person is the `owl:sameAs` some Flickr user has been established by the UserMap service, where a table of associated user identities has been transformed into RDF by a GRDDL XSLT transform. That Flickr user's photos have been crawled by using the ReLL description of the Flickr service and transforming this information into RDF. Finally, the metadata about the particular camera used for each photo has been extracted from each photo's information page, again through GRDDL XSLT. This means that the only information required to get to a connected graph that can be queried as shown in Figure 8b are ReLL descriptions, XSLT transforms for some resource types, and a composition service.

8 Implementation and Results

Figure 9 shows the components for harvesting triples from RESTful services. Services are described by ReLL generating XML documents that direct the actions of a Web crawler. Since REST services do not have "endpoints", a list of seed URIs is required. There is no guarantee that the whole resource graph will be covered since this depends on how well the resources of a service are connected. While crawling, a translator component is invoked for generating RDF triples. Translation is optional, a property file gives the translator the mappings between the resource's type (e.g., `person`) that is required to translate to RDF, and XSLT code. XSLT transforms are defined for ReLL description in order to generate domain triples (Layer 2), and for the representation contents in order to generate attributes (Layer 3). Resource URIs, resource types and link types are passed to the translator in order to assert individuals and properties. We use Sesame 2.0 as triple store and the system is implemented in Java.

Sesame supports named graphs as context, which is a fourth component that can be added to a triple, and it is possible to treat that fourth component as a semantic resource (*rdf:Resource*). There is no need to modify the triples, since context is manipulated trough Sesame's Java interface. Triples corresponding to the "upper" ontology and media types taxonomy are asserted into the triple store directly. Figure 10 presents the results of querying the triple store for `ischool:people/faculty/erikwilde` through the Sesame export functionality. In the first row the representation element is shown and is presented as the context element (quad) in the following items. For the triples in Layer 3 (such as those with `school:course-person`) a new triple is inferred, since the property is a subproperty of `rell:link`, and in this case there are not a context elements.

9 Conclusions

In this paper we propose a method and implementation for harvesting triples from services and service compositions that follow the REST architectural principles. Most Web sites fall into that category, which implies that a large dataset may be available to be translated to RDF. We propose a lightweight approach that places a strong emphasis on flexibility by decoupling the main components. That is, REST services do not require modifications and do not depend on existing ReLL descriptions for our approach, and ReLL descriptions do not contain information for the translation to RDF, they can be used independently for other purposes such as documentation or as the starting point for a service contract. Mapping is isolated in one layer and as far as possible can be configured by dynamic files. We believe that this approach is sufficiently generalized to be applied to various data sources provided that they follow the REST architectural principle.

Currently, we are not providing information about the representations in the RDF data, but we intend to continue this work in that direction. We consider that they may include time stamps indicating the last time

when the resource was crawled, entity tags (ETags) served by Web servers indicating whether the resource has changed since the last retrieval, or other HTTP metadata.

The two most challenging research questions we are facing is whether we can extend the architecture to support incremental harvesting, so that large services do not need to be completely recrawled for getting updates, and whether we can further extend this direction by supporting "on-demand service access", so that queries into the triple store are actually mapped to live services instead of using harvested triples. In that latter case, the triple store would essentially become a cache instead of a separate dataset, and the problem of deciding whether to serve harvested RDF or whether to selectively recrawl the underlying services might nicely translate into the more general problem of how to efficiently describe and use cacheable services on the Web.

References

[1] Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. RDFa in XHTML: Syntax and Processing — A Collection of Attributes and Processing Rules for Extending XHTML to Support RDF. World Wide Web Consortium, Recommendation REC-rdfa-syntax-20081014, October 2008.

[2] Areeb Alowisheq, David E. Millard, and Thanassis Tiropanis. EXPRESS: EXPressing REstful Semantic Services Using Domain Ontologies. In Bernstein et al. [7], pages 941–948.

[3] José Luis Ambite, Sirish Darbha, Aman Goel, Craig A. Knoblock, Kristina Lerman, Rahul Parundekar, and Thomas Russ. Automatically Constructing Semantic Web Services from Online Sources. In Bernstein et al. [7], pages 17–32.

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *6th International Semantic Web Conference (ISWC 2007)*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735, Busan, Korea, November 2007.

[5] Robert Battle and Edward Benson. Bridging the Semantic Web and Web 2.0 with Representational State Transfer (REST). *Journal of Web Semantics*, 6(1), 2008.

[6] Tim Berners-Lee, James A. Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[7] Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, Krishnaprasad, and Thirunarayan, editors. *8th International Semantic Web Conference*, volume 5823 of *Lecture Notes in Computer Science*, Chantilly, Virginia, October 2009. Springer-Verlag.

[8] Uldis Bojārs, John G. Breslin, Vassilios Peristeras, Giovanni Tummarello, and Stefan Decker. Interlinking the Social Web with Semantics. *IEEE Intelligent Systems*, 23(3):29–40, May 2008.

[9] Jeremy J. Carroll, Christian Bizer, Patrick Hayes, and Patrick Stickler. Named Graphs, Provenance and Trust. In Allan Ellis and Tatsuya Hagino, editors, *14th International World Wide Web Conference*, pages 613–622, Chiba, Japan, May 2005. ACM Press.

[10] Dan Connolly. Gleaning Resource Descriptions from Dialects of Languages (GRDDL). World Wide Web Consortium, Recommendation REC-grddl-20070911, September 2007.

[11] Cyrus Daboo, Bernard Desruisseaux, and Lisa Dusseault. Calendaring Extensions to WebDAV (CalDAV). Internet RFC 4791, March 2007.

[12] Lisa Dusseault. HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV). Internet RFC 4918, June 2007.

[13] Lisa Dusseault and James M. Snell. PATCH Method for HTTP. Internet Draft draft-dusseault-http-patch-15, October 2009.

[14] Joel Farrell and Holger Lausen. Semantic Annotations for WSDL and XML Schema. World Wide Web Consortium, Recommendation REC-sawsdl-20070828, August 2007.

[15] Roy Thomas Fielding, Jim Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach, and Tim Berners-Lee. Hypertext Transfer Protocol — HTTP/1.1. Internet RFC 2616, June 1999.

[16] Roy Thomas Fielding and Richard N. Taylor. Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, May 2002.

[17] Joe Futrelle. Harvesting RDF Triples. In Luc Moreau and Ian Foster, editors, *International Provenance and Annotation Workshop (IPAW 2006)*, volume 4145 of *Lecture Notes in Computer Science*, pages 64–72, Chicago, Illinois, May 2006. Springer-Verlag.

[18] Marc Hadley. Web Application Description Language (WADL). Technical Report TR-2006-153, Sun Microsystems, April 2006.

[19] Graham Klyne and Jeremy J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, February 2004.

[20] Jacek Kopecký, Karthik Gomadam, and Tomas Vitvar. hRESTS: An HTML Microformat for Describing RESTful Web Services. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 619–625, Sydney, Australia, December 2008.

[21] Jon Lathem, Karthik Gomadam, and Amit P. Sheth. SA-REST and (S)mashups: Adding Semantics to RESTful Services. In *First IEEE International Conference on Semantic Computing (ICSC 2007)*, pages 469–476, Irvine, California, September 2007.

[22] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. R. Payne, E. Sirin, N. Srinivasan, and K. Sycara. OWL-S: Semantic Markup for Web Services. Member Submission, W3C, 2004.

[23] Cesare Pautasso. Composing RESTful services with JOpera. In Alexandre Bergel and Johan Fabry, editors, *International Conference on Software Composition 2009*, volume 5634 of *Lecture Notes in Computer Science*, pages 142–159, Zürich, Switzerland, July 2009. Springer-Verlag.

[24] Cesare Pautasso and Erik Wilde. Why is the Web Loosely Coupled? A Multi-Faceted Metric for Service Design. In Quemada et al. [27], pages 911–920.

[25] Nicoleta Preda, Fabian M. Suchanek, Gjergji Kasneci, Thomas Neumann, Maya Ramanath, and Gerhard Weikum. ANGIE: Active Knowledge for Interactive Exploration. In *35th International Conference on Very Large Data Bases (VLDB 2009)*, pages 1570–1573, Lyon, France, August 2009. ACM Press.

[26] Eric Prud'Hommeaux and Andy Seaborne. SPARQL Query Language for RDF. World Wide Web Consortium, Recommendation REC-rdf-sparql-query-20080115, January 2008.

[27] Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl, editors. *18th International World Wide Web Conference*, Madrid, Spain, April 2009. ACM Press.

[28] Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Rubén Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Cristoph Bussler, and Dieter Fensel. Web Service Modeling Ontology. *Applied Ontology*, 1(1):77–106, January 2005.

[29] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. SOFIE: A Self-Organizing Framework for Information Extraction. In Quemada et al. [27], pages 911–920.

[30] Erik Wilde and Yiming Liu. Lightweight Linked Data. In *2008 IEEE International Conference on Information Reuse and Integration*, Las Vegas, Nevada, July 2008.

```
@prefix rell: <http://rell.org/rell/> .

rell:representation a rdfs:Class .
rell:resource a rdfs:Class .
rell:collection rdfs:subClassOf rell:resource .

rell:link a owl:ObjectProperty ;
   rdfs:domain rell:resource ;
   rdfs:range rell:resource .

rell:represents a owl:ObjectProperty ;
   rdfs:domain rell:representation ;
   rdfs:range rell:resource .
```

(a)

```
@prefix iana: <http://www.iana.org/assignments/media-types/> .

<iana:text/html> rdfs:subClassOf rell:representation , <iana:text> ;
   rdfs:label "The \"text/html\" Media Type" .
<iana:image/jpeg> rdfs:subClassOf rell:representation , <iana:image> ;
   rdfs:label "The \"image/jpeg\" Media Type" .
<iana:application/atom+xml> rdfs:subClassOf rell:representation ,
   <iana:application> ;
   rdfs:label "The \"application/atom+xml\" Media Type" .
<iana:application/xml> rdfs:subClassOf rell:representation ,
   <iana:application> ;
   rdfs:label "The \"application/xml\" Media Type" .
```

(b)

```
@prefix school: <http://rell.org/school/> .
@prefix iana: <http://www.iana.org/assignments/media-types/> .
school:person a rdfs:Class ;
   rdfs:subClassOf rell:resource ;
school:course a rdfs:Class ;
   rdfs:subClassOf rell:resource ;
school:course-page a rdfs:Class ;
   rdfs:subClassOf rell:resource ;
school:courselist a rdfs:Class ;
   rdfs:subClassOf rell:collection ;
school:person-html a rdfs:Class ;
   rdfs:subClassOf <iana:text/html> ;
   rell:represents school:person .
```

(c)

```
school:course-page-list a owl:ObjectProperty ;
   rdfs:domain school:course-page ;
   rdfs:range school:courselist ;
   rdfs:subPropertyOf school:page .

school:personcourse a owl:ObjectProperty ;
   rdfs:subPropertyOf rell:link ;

school:courses a owl:ObjectProperty ;
   rdfs:subPropertyOf rell:link ;
```

(d)

```
school:person-course a owl:ObjectProperty ;
   rdfs:domain school:person ;
   rdfs:range school:course ;
   rdfs:subPropertyOf school:personcourse .

school:course-page-course a owl:ObjectProperty ;
   rdfs:domain school:course-page ;
   rdfs:range school:course ;
   rdfs:subPropertyOf school:courses .
```

(e)

```
<http://www.ischool.berkeley.edu/people/faculty/erikwilde> a school:person ;
   school:person-website <http://dret.net/netdret/> ;
   school:person-course <http://www.ischool.berkeley.edu/programs/courses/242> .

<http://www.ischool.berkeley.edu/programs/courses/242> a school:course ;
   school:course-person <http://www.ischool.berkeley.edu/people/faculty/erikwilde> .
```

(f)

```
school:r391588838 a school:person-html ;
   rell:represents <http://www.ischool.berkeley.edu/people/faculty/erikwilde> .
```

(g)

Figure 5: N3 notation snippets of RDF triples generated for REST services

```
<http://www.ischool.berkeley.edu/people/faculty/erikwilde> a school:person ;
   vCard:FN "Erik Wilde" ;
   vCard:ADR _:node14m5kienpx1603 ;
   vCard:TITLE "Adjunct Professor" ;
   vCard:ORG _:node14m5kienpx1604 ;
   vCard:EMAIL _:node14m5kienpx1606 ;
   vCard:TEL _:node14m5kienpx1607 ;
   vCard:URL <http://dret.net/netdret/> ;
   vCard:PHOTO <http://www.ischool.berkeley.edu/files/imagecache/profile-pic/DSC_0176.JPG> ;
   school:person-website <http://dret.net/netdret/> ;
   school:person-course <http://www.ischool.berkeley.edu/programs/courses/242> ,
     <http://www.ischool.berkeley.edu/programs/courses/152> ,
     <http://www.ischool.berkeley.edu/programs/courses/190-waim> ,
     <http://www.ischool.berkeley.edu/programs/courses/290-wa> .
```

Figure 6: An individual's properties in N3 notation

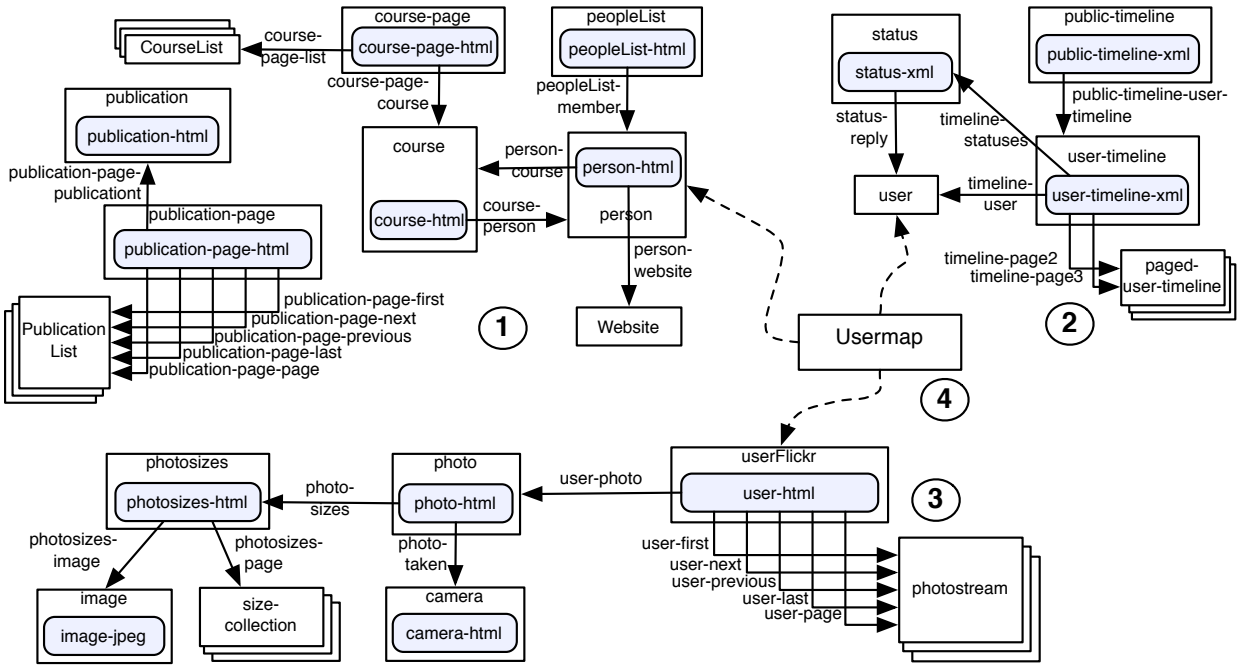

Figure 7: Composing four REST services, School, Twitter, Flickr and UserMap

```
<http://www.ischool.berkeley.edu/people/faculty/erikwilde> a school:person ;
   owl:sameAs <http://www.flickr.com/photos/dret/> , <http://twitter.com/dret> ,
   <http://twitter.com/users/show/dret.xml> ;
```

(a)

```
PREFIX dcterms:<http://purl.org/dc/terms/>
PREFIX flickr:<http://rell.org/flickr/>

SELECT DISTINCT ?person ?camera
WHERE
{?person owl:sameAs ?flickruser .
 ?picture dcterms:creator ?flickruser .
 ?picture flickr:photo-taken ?camera}
```

(b)

```
<school:people/faculty/erikwilde> <flickr:cameras/apple/iphone_3g/>
<school:people/faculty/erikwilde> <flickr:cameras/panasonic/dmc-tz5/>
<school:people/faculty/erikwilde> <flickr:cameras/panasonic/dmc-tz1/>
<school:people/faculty/erikwilde> <flickr:cameras/nikon/d80/>
```

(c)

Figure 8: N3 notation snippets for the composition scenario

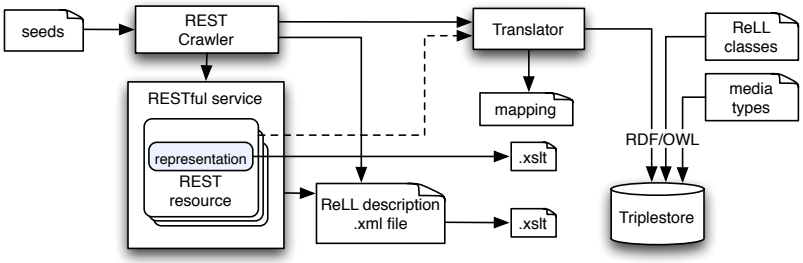

Figure 9: Implementation

Explore(ischool:people/faculty/erikwilde>)

Subject	Predicate	Object	Context
school:r391588838	rell:represents	ischool:people/faculty/erikwilde	
ischool:people/faculty/erikwilde	rdf:type	rdfs:Resource	
ischool:people/faculty/erikwilde	rdf:type	rell:resource	
ischool:people/faculty/erikwilde	rell:link	<http://dret.net/netdret/>	
ischool:people/faculty/erikwilde	rell:link	ischool:programs/courses/242	
ischool:people/faculty/erikwilde	school:website	<http://dret.net/netdret/>	
ischool:programs/courses/242	rell:link	ischool:people/faculty/erikwilde	
ischool:programs/courses/290-ssmels	rell:link	ischool:people/faculty/erikwilde	
ischool:people/faculty/erikwilde	owl:sameAs	<http://www.flickr.com/photos/dret/>	usermap:r689731148
ischool:people/faculty/erikwilde	owl:sameAs	<http://twitter.com/dret>	usermap:r689731148
ischool:programs/courses/290-ssmels	school:course-person	ischool:people/faculty/erikwilde	school:r196788832
ischool:programs/courses/242	school:course-person	ischool:people/faculty/erikwilde	school:r34379194
ischool:people/faculty/erikwilde	rdf:type	school:person	school:r391588838
ischool:people/faculty/erikwilde	school:person-course	ischool:programs/courses/242	school:r391588838
ischool:people/faculty/erikwilde	school:person-website	<http://dret.net/netdret/>	school:r391588838
ischool:people/faculty/erikwilde	vCard:ADR	_:node14m5kienpx1603	school:r391588838
ischool:people/faculty/erikwilde	vCard:EMAIL	_:node14m5kienpx1606	school:r391588838
ischool:people/faculty/erikwilde	vCard:FN	"ErikWilde"	school:r391588838
ischool:people/faculty/erikwilde	vCard:ORG	_:node14m5kienpx1604	school:r391588838
ischool:people/faculty/erikwilde	vCard:PHOTO	ischool:files/imagecache/profile-pic/DSC_0176.JPG	school:r391588838
ischool:people/faculty/erikwilde	vCard:TEL	_:node14m5kienpx1607	school:r391588838
ischool:people/faculty/erikwilde	vCard:TITLE	"AdjunctProfessor"	school:r391588838
ischool:people/faculty/erikwilde	vCard:URL	<http://dret.net/netdret/>	school:r391588838

Figure 10: Description for one User in the Composite Service