The Missing Links

How the Description Format RESTdesc Applies the Linked Data Vision to Connect Hypermedia APIs

Ruben Verborgh[1], Thomas Steiner[2], Rik Van de Walle[1], and Joaquim Gabarro[2]

[1] Ghent University – IBBT, ELIS – Multimedia Lab
Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium
{ruben.verborgh,rik.vandewalle}@ugent.be
[2] Universitat Politècnica de Catalunya – Department LSI
08034 Barcelona, Spain
{tsteiner,gabarro}@lsi.upc.edu

Abstract. "*Web 1.0 connected humans with machines. Web 2.0 connected humans with humans. Web 3.0 connects machines with machines.*"[1] On the one hand, an incredible amount of valuable data is described by billions of triples, machine-accessible and interconnected thanks to the promises of Linked Data. On the other hand, REST is a scalable, resource-oriented architectural style that, like the Linked Data vision, recognizes the importance of links between resources. Hypermedia APIs are resources, too—albeit dynamic ones—and unfortunately, neither Linked Data principles, nor the REST-implied self-descriptiveness of hypermedia APIs sufficiently describe them to allow for long-envisioned realizations like automatic service discovery and composition. We argue that describing inter-resource links—similarly to what the Linked Data movement has done for data—is the key to machine-driven consumption of APIs. In this paper, we explain how the description format RESTdesc captures the functionality of APIs by explaining the effect of dynamic interactions, effectively complementing the Linked Data vision.

1 The Linked Data Story is Only Half of the Story

Let us take a step back and think about the Web in its most abstract form. What we see are *resources*, with an unparalleled variety, and an ever increasing number of *links* between them [7]. Resources and their representations make up the essence of the Web [11], while the Linked Data vision made us all realize again the crucial role links play therein. Indeed, links have been the catalysts of the success of the human Web, and they continue to prove their strengths on the Semantic Web [5]. The representations of resources—and therefore data—are given meaning by links, corresponding to well-defined RDF predicates.

[1] As mentioned on Twitter by Inge Henriksen: https://twitter.com/ihenriksen/status/174510781352783872

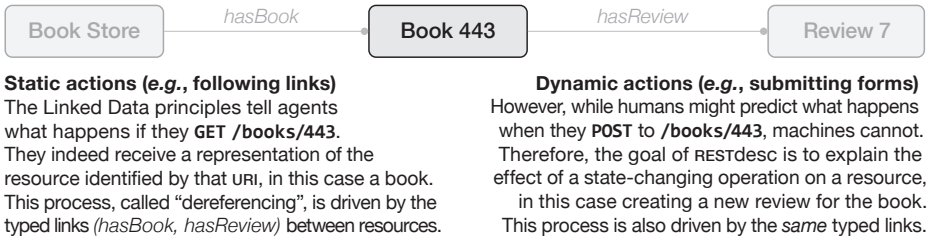

Fig. 1. RESTdesc complements Linked Data by explaining a hyperlink's *dynamic* functionality in machine-readable form. For instance, we can express with RESTdesc what happens when agents use `POST` on a linked resource instead of `GET`.

If links are so important, why don't we see them on the service-side of the Web yet? When Fielding redesigned the HTTP [10] specification, he had a resource-oriented model in mind where hypermedia drives Web applications: Representational State Transfer (REST, [11]). As he later clarified, the hypermedia constraint imposed by REST tells us that representations of a resource should contain the controls (*e.g.,* links and forms) to possible next steps or resources [9]. Consequently, modeling Web services or APIs the REST way leads to the same resources-and-links paradigm at the core of the human *and* the Semantic Web.

In all fairness, REST APIs—as defined by Fielding—are scarce. Hypermedia-driven APIs are vastly outnumbered by plain HTTP and RPC interfaces. However, this can in fact be compared to unstructured and unlinked data being currently more present on the Web than Linked Data. Therefore, the scarceness doesn't change the status of resource- and link-orientedness as well-suited model for automated agents to perform static *and* dynamic interactions on the Web.

Currently missing is a way for agents to know what effect a *state-changing* operation will have. Linked Data gives the answer for *information-retrieving* operations, known as *dereferencing*. Performing a `GET` operation on a resource's URI will provide the agent with information about that resource. But what happens when the agent performs a `POST` operation on the *same* resource? Since Fielding suggests the controls (*e.g.,* links and forms) should point to possible next steps or resources, it is obvious *how* the state change happens. However, *what* this state change will bring might be obvious to humans, but is still unknown to machines. Therefore, in this paper, we zoom in on how the description format RESTdesc [21] explains to agents what will happen if *state-changing* operations are performed on a resource, complementary to the Linked Data principles that explain the same for static operations. This complementary nature is illustrated in Fig. 1, which positions Linked Data and RESTdesc.

This paper starts by describing the differences and similarities of Linked Data and hypermedia APIs in the next section, zooming in on the gaps that need to be bridged. Section 3 illustrates the role RESTdesc can play herein by formally expressing the relationship between resources in a hypermedia API. We compare RESTdesc with other approaches in Section 4. Finally, Section 5 looks back on the discussed topics and ends by indicating the importance of hypermedia-driven APIs on the Web for autonomous agents.

2 A Joint Future for Linked Data and Hypermedia APIs

We start this section with an essential definition to avoid misunderstandings on the thin ice of REST, RESTlike, and unRESTful APIs:

Hypermedia Web APIs are interfaces to manipulate *resources* according to the HTTP method semantics, serving *representations* of these resources along with the *controls* to advance through the interface [9].[2]

Striking parallels between Linked Data and hypermedia APIs exist—and this is not a coincidence, since both are closely tied to the original visions and architecture of the Web. One of the common elements are **resources**: concepts in Linked Data are identified by one or multiple URIs, which, when requested through HTTP GET, lead to information about that concept. Hypermedia APIs are similarly structured as concepts or resources, with the constraints that every URI should identify a resource and that the HTTP methods should be used conform to the HTTP specification [10]. The semantics of the GET method have therein been defined as "obtaining the information identified by the URI", which, unsurprisingly, matches the Linked Data purpose.

The other common element are **links**: as the name implies, they play a vital role in Linked Data, and they are at the heart of the Semantic Web. Links are what gives a concept's data meaning beyond its own context. More concretely, if an agent does not understand what a data property means, it can look up that property because its link is an HTTP URI. The same applies to hypermedia APIs: the controls, telling us how other resources relate to the current resource, can be links. Details on the nature of the relation are conveyed by link types, which can have the same URIs as Linked Data properties [17].

In essence, one could see the whole Linking Open Data Cloud [6] as a large, distributed hypermedia application. This is in fact how its usage is encouraged: an agent starts from one resource and can make its way through the whole cloud, just by "following its nose", thanks to the links. However, it only provides a subset of the possibilities of what we expect from a hypermedia API: merely *retrieval* operations are supported. Yet, the role of links here remains important: browsing billions of triples in billions of resources would otherwise prove difficult.

An interesting aspect of REST is that it does not matter whether the resources and triples already exist. They can either be part of documents, or be the result of a service invocation—but the agent does not have to know and does not have to care. For example, a huge dataset of natural numbers has been made available as Linked Data [23], yet the information of each number is not static, but instead generated dynamically when an agent dereferences its URI. This dataset is thus what we would traditionally consider a "service", but thanks to the REST principles, it manifests itself as just another set of linked resources.

[2] Hypermedia APIs are synonymous to "REST APIs or services, *in the sense as defined by Fielding*" [11]. This last clarification is important, since many APIs that were given a "REST" label do *not*, or only partially, adhere to Fielding's definition, which is why we use the term "hypermedia API" to distinguish the *intended* meaning [13].

Nevertheless, we often associate the concept of services additionally with action-driven behavior, for example, allowing us to post a comment or order tickets. In a REST architectural model, these actions are captured by the modification or creation of resources, linked to existing resources. While these and similar actions are very common on the human Web and on the Web of services, the Semantic Web still struggles with state-changing operations [4]. Several mechanisms are there (*e.g.,* SPARQL `UPDATE` [12]), but issues such as authorization and security still impede wide adoption. Consequently, the Linked Data vision must in the meantime assume that the publisher and consumer sides are distinct, *i.e.*, that consumers of Linked Data will not need to perform write operations. This simplifying assumption has its benefits—just look at the overwhelming amount of data—but will not be sufficient for the vision of autonomous agents that require actions in the real world. Indeed, as the comment and ticket examples indicate, many interactions we perform daily involve write actions. Therefore, in the next section, we will look at the requirements of agents for browsing full hypermedia APIs, which offer both information-retrieving and state-changing operations.

3 RESTdesc Describes Hypermedia Links

As an example, let us consider the situation of Fig. 1. Starting from the book store's main URI, an agent discovers resources in a fully hypermedia-driven way. Its steps might be the following:

1. **`GET`** a representation of the index resource at `/`.
2. **Find** a `hasBook` link in this representation titled "*The Catcher in the Rye*".
3. **`GET`** a representation of this linked resource at `/books/443`.
4. **Find** a `hasReview` link in this representation.
5. **`GET`** a representation of this linked resource at `/books/443/reviews/7`.

This way of working is hypermedia-driven, because the agent only follows the representation-supplied controls (*e.g.*, links) to go from one step to the next.

This approach works perfectly so far, because the agent knows in each step what the result will look like, thanks to the Linked Data principles. Even if it should not, it can execute the `GET` request without any harm, since the HTTP specification states the `GET` method is *safe* [10]. What the agent cannot do, however, is carelessly issue a `POST` request in one of the steps, since *a*) it cannot predict what the result will be and *b*) testing what the result is can have unwanted consequences, as `POST` is *un*safe. Furthermore, it cannot determine what body it should send along with the `POST` request. Although some representation formats provide forms (*e.g.*, HTML and Atom), others lack form functionality (*e.g.*, RDF), but in either case, it is unclear how the result relates to the submitted data.

The RESTdesc description format [21] explains in a machine-processable way what functionality is offered by a certain hypermedia link. This enables agents to understand what data they can send along with a `POST` request and how this data will influence the outcome of the request. Listing 1 displays a description of the `hasBook` link type and serves as an illustration of several common aspects of RESTdesc descriptions.

```
@prefix ex: <http://example.org/book-store#>.
@prefix http: <http://www.w3.org/2011/http#>.

{
  ?store ex:hasBook ?book. 1
  ?review ex:author _:author;
          ex:rating _:rating;
          ex:contents _:text.
}
=>
{
  _:request http:methodName "POST"; 2
            http:requestURI ?book;
            http:body ?review;
            http:resp [ http:body ?book ].

  ?book ex:hasReview ?review. 3
}.
```

Listing 1. RESTdesc describes the act of posting a review by explaining the associated hypermedia link.

RESTdesc descriptions are expressed in Notation3 (N3, [2]), a small superset of RDF put forward by Tim Berners-Lee. N3 adds support for quantification, necessary to create statements concerning *all* resources instead of only specific ones. The description in Listing 1 consists of three parts and can be read as:

1 **IF** you obtain a book from a `hasBook` hyperlink
2 **THEN** you can make a `POST` request to that book's URI
3 to add a review with the supplied parameters to that book.

The logical foundations of N3 (N3Logic, [3]) define an operational semantics, *i.e.*, RESTdesc descriptions are N3 rules that can be instantiated and executed by a reasoner. Concretely, if the agent has been given the contents of a review *(author, rating, content)*, it can follow these hypermedia-driven steps:

1. **`GET`** the RESTdesc description of `hasBook`.[3]
2. **`GET`** a representation of the index resource at `/`.
3. **Find** a `hasBook` link in this representation titled "*The Catcher in the Rye*".
4. **Instantiate** the description with the review and found link.
5. **`POST`** the review, as instructed by the description, at `/books/443`.

Note again how only hypermedia controls are used to get from one step to the next. The added value of RESTdesc here is to explain the agent in advance what effect the `POST` request will have, so it can decide whether to execute this request. In real-world applications, RESTdesc descriptions can be used for goal-driven API compositions [22]. For instance, the user can supply the review parameters as input, and ask that it is submitted to a certain book.

[3] RESTdesc discovery, *i.e.*, how to `GET` RESTdesc descriptions, has been discussed earlier [21]. The agent could for example dereference the `hasBook` link.

4 Related Work

Description of Web services or APIs for automated use has been on the Web since before the advent of the Semantic Web (notably WSDL [8]), and played an important part during the beginning of the Semantic Web's inception. Several of the first initiatives are well-known: OWL-S [16], which evolved from DAML-S [1], and the conceptually different WSMO [15,19]. These formats target what are called "Big" Web services [18], which function in a message-passing or Remote Procedure Call (RPC) paradigm. While these models use Semantic Web elements such as ontologies, they predate the Linked Data vision and the recent revaluation of REST APIs. Neither OWL-S nor WSMO have stood the test of time, as extensive Web searches did not reveal substantial real-world usage. We therefore focus on more recent research projects that have design goals similar to RESTdesc, *e.g.*, a focus on functionality and/or hypermedia APIs.

Linked Open Services (LOS, [14]) have an HTTP API approach, in which SPARQL graph patterns identify the offered functionality. Part of the project's scope concerns the lifting and lowering of existing services, since many of them do not expose their data in a semantic format yet. A difference with RESTdesc is that LOS APIs are not committed to the hypermedia constraint, whereas the hypermedia-driven consumption of APIs is a central concept in RESTdesc.

Linked Data Services (LIDS, [20]) have a similar notion of input and output graphs. They use the input data to construct a resource's URI, as opposed to LOS, which sends input data in the request body. The result is an API whose interactions are thus in a sense solely *form*-based—the form structure being defined by the unbound variables in the input graph pattern. In addition to forms (not discussed in this paper), RESTdesc also aims to support the *link* part of the hypermedia control set.

5 Conclusion

The Linked Data vision strives to connect data on the Web, making it available in a machine-processable format. Hypermedia APIs similarly strive for connectedness of resources, but also consider the write side of interactions. Their goals are similar, and so are their tools: both make automated consumption of the Web available using the core principles of the HTTP architecture, featuring resources, representations, and links. However, dealing with state-changing operations requires automated agents to have expectations of what consequences their actions will have.

RESTdesc shows how existing Semantic Web technologies can be combined to explain the functionality of a Web API to those agents. It enables us to apply the Linked Data vision to hypermedia APIs by describing the meaning of links for state-changing operations. In that sense, it is a plea for more hypermedia APIs on the Web, as they beautifully incorporate the controls that future autonomous agents will need to browse the Web. Therefore, we believe it is time to transition today's services towards hypermedia APIs by adding the missing links.

Acknowledgments The described activities were funded by Ghent University, the Interdisciplinary Institute for Broadband Technology (IBBT), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research Flanders (FWO), and the European Union.

This work was partially supported by the European Commission under Grant No. 248296 FP7 (I-SEARCH project). Joaquim Gabarró is partially supported by TIN-2007-66523 (FORMALISM), and SGR 2009-2015 (ALCOM).

References

1. A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. DAML-S: Web service description for the Semantic Web. 2342:348–363, 2002.
2. T. Berners-Lee and D. Connolly. Notation3 (N3): A readable RDF syntax. W3C Team Submission, 2011. Available at `http://www.w3.org/TeamSubmission/n3/`.
3. T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf, and J. Hendler. N3Logic: A logical framework for the World Wide Web. *Theory and Practice of Logic Programming*, 8(3):249–269, 2008.
4. T. Berners-Lee, R. Cyganiak, M. Hausenblas, J. Presbrey, O. Seneviratne, and O. Ureche. Realising a read-write Web of Data, June 2009. Available at `http://web.mit.edu/presbrey/Public/rw-wod.pdf`.
5. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.
6. C. Bizer. The emerging Web of Linked Data. *Intelligent Systems, IEEE*, 24(5):87–92, Sept. 2009.
7. C. Bizer, T. Heath, and T. Berners-Lee. Linked Data – the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
8. F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. *Internet Computing,* IEEE, 6(2):86–93, Mar. 2002.
9. R. T. Fielding. REST APIs must be hypertext-driven. Untangled – Musings of Roy T. Fielding, Oct. 2008. Available at `http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven`.
10. R. T. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Request for Comments: 2616, June 1999. Available at `http://tools.ietf.org/html/rfc2616`.
11. R. T. Fielding and R. N. Taylor. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, May 2002.
12. P. Gearon, A. Passant, and A. Polleres. SPARQL 1.1 Update. W3C Working Draft, Jan. 2012. Available at `http://www.w3.org/TR/sparql11-update/`.
13. S. Klabnik. REST is over, Feb. 2012. Available at `http://blog.steveklabnik.com/posts/2012-02-23-rest-is-over`.
14. R. Krummenacher, B. Norton, and A. Marte. Towards Linked Open Services and Processes. In A. Berre, A. Gómez-Pérez, K. Tutschku, and D. Fensel, editors, *Future Internet – FIS 2010*, volume 6369 of *Lecture Notes in Computer Science*, pages 68–77. Springer Berlin / Heidelberg, 2010.
15. R. Lara, D. Roman, A. Polleres, and D. Fensel. A conceptual comparison of WSMO and OWL-S. In L.-J. Zhang and M. Jeckle, editors, *Web Services*, volume 3250 of *Lecture Notes in Computer Science*, pages 254–269. Springer Berlin, 2004.

16. D. Martin, M. Burstein, D. Mcdermott, S. Mcilraith, M. Paolucci, K. Sycara, D. L. Mcguinness, E. Sirin, and N. Srinivasan. Bringing semantics to Web services with OWL-S. *World Wide Web*, 10:243–277, Sept. 2007.
17. M. Nottingham. Web Linking. Request for Comments: 5988, Oct. 2010. Available at `http://tools.ietf.org/html/rfc5988`.
18. C. Pautasso, O. Zimmermann, and F. Leymann. RESTful Web services vs. "Big" Web services: making the right architectural decision. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 805–814, New York, NY, USA, 2008. ACM.
19. D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology. *Applied Ontology*, 1:77–106, January 2005.
20. S. Speiser and A. Harth. Taking the LIDS off data silos. In *Proceedings of the 6th International Conference on Semantic Systems*, I-SEMANTICS '10, pages 44:1–44:4, New York, NY, USA, 2010. ACM.
21. R. Verborgh, T. Steiner, D. Van Deursen, J. De Roo, R. Van de Walle, and J. Gabarro. Capturing the functionality of Web services with functional descriptions. *Multimedia Tools and Applications*, 2012. Available at `http://rd.springer.com/article/10.1007/s11042-012-1004-5`.
22. R. Verborgh, D. Van Deursen, E. Mannens, C. Poppe, and R. Van de Walle. Enabling context-aware multimedia annotation by a novel generic semantic problem-solving platform. *Multimedia Tools and Applications*, 2012. Available at `http://rd.springer.com/article/10.1007/s11042-010-0709-6`.
23. D. Vrandečić, M. Krötzsch, S. Rudolph, and U. Lösch. Leveraging non-lexical knowledge for the Linked Open Data Web. *5th Review of April Fool's day Transactions*, pages 18–27, 2010.