Future Challenges for Linked APIs

Steffen Stadtmüller, Sebastian Speiser, and Andreas Harth

Karlsruhe Institute of Technology, Institute of Applied Informatics and Formal Description Methods (AIFB)
firstname.lastname@kit.edu,

Abstract. A number of approaches combine the principles and technologies of Linked Data and RESTful services. Services and APIs are thus enriched by, and contribute to, the Web of Data. These resource-centric approaches, referred to as Linked APIs, focus on flexibility and the integration capabilities of Linked Data. We use our experience in teaching students on how to use Linked APIs to identify the existing challenges in the area. Additionally we introduce the *LAPIS catalogue*, a directory for Linked APIs as basis for the research to address the identified challenges.

1 Introduction

There is a strong movement in the Web toward a resourceful model of services [8] based on a constrained set of uniform operations. A major role plays Representational State Transfer (REST [3]) a software architecture style for distributed systems. In the case of HTTP clients apply operations to resources addressed via individual URIs. Flexibility, adaptivity and robustness are the major objectives of REST and are particularly useful for software architectures in distributed data-driven environments such as the web [7]. However, data sources and APIs are published according to different interaction models and with interfaces using non-aligned vocabularies, which makes writing programs that integrate offers from multiple providers a tedious task.

Linked Data unifies a standardised interaction model with the possibility to align vocabularies using RDF, RDFS and OWL. However, the interactions are currently constrained to simple data retrieval. The combination of Linked Data with REST architectures, allows to combine the advantages of both paradigms when offering functionality on the web: the easy data integration offered by Linked Data together with the flexibility of REST enables lightweight and adaptive services, called *Linked APIs*.

However, to realise the benefits from Linked Data-based APIs there are still a number of challenges to address. In this position paper we identify key topics as basis for discussion. Based on our experience with students that work with Linked APIs we describe existing hurdles for the adoption of Linked APIs (Section 3) and thus sketch the relevant research topics for the domain that have to be addressed by the community to achieve a pervasive web of Linked APIs.

To support the ongoing efforts to solve the existing challenges in the area of Linked APIs we introduce *LAPIS catalogue*, an open directory for Linked APIs (Section 4).

2 Motivation

According to the Richardson maturity model [8], REST is identified as the interaction between a client and a server based on three principles:

- The use of URI-identified resources.
- The use of a constrained set of operations, i.e., the HTTP methods, to access and manipulate resource states.
- The application of hypermedia controls, i.e., the data representing a resource contains links to other resources. Links allow a client to navigate from one resource to another during his interaction.

The idea behind REST is that applications, i.e., clients, using functionalities provided on the web, i.e., APIs, are not based on the call of API-specific operations or procedures but rather on the direct manipulation of exposed resource representations or the creation of new resource representations. A resource can be a real world object or a data object on the web. The representation of a resource details the current state of the resource. A manipulation of the state representation can imply that the represented resource is manipulated accordingly.

The flexibility of REST results from the idea that client applications do not have to know about all necessary resources. The retrievable representation of some known resources contains links to other resources that the client can discover during runtime. Clients can use such discovered resources to perform further interaction steps.

The Linked Data design principles[1] also address the use of URI-identified resources and their interlinkage. However, Linked Data is so far only concerned with the provisioning and retrieval of data. A combination of Linked Data with REST architectures is therefore an intuitive extension, to offer flexible functionality with easy data integration capabilities. In contrast to REST, Linked Data does distinguish explicitly between URI-identified objects (i.e., non-information resources) and their data representation (information resources). An extension of Linked Data with REST to allow for resource manipulation leads to read/write Linked Data, i.e., information resources can be accessed and manipulated. REST furthermore implies that a change of an information resource can result in a change of the corresponding non-information resource.

The extension of Linked Data with REST technologies has been explored [1, 13] and led recently to the establishment of the *Linked Data Platform*[2] W3C working group. Additionally many approaches [4, 5, 9, 6, 10–12] follow the motivation of REST and look beyond the exposure of fixed datasets using HTTP, SPARQL and RDF. These approaches are referred to as *Linked APIs* and extend the traditional use of HTTP in Linked Data, consistently with REST, by allowing all HTTP operations to be applied to Linked Data resources. Linked APIs view services primarily as RDF prosumers, i.e., the state of resources should be made available encoded in RDF, at least as an alternative via Content Negotiation. By interlinking output data with existing Linked Datasets, and other dynamic data, Linked APIs also contribute to the Web of Data.

[1] `http://www.w3.org/DesignIssues/LinkedData.html`
[2] `http://www.w3.org/2012/ldp/charter`

Linked APIs consider how the data that results from computation over input can carry explicit semantics. Many approaches base their service descriptions on the notion that Linked Data already provides a description for services' input and output requirements: the graph patterns provided by the SPARQL query language or the N3 notation [5, 9, 10, 12].

Increased value comes from combining data from multiple sources and functionality from multiple providers. Such combinations are fostered by the use of Linked Data, which enables an easy integration of the communicated data by the clients that use several Linked APIs.

3 Challenges for Linked APIs

We organised three seminars at the Institute AIFB where undergraduate students had to create applications based on Linked APIs. In total, we had 20 students that worked in groups of sizes between 2 and 4. The participating students had programming experience, but were relatively new to the field of Web APIs. Therefore the students are well suited to represent developers that consider using Web functionality for the first time. We collected the experiences made by the students to identify existing entry barriers of Linked APIs and derive key challenges that should be addressed by the research community to foster the adoption of the existing approaches.

Each student was tasked over the course of 4 months to develop at least one Linked Data-based REST API and to develop an application that makes use of at least two such APIs by integrating their data and composing their functionality. The students were allowed to leverage existing (not Linked Data-based) REST APIs for the development of a Linked API by wrapping them and offering their functionality as Linked API with the possibility for RDF data input and output. In the remainder of this paper we refer to the used pre-existing non-Linked Data APIs as *underlying APIs*.

At the end of the 4 months the students wrote a report about their experiences and lessons learned when working with Linked APIs. Since the students were not given a list of predefined answers, we clustered the problems (Table 1) as well as the perceived benefits when working with Linked APIs the students reported. Even though the collected experiences of the students can not be seen as a representative survey, they provide empirical indicators to existing problems in the field of Linked APIs, which imply important challenges that should be addressed by the research community. Many participants mentioned similar problems in their reports, which supports the claim that we identified recurring issues.

Table 1. Problems when working with Linked APIs as identified by 20 students

Problem	# Students
Response time of composed API	14
API limitations	14
Missing directories	12
No standard formalism for API descriptions	8
Complexity of RDF	3

One of the two most common reported problems was with regard to the *response time of composed APIs* (reported by 14 students). The slow response time of composed APIs could be on the one hand attributed to slow response times of underlying APIs that were used by the students. On the other hand the interaction between APIs was also a significant factor: A single call to a composed API could potentially result in hundreds of subsequent calls to the different Linked APIs, that had to be sequentially executed, thus causing significant delays before the initial call could be answered. We see one of the most important challenges for the research in the area of Web APIs in the development of methods to achieve a scalable interaction with APIs. This includes the definition of interaction models on which applications can build upon as well as systems to enable the combination and composition of offered functionalities. The quick identification of APIs that have to be called next within a defined interaction pattern, as well as the ability to identify sets of API calls that can be executed in parallel are crucial factors to address this challenge.

API limitations refer to reported problems originating from limitations, insufficient functionality and constraints of the underlying APIs (e.g., a maximum number of API calls per day or incomplete data). While these limitations are usually driven by external circumstances (e.g., business aspects), it is noteworthy, that 10 out of the 14 participants that reported to have such problems considered to use different APIs than originally intended because of the limitations. However, these participants reported problems identifying suitable APIs as a replacement. APIs often need to fulfill very specific (functional and non-functional) needs to allow developers to fulfill their goal when implementing web-based applications. Since the functionality of many web APIs is often only described with natural language texts, the task of finding a suited API for a specific problem is often carried out by trial and error. However, a trial and error approach does not allow to consider hundreds of potential candidates for a Web API. In addition to application developers, the identification of APIs with specific functionalities is also important for API providers. Since REST demands an interlinkage of API resources (potentially from different providers) and the usefulness of an API is increasing with these links, the task of finding related APIs becomes crucial for API providers to include their API in an ecosystem of APIs on the web. Many approaches to Linked APIs propose to leverage graph patterns to provide machine-readable descriptions of functionality and properties to mitigate the problem [5, 9, 12, 10]. However, finding suitable methods to allow an automated identification and comparison of functionalities offered by Linked APIs remains to be an important challenge.

Even though many of the participants used the possibility to describe Linked APIs with graph patterns, 8 participants had difficulties because they could not identify a definitive formalism for the descriptions (*No standard formalism for API descriptions*). Indeed, the research community should make an effort to define a common standard way of describing APIs, especially since most existing description mechanisms are already based on the same notion of utilising graph patterns. Such a standard should include a vocabulary for the description, a clear definition of the minimal set of properties that are to be described (e.g., input and output data), and a standard way of attaching a description to an API resource (e.g., retrievable via the HTTP OPTIONS method). The common description would not have to cover all possible aspects that can be described,

but should be flexible enough to allow for extensions to cover the specific needs of the individual Linked API approaches. Merely the aspects that are necessary to invoke the operations on the API resources have to be described in a unified way to foster the adoption of Linked APIs by developers.

Related to the problem of finding replacements for APIs, the participants also reported they were often not able to identify suitable information on where to find existing Linked APIs (reported by 12 participants). We classified these problems as *missing directories*. While this is not a research problem per se, we believe it is a fundamental issue to be addressed not only to enable research on the topic, but also to foster the adoption of Linked APIs by application developers (cf. Section 4).

Finally, three students reported that the complexity of RDF was too high to be useful. However these three participants reported similar problems with other data formats like XML or JSON. Therefore there problems can be attributed to lacking development experience especially with web technologies.

Summarising we identified the following challenges that have to be addressed:

- A common standard minimal description mechanism for Linked APIs based on graph patterns.
- Methods for an automated identification and comparison of functional and non-functional properties that leverage the descriptions.
- The development of methods and systems to enable a scalable interaction and composition of Linked APIs.

While these challenges remind of the high level tasks that have been addressed in traditional web service research [2] (service descriptions, discovery and composition), Linked APIs require different approaches due to their focus on a resource-centric interaction and the interlinkage of these resources. The resource-driven and interlinked design results in benefits for the implementation and use of APIs as the majority of the participants confirmed (Table 2) and can ultimately lead to a high adoption of Linked APIs.

Table 2. Perceived benefits when working with Linked APIs as identified by 20 students

Benefit	# Students
Easy data integration	17
High modularity of composed APIs	14
Simplicity of Interaction with API	9

4 LAPIS catalogue

One of the outcomes of our exploration of existing challenges when working with Linked APIs was that it is difficult to find the existing Linked APIs. To address this issue we release the *LAPIS catalogue*[3] as open directory, where providers can register

[3] `http://km.aifb.kit.edu/services/ckan/`

their Linked APIs. The *LAPIS catalogue* is based on CKAN[4], an open-source data portal software, developed by the Open Knowledge Foundation [5]. The objective of CKAN is to make it easy to publish, share and find data. It provides a powerful database for cataloguing datasets, with an intuitive web front-end and API.

Linked APIs can be registered in the *LAPIS catalogue* along with basic information like name, URI, author, license and maintainer. Additionally information can be provided about the supported content types, other APIs that are linked and underlying APIs, if the registered API is based on a wrapper. Due to the focus of CKAN on datasets, the registered APIs can be seen as Read/Write Linked Datasets (i.e., dynamic datasets that support more than one HTTP method for manipulation of the resources).

The *LAPIS catalogue* serves as hub for Linked APIs, where researchers, developers and API providers can find existing APIs to use in their individual tasks. Researchers in the field of Linked APIs can find relevant APIs for evaluation of their individual approaches (e.g., for interaction with APIs) or use the directory to survey the current developments and the adoption of their approaches (e.g., description mechanisms). Developers of applications based on Linked APIs can search and lookup APIs that can be employed in their applications. API Providers can use the *LAPIS catalogue* to promote their API for public use. However, also the directory of other Linked APIs can be useful for API providers: Since REST demands the interlinkage of resources, API providers require information about existing APIs to link their APIs to other resources and thus complementing the offered functionality.

Related popular directories are *The Data Hub*[6] and *Programmable Web*[7].

The Data Hub is a general directory for datasets that also provides information about links between datasets. While Read/Write Linked Datasets can be entered in The Data Hub, the much broader focus prevents The Data Hub to serve all the needs of the Linked API community. Specifically the tedious task of identifying the subset of registered datasets that support resource manipulation consistently with REST renders The Data Hub insufficient as an API repository. The related Linking Open Data cloud[8] even explicitly excludes datasets that are served by a service that produces RDF in response to input data, which is an inherent property of Linked APIs.

Programmable Web is a directory for Web APIs and is therefore more focused on the functionality aspect compared to The Data Hub. Similar to The Data Hub, Linked APIs can be registered with Programmable Web along with other kinds of APIs and web services (e.g., SOAP-based services). However, the lacking focus on resource-centric interlinked APIs results in missing relevant information for developers and providers that work with Linked APIs. Specifically lacking information about links between the registered APIs is a significant problem in that respect.

The *LAPIS catalogue* is not a competitor of The Data Hub or Programmable Web, but a complementing directory between them. Therefore it is specifically targeted at

– Read/WriteLinked Datasets (from the viewpoint of The Data Hub) or

[4] http://ckan.org/
[5] http://okfn.org/
[6] http://datahub.io/
[7] http://www.programmableweb.com/
[8] http://lod-cloud.net/

– Linked Data-based REST APIs (from the viewpoint of Programmable Web),

providing the relevant information from both areas. We will continue to maintain the *LAPIS catalogue* to foster the establishment of a pervasive web of Linked APIs.

5 Conclusion

In this position paper we sketched the main challenges for Linked APIs that need to be addressed to foster the adoption of Linked Data based REST APIs. We described the main problems in the work with Linked APIs from our experience when introducing students in the field and identified key topics as basis for discussion and further research. We believe, that the benefits that result from the combination of Linked Data and REST architectures are worth the effort to overcome the existing problems.

To foster the adoption of Linked APIs and research activities we introduced the *LAPIS catalogue*, an open directory for Linked APIs. A directory targeted at Linked APIs is an essential support mechanism for the uptake of Linked APIs and a fundamental requirement to intensive the research activities in the area.

Acknowledgments

This work was partially supported by the PlanetData NoE (FP7:ICT-2009.3.4, #257641) and by the German Ministry of Education and Research (BMBF) within the SoftwareCampus project framework.

References

1. Berners-Lee, T.: Read-Write Linked Data (August 2009), avaiable at `http://www.w3.org/DesignIssues/ReadWriteLinkedData.html`, accessed 26th November 2012
2. Bussler, C., Davies, J., Fensel, D., Studer, R. (eds.): The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004, Heraklion, Crete, Greece, May 10-12, 2004, Proceedings, Lecture Notes in Computer Science, vol. 3053. Springer (2004)
3. Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis, University of California, Irvine (2000)
4. Kopecky, J., Pedrinaci, C., Duke, A.: Restful write-oriented api for hyperdata in custom rdf knowledge bases. In: Next Generation Web Services Practices (NWeSP), 2011 7th International Conference on. pp. 199–204 (2011)
5. Krummenacher, R., Norton, B., Marte, A.: Towards Linked Open Services. In: 3rd Future Internet Symposium (September 2010)
6. Panziera, L., Comerio, M., Palmonari, M., Paoli, F.D., Batini, C.: Quality-driven extraction, fusion and matchmaking of semantic web api descriptions. J. Web Eng. 11(3), 247–268 (2012)
7. Pautasso, C., Wilde, E.: Why is the web loosely coupled?: a multi-faceted metric for service design. In: Conference on World Wide Web (WWW). pp. 911–920 (2009)
8. Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly Media (May 2007)

9. Speiser, S., Harth, A.: Integrating linked data and services with linked data services. In: Proceedings of 8th Extended Semantic Web Conference (ESWC) (2011)
10. Stadtmüller, S., Speiser, S., Harth, A., Studer, R.: Data-fu: A language and an interpreter for interaction with read/write linked data. In: Proceedings of the 22th international conference on World Wide Web. WWW '13 (2013)
11. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: Rapidly integrating services into the linked data cloud. In: Proceedings of the 11th International Semantic Web Conference (ISWC) (2012)
12. Verborgh, R., Steiner, T., Deursen, D.V., de Walle, R.V., Valls, J.G.: Efficient Runtime Service Discovery and Consumption with Hyperlinked RESTdesc. In: The 7th International Conference on Next Generation Web Services Practices (2011)
13. Wilde, E.: REST and RDF granularity (2009), available at `http://dret.typepad.com/dretblog/2009/05/rest-and-rdf-granularity.html`