Web Semantics: Science, Services and Agents on the World Wide Web 29 (2014) 12–18

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

API-centric Linked Data integration: The Open PHACTS Discovery Platform case study

Paul Groth[a,*], Antonis Loizou[a], Alasdair J.G. Gray[d], Carole Goble[b], Lee Harland[c], Steve Pettifer[b]

[a] *Department of Computer Science, VU University Amsterdam, Amsterdam, Netherlands*
[b] *School of Computer Science, University of Manchester, Manchester, UK*
[c] *Connected Discovery, UK*
[d] *Department of Computer Science, Heriot-Watt University, Edinburgh, UK*

ARTICLE INFO

Article history:
Received 31 August 2013
Received in revised form
4 March 2014
Accepted 24 March 2014
Available online 8 April 2014

Keywords:
Linked Data
Data integration
Pharmacology
Data spaces
API

ABSTRACT

Data integration is a key challenge faced in pharmacology where there are numerous heterogeneous databases spanning multiple domains (e.g. chemistry and biology). To address this challenge, the Open PHACTS consortium has developed the Open PHACTS Discovery Platform that leverages Linked Data to provide integrated access to pharmacology databases. Between its launch in April 2013 and March 2014, the platform has been accessed over 13.5 million times and has multiple applications that integrate with it. In this work, we discuss how Application Programming Interfaces can extend the classical Linked Data Application Architecture to facilitate data integration. Additionally, we show how the Open PHACTS Discovery Platform implements this extended architecture.

1. Introduction

A central challenge in modern information systems is the need to interrogate information combined from multiple sources [1]. This is particularly evident in the life sciences where the need to cross boundaries of data is central to the science itself [2]. For example, in drug discovery (i.e. pharmacology), one central task is to connect information about chemistry to biological information such that researchers can determine the potential impact of a chemical on a biological system.

A key difficulty in data integration for the life sciences is the number of databases available and the variability of their schemata. Even when integrating across a small number of schemata, these may vary over time. Essentially, as the science itself changes so does the data model of the underlying datasets. For example, in pharmacology the notion of a target (i.e. what biological system is impacted by a cell) was largely defined as a particular protein but now drugs are being developed that impact collections of proteins — this changes how one models the notion of a target in a chemical experiments database. To address environments where there are a large number of heterogeneous datasets with varying schemata, Halevy introduced the notion of dataspaces [3]. Instead of designing a global schema and integrating data up-front, data is integrated in a pay-as-you-go fashion where mappings are driven by the needs of the user. Bizer has argued that Linked Data represents an example of a dataspace at a global scale [4]. Indeed, Linked Data has several benefits for creating dataspace style systems:

1. the use of RDF removes issues of syntactic data integration[1];
2. by leveraging a global naming system (e.g. URLs) it allows for the referencing of identities across datasets — making the expression of mappings easier;
3. schemata (e.g. RDFs and OWL) and data are accessible using the same protocols and query languages (e.g. SPARQL);

These characteristics allow effort to be distributed across parties. It is no longer the consumer or producer of data that necessarily needs to be responsible for data integration, third parties can contribute as well [4]. However, a question still arises for any of these contributors, how much price should one pay and when? Translated: given a large number of heterogeneous datasets

* Corresponding author. Tel.: +31 20 598 7742; fax: +31 20 598 7728.
E-mail addresses: pgroth@gmail.com, p.t.groth@vu.nl (P. Groth).

[1] One can dump RDF into a triple store and begin querying.

http://dx.doi.org/10.1016/j.websem.2014.03.003
1570-8268/© 2014 Elsevier B.V. All rights reserved.

where should the investment in designing schemata, producing mappings, and harmonizing data be made?

In this paper, we describe the central role that an Application Programming Interface (API) has made in answering this question for the Open PHACTS Discovery Platform — a Linked Data integration system for pharmacology data. Specifically, we concentrate on how APIs[2] can be seen as an extension to the classical Linked Data application architecture. Concretely, the contributions of this paper are:

- an extension of the Linked Data Application Architecture (LDAA) [5] to incorporate APIs;
- a description of the Open PHACTS Discovery Platform as a case study in applying this extended LDAA.

The use of an API has facilitated the uptake of the platform. The platform was launched April 21, 2013.[3] As of March 1, 2014, the platform has been accessed 13.5 million times[4] and integrated into 12 applications.[5]

This work builds on prior work. The goals and motivation for the Open PHACTS project as a whole are described in [6]; while [7] describes the overall architecture and implementation of the platform in terms of a series of design decisions. In contrast, this paper focuses on the role of APIs with respect to the data integration approach and its relationship to standard Linked Data application design. The API to the platform is itself a consequence of a requirements gathering process focused on business questions from a number of pharmaceutical companies and academic institutions [8].

The rest of this paper is organized as follows. We begin with a motivation for the use of APIs to address pharmacology data integration. This is followed by the description of an Extended Linked Data Application Architecture. This section also discusses questions that these extensions can help answer. We then describe the Open PHACTS Discovery Platform in terms of this extended architecture. Finally, we discuss related work and conclude.

2. Motivation

While the need for data integration has a long history [9], this work is specifically motivated by the needs of pharmacology researchers within the Open PHACTS project[6] [6]. Data integration is a prerequisite for modern drug discovery as researchers need to make use of multiple information sources to find and validate potential drug candidates. Data integration presents a challenge to these scientists from two perspectives:

1. *Structural Data Integration*: the need to repeatedly perform error-prone and tedious mechanical integration;
2. *Semantic Data Integration*: the intellectual complexity in developing models or views that span different domains (e.g. biology and chemistry).

The first challenge, while still requiring investment, is somewhat mitigated by the use of Linked Data technologies. In particular, the use of RDF reduces the need for dealing with syntactic heterogeneity issues (e.g. two RDF files can be concatenated and queried over). Likewise, the use of a standard graph data model and URIs makes it easy to refer to existing datasets. LDIF [10] is an example of a platform that caters for the full stack of integrating data removing some of the issues with mechanical data integration. Systems such as Bio2RDF, Chem2Bio2RDF and Linked Life Data have shown the efficacy of this approach in being able to query and use large quantities of biomedical data. However, even with some integration at the schema level, there is still the need to compose queries that span these datasets. These can often be difficult to formulate especially if the datasets are heterogeneous in nature, which is the common case in pharmacology (e.g. pathway, genetic and chemistry datasets) [11].

The second challenge is more difficult. While the reuse of ontologies (e.g. FOAF, Dublin Core) increases semantic interoperability, many datasets do not adopt the same ontologies or the ontologies are used inappropriately [12]. Furthermore, in complicated domains it takes time to develop and maintain ontologies [13]. The work in developing ontologies is essential. However, consensus and agreement take time and in some cases the particular views on a concept may always be contextual. Agreeing on a common ontology that spans multiple domains, addresses multiple consumers and that supports multiple applications would be a large time investment and not practical in an environment with an ever larger number of changing datasets.

To address the issues above, we adopt the approach suggested in [2], viz. to follow a lightweight integration focused on APIs. There are a number of arguments for the use of APIs. First, APIs implement the classic idea of information hiding. Applications are buffered from changes to the underlying schemata and datasets by acquiring data through a well-defined interface. Second, an API bridges the gap between the domain data models and what is necessary for an application to provide end-user functionality. As [2] argues, the interpretation of the data is dictated by the needs of the application rather than any model. Developers need not worry about constructing complex queries but can instead interact with integrated data through a common interface. Finally, APIs provide a practical mechanism to implement dataspace style integrations. Concretely, this takes the form of either including new data within existing API calls or adding new data. Such additions can be driven by application requirements.

Using APIs and in particular web service APIs, also addresses deployment issues. For example, it allows for the use of off-the-shelf management environments to handle things like documentation, metrics and even caching. Additionally, it protects the database infrastructure from incorrect or complex queries (i.e. no one exposes SQL endpoints completely to the Web). Indeed most web applications that expose their data do this through APIs. Finally, there is a larger set of developers who are familiar with REST-like APIs than SPARQL or Linked Data access via crawling. Given the benefits of APIs, we now look at how these can be combined with the LDAA to facilitate data integration.

3. An extended LDAA

In this section, we begin by presenting the existing LDAA and enumerating integration questions that it leaves to application developers. We then discuss our extensions. Finally, we discuss how the extended LDAA can address these questions.

3.1. The LDAA

Fig. 1 depicts the LDAA. At the lowest level is the publication layer, where datasets are exposed as Linked Data. This can be done by wrapping existing databases (LD Wrapper), exposing data through RDFa, or just publishing RDF directly. This publication is what creates the Web of Data.

[2] We note that when we refer to APIs in this paper, we mean REST-like Web Service APIs.

[3] 4th Open PHACTS Community Workshop: http://www.openphacts.org/4-london.

[4] The statistics reported here are the total number of hits on the API as reported by the 3scale API dashboard.

[5] http://www.openphactsfoundation.org/apps.html.

[6] http://www.openphacts.org.

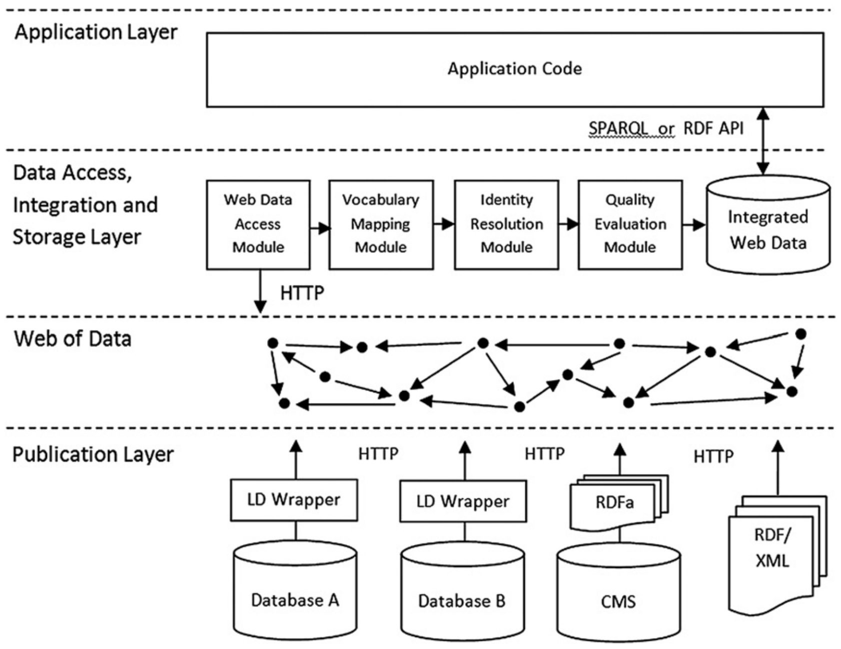

Fig. 1. The LDAA from Section 6.3 of [5].

A central part of a Linked Data application is how to integrate and use data from the Web of Data. The LDAA deals with this in its Data Access, Integration and Storage Layer. This layer consists of four modules arranged in a pipeline that leads to an integrated set of Web Data. The modules are:

Web Access Module: acquires data from the Web of Data using techniques such as SPARQL queries, federated querying, crawling and linked data specific search engines.

Vocabulary Mapping Module: maps the data from the various data sources into a single uniform vocabulary (i.e. schema mapping) for use by applications. This is often accomplished by the use of ontology mappings or more complex structural mapping languages.

Identity Resolution Module: links data instances between datasets. While many linked datasets assert mappings between instances using for example owl:sameAs links, there are still cases where instances are not linked. Linking is achieved through instance mapping (i.e. record linkage).

Quality Evaluation Module: filters out erroneous or non-applicable information coming from the open web. This module can use various techniques (e.g. content heuristics, provenance, or network measures).

The final layer, within the architecture, is the Application Layer. Here applications are suggested to access integrated data via SPARQL or RDF APIs.

The architecture highlights the key parts of a Linked Data Application and highlights the role of integration modules in the design of these applications. However, there a number of questions that the architecture leaves open to linked data application developers. Essentially, these are questions about when and how much developers should pay – in terms of effort – for data during different data integration tasks.

1. Does one need to integrate all the data retrieved through the access mechanism?
2. What should the single integrated vocabulary be? How should one decide?
3. How does one communicate how to use the vocabulary?
4. When does one expand the vocabulary if underlying datasets have changed?
5. What instance mappings are applicable to this application? What if we have more than one application using the integrated data?

3.2. *The extended architecture*

Through the introduction of APIs, we believe Linked Data application developers and in particular those looking at providing integrated data as a service can answer these questions. Fig. 2 shows an extended LDAA. We focus on the top two layers; the others being unchanged. As with the original diagram, arrows represent data flow between components.

We begin by describing each additional component of the architecture. We then revisit the existing components and discuss any modifications of their role. Finally, we describe how this architecture helps to answer the above questions.

Declarative API Definition: A key realization in the design of the Open PHACTS Discovery Platform was that access to data is almost always *mediated* by some end-user application (i.e. application code in Fig. 2). Users rarely access data directly or even compose queries directly on the data itself. They usually acquire and munge the data together using tools — statistical packages such as R, workflow systems such as Knime or analytics applications such as Spotfire. Thus, taking a tool centric approach can help in defining what data to integrate and how to integrate. When discussing requirements with tool developers, it is often easier to discuss the methods by which they *access* the data instead of the data model of any eventual integrated data. The role of the Declarative API Definition is to provide an artifact that describes precisely the link between access methods (i.e. the API) and the underlying data. This declarative definition should be represented using RDF.

API Docs: Documentation of the API is essential. It allows application developers to understand how they can acquire and use the integrate web of data. The generation of API documentation is driven by the declarative API definition.

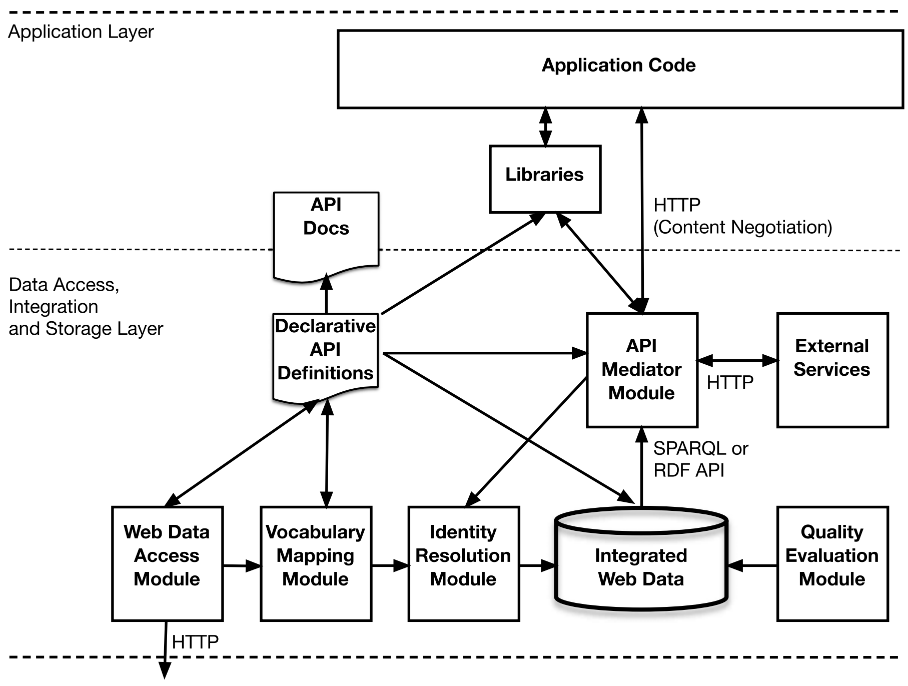

Fig. 2. Extended Linked Data application architecture.

API Mediator Module: This module implements the API definition. It mediates between the application and the integrated web data. Its most simple role is to translate between API calls and SPARQL queries. This module also allows for flexibility in provisioning data. For example, applications can access data using multiple formats through content negotiation (e.g. returning JSON or TSV formats). It also allows the transparent integration of processing within the provisioning pipeline, for example, computing complex filters or contacting external services. Finally, it allows for the integration to be modified on the fly based on API requests as we discuss later.

Libraries: The last additional component in the extended architecture, is the libraries module. While most programming languages provide ready access to libraries for using Web services, with the addition of a declarative API specification, it becomes easier to provide programming language specific libraries. Such libraries lower development time for developers interacting with the API.

The introduction of the API changes the role of the existing components within the architecture. For example, the API Definition can guide the instantiation of the Web Data Access Module; helping to determine which datasets are necessary from the Web of Data in order to fulfill the API's requirements. The role of the Vocabulary Mapping Module also changes; instead of having to design a single target vocabulary that covers all the integrated data, a vocabulary can be designed that focuses on the API's requirements. This often reduces the complexity of the vocabulary mapping as tool developers often want to acquire the data in simple formats, such as CSV, so that they can be easily displayed to the user or put in analysis environments that require tabular data structures. In a sense, the target vocabulary can be viewed less as a full fledged vocabulary and more as a result format.

The API also changes how applications can interact with the integration process itself, in particular, with respect to the Identity Resolution Module. Instead of identity between data items being fixed at integration time, identity can be determined or applied at runtime. Thus, there is a feedback loop between API Mediator and Identity Resolution Modules. This loop also changes where the Quality Evaluation Module sits. We see it as a module that is running constantly over the integrated data as it changes in response to different API queries. This is in contrast to a onetime quality evaluation in the original LDAA.

3.3. Addressing the integration questions

We now discuss how this extension can help address the integration questions enumerated above:

Does one need to integrate all the data retrieved through the access mechanism? The introduction of an API Definition helps guide the selection of both datasets as well as the data with that. For example, within Open PHACTS, the DrugBank dataset is only used for information about the medicinal usage of the drug and not information about a drug's chemistry.

What should the single integrated vocabulary be? How should one decide? Again the API Definition constraints the needed vocabulary. Instead of trying to cater for all potential queries over an integrated dataset, the vocabulary (i.e. the definition of the result format returned from API calls), can focus on the needs of application developers. The discussion about necessary data can be had in terms of how the data can be most easily *accessed* by the applications instead of *modeled*. We often refer to the vocabulary of API results as a *common sense vocabulary* in that it needs to convey how the data can be used by application developers and not necessarily all the scientific nuances of the data.

How does one communicate how to use the vocabulary? In the existing LDAA, one would most likely publish an ontology or vocabulary description but developers would still need to figure out how to compose queries to extract the data they need. This can be solved through example queries,[7] but these are often complex and sometimes hard to generalize. API Documentation is a simple way to communicate how to use the responses from accessing it.

7 See, for example, http://www.wikipathways.org/index.php/Help:WikiPathways_Sparql_queries.

A majority of developers are used to web service APIs and their corresponding documentation.

When does one expand the vocabulary if underlying datasets have changed? Again, the API Definition plays the role of guiding how datasets are used. Even if datasets change, the API only expands if tool developers require it.

What instance mappings are applicable to this application? What if we have more than one application using the integrated data? Elsewhere [14], we have pointed out that different applications may have different perspectives in terms of what entities are considered equal. By enabling applications to dynamically adjust identity resolution on the fly through an API, instance mappings can be used more flexibly.

3.4. Ramifications of the extended LDAA

While introducing an API in the LDAA addresses the aforementioned integration questions, it does have ramifications on other application capabilities. It limits the flexibility of application builders. They cannot write arbitrary queries over the integrated data. It pushes more work on the data integration platform developer as they are now responsible for maintaining SPARQL queries over the set of integrated data sources. It can also make it more difficult to perform automated Linked Data retrieval because of the additional API invocation step necessary. We note, however, that all data is still provided as RDF using dereferenceable URLs. Again, the LDAA still requires the publication of data in some Linked Data form that is accessible. We see APIs as an augment to standard Linked Data publication practices.

We now describe the Open PHACTS Discovery Platform as a case study of applying this extended LDAA.

4. The open PHACTS discovery platform

The Open PHACTS Discovery Platform provides integrated access to 11 Linked Datasets covering information about chemistry, pathways, and proteins.[8] Here, we describe version 1.3 of the platform, this is an update from the version 1.2 released in April 2013. These versions have the same architecture — version 1.3 has additional datasets and API calls. Documentation for the platform is available at http://dev.openphacts.org/docs/.

We encourage readers to look at these documentation directly or to look at our developer support video[9] that walks through how to create a simple pharmacology application with three service calls. To give an intuition as to the difference between using an API and writing a corresponding SPARQL query, the call for getting information on a compound requires one parameter[10] whereas the corresponding SPARQL, see the Inline Supplementary SPARQL Query 1, is 100 lines long.

Inline Supplementary SPARQL Query 1 can be found online at http://dx.doi.org/10.1016/j.websem.2014.03.003.

We now discuss the mapping between the platform and the Extended LDAA by first describing the API Definition and Mediator Modules.

8 For a full list of available datasources with VoID descriptions, see https://github.com/openphacts/ops-platform-setup/tree/master/data-sources.

9 http://www.youtube.com/watch?v=8XV0kC7PuSU.

10 The URL for getting information on the compound Sorafenib is as follows: https://beta.openphacts.org/1.3/compound?uri=http%3A%2F%2Fwww.conceptwiki.org%2Fconcept%2F38932552-111f-4a4e-a46a-4ed1d7bdf9d5&app_id=0e939a76&app_key=1004d9ef5f4ee1ab0bbfc02b623cb955. We note that the URL uses an app_id and app_key that are publicly available, however, in the future if these keys are abused the URL will no longer work.

4.1. API definition, mediator and vocabulary mapping implementation

The Open PHACTS Discovery Platform API has been implemented in PHP as an extension to Puelia,[11] an implementation of the Linked Data API (LDA[12]) vocabulary.

The LDA is the instantiation within the platform of the *Declarative API Definition* defined within the extended LDAA. In short, each LDA endpoint (e.g. API method) is associated with a URL and defined in an RDF configuration file, which specifies the SPARQL query to be issued on the triple store, and allowed HTTP parameter names, the values of which generate SPARQL `FILTER` statements. LDA endpoints are characterized by whether they return data about a single resource, or a list of resources of the same type.

While the provision of an API is motivated by the same rationale that drove the development of the LDA vocabulary, a significant difference between the two is that Open PHACTS is strongly focused on data integration while the LDA itself aimed to provide easy to access representations of core resources inside single datasets.[13]

As such the original specification does not allow the creation of views over the data that do not match the original schema. In contrast, the ability to project information retrieved based on different, dataset specific, schemata into a single, consistent result format is a central requirement in a data integration setting. We have therefore extended the specification to allow declaring graph patterns that should appear in a `CONSTRUCT` clause separately to those that should appear in a `WHERE` clause. This allows us to implement the *vocabulary mapping* directly within the specification itself.

Additionally, the LDA specification assumes that both methods and resources can be uniquely identified through a combination of the URL path and HTTP parameters, which conflicts with users being able to retrieve data using already known equivalent identifiers from data sources that may or not be present in the integrated data. Thus, in the Open PHACTS Discovery Platform API, the URL path only identifies the API method, and a special HTTP parameter called 'uri' is used to allow the identification of resources by arbitrary URIs. Any additional HTTP parameters are then interpreted as filters, as in the original LDA specification. The LDA specifications used in the Open PHACTS Discovery platform are available at: https://github.com/openphacts/OPS_LinkedDataApi/tree/master/api-config-files.

Thus, to add a new API call, we simply define a new LDA configuration. The SPARQL query is then used to select or aggregate data from across the multiple datasets. To extend an existing API call, one simply extends the SPARQL query.

The *API Mediator Module* is implemented by extending Puelia and exposing the API using the 3scale service. Puelia is a PHP implementation of the LDA specification and facilitates the conversion of the SPARQL XML or RDF result of the queries into a variety of formats. On the one hand publishing the results of API queries as RDF under stable HTTP URLs is attractive as such fragments can be mined and reused in third party knowledge bases who would like to link their content to Open PHACTS. Moreover, it provides a non ambiguous provenance for the results as the identifiers and predicates used can be use to query over the original sources. On the other hand our experience confirms that the most popular format amongst application developers is JSON, with simple XML a distant second, while RDF is considered complicated for the purpose of user interaction and presentation. Puelia creates consistent, distinct URIs for each format. We have

11 https://code.google.com/p/puelia-php/.

12 http://code.google.com/p/linked-data-api.

13 See http://code.google.com/p/linked-data-api/wiki/AssumptionsAndGoals.

extended Puelia to wrap external web services. In particular, chemistry structure searching is implemented by calls to external services. This wrapping ensures that developers have one URL for all methods provided by the platform and there is a consistent result format for all methods.

The API is exposed using the 3scale API Management Platform.[14] This platform supports developer accounts, API tracking and monetization, documentation portals as well policy enforcement. This is another benefit of using REST-like APIs — this sort of infrastructure is available and can be used without the cost of development overhead.

4.2. API documentation and libraries

This declarative definition of API methods is beneficial since it allows for automatically generating *API documentation* that in turn integrates well with code generation frameworks. The documentation is generated by converting the LDA specifications to Swagger[15] documentation that can be accessed through our 3scale portal.

There are a several *libraries* for the Open PHACTS API including Javascript[16] and Ruby.[17] The use of web services has also made it easy to integrate with workflow systems.[18]

4.3. Identity resolution and quality evaluation

A comprehensive description of the *identity resolution module* of Open PHACTS is given in [14]. Quickly summarized, the API Mediator can contact this module to activate particular mappings between resources based on optional parameters provided within the API. For instance, one can select whether to use mappings that state that resources are the same because they have equivalent chemistry (e.g. share the same InCHI key) or that they are the same based upon string similarity between labels (e.g. matching on rdfs:label).

Quality evaluation is primarily focused on pre-selection of datasets for integration as well as a domain specific validation of chemistry.

5. Related work

There has been a large amount of work in applying semantic technologies to the biomedical domain and in particular pharmacology [15]. The findings of the Health Care and Life Sciences interest group with regards to the availability of data sources, their potential for linking, and present recommendations for best practices in publishing pharmacology data as Linked Data are presented in [16]. The Bio2RDF [17], Neurocommons [18], Linking Open Drug Data (LODD) [19], Linked Life Data [20] and Chem2Bio2RDF [21] projects have all made significant sets of biology and chemistry data available in RDF. We built upon the comprehensive work of the community in creating RDF-based data sources to power the Open PHACTS Discovery platform. The key difference with our work and these initiatives is the focus on a developer friendly API. Additionally, Open PHACTS has focused on the special role of chemistry in pharmacology linked data integration, an issue that is now coming into focus [22].

Data integration is a large topic and has been addressed extensively in the database literature [23,24]. Linked data extends these notions through the adoption of web standards and best practices [4].

A number of works have looked at the connection between REST Web Service APIs and Linked Data. For example, exposing REST APIs as Linked Data [25,26] or using REST to read and write Linked Data.[19] The Services and Applications over Linked APIs and Data workshop[20] provides a good overview of current developments. Our contribution is to describe how APIs combined with Linked Data facilitate data integration in a concrete case study.

6. Conclusion

Many Linked Data success stories such as data.gov.uk, the Ordnance Survey and Europeana provide REST-like APIs. The experience of launching the Open PHACTS Discovery Platform confirms the importance of such access. Moreover, in this paper, we have shown how APIs are not just an adjunct to the design of a Linked Data application but can be a crucial part of the development of the application, particularly, with respect to integrating data. An important benefit of the extension to the LDAA is to provide a framework for answering questions about how and when to integrate in a pay-as-you-go fashion. Concretely, we presented an extension of the reference Linked Data Application Architecture to include APIs and discussed how the Open PHACTS Discovery Platform implements this architecture. We believe that the Open PHACTS Discovery Platform is a step toward true "smashups" [2] — leveraging Linked Data combined with APIs to enable the quick development of third party applications that work over multiple datasets.

Acknowledgments

We would like to thank the developers and users who have tested and used the platform as we have developed it.

The work in this paper was supported by the Innovative Medicines Initiative Joint Undertaking under grant agreement number 115191, resources of which are composed of financial contribution from the European Unions Seventh Framework Programme (FP7/20072013) and EFPIA companies in kind contribution.

References

[1] P.A. Bernstein, L.M. Haas, Information integration in the enterprise, Commun. ACM 51 (9) (2008) 72–79.
[2] C. Goble, R. Stevens, State of the nation in data integration for bioinformatics, J. Biomed. Inform. 41 (5) (2008) 687–693.
[3] A.Y. Halevy, M.J. Franklin, D. Maier, Principles of dataspace systems, in: Proceedings of the Twenty-Fifth Symposium on Principles of Database Systems, PODS 2006, ACM, New York, NY, USA, 2006, pp. 1–9.
[4] C. Bizer, Interlinking scientific data on a global scale, Data Science Journal 12 (2013) GRDI6–GRDI12.
[5] T. Heath, C. Bizer, Linked Data: Evolving the Web into a Global Data Space, first ed., in: Synthesis Lectures on the Semantic Web: Theory and Technology, vol. 1, Morgan & Claypool, 2011.
[6] A.J. Williams, L. Harland, P. Groth, S. Pettifer, C. Chichester, E.L. Willighagen, C.T. Evelo, N. Blomberg, G. Ecker, C. Goble, B. Mons, Open PHACTS: semantic interoperability for drug discovery, Drug Discov. Today 17 (21–22) (2012) 1188–1198.
[7] A.J. Gray, P. Groth, A. Loizou, S. Askjaer, C. Brenninkmeijer, K. Burger, C. Chichester, C.T. Evelo, C. Goble, L. Harland, et al., Applying linked data approaches to pharmacology: architectural decisions and implementation, Semantic Web.

14 http://www.3scale.net.
15 https://developers.helloreverb.com/swagger/.
16 https://github.com/openphacts/ops.js.
17 https://github.com/openphacts/ops_gems.
18 https://dev.openphacts.org/workflow.

19 See the Linked Data Platform Working Group http://www.w3.org/2012/ldp/wiki/Main_Page.
20 http://salad2013.linkedservices.org.

[8] K. Azzaoui, E. Jacoby, S. Senger, E.C. Rodrguez, M. Loza, B. Zdrazil, M. Pinto, A.J. Williams, V. de la Torre, J. Mestres, M. Pastor, O. Taboureau, M. Rarey, C. Chichester, S. Pettifer, N. Blomberg, L. Harland, B. Williams-Jones, G.F. Ecker, Scientific competency questions as the basis for semantically enriched open pharmacological space development, Drug Discov. Today 18 (1718) (2013) 843–852.

[9] P. Ziegler, K.R. Dittrich, Three decades of data integration—all problems solved? in: R. Jacquart (Ed.), IFIP Congress Topical Sessions, Kluwer, 2004, pp. 3–12.

[10] A. Schultz, A. Matteini, R. Isele, P.N. Mendes, C. Bizer, C. Becker, LDIF—a framework for large-scale linked data integration, in: 21st International World Wide Web Conference, WWW 2012, Developers Track, Lyon, France, 2012.

[11] A. Loizou, P.T. Groth, On the formulation of performant sparql queries, CoRR arxiv:abs/1304.0567.

[12] P. Jain, P. Hitzler, P.Z. Yeh, K. Verma, A.P. Sheth, Linked data is merely more data, in: AAAI Spring Symposium: Linked Data Meets Artificial Intelligence, AAAI, 2010.

[13] D. Baorto, L. Li, J.J. Cimino, Practical experience with the maintenance and auditing of a large medical ontology, J. Biomed. Inform. 42 (3) (2009) 494–503.

[14] C.Y.A. Brenninkmeijer, C. Goble, A.J.G. Gray, P. Groth, A. Loizou, S. Pettifer, Including co-referent uris in a sparql query, in: Fourth International Workshop on Consuming Linked Data, COLD2013, 2013.

[15] E.K. Neumann, E. Miller, J. Wilbanks, What the semantic web could do for the life sciences, Drug Discov. Today: BIOSILICO 2 (6) (2004) 228–236.

[16] M. Samwald, A. Coulet, I. Huerga, R.L. Powers, J.S. Luciano, R.R. Freimuth, F. Whipple, E. Pichler, E. Prud'hommeaux, M. Dumontier, M.S. Marshall, Semantically enabling pharmacogenomic data for the realization of personalized medicine, Pharmacogenomics 13 (2) (2012) 201–212.

[17] F. Belleau, M.-A. Nolin, N. Tourigny, P. Rigault, J. Morissette, Bio2RDF: towards a mashup to build bioinformatics knowledge systems, J. Biomed. Inform. 41 (5) (2008) 706–716.

[18] A. Ruttenberg, J.A. Rees, M. Samwald, M.S. Marshall, Life sciences on the Semantic Web: the Neurocommons and beyond, Briefings Bioinform. 10 (2) (2009) 193–204.

[19] M. Samwald, A. Jentzsch, C. Bouton, C. Kallesoe, E. Willighagen, J. Hajagos, M. Marshall, E. Prud'hommeaux, O. Hassanzadeh, E. Pichler, S. Stephens, Linked open drug data for pharmaceutical research and development, J. Cheminform. 3 (1) (2011) 19+.

[20] V. Momtchev, Linked life data: knowledge extraction and semantic data integration in the pharmaceutical domain, in: larKC Pharma Workshop at High Performance Computing Center Stuttgart, HLRS, Stuttgart, Germany, 4, 2011.

[21] B. Chen, X. Dong, D. Jiao, H. Wang, Q. Zhu, Y. Ding, D. Wild, Chem2Bio2RDF: a semantic framework for linking and data mining chemogenomic and systems chemical biology data, BMC Bioinform. 11 (1) (2010) 255.

[22] J.G. Frey, C.L. Bird, Cheminformatics and the semantic web: adding value with linked data and enhanced provenance, Wiley Interdiscip. Rev. Comput. Mol. Sci. 3 (5) (2013) 465–481.

[23] A.Y. Halevy, A. Rajaraman, J.J. Ordille, Data integration: the teenage years, in: Proceedings of 32nd International Conference on Very Large Data Bases, VLDB, ACM, New York, NY, USA, 2006, pp. 9–16.

[24] M. Lenzerini, Data integration: a theoretical perspective, in: Proceedings of 21st ACM Symposium on Principles of Database Systems, PODS 2002, ACM, New York, NY, USA, 2002, pp. 233–246.

[25] S. Speiser, A. Harth, Integrating linked data and services with linked data services, in: The Semantic Web: Research and Applications, Springer, 2011, pp. 170–184.

[26] M. Taheriyan, C.A. Knoblock, P. Szekely, J.L. Ambite, Rapidly integrating services into the linked data cloud, in: The Semantic Web–ISWC 2012, Springer, 2012, pp. 559–574.