A proposal of architecture and process of deployment for Linked Data projects

Francisco Cifuentes-Silva, José María Álvarez Rodriguez, and Jose Emilio Labra Gayo

Universidad de Oviedo, WESO Research Group,
C/ Calvo Sotelo s/n, 33007, Oviedo, España.
francisco.cifuentes@weso.es, josem.alvarez@weso.es, labra@uniovi.es,
http://www.weso.es

Abstract. The establishment of Linked Data on the Web principles[5] and the emerging Linking Open Data project [1], to establish solid foundations for the growth of the concept of Web of Data. However, although these work define the form (Linked Data) and the target (Web of data), is fuzzy yet the definition of a components architecture that give support to the implantation of these technologies in production environments, also it is fuzzy a implementation process related to this architecture, so that together enable both the publishing as the maintenance of semantic data on the Web. In this paper we describe a first approach about an architecture and one adoption process of technologies that enable the publishing and the maintenance of Linked Data from a general perspective. Then, we define a set of components that allow the basic functions in a Linked Data environment, describing their main features and functions, and the sequentially in that must be implemented. Finally, we briefly review similar approaches to our that have been used as base for this work.

Keywords: Semantic Web, Linked Data, Methodology, Semantic Web Methodology

1 Introduction and Motivation

Each year, there are more organizations in the world that openly publish datasets of Linked Data, queryable from any point on the Web. It is estimated that there are published officially, to April of 2011, over 35,8 billion of triples [2] distributed in approximately 280 datasets throughout the world. The interest about why make available openly the data has multiple justifications:

- To generate trust, promoting transparency in the information
- To facilitate studies and research
- Open systems facilitate external contributions

[1] http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
[2] http://www4.wiwiss.fu-berlin.de/lodcloud/

– The public data belong to the nation, they are conceived through taxes to citizens.

Supporting the adoption of semantic Web technologies, currently, there exists a great of tools oriented to creation, publication and management of data, and a big subset for our interest object, Linked Data. However, an important weakness in this area of web engineering is that it has not been established completely a formal reference that integrates, in general aspects, the necessary infrastructure in terms of components, and the order that this infrastructure must be implanted. This lack implies a slower technological adoption, covering both public sector and private sector[8]. Although currently exist general approaches related to publishing and consume of Linked Data[6, 20], our proposal adds aspects related to maintenance, use and adaptation of Linked Data contents in a general context. Under this scenario, our main contribution is the description of a base architecture and an adoption process of Semantic Web technologies that enable a Linked Data environment. Therefore, we define an architectural view of technological components that support the publishing and maintenance of Linked Data, and we describe the needed process for the implementation and deployment of this components infrastructure. In this way, in a first stage we define the architecture and their components, and subsequently we describe the sequentiality of implementation, providing a reference that may be applied in multiples contexts.

2 Linked Data Infrastructure

The Fig. 1 shows our architectural proposal for the Linked Data implementation. The model is defined based on the following components or layers:

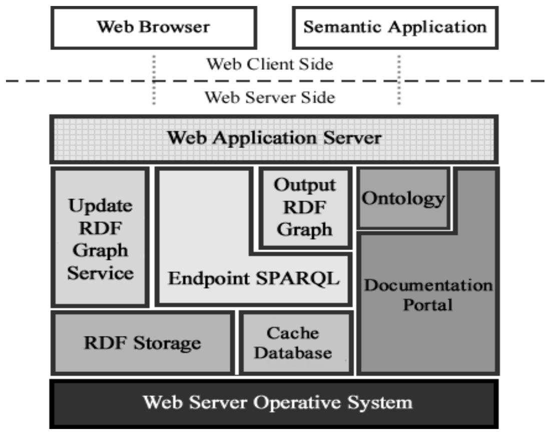

Fig. 1. Our proposal of Linked Data architecture

- **RDF Storage System**[13]: A component used for the RDF triples management. Over this, will work the SPARQL Endpoint and the Output RDF Graph.
- **Cache database**: A database management system optimized for cache, that, among others tasks, has as aim store queries and results from the RDF Storage System, achieving improve the response time in complex SPARQL queries that implies reasoning. This component should be used at least by the SPARQL Endpoint.
- **Documentation Web Portal**: This Web portal has three targets, to contain the ontologies of the domain model, to contain the documentation about URIs schema and RDF graph for developers, and finally to be the access point for all the services and resources of the Linked Data infrastructure.
- **Ontology**: One or a set of ontologies that model the domain over which Linked Data are generated.
- **Output RDF Graph**: Also called "Linked Data Frontend", is an application that generates RDF triples, for URIs previously designed, or in simple terms, an application that generates a RDF graph over an HTTP URI. For simplify the model, we considerate that for the data collection, this tool makes queries to SPARQL Endpoint, processing and showing results in URIs defined for the graph model.
- **Endpoint SPARQL**: A tool that meets the SPROT specification (SPARQL Protocol for RDF)[11], which allows executing SPARQL[18] queries over a RDF Graph, generating consumables results for clients. Under this tool in our architecture, is defined both a RDF Storage System and a Cache database, and over this, the Output RDF Graph.
- **Update RDF Graph Service**: this component has as aim, to make the updates from the corporative database to RDF Storage System. Process such as automatic create, update and delete of RDF triples, are tasks of this component.

3 Adoption Process

For the implementation of the architecture that we defined above, we propose an adoption process in the phases defined in the Fig. 2, which shows each one of the phases applied in a temporal dimension. Now, we going to explain each phase of the adoption process.

3.1 Contextualization

In this phase should be identified the application context and nature of the data to publish. Using a method of requirements description such as UML notation or similar, we purpose to define from a high level, a systemic view based on three elements:

1. **What data will be provided**: to describe the domain model of data that will be published, considering referencing external data sources. This involves the description of the domain model in form of classes, attributes and

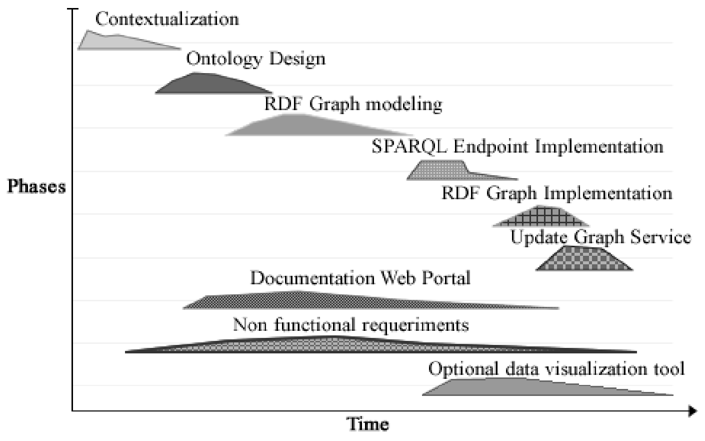

Fig. 2. Our proposal of process for deployment of Linked Data

relationships that can be expressed, for instance in a first approach using UML notation, whatever subsequently will be converted to RDF.

2. **How will be delivered the data**: to describe access models and consultation of the data, giving priority to the use of open standards such as REST or SOAP. Also must be defined the output formats in that the data will be provided, for instance, common output formats are HTML, RDF, XML, JSON, N3 or YAML.
3. **Who will consume the data**: to describe who will consume the data, if there are specific agents, specific own or third party applications, or visualization tools oriented to humans.

As a result of this stage, will be obtained a first approximation of system that will be detailed in a document of analysis that consider the three points mentioned above.

3.2 Ontology Design

This phase includes the definition of vocabularies or ontologies required commensurate with the domain model established in the previous phase. This stage is of vital importance in our semantic system, due to with this model will be possible apply validations, logical rules and inferences about our data. For doing this, we consider two possible alternatives:

1. **Use any existing model**: If the domain model we need using has been modeled by others as ontology and is available for use, the best choice will be to use total or partially without extra effort. To determine if the domain model has been modeled before, you can use search engines of semantic material such as Falcons [3], Swoogle[4], Watson[5] or similar.

[3] http://ws.nju.edu.cn/falcons/
[4] http://swoogle.umbc.edu/
[5] http://kmi-web05.Open.ac.uk/WatsonWUI/

2. **Design a model reusing others**: If the domain model we need to model is not modeled, will necessary to design an ontology that allows to represent the problem. For the design, there ontology matching techniques as defined in [10, 14, 15] that enable reuse. In this point take special meaning the principles of "reuse, not reinventing" and "mix freely" as defined in [6].

For the implementation of the ontology, the ideal is to use OWL [17], RDF Schema [19] or a combination of them, whereas RDF Schema allows less expressiveness that OWL, since through RDF Schema is possible express only classes, properties and hierarchies between these elements, while OWL adds the definition of negations, quantifiers, cardinalities and attributes of properties.

Other point to consider at this phase is the establishment of a HTTP URI and a prefix for the publication of the ontology. We suggest in this aspect, leave all the ontologies under a directory "ontologies/". With respect to the prefix, if it is an organization, ideally the prefix should be related to the name or acronym that will be unique for publication on the Web. On the other hand, if we are modeling in the ontology only one part of the domain of the organization, we recommend implementing the ontology separately referenced from the ontology root. In this way, the prefix of the new part of the organization will be composed with the ontology base prefix concatenated with a prefix of the sub-domain model. Applying this design criteria, will be possible to extend without limits the ontology and the application domains through sub-ontologies. Finally, we believe that the ontology should be commented and documented in English language and in the language of context of the project.

3.3 RDF Graph modeling

In this phase must be defined the HTTP URIs of the RDF Graph, and the data structure of the information which will be accessed in each request. Then, the first step will define the URI patterns that will form the RDF Graph. For this, there are many approaches that support the modeling through URI design patterns such as the book by Davis and Dodds[9], topic too studied by Berners-Lee in works about URIs such as [3] and Cool URIs[4]. Another recommendation for this topic, is hide the implementation details, designing URIs based on abstractions that enable the legibility, such as is described in [6]. At the same time of designing the URIs pattern of RDF Graph, should be designed the data that will be delivered in each URI access. Our approach considers that the data to deliver in each request must be the modeled in the ontology or vocabulary related with the entity defined in the URI. For defining the data output is a well practice define an example of URI describing the output RDF. For this exercise could be used Notation 3 syntax, more human readable. Once that are designed all URIs pattern, and RDF outputs, the RDF Graph is modeled almost entirely.

Others important topics for to consider are the content negotiation at the time of deliver the content and the data visibility. In the first case, works as [6, 12] have developed methods that enable this feature applied Linked Data environments. About the data visibility, for helping to crawlers of search engines will be very useful the creation of a semantic site map, such as explain [6]. In technical words, an robots.txt file of the web site, where should be considered at least the SPARQL Endpoint location, the base URI of web site and Linked Data sources, and the location of a data dump if exists.

Our purpose includes too the internationalization of URI patterns when the project contextualization is out of english language. There are application contexts that require the publishing of data in the context language, for instance the public administrations. This requirement, involves maintain the URIs in the project context language, adding a translation to the static parts of each pattern to english language, achieving in this way maintain a standardized access for all the possible consumers of the data.

3.4 SPARQL Endpoint Implementation

The next phase involves the implementation of the named SPARQL Endpoint. Commonly the organizations maintains and manages their data in corporative databases which access the applications. For enable the use of this data in the Linked Data context, will be necessary to perform an extraction, transformation and loading process, also called for short ETL[16] in the Business Intelligence context, or at least a transformation process from the corporative database to RDF data. Then, in this phase we observe two main tasks:

- **Extraction, transformation and loading of data**: this task involves take the data from the organization data environment to the publishing on the Web. We recommend the use of ETL tools such as Kettle[6] used in Business Intelligence context, because is free for use and offers improved features for making the job in an automatic way.
- **SPARQL Endpoint implementation**: This task can be divided giving implementation to three elements: a database management system with a SPARQL query defined as input, and one reasoner for the logic operations if reasoning capabilities are required. Our solution purpose the use of an RDF storage system like the defined in [13], where the reasoning capabilities are implemented in an architecture layer of the component. In the practice, currently exists implementations for this component such as Sesame[7], Openlink Virtuoso[8], RDFStore[9], 4Store[10] or Redland[11].

3.5 RDF Graph implementation

In this phase must be perform the implementation of the component that generates a RDF Graph over the previously designed HTTP URIs. Basically, the

[6] http://kettle.pentaho.com
[7] http://www.Openrdf.org/
[8] http://virtuoso.Openlinksw.com/
[9] http://rdfstore.sourceforge.net/
[10] http://4store.org/
[11] http://librdf.org/

application must receive an HTTP request in a defined HTTP URI, and must return a RDF data set in some format representation established by one parameter. Innerly, the application should be communicated with the SPARQL Endpoint, performing SPARQL queries for each HTTP request. Although, this application could be developed completely, there are many alternatives of free software that implement the functionality (also called Linked Data Frontend), such as Pubby[12], Elda[13], or djubby[14], latter used in architectures where the Linked Data publish is based on mapping from relational databases.

3.6 Update Graph Service

In this phase must be implemented a tool that allows maintain updated the RDF graph through the time. The main idea is design an autonomous service with the capability of add the new data from the corporative database to the RDF storage system. For the implementation, an alternative is to compose the tool using the ETL transformations defined formerly, but changing the manual execution of the process for programmations of automated executions. Another alternatives for this task could be the implementation of external applications based on web services, sockets or other communication methods.

3.7 Documentation Web Portal

This phase has as aim the implementation of a web portal that allows the publishing of documentation and support for the data consume. The purpose contents of the portal are defined as follows:

- Ontologies or Vocabularies modeled.
- Documentation guide for the consume of published data.
- A tutorial or user guide for the SPARQL Endpoint, that includes examples of queries.
- The necessary contents that allows the access to all the Linked Data infrastructure.

About the most important features of the web portal, we consider key the internationalization when the context language of project is not the english. In this case should exist a main version of the portal in the context language, as well as an english version that enables visibility of the project for the rest of the world.

3.8 Non functional requirements

The Linked Data support infrastructure implies the implementation of some non functional requirements that offers some advantages in production. For instance, we consider relevant the cache management. This feature will allow to

[12] http://www4.wiwiss.fu-berlin.de/pubby/
[13] http://elda.googlecode.com/hg/deliver-elda/src/main/docs/index.html
[14] http://code.google.com/p/djubby/

improve response times in SPARQL queries, specially when query implies reasoning capabilities or high data volume. In our architecture, is considered a cache management between the RDF storage system and the output RDF graph. Another very important non functional requirement is the security. For instance, the Update Graph Service should transfer the news triples from the corporative database to the RDF storage system in a secure way over Internet, in this case must not be possible to change the data if is intercepted the data stream.

3.9 Optional Data visualization tool

Possibly the most attractive applications when there are data in Linked Data format, are the visualizations or mashups, more yet in Web 2.0[1] contexts in transition to Web 3.0 like current. Whereas under this context is possible interoper with diverse sources, without necessity of consuming data through API or transformations from specific formats, is a good idea to design a mashup or visualization that represents data in an attractive form, and at the same time, give more added value to the project. For the implementation of Linked Data visualization tools, there are free alternatives as Linked Data Browsers[7], Faceted Navigation Tools and Mashups. Some examples of this type of tools are Tabulator[15], Marbles[16], FOAFNaut[17], DERI Pipes[18], Openlink Data Explorer[19], Disco[20], Zitgist[21] or gFacet[22].

4 Related work

Some works that have been our fundamental base are mentioned as follows:

In [2] Berners-Lee gives a justification about why put the data government as Linked Data. Additionally, this work gives ideas and recommendations about how to put Linked Data for all in a open way.

In [6] are described the stages related to Linked Data publishing in a general way, defining main tasks that include negotiation of content, ontologies and URIs design, and contextualization, however, this approach does not consider the ETL process, the update graph service and the points of documentation and internationalization depending on the context.

In [12] is established a detailed introduction of technical aspects mainly related to publishing and consume of Linked Data. Through an use scenario, they bring an overview of methods, frameworks and good practices related to implementation of Linked Data in a generic context. In complement with this work,

[15] http://www.w3.org/2005/ajar/tab
[16] http://marbles.sourceforge.net/
[17] http://crschmidt.net/semweb/foafnaut/
[18] http://pipes.deri.org/
[19] http://ode.Openlinksw.com/
[20] http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/
[21] http://zitgist.com/
[22] http://code.google.com/p/gfacet/

we add the definition of an architecture and an adoption process that enable the sequentiality of deployment of the components of the architecture in a general context.

In [21] is presented a case of study in the scope of RDF graph modeling. In this work, are described the adoption of the SKOS ontology, the concepts of content negotiation and other technical aspects related to our work.

Finally, have been important references for our work, the documentation of the implementations released by The Library of Congress of the United States[23] and for initiatives developed in United Kingdom and Spain such as the Opening Up Government project[24], and the Spanish Association of Linked Data[25].

5 Conclusions

Our work provides the definition of a components architecture and a deployment process of Linked Data in a generic context. For this approach, we have relied in the most common needs of a technological support infrastructure of Linked Data with all its implications. As result, we have generated a systemic view, where we define the functionality of a set of components inner an architecture, and the main tasks with their sequentiality of implantation, including even tools, always in Linked Data context. We believe that adapting our proposal architecture and implantation process, there will be a clear reference at the time of define both requirements in the implementation and technical implications in a Linked Data project.

References

1. Ankolekar, A., Krtzsch, M., Tran, T., Vrandecic, D.: The two cultures: Mashing up web 2.0 and the semantic web. Web Semantics: Science, Services and Agents on the World Wide Web 6(1), 70 – 75 (2008), semantic Web and Web 2.0
2. Berners-Lee, T.: Putting government data online - design issues (Jun 2009), http://www.w3.org/DesignIssues/GovData.html
3. Berners-Lee, T.: Univeral resource identifiers – axioms of web architecture (1996), http://www.w3.org/DesignIssues/Axioms.html
4. Berners-Lee, T.: Cool uris don't change (1998), http://www.w3.org/Provider/Style/URI.html
5. Berners-Lee, T.: Linked data - design issues (Jul 2006), http://www.w3.org/DesignIssues/LinkedData.html
6. Bizer, C., Cyganiak, R., Heath, T.: How to publish Linked Data on the Web (Oct 2008), http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/
7. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. International Journal on Semantic Web and Information Systems pp. 1–26 (Mar 2010)

[23] http://id.loc.gov/authorities/about.html
[24] http://Data.gov.uk/Linked-Data
[25] http://www.aelid.es/

8. Coskun, G., Streibel, O., Paschke, A., Schäfermeier, R., Heese, R., Luczak-Rösch, M., Oldakowski, R.: Towards a corporate semantic web. In: International Conference on Semantic Systems (I-SEMANTICS '09). Graz, Austria (09/2009 2009)
9. Davis, I., Dodds, L.: Linked data patterns (2010), http://patterns.dataincubator.org/book/
10. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: Semantic schema matching. In: Meersman, R., Tari, Z. (eds.) On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, Lecture Notes in Computer Science, vol. 3760, pp. 347–365. Springer Berlin / Heidelberg (2005)
11. Grant, K., Lee, C., Feigenbaum, Torres, E.: SPARQL protocol for RDF (2008), http://www.w3.org/TR/2008/REC-rdf-sparql-protocol-20080115/
12. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool (Jan 2011), http://linkeddatabook.com/editions/1.0/
13. Hertel, A., Broekstra, J., Stuckenschmidt, H.: RDF Storage and Retrieval Systems (2008)
14. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Ontology matching with semantic verification. Web Semantics: Science, Services and Agents on the World Wide Web 7(3), 235 – 251 (2009), the Web of Data
15. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: the state of the art. The Knowledge Engineering Review 18(01), 1–31 (2003), http://dx.doi.org/10.1017/S0269888903000651
16. Kimball, R., Caserta, J.: The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. Wiley, Indianapolis, IN (2004)
17. McGuinness, D.L., van Harmelen, F.: OWL web ontology language overview (2004), http://www.w3.org/TR/owl-features/
18. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF (2008), http://www.w3.org/TR/rdf-sparql-query/
19. RDF Working Group, W.: RDF - semantic web standards (2004), http://www.w3.org/RDF/
20. Sequeda, J., Hartig, O., Sinclair, P., Taylor, J.: ISWC 2009 Tutorials/How to consume linked data on the web - iswc2009 (Oct 2009)
21. Summers, E., Isaac, A., Redding, C., Krech, D.: Lcsh, skos and linked data. In: Proceedings of the 2008 International Conference on Dublin Core and Metadata Applications. pp. 25–33. Dublin Core Metadata Initiative (2008), http://portal.acm.org/citation.cfm?id=1503418.1503422