Integrating FLOSS repositories on the Web

Aftab Iqbal Richard Cyganiak
Michael Hausenblas

DERI Technical Report 2012-12-10
December 2012

DERI Galway
IDA Business Park
Lower Dangan
Galway, Ireland
http://www.deri.ie/

DERI Technical Report

DERI Technical Report 2012-12-10, December 2012

Integrating FLOSS repositories on the Web

Aftab Iqbal[1] Richard Cyganiak[1] Michael Hausenblas[1]

Abstract. This paper provides a novel approach to the problem of integrating data from multiple code forges of FLOSS. We review the current problems in integrating the data from multiple forges and argue that Semantic Web technologies are suitable for representing knowledge contained in code forges. Further, we show the advantage of linking the metadata of projects to other data sources on the Web which will enable querying extra information from the Web. The paper briefly describes how the modeling is achieved and what benefits can be obtained by enabling linking to other relevant data sources already available on the Web.

Keywords: Web of Data; Software Engineering; FLOSS; Repositories; Data Integration.

[1]DERI, National University of Ireland, Galway, Ireland. firstname.lastname@deri.org

Acknowledgements: This work has been carried out in the Linked Data Research Centre (LiDRC) and has been supported by Science Foundation Ireland under Lion 2 and the European Commission's FP7 Support Action LOD-Around-The-Clock (LATC), project no. 256975, Intelligent Information Management (ICT-2009.4.3).

Contents

1 Introduction

Free/Libre open source software (FLOSS) often use a variety of tools to help manage their projects. These tools include managing project code, a place for the users to download the project releases, discussion forums, bug trackers, mailing lists etc. A FLOSS project team may choose to host the project on their own code repository or may choose to host the project on one of the many available online code forges (such as Sourceforge[1], Savannah[2], Github[3] etc.). These code forges host hundreds of thousands of projects. Each project code repository along with other tools leave a detailed trace about the activities being carried out during the whole life span of the project [GBICS10]. Much software development research has been carried out on gathering metrics and developing empirical studies based on data retrieved from these project repositories. However, researchers sometimes find it difficult to make sense of all the data for a research study due to the sheer size of code forges, the amount of data each project holds, the heterogeneity of the projects being studied and harvesting or crawling the code forges which is a daunting task. To provide the researchers easy access to the project's data, two research projects were initiated (with slightly different objective) by the FLOSS research community which are FLOSSmole[4] and FLOSSMetrics[5], also known as "repository of repositories (RoR)". These RoRs were created to consolidate metadata and analysis of projects from a variety of code forges into a centralized place for use by the researchers in academia and industry.

In this paper, we take into consideration only project's metadata from the code forges which are made available to download by the FLOSSmole community. Further, we only study Googlecode and Sourceforge data for this paper, although our methods extend to other code forges as well. The contribution of this paper is twofold: first we discuss the challenges faced while integrating metadata from different code forges. Later we propose our approach of modeling the forges using a common model and representing project's metadata in a standard format. Further, we show the benefits of interlinking the project's metadata to other data sources which are available on the Web using some example scenarios.

The paper is structured as follows: in section 2, we discuss the challenges in integrating data from code forges and outline a few use-case scenarios which can be addressed via interlinking forges to each other. In section 3, we describe our approach of modeling the code forges and give an outline of the overall architecture. We present some preliminary results from a small study which will be presented in section 4. Finally, we conclude in section 5 and outline future steps.

2 Motivation

Flossmole [HCC06] crawls data from 24 different code forges and makes each forge data dump accessible to the community as flat delimited files, as SQL dumps or as direct database access respectively. Although the data extracted from each forge is complete, cleaned and well described, integrating knowledge across multiple forges is still a big challenge. Each code forge has its own database schema and represents data elements differently. Code forges sometimes use different ter-

[1] `https://sourceforge.net/`
[2] `http://savannah.gnu.org/`
[3] `https://github.com/`
[4] `http://flossmole.org/`
[5] `http://www.flossmetrics.org/`

minologies to represent same things like project topics, operating systems, programming languages etc. For example, Googlecode defines the term **Mac** to associate "Macintosh" as an operating system to the project's metadata but Sourceforge defines it as **OS X**. Although both refer to the same operating system, different terms are used. Hence, we are required to define some kind of classification/mapping which explicitly states that the two terms are semantically similar and belong to the operating system family.

Associating metadata to a project is also being handled differently in different code forges. For example, Sourceforge has provided a granular hierarchical structure of associating metadata to a project by allowing developers to select attributes from various category lists (i.e., operating systems, programming languages, database environments etc.). Compared to Googlecode, there is no proper hierarchical structure for associating metadata to a project which makes it harder to integrate the two forges based on similar metadata. At a schema level, the two forges do not share common semantics for associating attributes to the project. We cannot say that the column called **project_operating_system.description** in Sourceforge holds the same meaning as the column called **gc_project_labels.label** in Googlecode. The reason is column **gc_project_labels.label** hold attributes from various categories (i.e., programming languages, project topics, Databases, etc.). Though the data from each code forges have been made available in a relational format by FLOSSmole, the challenge is to define a common model which can be used by all code forges to represent project's metadata.

Further challenges upfront in integrating forges as mentioned in [Con06] are: forges may have projects with same names, large open source projects may have sub-projects hosted on different forges (e.g. Apache projects hosted on GitHub[6]), same developers may have different ID(s) across multiple forges, different developers may have same name across multiple forges and developers may use same ID(s) across multiple forges.

It is worth mentioning that code forges are somehow interconnected but rather of implicit nature (developers contributing to projects across multiple forges, for example). We hence need not only make the interconnections across forges explicit, but also allow connecting to relevant data on the Web. Having such an explicit representation of the connections between multiple forges, we will be able to support certain use-case scenarios as follows:

1. **Tracing developer activities across multiple code forges.** Linking same developer's profile across multiple repositories will enable us to discover the activities of a developer in different forges. It further allows to rank forges based on the number of projects a developer is working in a particular forge.

2. **Is the popularity level of a particular code forge increasing or decreasing?** It would be interesting to study if developers are migrating from one code forge to the other. For example, if developers consider hosting new open source projects on Github rather than Googlecode or Sourceforge. It will lay down the foundation for studying community dynamics across code forges.

3. **Interlinking similar projects across multiple forges.** Sometimes it happens that the projects are migrated from one code forge to another. Linking those project will enable full history of the project across forges.

[6] `https://github.com/apache/`

4. Assuming that the **project's metadata is interlinked to other data sources on the Web**, further information about a particular entity can be provided. For example, if **Algol 68**[7] has been associated as a programming language to a particular project than the developer would be able to get further information about **Algol 68** from the Web (for example, from Wikipedia[8]).

3 Methodology and Architecture

Aforementioned, the usage of a common model and standard format to represent project's metadata from multiple forges would allow for better integration. One may think of questions like: what is the best way to express the knowledge so that it can be integrated easily across multiple code forges? Can the knowledge be further used to link to other data sources which contains extra information about a certain entity? Can it be done in an automated fashion?

We propose to use Semantic Web technologies to represent FLOSS data from different code forges. As such, we propose to use RDF [KCM04] (Resource Description Framework) as the core, target data model. Further the RDFS and OWL standards can be used to well-define the vocabulary needed to describe the data (i.e., classes and properties). Once modeled in RDF, the data can be indexed and queried using the SPARQL query standard and associated tools. Additionally, the more recent SKOS[9] standard can be used to model hierarchical concept schemes. Finally, the integrated data can be published on the Web using Linked Data priniciples[10] allowing third parties to discover and subsequently crawl the knowledge, and also allowing to interlink with background information available remotely on the Web. For space reasons, we refer the readers to [HB11] for details on how these standards would be used. Instead here we focus on the use of SKOS and Linked Data principles to illustrate part of the modeling process.

We looked into Sourceforge and Googlecode's hierarchical structure of assigning attributes to a project and found that Sourceforge is following a proper hierarchy of categorizing the project's attributes. Hence, we decided to do the modeling based on the hierarchical organization of project attributes in Sourceforge rather than Googlecode. We examine the modeling of forges and interlinking approach to other data sources in the following. An excerpt of an exemplary RDF representation of few operating systems classification using SKOS vocabulary is shown in listing 1[11].

The SKOS vocabulary allows aggregating concepts/terminologies into a single concept scheme. For example, in the above listing we defined few desktop operating system under one scheme. The benefit of using SKOS vocabulary is that the concept schemes are extensible so if a code forge has a desktop operating system which is not listed in the core concept scheme than it can be added easily (see listing 2) or the forge can define it as a SKOS concept in a secondary concept scheme and link it to the core SKOS concept scheme using `skos:inScheme` property. This approach enables us to keep the desktop operating systems from different code forges under one scheme.

Referring to the previous example (cf. section 2) of using different terminologies for the same concept, SKOS vocabulary offers to use the `skos:exactMatch` property to define that the terms

[7] `http://www.algol68.org/`

[8] `http://en.wikipedia.org/wiki/ALGOL_68`

[9] `http://www.w3.org/2004/02/skos/`

[10] `http://www.w3.org/DesignIssues/LinkedData.html`

[11] We encourage readers to have a look at the concept schemes we defined, which are available at : http://ld2sd.deri.org/linkedfloss/schemes/

```
@prefix base: <http://ld2sd.deri.org/linkedfloss/schemes/os#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
base:desktop_OS a skos:ConceptScheme ;
        skos:prefLabel "Desktop Operating Systems"@en ;
        skos:hasTopConcept base:linux ;
        skos:hasTopConcept base:net_bsd ;
        skos:hasTopConcept base:os_x .
base:net_bsd a skos:Concept ;
        skos:inScheme base:desktop_OS ;
        skos:prefLabel "NetBSD"@en .
...
```

Listing 1: An exemplary Sourceforge modeling.

```
base:desktop_OS a skos:ConceptScheme ;
        ...
    skos:hasTopConcept base:vista ;
    ...
base:vista a skos:Concept ;
        skos:inScheme base:desktop_OS ;
        skos:prefLabel "Windows Vista"@en .
...
```

Listing 2: An exemplary addition of a new concept to the existing concept scheme.

Mac and **OS X** are semantically similar. So far we have shown how to do the modeling of forges, the next is how to interlink the terminologies or concepts to other data sources. For example, we know that "Windows Vista" is an operating system and it would be beneficial if we interlink this concept to the relevant data source on the Web (for example, to the Wikipedia entry[12]). By interlinking the two data sources together we will be able to get information about "Windows Vista" from Wikipedia. In order to extract structured information from Wikipedia, the research community has developed DBPedia [ABK+07]. DBPedia allows to ask queries against Wikipedia, and to link other data sets on the Web to Wikipedia data. We can interlink the concepts we defined (e.g. base:vista) to the corresponding DBPedia entity using an `owl:sameAs` property indicating that these URIs actually refer to the same entity (see listing 3).

```
...
base:vista a skos:Concept ;
        skos:inScheme base:desktop_OS ;
        skos:prefLabel "Windows Vista"@en ;
    owl:sameAs <http://dbpedia.org/resource/Windows_Vista> .
...
```

Listing 3: An interlinking example.

In Fig. 1 the overall architecture of our approach is depicted. The architecture basically covers the layers as described in the following:

1. The project's metadata from different code forges is being crawled periodically by the FLOSSmole and FLOSSmetrics RoRs, yielding database dumps of each code forge.

[12] `http://en.wikipedia.org/wiki/Windows_Vista`

2. Study the database schema of each forge and model it using a common vocabulary, such as SKOS.
3. Produce the metadata of each project from each forge in RDF[13].
4. Interlink the metadata of projects with each other, across forges and to the LOD cloud[14], where necessary. This enables one to answer many research questions (for example, discussed in section 2).

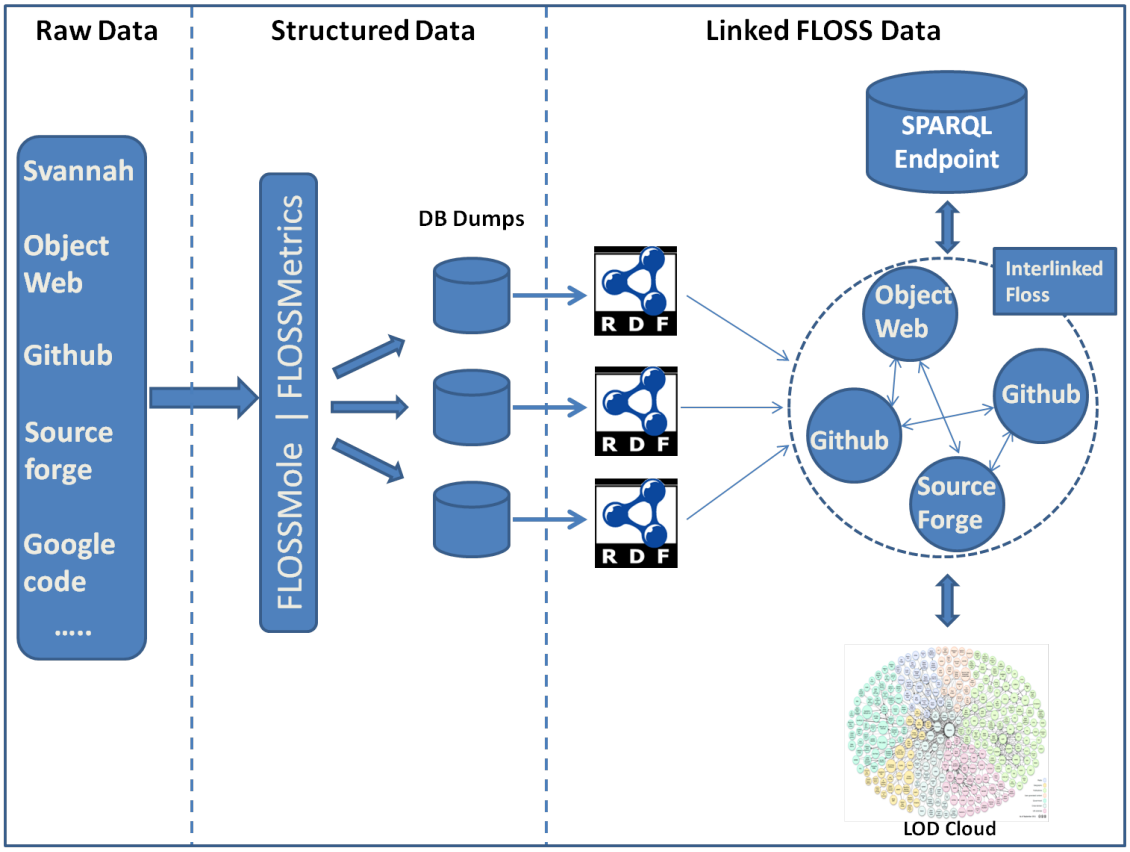

Figure 1: Architecture.

4 Preliminary Findings

In this section, we will argue for linking code forges to other potential data sources on the Web using some examples. We have discussed in the previous section about interlinking project's metadata to their relevant terms in Wikipedia. We will now show that we can also interlink the project from a code forge to its existence in other data sources on the Web. Ohloh[15] for example, is a free

[13] e.g. `http://ld2sd.deri.org/linkedfloss/sourceforge/project/filezilla.rdf`
[14] `http://lod-cloud.net/`
[15] `http://www.ohloh.net/`

wiki for open source software and people. Ohloh provides statistical services about open source projects by using data hosted in their directory. An RDF wrapper[16] (known as RDFohloh) around Ohloh has also been developed by the Semantic Web research community, which provides Linked Data from Ohloh. The RDF data produced from RDFohloh can be interlinked to the relevant project's RDF data from code forges using the `owl:sameAs` property. This interlinking will allow anyone to retrieve more information from Ohloh like for example, analysis summary of the project, geo location of a particular developer, total number of commits made by a particular developer, developer's kudo rank and other projects he is contributing to etc., using a SPARQL query[17].

We also performed a preliminary analysis to identify the overlap across different code forges based on developers. We extracted a list of people (i.e., developers, contributors, bug reporters, users etc.) from the bug tracker of Apache[18]. For this small experiment, we only consider 27 random Apache projects and strictly matched their <*name,email*>(excluding the domain after "@") pair against all the developer <*name,email*>(excluding the domain after "@") pairs available in the database dump of Sourceforge. Names are not available for the developers in the database dump of Googlecode provided by FLOSSmole, so we consider matching only developer ID(s) in the case of Googlecode. The results of the experiment is shown in Table 1. The analysis may contain errors. The reason is that there may be different developers with identical <*name,email*>(excluding the domain after "@") pair exists in different forges. Also we have less confidence on our matching results against Googlecode because we only matched developer ID(s). A more thorough validation is required before interlinking developers across forges which is not in the current scope of this paper. We only wanted to highlight that there are overlapping across code forges and these overlapping can be made explicit by interlinking developers, projects etc., hence enabling an interlinked FLOSS ecosystem.

Forge A (developers)	**Forge B (developers)**	**Developers found**	**Overlap Ratio (%)**
Apache (29,860)	Sourceforge (256,516)	1,480	4.95
Apache (29,860)	Googlecode (203,691)	1,041	3.48
Googlecode (203,691)	Sourceforge (256,516)	16,351	8.02

Table 1: Overlapping between forges based on developers.

For our analysis, we only consider people who are involved in communication on the bug tracking systems of 27 different Apache projects. Few of them may not be the actual developers of those Apache projects. Thus we wanted to show that the developers who exists on Sourceforge or Googlecode are somehow contributing (bugs reporting, bugs commenting, patches, etc.) to the Apache projects and their contributions can be made explicit by interlinking the forges to each other.

5 Conclusions and Future Work

We have motivated and proposed a novel approach of integrating project's metadata across different code forges. We argue that a common model definition is required to enable integration across

[16] `http://rdfohloh.wikier.org/`

[17] Due to space limitations, please refer to http://ld2sd.deri.org/linkedfloss/sparql/example.html for an exemplary SPARQL query

[18] `https://issues.apache.org/bugzilla/`

different code forges. We proposed and supported through examples that Semantic Web technologies (Linked Data in particular) allow integrating knowledge not only across different forges but also to other relevant data sources available on the Web.

We have made some initial steps towards realizing this whole integration vision, although much more work has to be done. We plan to extend our modeling by incorporating more code forges and to produce a generic model which will cover almost all code forges. Additionally we improve the interlinking, yielding higher-quality links between the forges and to other relevant data sources. We also plan to host a SPARQL endpoint[19] which will allow research community to query knowledge from multiple forges and help them in developing new empirical studies based on integrated forge ecosystem. Finally, we will also integrate the projects analysis which are made available publicly by the FLOSSMetrics community by interlinking FLOSSMetrics and FLOSSmole RoRs.

References

[ABK+07] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[Con06] Megan Conklin. Beyond low-hanging fruit: Seeking the next generation in floss data mining. In *proceeedings of OSS*, pages 47–56, 2006.

[GBICS10] J.M. Gonzalez-Barahona, D. Izquierdo-Cortazar, and M. Squire. Repositories with public data about software development. *Int. J. Open Source Software and Processes*, 2(2):1–13, 2010.

[HB11] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space (1st edition). *Synthesis Lectures on the Semantic Web: Theory and Technology*, 1(1):1–136, 2011.

[HCC06] J. Howison, M. Conklin, and K. Crowston. Flossmole: A collaborative repository for floss research data and analyses. *International Journal of Information Technology and Web Engineering*, 1(3):17–26, 2006.

[KCM04] G. Klyne, J. J. Carroll, and B. McBride. Resource Description Framework (RDF): Concepts and Abstract Syntax). W3C Recommendation 10 February 2004, RDF Core Working Group, 2004.

[19] `http://semanticweb.org/wiki/SPARQL_endpoint`