<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Integrating Open Source Software Repositories on the Web through Linked Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Iqbal, A; Decker S</td>
</tr>
<tr>
<td><strong>Publication Date</strong></td>
<td>2015</td>
</tr>
<tr>
<td><strong>Publication Information</strong></td>
<td>Iqbal, A; Decker S (2015) Integrating Open Source Software Repositories on the Web through Linked Data IEEE International Conference on Information Reuse and Integration (IRI) San Francisco, CA,</td>
</tr>
<tr>
<td><strong>Item record</strong></td>
<td><a href="http://hdl.handle.net/10379/5444">http://hdl.handle.net/10379/5444</a></td>
</tr>
<tr>
<td><strong>DOI</strong></td>
<td><a href="http://dx.doi.org/10.1109/IRI.2015.27">http://dx.doi.org/10.1109/IRI.2015.27</a></td>
</tr>
</tbody>
</table>
Integrating Open Source Software Repositories on the Web through Linked Data

Aftab Iqbal
Insight Centre for Data Analytics
National University of Ireland, Galway (NUIG)
IDA Business Park, Lower Dangan, Galway, Ireland
Email: aftab.iqbal@insight-centre.org

Stefan Decker
Insight Centre for Data Analytics
National University of Ireland, Galway (NUIG)
IDA Business Park, Lower Dangan, Galway, Ireland
Email: stefan.decker@insight-centre.org

Abstract—In this paper, we propose a novel approach to the problem of integrating code forges based on metadata (e.g., programming language, database, intended audience, operating system), similar software projects and software developers. We review the current problems in integrating metadata of different code forges and argue that Semantic Web technologies are suitable for representing and integrating knowledge contained inside these code forges. Further, we show the advantages of interlinking metadata of software projects to other relevant data sources on the Web, which will enable querying more information from the Web. Moreover, we compute the overlapping between code forges based on similar software developers and argue the benefit of interlinking similar software developers and software projects across different code forges through examples.

Keywords—Linked Data; Semantic Web; FLOSS; Data Integration;

I. INTRODUCTION

With the success and adoption of Free/Libre Open Source Software (FLOSS) development, we have seen a tremendous growth in the availability and usage of different code forges [1]. Code forges provide different kinds of features in order to keep existing software projects and attract more software projects. These features include managing software project code, a place for the users to download the software project releases, discussion forums, bug trackers, mailing lists etc. A software project team may choose to host the software project on their own code repository or may choose to host the software project on one of the many available online code forges (such as SourceForge\(^1\), Savannah\(^2\), GitHub\(^3\) etc.). Each software project code repository along with other tools leave a detailed trace about the activities being carried out during the whole life span of the software project [2]. As pointed out by Conklin in [3], it is still surprisingly difficult to obtain and abstract information from these software repositories in order to answer even simple questions like:

- How many software developers are working on a software project?
- How big the community is surrounding a software project?
- How many contributors are contributing to a software project?
- What is the development ratio per software developer?
- Is the software project flourishing?

These are some of the many questions that are hidden deep inside the software repositories and usually have in the minds of software developers before joining or start contributing to a software project. Having answers to such questions can give a clear picture to the newcomers or other interested users/software developers of the software project. Much software development research has been carried out on gathering metrics and developing empirical studies based on the data retrieved from these software project repositories. However, researchers sometime finds 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 software project holds, the heterogeneity of the software projects being studied and harvesting or crawling the code forges, which is a daunting task. In order to provide the researchers an easy access to the software project’s data, two research software projects were initiated (with slightly different objective) by the open source software research community, which are FLOSSmole\(^4\) [4], [5] and FLOSSMetrics\(^5\) [6], also known as “repository of repositories (RoR)”. These RoRs were created to consolidate metadata (FLOSSmole) and analysis of software projects (FLOSSMetrics) 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 that are made available to download by the FLOSSmole community. Further, we only study Googlecode, GitHub and SourceForge metadata from the integration perspective in this paper, although our meth-

\(^1\)<http://sourceforge.net/>
\(^2\)<http://savannah.gnu.org/>
\(^3\)<https://github.com/>
\(^4\)<http://flossmole.org/>
\(^5\)<http://www.flossmetrics.org/>
ods extend to other code forges as well. The contribution of this paper is twofold: first we discuss the challenges faced while integrating metadata of different code forges. Later, we propose our approach to model code forges metadata using a common model and to represent software project’s metadata in a standard format. Further, we show the benefits of interlinking software project’s metadata to other data sources that are available on the Web as well as interlinking similar software projects across different code forges, using some example scenarios.

II. Motivation

FLOSSmole [5] crawls only meta information about software projects (e.g., software project name, description, url, no. of software developers, programming language, operating system, license) and software developers (e.g., software developer name, software developer ID, software projects) from different code forges and makes each code forge data accessible to the community as flat delimited files, SQL dumps or direct database access respectively. Although, the data extracted from each code forge is complete, cleaned and well described, integrating knowledge across different code forges is still a big challenge. Each code forge has its own database schema and represents data elements differently. Code forges sometimes use different terminologies or keywords while describing various categories (e.g., software project topics, operating systems, programming languages) for a particular software project. For example, Googlecode defines the term Mac to associate “Macintosh” as an operating system to the software project’s metadata but SourceForge defines it as OS X. Although both refers to the same operating system, different terms are used. Hence, we require methods and techniques to define some kind of classification/mapping, which explicitly states that the two terms are semantically similar and belong to the same operating system family. By doing so, one would be able to run a query on different code forges in order to retrieve a list of software projects that supports “Macintosh” operating system.

Associating metadata to a software project is also being handled differently in different code forges. For example, SourceForge provides a granular hierarchical structure for associating metadata to a software project by allowing software developers to select attributes from various category lists (e.g., operating systems, programming languages, database environments). Comparing it to Googlecode and GitHub, there is no proper hierarchical structure for associating metadata to a software project, which makes it harder to integrate the code forges based on similar metadata. Furthermore, the code forges do not share common semantics for associating attributes to a software project at the database schema level. For example, we cannot say that the database table column named “description” in SourceForge holds the same meaning as the database table column named “label” in Googlecode. The reason is column “label” hold values from various categories (e.g., programming languages, software project topics, Databases).

Further challenges upfront in integrating code forges as mentioned by [3] are: (1) code forges may have software projects with similar names. Large open source software projects may have sub-software projects hosted on different code forges or software projects may be hosted on multiple code forges exploiting different infrastructures. For example, Apache Software Foundation provides its own infrastructure to host Apache software projects but it also provide mirrors to various Apache software projects on GitHub for those software developers who prefer git versioning system over svn versioning system. (2) software developer identification in different code forges. With the existence of similar software projects on multiple code forges, software developers are often found on multiple code forges. However, software developer identification might be a problem because software developers may have used different IDs for different code forges, different software developers may have similar names/IDs across different code forges or software developers may have used similar IDs across different code forges.

Apart from the challenges upfront while integrating knowledge about software developers and software projects across different code forges, it may not be wrong to state that code forges are somehow interconnected but rather of an implicit nature (for example, software developers contributing to different software projects across different code forges). Hence, we need to make the interconnections among code forges explicit and further allow connecting to other relevant data on the Web.

Having such an explicit representation of the interconnections between different code forges, we will be able to support certain use case scenarios, some of which are listed in the following:

- **Retrieving complete history of a software project distributed across code forges.** Sometimes software projects are migrated from one code forge to another over the period of time. Linking similar software project across code forges will enable full development history of the software project across different code forges.
- **Tracing software developer activities across different code forges.** Linking software developer’s profile across different code forges will enable us to trace the activities of a software developer across code forges. It will further allows us to rank code forges based on the number of software projects he/she is working on in a particular code forge.
- **Is the popularity level of a particular code forge increasing or decreasing?** Assuming that code forges

6http://git-scm.com/
7http://subversion.apache.org/
are connected to each other, it would be interesting to investigate if software developers are migrating from one code forge to the other. For example, if software developers are considering hosting new software projects on Github rather than Googlecode or SourceForge. This use case will further lay down foundations to study open source software community dynamics across code forges.

- Assuming that the software 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 is associated as a programming language to a particular software project then the user would be able to get further information about Algol 68 from the Web (for example, from Wikipedia).

In order to deal with the interoperability and integration issues to realize different use case scenarios, we show modeling of code forges metadata based on a standard format in the following section.

III. MODELING CODE FORGE METADATA

Aforementioned, the usage of a common model and standard format to represent software project’s metadata from multiple code forges would allow 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 that contains extra information about a certain entity? Can it be done in an automated fashion? How easy will it be to explore related knowledge from different data sources?

In order to tackle these issues, we propose to use Semantic Web technologies to represent FLOSS data that exists in different code forges. As such, we propose to use RDF [7] (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 standard can be used to model hierarchical concept schemes. Finally, the integrated data can be published on the Web using Linked Data principles allowing third parties to discover and subsequently crawl the knowledge, and also allowing to interlink with background information available remotely on the Web. We refer the readers to [8] for details on how these standards would be used and put our focus on the use of SKOS as well as Linked Data to illustrate part of the modeling process.

We looked into SourceForge, Googlecode and Github’s hierarchical structure in order to identify the process of associating attributes to a software project and found that SourceForge is following an organized hierarchical structure of categorizing the software project’s attributes. Hence, we decided to do the modeling based on the hierarchical organization of software project attributes in SourceForge rather than Googlecode or Github. We examine the modeling and interlinking of code forges to other data sources in the following. An excerpt of an exemplary RDF representation of an operating system classification using SKOS vocabulary is shown in Listing 1.

```xml
<prefix base='http://www.algol68.org'/>
<prefix skos='http://www.w3.org/2004/02/skos/core'/>
<base:modern a skos:ConceptScheme;/>
<skos:prefLabel "Modern (Vendor-Supported) Desktop Operating System">en;
<skos:hasTopConcept base:linux;
<skos:hasTopConcept base:net-bud;
<skos:hasTopConcept base:io-os;
<skos:hasTopConcept base:mac;
<skos:hasTopConcept base:so;
<skos:hasTopConcept base:vista;
<skos:hasTopConcept base:windows;
<base:modern a skos:Concept;
<skos:inScheme base:modern;
<skos:prefLabel "Linux">en;
<skos:prefLabel "OS X">en;
<skos:prefLabel "Mac">en;
<skos:prefLabel "Windows Vista">en;
<skos:prefLabel "Most Recently Supported">en;
<skos:exactMatch http://en.wikipedia.org/wiki/Linux>
<skos:exactMatch http://en.wikipedia.org/wiki/Mac>
<skos:exactMatch http://en.wikipedia.org/wiki/Mac>
```

Listing 1. An exemplary SKOS concept scheme defining different operating systems for SourceForge.

The SKOS vocabulary is used to aggregate concepts/terminologies into a single concept scheme. For example, in Listing 1 we have defined some operating systems under one concept scheme. The benefit of using SKOS vocabulary is that the concept schemes are easily extensible. Therefore, if a code forge later adds a new operating system, which is not listed in the existing concept scheme then it can be added easily (see Listing 2, line #3 and lines #5–8). Moreover, a code forge can define its own SKOS concept scheme and link it to the core SKOS concept scheme using skos:inScheme property. This approach allows to keep all predefined operating systems from different code forges under one scheme.

```xml
<base:modern a skos:ConceptScheme;/>
<skos:hasTopConcept base:vista;
<skos:hasTopConcept base:net-bud;
<skos:hasTopConcept base:io-os;
<skos:hasTopConcept base:mac;
<skos:hasTopConcept base:windows;
<skos:hasTopConcept base:linux;
<skos:hasTopConcept base:modern;
<skos:prefLabel "Windows Vista">en;
```

Listing 2. Adding a new concept to the existing SKOS concept scheme.

Referring to the previous example (cf. Section II) of using different terminologies for the same concept across code forges, SKOS vocabulary offers to use skos:exactMatch property in order to express that the terms Mac and OS X are semantically similar.

8http://www.algol68.org/
9http://en.wikipedia.org/wiki/ALGOL_68
10http://www.w3.org/2004/02/skos/
11http://www.w3.org/DesignIssues/LinkedData.html

12We encourage readers to have a look at the concept schemes, which are available at: http://srvgal85.deri.ie/linkedfloss/schemes/
We have shown modeling of code forges based on metadata, the next is to interlink metadata to other data sources on the Web. Referring back to Listing 2, we know that “Windows Vista” is an operating system and it would be beneficial if we interlink this concept to other relevant data source on the Web (for example, to the Wikipedia entry\(^\text{13}\)). By interlinking the two data sources together, we will be able to retrieve information about “Windows Vista” from Wikipedia. In order to extract structured information from Wikipedia, the research community has developed DBpedia [9]. DBpedia allows to ask queries against Wikipedia, and to link other datasets on the Web to the Wikipedia data. We can interlink the concepts defined (e.g., base:vista in this example) to the corresponding DBpedia entity using an owl:sameAs property indicating that these URIs actually refer to the same entity (see Listing 3).

Now that we have defined the metadata of code forges using SKOS vocabulary and further provided an example of interlinking metadata to other relevant data sources on the Web, the next step is to use the meta information while publishing RDF description of software projects that are contained inside FLOSSmole database dumps. Due to the large number of software projects contained inside FLOSSmole database dumps, we have developed custom written scripts to automatically publish each software project as RDF statements based on the information provided by FLOSSmole. We show RDF representation of 1 software project taken from the SourceForge database dump, as an example in Listing 4. We use different RDF vocabularies to describe meta information of software projects as RDF statements. doap (description of a project) vocabulary is specifically designed to describe meta information of a software project, therefore we use various properties from doap vocabulary to describe information of a software project as RDF statements. Moreover, we use foaf vocabulary to describe meta information of a software developer as shown in Listing 4. As each code forge has its own database schema, the underlying RDF description for software projects of each code forge slightly differs from each other.

### IV. INTERLINKING CODE FORGES: PRELIMINARY FINDINGS

In this section, we will argue the benefit of interlinking code forges to each other and to other potential data sources on the Web through examples. Before that, we show the total number of software projects and software developers that exist in the FLOSSmole database dumps of different code forges in Table I.

<table>
<thead>
<tr>
<th>Code Forge</th>
<th>Software Projects</th>
<th>Software Developers</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceForge</td>
<td>204439</td>
<td>256316</td>
</tr>
<tr>
<td>Googlecode</td>
<td>107286</td>
<td>203691</td>
</tr>
<tr>
<td>GitHub</td>
<td>155326</td>
<td>119196</td>
</tr>
</tbody>
</table>

Table I: Total number of software projects and software developers found in each code forge based on FLOSSmole database dump.

Once the software projects and software developers of each code forge under consideration are published as RDF datasets using our custom written scripts, we loaded the RDF datasets into our public SPARQL endpoint\(^\text{14}\). We discussed previously about interlinking code forges metadata to their relevant terms in DBpedia. Therefore, we show the total number of owl:sameAs links that are established between DBpedia and relevant code forges metadata terms that are defined using SKOS vocabulary in Table II.

<table>
<thead>
<tr>
<th>Code Forge</th>
<th>owl:sameAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceForge</td>
<td>542</td>
</tr>
<tr>
<td>Googlecode</td>
<td>382</td>
</tr>
<tr>
<td>GitHub</td>
<td>123</td>
</tr>
</tbody>
</table>

Table II: Interlinking code forge metadata terms with DBpedia dataset.

The owl:sameAs links were created by manually checking the terms in a code forge and their corresponding entry

\(^\text{13}\)http://en.wikipedia.org/wiki/Windows_Vista

\(^\text{14}\)http://linkedcodeforges.srvgal85.deri.ie/sparql
in DBpedia. However, duplicate detection algorithms and frameworks like Silk [10], Swoosh [11], Duke etc., could also be taken into account for establishing the owl:sameAs links but it is not in the current scope of this work. The results (cf. Table II) shows that SourceForge has defined a variety of terms to associate metadata to a software project in contrast to Googlecode and GitHub. We have done interlinking with DBpedia dataset only however other potential LOD datasets\(^\ast\) can also be taken into account to enrich the interlinking between code forges metadata and LOD datasets. This is particularly beneficial in querying relevant information about a particular entity from multiple data sources. For example, a software project is written in Java programming language and its meta information is interlinked with DBpedia using owl:sameAs property, one could query DBpedia to find more information about Java programming language. For example:

- who is the designer/software developer?
- when was it created?
- what is the license type?
- what operating system does it support?

Another benefit of interlinking is the ease of querying software projects across code forges that comes under a specific category. For example, a simple query is to find out the total number of software projects hosted on different code forges that are implemented in a specific programming language. We consider 5 programming languages and show the number of software projects that are implemented using these programming languages on different code forges, in Table III. The results shown in Table III can be further narrowed down by specifying other meta information in the query (e.g., audience, database, operating system).

![Table III](image)

**Table III**

<table>
<thead>
<tr>
<th>Programming Language</th>
<th>Googlecode (software projects)</th>
<th>GitHub (software projects)</th>
<th>SourceForge (software projects)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript</td>
<td>19,760</td>
<td>22,331</td>
<td>8,975</td>
</tr>
<tr>
<td>Ruby</td>
<td>2,163</td>
<td>28,655</td>
<td>1,563</td>
</tr>
<tr>
<td>Delphi</td>
<td>418</td>
<td>135</td>
<td>3,023</td>
</tr>
<tr>
<td>MatLab</td>
<td>393</td>
<td>181</td>
<td>441</td>
</tr>
<tr>
<td>Assembly</td>
<td>182</td>
<td>196</td>
<td>2,058</td>
</tr>
</tbody>
</table>

A. Interlinking Software Project to Other Relevant Data Resources

In this subsection, we provide an example of interlinking a software project hosted on a code forge to its existence in other data sources on the Web. For example, Ohloh\(^\d\) (now known as OpenHUB) is a free wiki for open source software and people. Ohloh provides analytical information and services about open source software projects by crawling data from a variety of code forges. An RDF wrapper\(^\d\), a.k.a., RDFohloh [12], has developed by the Semantic Web research community, which provides Linked Data version of Ohloh.

The RDF description of a software project generated by RDFohloh can be interlinked to its relevant software project’s RDF description generated from the code forge using an owl:sameAs property. This interlinking will allow anyone to retrieve statistical and other relevant information about a software project or software developer from Ohloh as shown in the example below.

Let’s take the example of FileZilla software project hosted on SourceForge and its statistical information is also available at Ohloh. Let’s assume that we have the metadata of FileZilla software project available in RDF (cf. Listing 4) and the RDF description of FileZilla software project from Ohloh is made available using RDFOhloh\(^\d\). Assuming that both RDF datasets are interlinked based on software developer IDs, one can query the statistical information about FileZilla software project, geo location of a software developer, total number of commits made by a software developer, software developer’s kudo rank and other software projects he/she is contributing to etc., from Ohloh. An excerpt of an exemplary SPARQL query retrieving information about a FileZilla software developer from Ohloh is show in Listing 5.

```
PREFIX doap: <http://usefulinc.com/ns/doap#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX ohloh: <http://rdfohloh.wikier.org/ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>

SELECT ?name ?kudorank ?lat ?long
WHERE {
  ?prj doap:name "FileZilla" .
  ?prj doap:developer ?d .
  ?geolocation geo:lat ?lat .
}
```

Listing 5. SPARQL query that retrieves profile information about a SourceForge software developer from Ohloh.

The result returned by the query listed in Listing 5 is shown in Table IV. By interlinking a software project hosted on a code forge with Ohloh [13], we enable to track not only the software developer’s development activity on that particular software project by mining information hidden deep inside the software repositories but also can get statistical information about software developer’s development activity.

B. Software Developers Overlapping Across Code Forges

We now perform a preliminary study to identify the overlap across different code forges based on similar software

\(^\d\)http://code.google.com/p/duke/
\(^\d\)http://lod-cloud.net/
\(^\d\)http://www.ohloh.net/
developers. For this preliminary study, we also considered 27 random Apache software projects and extracted 29,860 distinct <name,email> pairs of people (i.e., software developers, contributors, bug reporters, users etc.) from the bug repository of Apache Software Foundation. We strictly matched their <name,email> (excluding the domain after “@”) pairs against all software developer <name,email> (excluding the domain after “@”) pairs available in the database dump of SourceForge. The database dump of Googlecode provided by FLOSSmole does not contain naming information about software developers, therefore we considered matching only software developer IDs in the case of Googlecode. The results of the study is shown in Table V. The analysis may contain errors because there may be different software developers with identical <name,email> (excluding the domain after “@”) pair exists in different code forges. Furthermore, we have less confidence on our matching results against Googlecode because we only matched software developer IDs. A more thorough validation is required before interlinking software developer IDs across code forges, which is not in the current scope of this work. However, our preliminary study shows that there are overlappings across different code forges and these overlappings can be made explicit by interlinking software developers and software projects, hence enabling an interlinked FLOSS ecosystem.

<table>
<thead>
<tr>
<th>Forge A (software developers)</th>
<th>Forge B (software developers)</th>
<th>Developers Found</th>
<th>Overlap Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache (29,860)</td>
<td>Sourceforge (256,516)</td>
<td>1,480</td>
<td>4.95</td>
</tr>
<tr>
<td>Apache (29,860)</td>
<td>Googlecode (203,691)</td>
<td>1,041</td>
<td>3.48</td>
</tr>
<tr>
<td>Googlecode (203,691)</td>
<td>Sourceforge (256,516)</td>
<td>16,351</td>
<td>8.02</td>
</tr>
</tbody>
</table>

Table V
OVERLAPPING BETWEEN CODE FORGES BASED ON SOFTWARE DEVELOPERS.

For this particular study, we considered only those people who are involved in communication on the bug tracking system of 27 different Apache software projects. Most of them may not be the actual software developers of those Apache software projects. However, we wanted to show that the software developers who exists on SourceForge or Googlecode are somehow contributing (i.e., bugs reporting, bugs commenting, source code patches, etc.) to the Apache software projects or part of Apache software projects too and their contributions can be made explicit by interlinking them.

V. DISCUSSION

In the previous section, we discussed briefly the implicit connection between code forges based on software developers existence in multiple code forges. Further, we showed the advantage of interlinking software projects to other relevant data sources (e.g., Ohloh) on the Web, hence allowing to retrieve more information relevant to a software project or software developer. In the following, we further support the idea of interlinking software projects across different code forges by presenting a use case. We emphasize that we do not propose any heuristics or algorithm for efficient software project matching but instead advocate the benefits of interlinking code forges through an example.

UseCase: Interlinking similar software projects across multiple code forges:

In order to identify the existence of similar software projects across code forges, different heuristics could be adopted as proposed by [14]. For example, software project’s metadata (e.g., name, description, URL, license, operating system, programming language) can be taken into account to match software projects across code forges. However, we take into account only the software project URL as a matching attribute in order to identify similar software projects across code forges. Our simple matching approach is explained in the following:

Software projects hosted on Googlecode typically have URL of the form: http://code.google.com/p/\(x\), where \(x\) is the name of the software project. We query all software projects hosted on SourceForge that contains Googlecode software project URLs. For all those software projects where Googlecode software project URLs are found, we establish an owl:sameAs link between the URIs of both software projects. For an example, we consider Quadra software project, which is a multiplayer action puzzle game and is hosted on SourceForge and Googlecode. Assuming that the software projects of SourceForge and Googlecode are available as RDF datasets, we link both software projects using an owl:sameAs property as shown in Listing 6.

```
prefix owl: <http://www.w3.org/2002/07/owl#> .
prefix sourceforge: <http://example.org/linkedfloss/googlecode/project/quadra> .
prefix linkedfloss: <http://example.org/linkedfloss/sourceforge/project/quadra> .
```

Listing 6. An interlinking example connecting similar software project on Googlecode and SourceForge.

Matching software projects hosted on SourceForge against software projects in other code forges is achieved in a slightly different way. SourceForge usually have two different URLs for a single software project. One

22https://issues.apache.org/bugzilla/
23https://code.google.com/p/quadra/
24http://sourceforge.net/projects/quadra/
and exploiting software project-related information across code forges but to show the benefits of interlinking matching attributes, which are mentioned by [14]. However, the number of matching candidates if we take into account other hosted on different code forges where ple,

software projects based on software project code forge (cf. Table I). However, one advantage of matching projects based on the software project may exist on existence of which is the reason we do not provide any results of the does not have any information for software project The popularity of increase in the preference of software developers working that are found on GitHub. These approaches in a pairwise manner on Googlecode, GitHub, SourceForge code forges. The results achieved through matching software projects based on URL across different code forges are shown in Table VI.

<table>
<thead>
<tr>
<th>Source Forge</th>
<th>Code Forge</th>
<th>Target Code Forge</th>
<th>Projects Matched</th>
</tr>
</thead>
<tbody>
<tr>
<td>Googlecode</td>
<td>SourceForge</td>
<td>177</td>
<td></td>
</tr>
<tr>
<td>Googlecode</td>
<td>GitHub</td>
<td>595</td>
<td></td>
</tr>
<tr>
<td>SourceForge</td>
<td>GitHub</td>
<td>699</td>
<td></td>
</tr>
<tr>
<td>GitHub</td>
<td>SourceForge</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

Table VI 

Matching similar software projects across different code forges based on their URLs.

The results in Table VI shows existence of good number of Googlecode and SourceForge software projects that are found on GitHub in contrast to GitHub software projects that are found on SourceForge. This can be due to the increase in the preference of software developers working in a GitHub environment or an indication of the growing popularity of GitHub over Googlecode and SourceForge. The Googlecode database dump (provided by Flossmole) does not have any information for software project URL, which is the reason we do not provide any results of the existence of SourceForge and GitHub software projects that may exist on Googlecode. The number of matched software projects based on the software project URL is relatively low compared to the number of software projects hosted on each code forge (cf. Table I). However, one advantage of matching software projects based on software project URL is that it also matches those software projects where the software project names are different across code forges. For example, libxls25 and libxls_cmake26 are same software projects hosted on different code forges where libxls_cmake contains software project URL of libxls. We may achieve a high number of matching candidates if we take into account other matching attributes, which are mentioned by [14]. However, our goal in this paper is to not increase the number of links across code forges but to show the benefits of interlinking and exploiting software project-related information across different code forges.

Given that similar software projects are interlinked across different code forges through owl:sameAs property, we are able to extract a list of software developers who are working on the same software project across different code forges as shown in Listing 7.

Listing 7. SPARQL query that retrieves a list of software developers working on the same software project (i.e., quadra) across different code forges.

```
PREFIX doap: <http://usefulinc.com/ns/doap#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?developers {
  ?prj a doap:Project .
  {?prj doap:name "quadra" .}
  {?prj owl:sameAs ?prjAlt .} UNION {?prjAlt owl:sameAs ?prj .}
  {?prjAlt doap:developer ?developers .}
}
```

An interesting fact we see in Table VII is the existence of similar software developers in multiple code forges due to the existence of similar software project. We see that some software developers (i.e., pphaneuf, rveilleux and slajoie) in Table VII holds account IDs on both code forges (i.e., Googlecode and SourceForge). Therefore, existence of software developers in multiple code forges can be exploited by interlinking their respective IDs.

**VI. Conclusion and Future Work**

We have motivated and proposed a novel approach of integrating code forges based on software project’s metadata, similar software projects and software developers. We highlighted the issues in integrating metadata of different code forges and argue that a common model definition is required to enable integration across different code forges. We proposed and supported with examples that Semantic Web technologies (Linked Data in particular) allow integrating knowledge not only across different code forges but also to other relevant data sources available on the Web. We considered only SourceForge, Googlecode and GitHub as an

---

25 http://sourceforge.net/softwareprojects/libxls/
26 https://github.com/wiglot/libxls_cmake
example for modeling, however there are several other code forges available on the Web. Therefore, one potential future work direction is to extend the common model definition by taking into account other code forges.

We have made some initial steps towards realizing this whole integration vision, although much more work needs to be done. We have demonstrated interlinking software projects across different code forges but not much information can be queried due to the limited data available for each code forge, which is provided by the FLOSSmole community. FLOSSmole provides only meta information about software projects, therefore we are not able to query the artifacts (i.e., bug, source control commit logs, emails etc.) underlying software repositories in order to discover valuable information. However, if in the future FLOSSmole publish artifacts level information of every software project for different code forges then it can be easily converted to RDF datasets and further integration can be achieved [15].

In order to demonstrate the real benefits of integrating code forges (based on software projects and software developers information) to each other, we are required to extract detailed information relevant to a software project that are hidden deep inside software repositories of code forges. This would allow interested users to find answers to various questions:

- how much percentage of development is carried out on a software project across different code forges?
- what is the contribution of a software developer in different software projects across code forges?
- how big the community is around a particular software project across different code forges?

ACKNOWLEDGMENT

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

REFERENCES


