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[6] over svn versioning system[7]. (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

[6]http://git-scm.com/

[7]http://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**[8] 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*[9]).

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[10] standard can be used to model hierarchical concept schemes. Finally, the integrated data can be published on the Web using Linked Data priniciples[11] 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.

[8]http://www.algol68.org/

[9]http://en.wikipedia.org/wiki/ALGOL_68

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

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

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[12].

```
@prefix base: <http://srvgal85.deri.ie/linkedfloss/schemes/os#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .

base:modern a skos:ConceptScheme;
skos:prefLabel "Modern (Vendor-Supported) Desktop Operating Systems"@en;
skos:hasTopConcept base:linux;
skos:hasTopConcept base:net-bsd;
skos:hasTopConcept base:os-x
.
base:linux a skos:Concept;
skos:inScheme base:modern;
skos:prefLabel "Linux"@en
.
...
```

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.

```
base:modern a skos:ConceptScheme;
...
skos:hasTopConcept base:vista;
...
base:vista a skos:Concept;
skos:inScheme 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.

[12]We encourage readers to have a look at the concept schemes, which are available at: http://srvgal85.deri.ie/linkedfloss/schemes/

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

Listing 3. An interlinking example connecting an operating system to its relevant entity in DBpedia.

We have shown modeling of code forges based on metadata, the next is to interlink the 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[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

[13]http://en.wikipedia.org/wiki/Windows_Vista

```
@prefix doap: <http://usefulinc.com/ns/doap#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<http://example.org/linkedfloss/sourceforge/project/filezilla> a doap:Project;
dc:title "FileZilla";
doap:created "2001-02-27T00:00:00.0";
doap:description "FileZilla is a cross-platform graphical FTP, FTPS and SFTP
      client a lot of features, supporting Windows, Linux, Mac OS X and more.
      FileZilla Server is a reliable FTP server for Windows.";
doap:developer <http://example.org/linkedfloss/sourceforge/author/bog>,
        <http://example.org/linkedfloss/sourceforge/author/civ>,
        <http://example.org/linkedfloss/sourceforge/author/ean>;
doap:homepage <http://sourceforge.net/software projects/filezilla>;
doap:license <http://www.gnu.org/licenses/gpl-1.0.html>;
foaf:homepage <http://filezilla.sourceforge.net>;
dcterms:subject <http://example.org/linkedfloss/schemes/audience#desktop>,
        <http://example.org/linkedfloss/schemes/audience#developers>,
        <http://example.org/linkedfloss/schemes/topic#social_networking>,
        <http://example.org/linkedfloss/schemes/translation#french>,
        <http://example.org/linkedfloss/schemes/translation#english> .

<http://example.org/linkedfloss/sourceforge/author/bog> a foaf:Person;
foaf:accountName "bog";
foaf:mbox_sha1sum "9080a9d64e29e35892c9517bd95e2e8e4f5b13f6";
foaf:name "Tim" .
...
```

Listing 4. RDF description of a SourceForge software project based on the data provided by FLOSSmole.

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.

Code Forge	Software Projects	Software Developers
SourceForge	204439	256516
Googlecode	167286	203691
GitHub	155326	119196

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[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.

Code Forge	owl:sameAs
SourceForge	542
Googlecode	382
GitHub	123

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

[14]http://linkedcodeforges.srvgal85.deri.ie/sparql

in *DBpedia*. However, duplicate detection algorithms and frameworks like Silk [10], Swoosh [11], Duke[15] 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[16] 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*).

Programming Language	***Googlecode*** (*software projects*)	***GitHub*** (*software projects*)	***SourceForge*** (*software projects*)
JavaScript	19,760	22,331	8,975
Ruby	2,163	28,655	1,563
Delphi	418	135	3,023
MatLab	393	181	441
Assembly	162	196	2,058

Table III
SOFTWARE PROJECTS DEVELOPED USING DIFFERENT PROGRAMMING LANGUAGES ON CODE FORGES.

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*[17] (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[18], 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[19] hosted on *SourceForge* and its statistical information is also available at *Ohloh*[20]. 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*[21]. 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 sioc: <http://rdfs.org/sioc/ns#>
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 {
       ?prj doap:name "FileZilla" .
       ?prj doap:developer ?d .
       ?d owl:sameAs <http://rdfohloh.wikier.org/user/19727#person> .
       ?dev foaf:name ?name .
       ?dev ohloh:kudo-rank ?kudorank .
       ?f foaf:holdsAccount ?dev .
       ?f foaf:based_near ?geolocation .
       ?geolocation geo:lat ?lat .
       ?geolocation geo:long ?long .
}
```

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

[15]http://code.google.com/p/duke/
[16]http://lod-cloud.net/
[17]http://www.ohloh.net/
[18]http://rdfohloh.wikier.org/
[19]https://sourceforge.net/softwareprojects/filezilla/
[20]http://www.ohloh.net/p/filezilla
[21]http://rdfohloh.wikier.org/project/FileZilla.rdf

Name	Kudorank	Lat	Long
Tim	10	39.234651	11.432376

Table IV
GEO LOCATION AND RANK OF A SOFTWARE DEVELOPER RETRIEVED FROM OHLOH.

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*[22]. 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.

Forge A (software developers)	**Forge B** (software 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 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.

[22]https://issues.apache.org/bugzilla/

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*[23] and *Googlecode*[24]. 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.

```
1 @prefix owl: <http://www.w3.org/2002/07/owl#> .
2
3 <http://example.org/linkedfloss/googlecode/project/quadra> owl:sameAs
4 <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

[23]https://code.google.com/p/quadra/
[24]http://sourceforge.net/projects/quadra/

URL refers to the actual software project directory where software repositories, software project's metadata and software developer's information are available (e.g., *http://sourceforge.net/projects/x*), while the other URL refers to the HTML website describing the software project (e.g., *http://x.sourceforge.net/*). The software development teams may use any of the two URLs as a homepage for the software projects that are hosted on other code forges. Therefore, we match software projects based on both URLs. We apply 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.

Source Code Forge	Target Code Forge	Projects Matched
Googlecode	*SourceForge*	177
Googlecode	*GitHub*	595
SourceForge	*GitHub*	699
GitHub	*SourceForge*	25

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 comparing 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, *libxls*[25] and *libxls_cmake*[26] 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.

```
1 PREFIX doap: <http://usefulinc.com/ns/doap#>
2 PREFIX owl: <http://www.w3.org/2002/07/owl#>
3
4 select ?developers {
5        ?prj a doap:Project .
6        ?prj doap:name "quadra" .
7        {?prj owl:sameAs ?prjAlt .} UNION {?prjAlt owl:sameAs ?prj .}
8        ?prjAlt doap:developer ?developers .
9 }
```

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

The results of the SPARQL query listed in Listing 7 are shown in Table VII. The results in Table VII shows that four software developers are found on *Googlecode* developing the "*quadra*" software project while other software developers are found on *SourceForge*.

Software Developers
http://example.org/linkedfloss/googlecode/author/pphaneuf
http://example.org/linkedfloss/googlecode/author/rveilleux
http://example.org/linkedfloss/googlecode/author/slajoie
http://example.org/linkedfloss/googlecode/author/dgryski
http://example.org/linkedfloss/sourceforge/author/pphaneuf
http://example.org/linkedfloss/sourceforge/author/rveilleux
http://example.org/linkedfloss/sourceforge/author/stanb
http://example.org/linkedfloss/sourceforge/author/nypon
http://example.org/linkedfloss/sourceforge/author/pihvi
http://example.org/linkedfloss/sourceforge/author/slajoie
http://example.org/linkedfloss/sourceforge/author/plombe
http://example.org/linkedfloss/sourceforge/author/roncli

Table VII
LIST OF SOFTWARE DEVELOPERS WORKING ON THE SAME SOFTWARE PROJECT ACROSS DIFFERENT CODE FORGES.

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

[1] M. Squire and D. Williams, "Describing the software forge ecosystem," in *Proceedings of the 2012 45th Hawaii International Conference on System Sciences*, ser. HICSS '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 3416–3425. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2012.197

[2] J. Gonzalez-Barahona, D. Izquierdo-Cortazar, and M. Squire, "Repositories with public data about software development," *Int. J. Open Source Software and Processes*, vol. 2, no. 2, pp. 1–13, 2010.

[3] M. Conklin, "Beyond low-hanging fruit: Seeking the next generation in floss data mining." in *proceeedings of OSS*, 2006, pp. 47–56.

[4] J. Howison and M. S. Conklin, "Ossmole: A collaborative repository for floss research data and analyses," in *1st International Conference on Open Source Software*, Genova, Italy, 2005.

[5] 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*, vol. 1, no. 3, pp. 17–26, 2006.

[6] I. Herraiz, D. Izquierdo-Cortazar, and F. Rivas-Hernández, "Flossmetrics: Free/libre/open source software metrics," in *CSMR*, 2009, pp. 281–284.

[7] G. Klyne, J. J. Carroll, and B. McBride, "Resource Description Framework (RDF): Concepts and Abstract Syntax)," http://www.w3.org/TR/rdf-concepts/, RDF Core Working Group, W3C Recommendation 10 February, 2004.

[8] T. Heath and C. Bizer, "Linked data: Evolving the web into a global data space (1st edition)," *Synthesis Lectures on the Semantic Web: Theory and Technology*, vol. 1, no. 1, pp. 1–136, 2011.

[9] 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*, ser. ISWC'07/ASWC'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 722–735. [Online]. Available: http://dl.acm.org/citation.cfm?id=1785162.1785216

[10] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov, "Silk a link discovery framework for the web of data," in *2nd Workshop about Linked Data on the Web (LDOW2009)*, Madrid, Spain, 2009.

[11] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom, "Swoosh: a generic approach to entity resolution," *The VLDB Journal*, vol. 18, no. 1, pp. 255–276, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1007/s00778-008-0098-x

[12] S. Fernández, "Rdfohloh, a rdf wrapper of ohloh," in *1st workshop on Social Data on the Web (SDoW2008), co-located with ISWC2008, Karlsruhe, Germany*, 2008.

[13] A. Iqbal and M. Hausenblas, "Integrating developer-related information across open source repositories," in *IEEE 13th International Conference on Information Reuse and Integration (IRI)*, 2012.

[14] M. Conklin, "Project entity matching across floss repositories," in *3rd International Conference on Open Source Systems*, Springer. Limerick, IE: Springer, 2007, pp. 45–57.

[15] A. Iqbal, O. Ureche, M. Hausenblas, and G. Tummarello, "LD2SD: Linked Data Driven Software Development," in *Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering (SEKE 2009)*, Boston, USA, 2009.