Integrating Developer-related information across Open Source Repositories

Aftab Iqbal and Michael Hausenblas
Digital Enterprise Research Institute (DERI)
National University of Ireland, Galway (NUIG)
IDA Business Park, Galway, Ireland
Email: {aftab.iqbal , michael.hausenblas}@deri.org

Abstract

Software developers use various software repositories in order to interact with each other or to solve software related problems. They are required to adopt an identity for each of the software repositories they wanted to use. Quite often developers are also found on different code forges developing open source projects. It is worth mentioning that the information relevant to the developers are distributed on the Web among different data sources each requires an ID and an authentication mechanism. In this paper, we propose to interlink the identities of a developer across different data sources on the Web. Further, we show the benefit of integrating developer-related information from different data sources using some real-world scenarios.

1. Introduction and Motivation

Software developers use tools with underlying repositories to support the collaboration of distributed software development. In order to interact with the many software repositories that are part of an open-source project, software developers usually require to adopt an identity for each repository. Often, they use multiple identities for the same repository [10]. Research has shown that these software repositories contain rich amount of information about software projects. By mining the information contained in these software repositories, practitioners can depend less on their experience and more on the historical data [5]. However, software repositories are commonly used only as record-keeping repositories and rarely for design decision processes [4]. 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 developers are working on the project? How big the community is surrounding a project? How many contributors are contributing to the project? What is the development ratio per developer? Is the project flourishing? and many more. These are few of the many questions which are hidden deep in the software repositories and developers usually have in mind before joining or start contributing to a project. Having answers to such questions can give a clear picture to the newcomers or other interested users/developers of the project.

Apart from mining information from software repositories in order to answer questions which are mentioned above, there also exist analytic services which provides detailed analysis on different open source projects. One good example of such a service is Ohloh[1]. Ohloh is a free, public software directory which monitors up-to-date development activity of open source projects. Ohloh allows software developers to join (i.e., adopt an identity) and claim their commits on existing projects and also add projects not yet on Ohloh, in order to assemble a complete profile of their open source project contributions. Ohloh provides several types of information about an open source project. For example, it provides detailed analysis of per developer commit ratio to the project, per programming language commit ratio to the project, longevity of the project and software metrics such as total lines of source code, commit statistics, comment ratio etc. Other global statistics like programming-language usage are also provided. Ohloh provides all types of information which a user, developer or project manager is interested or keen to know. At the time of writing this paper Ohloh indexed `552,103` open source projects connecting more than `1,534,084` open source developers/-contributors making it a valuable data source for collecting up-to-date metrics about open source projects.

With the success and adoption of open source software development, we have seen a tremendous growth in the availability of different code forges. Different forges provide different kinds of features in order to keep existing projects and attract more projects. Because of this, an open source project sometimes developed at multiple code forges. For example, Apache Software Foundation (ASF)[2]

[1] `http://www.ohloh.net`
[2] `http://apache.org`

manage projects at their own infrastructure but also provide github mirrors for developers who prefer git[3] over svn[4] versioning system. Therefore certain developers use the GitHub[5] infrastructure to contribute to the development of Apache projects. At the time of writing this paper, ASF host mirrors of approximately 320 different Apache projects on GitHub[6]. Further, an open source project sometimes migrate between different code forges during its entire time period. These code forges also require projects and developers to adopt an identity in order to host a project and to keep track of developers development activity. Eventually, developers end up in developing multiple projects in different code forges. For example, Stefan Bodewig[7] who is a member of ASF and is contributing to many Apache projects[8], has also developed few projects on GitHub[9] and SourceForge[10]. Hence, the history of open source project development and developer's contribution to different open source projects are distributed across multiple code forges.

It is worth mentioning that there is an implicit connection between the developer's activity in different software repositories (i.e., mailing lists, bug tracking systems, source control etc.) hosting a particular project, project development statistics (available via Ohloh), activity on the social media platforms and involvement in multiple projects on different code forges. The developers are required to adopt an identity for each of the repositories they wanted to use. For example, they are required to adopt an email address in order to send an email to the project mailing list, adopt an ID to interact with others on social media platforms, adopt an ID on a particular code forge, adopt an ID to push commits to the source code repository, adopt an ID to comment on the bug in a bug tracking system etc. Each repository implemented its own proprietary ID management in order to authenticate developers to log on and its own proprietary user profile system to manage information about developers. Hence the information relevant to developers are distributed on the Web among different data sources (i.e., social platforms, code forges, software repositories etc.). The different types of identities developer adopts are OpenID[11], WebID[12], email address etc. We need hence not only make the interconnection between developer identities among different software repositories within a project explicit but also allow connecting it to other related data sources available on the Web. Having such an explicit representation of the interconnection between the data sources available we will be able to support certain scenarios often found in the software development process:

1. **Synthesis Scenarios**
   - A developer could effectively query the co-developers activities in different software repositories of the project.
   - A developer could learn about the expertise of co-developers in different programming languages.
   - A developer could easily track the contribution of co-developers in different projects.
2. **Analysis Scenarios**
   - Different programming languages used in the project and the ratio of commits relevant to each programming language.
   - Development ratio of a project across multiple code forges.
   - Developer's contribution statistics on each project.

The contribution of this paper is twofold: first, we identified the different types of identities which developers are using in different data sources and provide a simple yet effective approach to interlink identities of the same developer found in different data sources. It will enable developers and other interested users to not only query facts which are hidden deep inside the software repositories but also allow to query development statistics as well as development activity of a developer across multiple code forges. Further, we show different use case scenarios which can be easily addressed by integrating data from multiple data sources.

The paper is structured as follows: in Section 2 we present use cases which describes the benefit of data integration from multiple sources. Then, we introduce the overall architecture of data extraction, transformation and interlinking process in Section 3. We report on exemplary queries and their results in Section 4. In Section 5 we review related work and compare it to our approach. Finally, we conclude in Section 6 and outline future steps.

2. Use Cases

In the following we describe real-world scenarios from the software development domain that can benefit from our methodology. By establishing the identity of developers throughout two or more data sources on the Web (for example a code forge, social media platform and Ohloh) we can integrate the necessary information to meet the requirements of the following, non-exhaustive list of application scenarios:

[3] http://git-scm.com/
[4] http://subversion.apache.org/
[5] https://github.com
[6] https://github.com/apache
[7] http://stefan.samaflost.de/
[8] http://people.apache.org/committer-index.html#bodewig
[9] https://github.com/bodewig
[10] http://sourceforge.net/users/bodewig
[11] http://openid.net/
[12] http://www.w3.org/wiki/WebID

Identifying a potential contributor Ryan is the initiator of an open source project dealing with dynamic Web content. He would like to extend the codebase and is looking for one or more developers he can approach based on a profile he defines as: "Must be familiar with HTTP and REST and should have at least five years experience in back-end development. Working experience with JavaScript is a plus". How can Ryan, based on information from GitHub, Geekli.st[13], Twitter and the project mailing list, find appropriate candidates he can approach?

Supporting team changes Mary, a developer for a software company, has to relocate in the middle of a project. Julie, her supervisor, needs to hire someone who can replace Mary. Julie wants to analyse Mary's expertise and recent activities: assigned bugs, committed code, mailing list and blog posts, etc. What Julie ultimately wants is to enable the new hire to hit the ground running, making the new team member as productive as possible from day one by benefitting from Mary's experience.

Selecting a project for corporate sponsorship The board of VOZ, a big, multinational company has identified an opportunity to strategically (that is, both in-kind and financially) sponsor an open source project dealing with a MapReduce implementation. Ken, a new VP for this area, is responsible to suggest an open source project to the board. Ken has access to the code repositories and issue trackers, the mailing lists, a few blog posts and white papers of the projects. How can Ken assess which of the many open source projects is trustworthy? Which project has a mature base and a healthy community? What project fits best both with VOZ's company and technology culture? How can Ken rank the candidate projects with as little manual work as possible involved, based on objective criteria?

3. Design and Architecture

In this section, we describe the data sources we used to extract information and the usage of a common model and standard format to represent extracted information from different data sources to support 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 different data sources? 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 data from multiple data sources. As such, we propose to use RDF [7] (Resource Description Framework) as the core, target data model. Once modeled in RDF, the data can be indexed and queried using the SPARQL query standard and associated tools. Finally, the integrated data can be published on the Web using Linked Data principles[14] allowing third parties to discover and subsequently crawl the knowledge, and also allowing to interlink with background information available remotely on the Web. 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 and developer's information from different data sources are extracted and transformed into RDF, yielding RDF data sets.
2. Interlink the RDF data sets with each other and across different data sources, where necessary.
3. Load the interlinked RDF data sets into an SPARQL endpoint. This enables one to query the interlinked data sources in order to address many use cases (cf. Section 2).

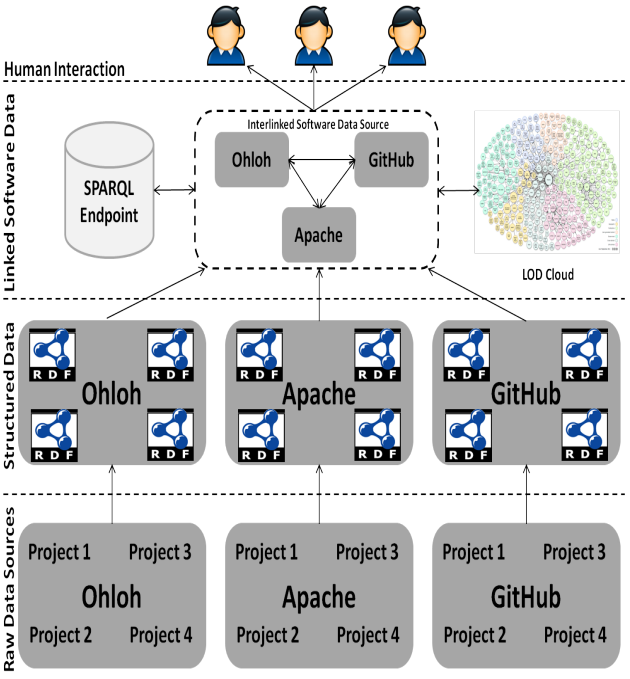

Figure 1. Architecture.

3.1 Transforming Data Sources into RDF

We considered Apache ANT[15] project repositories, GitHub and Ohloh as our primary data sources. We gen-

[13] http://geekli.st/

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

[15] http://ant.apache.org

erated RDF data sets from mailing list archives, bug tracking systems, source control repositories and source code of Apache ANT project. Further, we extracted project and developer related information from GitHub and Ohloh, producing more RDF data sets.

In order to generate and interlink information from Apache ANT project repositories, we used a Linked Data-driven approach for extracting and interlinking information, as we have argued elsewhere [6]. The overall concept of *Linked Data Driven Software Development* (LD2SD) is to extract information from software repositories of a particular project in RDF format by assigning URIs to each project entity (i.e., bug, email, developer id etc.) and interlink the URIs where necessary. Our LD2SD approach currently generates RDF data sets from bug tracking systems, mailing list archives, source control commit logs and the Java source code of a particular project. An excerpt of an exemplary RDF representation of a source code file is shown in Listing 1. [16]

```
1  @prefix : <http://example.org/prj/org/> .
2  @prefix b: <http://vocab.deri.ie/linkedfloss#> .
3  @prefix owl: <http://www.w3.org/2002/07/owl#> .
4  :connect a b:JavaClass;
5  b:imports "java.io.IOException";
6  b:author <http://example.org/ant/author/bodewig>;
7  b:package <http://example.org/prj/org/>;
8  b:hasMethod :connect#send;
9  b:hasMethod :connect#write;
10 b:hasAttribute "_port"
11 .
12 :connect#write a b:JavaMethod;
13 b:parameter-type "byte[]";
14 b:parameter-name "buff"
15 .
```

Listing 1. An Exemplary Java Source in RDF.

Listing 1 describes classes or packages which are imported (see `line#5`), the author of the class (see `line#6`), the methods defined in the class (see `line#8-9`) and the class variables (see `line#10`). Further, the parameter names and types along with the JavaDocs associated with each method definition is also extracted (see `line#12-14`).

GitHub provides an API[17] access over HTTPS and allows to send and receive data as JSON. We used the API to extract projects and developers information in JSON and converted it into RDF data sets. We do not go into details of the metadata which is provided by the API but recommend interested readers to have a look at their API tutorial[18]. An excerpt of an exemplary RDF representation of a developer information extracted from GitHub is shown in Listing 2. The information extracted for a particular developer describes the developers he/she is following and is being followed (see `line#4`), the projects he/she is working on (see `lines#5-6`) and basic profile information (see `lines#7-9`).

```
1  @prefix : <http://vocab.deri.ie/linkedfloss#> .
2  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
3  <http://example.org/gh/dev/bodewig> a foaf:Person;
4  :followers <http://example.org/gh/dev/larrys>;
5  :repo <http://example.org/gh/prj/Ant4NantAndMSBuild>;
6  :repo <http://example.org/gh/prj/ant>;
7  :location "Germany";
8  foaf:accountName "bodewig";
9  foaf:name "Stefan Bodewig"
10 .
```

Listing 2. An Exemplary Developer Information extracted from GitHub in RDF.

An excerpt of an exemplary RDF representation of a project extracted from GitHub is shown in Listing 3. The project information extracted from GitHub (cf. Listing 3) describes some basic information about the project (see `lines#6-10`), core developers of the project (see `line#11`), the developers who forked the project (i.e., contributors of the project) and source control commits relevant to the project (see `lines#12-17`).

```
1  @prefix : <http://vocab.deri.ie/linkedfloss#> .
2  @prefix doap: <http://usefulinc.com/ns/doap#> .
3  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
4  @prefix b: <http:baetle.googlecode.com/svn/ns/#> .
5  <http://example.org/gh/prj/ant> a doap:Project;
6  :forkedby <http://example.org/gh/dev/terabyte>;
7  :watchers <http://example.org/gh/dev/bodewig>;
8  doap:description "Apache Ant";
9  doap:name "ant";
10 doap:programming-language "Java";
11 doap:developer <http://example.org/gh/dev/bodewig>;
12 :commits <http://example.org/gh/commits/0035b01af>,
13 <http://example.org/gh/commits/3bf8ef5f4>
14 .
15 :0035b01af a baetle:Committing;
16 b:author "stefan.bodewig";
17 b:summary "simple build file to create ..."
18 .
```

Listing 3. An Exemplary Project Information extracted from GitHub in RDF.

Ohloh provides a RESTful API to the Ohloh open source directory and returns XML-formatted data in response to HTTP GET requests. We used the Ohloh API to get statistical information about projects and developers in XML format and converted it into RDF data sets. For details on the metadata provided by Ohloh API we recommend interested

[16] The URIs used in the Listings are just for illustration purposes and are not dereferenceable.

[17] `https://api.github.com`

[18] `http://developer.github.com/v3/`

readers to have a look at their API tutorial[19]. An excerpt of an exemplary RDF representation of a developer information extracted from Ohloh is shown in Listing 4. The information extracted for a particular developer describes the basic profile information (see lines#4-9) along with the projects he is working on (see lines#11-13).

```
1  @prefix : <http://vocab.deri.ie/linkedfloss#> .
2  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
3  <http://example.org/ohloh/dev/bodewig> a foaf:Person;
4  :kudo-rank "9";
5  :location "Germany";
6  foaf:based_near [
7         geo:lat "51.191261";
8         geo:long "6.442066" ];
9  foaf:name "Stefan Bodewig"
10 .
11 :repo <http://example.org/ohloh/prj/ant>,
12 <http://example.org/ohloh/prj/Ant4NantAndMSBuild>,
13 <http://example.org/ohloh/prj/xmlunit>
14 .
```

Listing 4. An Exemplary Developer Information extracted from Ohloh in RDF.

An excerpt of an exemplary RDF representation of a project information extracted from Ohloh is shown in Listing 5. The project information extracted from Ohloh describes the number of users who uses the project, total number lines of code, developers contributed to the project and the name of the project (see lines#6-9) etc. Moreover, the total number of commits made by a particular developer, basic information about him/her and the total number of commits made using different programming or scripting languages is also extracted (see lines#12-25), which will help in identifying the expertise of a developer in a certain programming or scripting language.

3.2 Interlinking RDF Data Sources

From the Listings, we are able to conclude that the RDF fragments are talking about two different entities, i.e., a project named "Apache Ant" and a developer named "Stefan Bodewig". We can interlink these RDF fragments using an owl:sameAs property indicating that these URIs actually refers to the same entity (see Listing. 6). Through interlinking Apache Ant repositories, GitHub and Ohloh RDF data sources, we will be able to query the projects which "Stefan Bodewig" is developing at GitHub, his development activity (e.g., last month commits, bug fixes, social interaction etc.) in Apache Ant repositories and his project development ratio in different programming languages (available via Ohloh).

[19] http://meta.ohloh.net/getting_started/

```
1  @prefix : <http://example.org/ohloh/prj/ant#> .
2  @prefix ls: <http://vocab.deri.ie/linkedfloss#> .
3  @prefix doap: <http://usefulinc.com/ns/doap#> .
4  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
5  <http://example.org/ohloh/prj/ant> a doap:Project;
6  ls:users "1174";
7  ls:loc "218797";
8  doap:developer :15491947040395;
9  doap:name "Apache Ant"
10 .
11 :15491947040395 a foaf:Person;
12 ls:commits "3994";
13 ls:language_fact :15491947040395_5 ,
       15491947040395_3;
14 foaf:accountName "bodewig";
15 foaf:name "Stefan Bodewig"
16 .
17 :2528958348267147_5
18 ls:commits "2501";
19 ls:language-id "5";
20 doap:programming-language "Java"
21 .
22 :2528958348267147_3
23 ls:commits "1311";
24 ls:language-id "3";
25 doap:programming-language "XML"
26 .
```

Listing 5. An Exemplary Project Information extracted from Ohloh in RDF.

```
1  @prefix owl: <http://www.w3.org/2002/07/owl#> .
2
3  <http://example.org/gh/dev/bodewig> owl:sameAs
4  <http://example.org/ohloh/dev/bodewig>,
5  <http://example.org/ant/author/bodewig> .
6
7  <http://example.org/ohloh/prj/ant> owl:sameAs
8  <http://example.org/gh/prj/ant> .
```

Listing 6. An Exemplary Interlinking.

In order to interlink software repositories of a particular project, we wrote our own scripts. For example, the *log extractor* generates the RDF data sets from source control commit logs and further, links it to the RDF data sets of bugs and source-code where necessary. An excerpt of an exemplary RDF representation of a source control log is shown in Listing 7. It exploits the convention used by the developers mentioning bug IDs in the summary of a commit while committing changes to the source control repository. The *log extractor* uses simple text search algorithm to search for certain phrases commonly used by developers such as, bug#xxxx in the summary of a source control logs. When a bug ID is detected, the *log extractor* adds a triple using property b:fixes to interlink the source control log to that particular bug (see line#8). The *log extractor* also links the source code file URL on source control repository URL to the meta-information of that particular source code file using owl:sameAs property (see line#11).

```
@prefix : <http://example.org/apache/prj/SVN/> .
@prefix b: <http://baetle.googlecode.com/svn/ns/#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
:39823 a b:Committing;
b:modified
     <http://example.org/prj/svn/org/connect.java>;
b:author
     <http://example.org/apache/prj/author/bodewig>;
b:revision "39823";
b:fixes <http://example.org/apache/prj/Bug/123>;
b:summary "this patch fixes bug#123.";
<http://example.org/prj/svn/org/connect.java> a
     b:JavaSource;
owl:sameAs <http://example.org/prj/org/connect>
.
```

Listing 7. An Exemplary Source Control Interlinking.

Generating `owl:sameAs` links between the developers of Ohloh to the Apache or GitHub developers is straight forward. Ohloh contains statistics of a particular project by analyzing source code repositories which means that the developer names at Ohloh will be same as the developer names in the source code repository of that particular project. In order to generate a set of `owl:sameAs` links between an Apache project and the Ohloh project, we extracted a list of developers who committed on the source control repository of an Apache ANT project by querying the source control RDF data sets. Further, we queried the Ohloh RDF data set to retrieve a list of developers of Apache ANT project. Finally we compared both lists of developers using string similarity measures in order to generate `owl:sameAs` links between them. There are in total `46` developers who committed code to the source control repository of Apache ANT project from the beginning of the project to date, among them we found `45` developer names in the Ohloh data set hence producing `45` links between Apache ANT and the respective Ohloh data set. This interlinking enables us to not only query the activities of a particular developer in Apache ANT project but also allows to see the commits ratio of a developer in different programming languages used in the project.

In order to generate links between Ohloh and GitHub developers, we took a subset of `153` random projects from Ohloh and `4414` projects from GitHub. We extracted a list of developers who worked on the GitHub projects under consideration and compared it with the developers of selected `153` Ohloh projects. The string similarity approach resulted in `196 owl:sameAs` links between Ohloh and GitHub data sets. This enables us to not only query the development activity of developers in GitHub project but also allows to query their contribution statistics which are stored by Ohloh.

4. Preliminary Experimental Results

We are currently in the phase of preparing a ground truth in order to validate our approach of interlinking data sources as well as comparing it to other duplicate detection algorithms and frameworks like Silk [11], Swoosh [1], Duke [20] etc. We will address in this section few use cases which we discussed in Section 2 and show how linked data sets can be used to exploit them.

In order to show the benefit of integrating developer related information from different data sources, we hosted a SPARQL endpoint[21] which contains the RDF data sets from GitHub, Ohloh and Apache projects. We will use this SPARQL endpoint to run SPARQL queries which are presented in this section. We start with a simple query to lists all projects on which a developer is working on or has worked in the past (cf. Listing 8).

```
PREFIX ls: <http://vocab.deri.ie/linkedfloss#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

select distinct ?repo {
?dev ls:repo ?repo .
?dev foaf:accountName "bodewig".
}
```

Listing 8. Developer Projects Query Pattern.

In our current settings, we extracted Apache ANT developers data from one code forge only (i.e., GitHub). In the near future, we will incorporate other code forges (i.e., SourceForge, GoogleCode etc.) and further apply interlinking approach in order to get an extensive list of projects on which developer is working or has worked in the past. Given that the Ohloh data set contains the development statistics of a developer in different programming languages as part of a particular project, it is easy to query the number of commits he made to a particular project using different programming languages as shown in Listing 9.

The results of the SPARQL query (cf. Listing 9) is shown in Table 1 which makes it easier to understand the expertise of a particular developer in different programming languages based on the number of commits he made to the project. For example, the result shows that the developer has most experience in "Java" comparing to "MetaFont" programming language. It also address to a certain extent our first use case scenario outlined in Section 2.

It is very likely that the developer has contributed to other open source projects which are also indexed by Ohloh. Hence, one can also query the development statistics of all the programming languages which a developer has used in developing different open source projects. The query in

[20] `http://code.google.com/p/duke/`
[21] `http://linkedfloss.srvgal85.deri.ie/sparql`

```
PREFIX ls: <http://vocab.deri.ie/linkedfloss#>
PREFIX doap: <http://usefulinc.com/ns/doap#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf:
     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

select ?commits ?language {
?id foaf:accountName "Stefan Bodewig" .
?repo doap:developer ?id .
?repo doap:name "Apache Ant" .
?id ls:language-fact ?languageFact .
?languageFact ls:commits ?commits .
?languageFact doap:programming-language ?language .
}
```

Listing 9. Developer Commits Query Pattern.

Programming Language	Commits
Java	2501
HTML	1404
XML	1311
JavaScript	257
Shell Script	74
XSL Transformation	30
DOS Batch Script	28
CSS	23
Perl	8
Python	7
C#	6
XML Schema	1
MetaFont	1

Table 1. Developer Commits to the Project based on Programming Language.

Listing 10 returns an average commit ratio of a developer in different programming languages based on all the projects he worked. The results returned by the query (cf. Listing 10) gives an idea about the expertise level of a developer in different programming languages as shown in Table 2.

Further, one can query the total number of commits made by all the developers to the project in different programming languages. It will help newcomers (i.e., volunteers) of the open source project to get an insight of which programming or scripting language they could potentially use to start contributing to the project. In fact, most or all of the questions pointed out by Conklin (cf. Section 1) can be answered by simple queries. Enabling such integration will not only allow developers to query abstract level information (e.g., number of commits made by a developer to the project) about the project or developer but also allow to query information which is hidden deep inside the project repositories (e.g., contribution of a developer in the last release of the project). We have tried to show the benefit of integrating developer-related data from different data

```
PREFIX ls: <http://vocab.deri.ie/linkedfloss#>
PREFIX doap: <http://usefulinc.com/ns/doap#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf:
     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

select ?language (AVG(xsd:int(?commits)) AS
     ?commit_average)
where {
?id foaf:accountName "Stefan Bodewig" .
?repo doap:developer ?id .
?id ls:language-fact ?languageFact .
?languageFact ls:commits ?commits .
?languageFact doap:programming-language ?language .
OPTIONAL {?languageFact ls:comment-ratio
     ?commentRatio }
} group by ?language
```

Listing 10. Developer's Average Commit Ratio Query Pattern.

Programming Language	Commits Average
Java	327.00
JavaScript	257.00
HTML	171.88
XML	129.57
Shell Script	37.50
DOS Batch Script	28.00
C#	27.00
CSS	11.33
XSL Transformation	8.00
Perl	8.00
Python	7.00
XML Schema	1.50
MetaFont	1.00
Ruby	1.00

Table 2. Developer's Average Commit Ratio based on Programming Language.

sources in order to serve a variety of use cases often found in the software development domain.

5. Related Work

To the best of our knowledge, there are only a few published works on identifying and relating the different identities that developers use to interact with different tools in the field of software engineering. In [2], Bird et al. proposed an approach to produce a list of *<name,email>* identifiers by parsing the emails and clustering them. The clustering algorithm to measure the similarity between every pair of IDs is based on string similarity between names, between emails, between names and emails, etc. We also use the string similarity approach in order to interlink the data sources but

our scope is broader in a sense that we are not applying the heuristics within a single project but across different data sources.

Robles et al. [10] discusses the problem of developer identification in general, but the work lacks in details about the heuristics they propose to identify and match the different identities of developers. The authors propose a technique to build one identity from another by extracting the "real life" name from email addresses, such as *nsurname@domain.com, name.surname@domain.com* etc. Their approach also relies on string similarity algorithm. In general, the problem is related to duplicate detection. The duplicate detection frameworks provide several techniques to effectively solve matching different entities. In this regard, Kopcke et al. [8] analyzed and compared 11 different duplicate detection frameworks. While research in this area mostly refers to identifying duplicates in the same data set, the techniques might be mapped to the case of matching over different data sets. However, they are tailor-made for identifying different IDs of the same developer inside one repository. Naumann et al.[9] provides a nice overview of this research direction.

In the Semantic Web domain, Volz et al. [11] proposed an interlinking framework also known as SILK framework which generates link between two RDF data sets based on some string similarity measures specified by the user. Their interlinking framework supports different string similarity algorithms to compare if two different RDF resources are similar or not. In a next step, we will assess to what extent we can use the SILK framework to link different data sources in the near future.

6. Conclusion

We have motivated and proposed a simple yet effective approach of integrating developer-related information from different data sources on the Web. We have argued that Semantic Web technologies allow integrating and querying information across different data sources and have illustrated this through a number of real-world examples.

We have made some initial progress in integrating different data sources using string similarity based interlinking approach. Currently, we are in the phase of preparing a ground truth and will compare our approach with other duplicate detection algorithms and frameworks. Additionally, we improve the interlinking approach, yielding higher quality and quantity links between different data sources. We also plan to extract more project and developer-related information from different data sources (i.e., Apache, GitHub, Ohloh, SourceForge, GoogleCode etc.), transform them into RDF data sets, interlink them and host them via our SPARQL endpoint. In the near future, we also plan to provide an application on top of our interlinked data sources in order to address certain issues/use cases often found in software development processes.

References

[1] 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*, 18(1):255–276, Jan. 2009.

[2] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *MSR '06: Proceedings of the 2006 international workshop on Mining software repositories*, pages 137–143, New York, NY, USA, 2006. ACM.

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

[4] S. Diehl, H. C. Gall, and A. E. Hassan. Guest editor's introduction: Special issue on mining software repositories. *Empirical Softw. Eng.*, 14(3):257–261, 2009.

[5] A. E. Hassan. The Road Ahead for Mining Software Repositories. In *Future of Software Maintenance (FoSM) at Int. Conf. on Software Maintenance(ICSM)*, 2008.

[6] A. Iqbal, O. Ureche, M. Hausenblas, and G. Tummarello. LD2SD: Linked Data Driven Software Development. In *Int. Conf. on Software Engineering and Knowledge Engineering (SEKE 09)*, 2009.

[7] 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.

[8] H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data Knowl. Eng.*, 69(2):197–210, Feb. 2010.

[9] F. Naumann and M. Herschel. An introduction to duplicate detection. *Synthesis Lectures on Data Management*, 2(1):1–87, 2010.

[10] G. Robles and J. M. Gonzalez-Barahona. Developer identification methods for integrated data from various sources. In *Proceedings of the 2005 international workshop on Mining software repositories*, MSR '05, pages 1–5, New York, NY, USA, 2005. ACM.

[11] 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.