<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Integrating Developer-related information across Open Source Repositories</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Iqbal, Aftab; Hausenblas, Michael</td>
</tr>
<tr>
<td><strong>Publication Date</strong></td>
<td>2012</td>
</tr>
<tr>
<td><strong>Item record</strong></td>
<td><a href="http://www.deri.ie/content/integrating-developer-related-information-across-open-source-repositories">http://www.deri.ie/content/integrating-developer-related-information-across-open-source-repositories</a>; <a href="http://hdl.handle.net/10379/4510">http://hdl.handle.net/10379/4510</a></td>
</tr>
</tbody>
</table>

Downloaded 2021-05-29T23:47:28Z

Some rights reserved. For more information, please see the item record link above.
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, open source projects 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 over SVN versioning system. Therefore certain developers use the GitHub 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. 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 who is a member of ASF and is contributing to many Apache projects, has also developed few projects on GitHub and SourceForge. 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, WebID, 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:

---

1. **http://git-scm.com/**
2. **http://subversion.apache.org/**
3. **https://github.com/**
4. **https://github.com/apache/**
5. **http://stefan.samaflost.de/**
7. **https://github.com/bodewig**
8. **http://sourceforge.net/users/bodewig**
9. **http://openid.net/**
10. **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, Geeklist, 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 analyze 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 principles14 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 ANT15 project repositories, GitHub and Ohloh as our primary data sources. We gen-

---

13http://geeklist/
14http://www.w3.org/DesignIssues/LinkedData.html
15http://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.  

Listing 1. An Exemplary Java Source in RDF.

```java
@prefix b: <http://example.org/prj/org/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
:connect a b:JavaClass;
:bimports "java.io.IOException";
:bpackage <http://example.org/prj/org/?>/;
:bhasMethod :connect#write;
:bhasAttribute "_port" .
:connect#write a b:JavaMethod;
:bparameter-type "byte[]";
:bparameter-name "buff" .
```

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).

```java
@prefix : <http://example.org/prj/ant/> .
@prefix b: <http://example.org/prj/org/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix b: <http://example.org/prj/ant/> .
@prefix doap: <http://example.org/prj/ant/>.
:connect a b:JavaClass;
:bimports "java.io.IOException";
:bpackage <http://example.org/prj/org/?>;
:bhasMethod :connect#write;
:bhasAttribute "_port" .
:connect#write a b:JavaMethod;
:bparameter-type "byte[]";
:bparameter-name "buff" .
```

Listing 3. An Exemplary Project 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).

```java
@prefix : <http://example.org/prj/ant/> .
@prefix b: <http://example.org/prj/org/> .
@prefix doap: <http://example.org/prj/ant/>.
@prefix b: <http://example.org/prj/ant/> .
:connect a b:JavaClass;
:bimports "java.io.IOException";
:bpackage <http://example.org/prj/org/?>;
:bhasMethod :connect#write;
:bhasAttribute "_port" .
:connect#write a b:JavaMethod;
:bparameter-type "byte[]";
:bparameter-name "buff" .
```

### Annotating Metadata in Software Repositories

GitHub provides an API 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. An excerpt of an exemplary RDF representation of a developer information extracted from GitHub is shown in

---

16The URIs used in the Listings are just for illustration purposes and are not dereferenceable.
17https://api.github.com
readers to have a look at their API tutorial\(^9\). 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).

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

```java
PREFIX foaf: <http://xmlns.com/foaf/0.1/> ;
PREFIX ls: <http://usefulinc.com/ns/ls#> ;
PREFIX doap: <http://usefulinc.com/ns/doap#> .
PREFIX owl: <http://www.w3.org/2002/07/owl#> .
PREFIX oh: <http://example.org/ohloh/> .
PREFIX repo: <http://example.org/ohloh/prj/> ;
PREFIX :location: <http://example.org/ohloh/prj/> ;
PREFIX :kudo-rank: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloh/prj/> ;
PREFIX :ls: <http://example.org/ohloh/prj/> ;
PREFIX :http: <http://example.org/ohloo
Generating `owl:sameAs` links between the developers of Ohloh to the Apache or GitHub developers is straightforward. 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 that particular project. 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).

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 as which a developer has used in developing different open source projects. The query in

\(^{20}\)http://code.google.com/p/duke/

\(^{21}\)http://linkedfloss.srvga185.deri.ie/sparql
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 \(<\text{name},\text{email}\)> 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