### Abstract
The MetricsGrimoire system is composed by a set of tools designed to retrieve data from repositories related to software development. Their aim is to produce organized databases suitable for easy querying with research and industrial purposes. The data in those databases have a similar structure, to easy cross-database studies, and can be enriched with information such as linkage of the multiple identities of actors, or their affiliation. This paper presents the general structure of those databases, and a collection of up-to-date database dumps that are publicly available. They correspond to two well-known projects, OpenStack, and Eclipse, including data from source code management repositories, issue tracking systems, mailing lists, and code review systems.

### I. Description
The data presented in this paper corresponds to the contents of several repositories related to software development in two major free / open source software projects: Eclipse and OpenStack. The data has been checked and curated with the help of developers from the corresponding projects, and includes merged identities (unique identities that correspond to persons which may use different identifiers in different repositories) and affiliations (hiring organizations) for a large fraction of developers. For both projects, the schemas used in the databases are mainly the same, so they can be considered homogeneous and a good base for comparative analysis.

In particular, the projects and repositories involved are:
- **OpenStack**\(^1\), with information from the following repositories: source code management (git), issue tracking (Launchpad), mailing lists (archived in mbox format), and code review (Gerrit). See Table I for a summary of data in the corresponding databases.
- **Eclipse**\(^2\), with information from the following repositories: source code management (git), issue tracking (Bugzilla), mailing lists (archived in mbox format), and code review (Gerrit). See Table II for a summary of data in the corresponding databases.

For each kind of repository and project, a database with all information in repositories of that kind is maintained. For example, the information for all git repositories in one project is found in the database `source_code`. For each project a directory

\(^1\)http://openstack.org (March 29, 2015).
\(^2\)http://eclipse.org (March 29, 2015).
\(^3\)http://gsyc.es/~jgb/repro/2015-msr-grimoire-data

### II. Retrieval
To retrieve the data, the following MetricsGrimoire tools were used:
• CVSAnalY\textsuperscript{4}, to retrieve data from git repositories.
• Bicho\textsuperscript{5}, to retrieve data from Bugzilla, Launchpad (issue tracking) and Gerrit (code review) repositories.
• MailingListStats\textsuperscript{6}, to retrieve data from mailing lists.

A description of how CVSAnalY, Bicho and MailingListStats work, and the process they follow to retrieve, organize and store the data can be found in [1]\textsuperscript{7}.

The tools were run with standard options, to gather all the data from the repositories, and store it in MySQL databases. In addition, some extra tables are produced for adding information about identities that correspond to the same person (unique identities), and affiliation for most of the developers. Those tables are maintained in part by some scripts that apply some heuristics, an include information from the projects themselves (such as affiliation documents maintained by the corresponding Foundations). Manual screening of the data, and verification with the project is done on a periodic basis.

For both projects the data is being gathered daily, and updated database dumps produced several times per week. These updated versions can be found at the database dump page.\textsuperscript{8} These tools are pre-existent to this paper, and were not developed specifically for it. They allow, for example, to find commits or messages authored by people affiliated to a certain company.

The basic ideas of how to store information about unique identities (those that allow for merging the same identities of a person) were described in [4].

The information about unique identities is stored in the following tables:

• upeople. Table for storing information about each unique identity, such as the canonical name that will be used for it. This table is common for a whole project (OpenStack or Eclipse), and stored in the CVSAnalY (source_code) database.
• people. Table for storing information about each identity found in repositories, such as the canonical name for the identity. Each database (corresponding to a kind of repository) has one people table.
• people_upeople. Correspondence between regular identities (those found in the repositories, stored in the people table) and “unique” identities (corresponding to a single person, stored in upeople). Each unique identity may correspond to several regular identities. Therefore, this table will have one row per each identity that can be merged in the same unique identity. Each database in the collection has one of these tables, for the identities found in it.
• Identities in other tables. When any table needs to link to an identity, it will use the corresponding identifier in the people table. For example, the scmlog table in the git database, which has an entry per commit, features two links to people: the author identifier and the committer identifier.

Information in people tables can be used as well to track the same identity in several kinds of repositories. For example, joe.smith@foo.com can be searched in all people tables to know in which ones that identity was active.

When considering unique identities, the common upeople table can be used to identify developers in any repositories, by joining the people_upeople table.

The information about how developers are affiliated to organizations is maintained by linking it to unique identities. Each affiliation entry will in fact link a unique identity to an organization during a certain time span. For a complete landscape the following tables are involved, all of them unique for a whole project, stored in the source_code database:

• companies. This table includes one entry per company, with its information: for example, its canonical name.
• upeople. The same upeople table described above.
• upeople_companies. This table link unique identifiers for developers with the organization to which they are affiliated, including the starting and finishing date for that affiliation as well.

Since the upeople table is linked to the identities found in repositories, via the corresponding people_upeople tables, it can be joined to link companies to those identities. This allows, for example, to find commits or messages authored by people affiliated to a certain company.
IV. LIMITATIONS

There are several known issues which cause some problems when analyzing the data:

- The information stored in both CVSAnalY databases correspond to git repositories. However, there is an important difference: in the case of OpenStack, all the information really corresponds to git. Meanwhile, in the case of Eclipse, the system started using CVS repositories, which were later migrated to Subversion, and still later to git. Therefore, a large fraction of the information really conforms to the CVS and Subversion abstractions, which are different from those in git. For example, CVS lacks atomic commits, which means that each commit corresponds to a change in a single file, while in Subversion and git a single commit can perfectly include changes to several files. The way of dealing with branches, for example, is very different between CVS, Subversion and git as well.

- If the uppeople table for each project were perfect, each entry would correspond to a real person. In general, due to the inexact nature of the heuristics and data sources used to maintain it, some duplicate (non unique) entries could exist. With time, this table is refined with new unique identities, allowing a more perfect merging of identities corresponding to the same person. In the current version, unique identities are only checked for developers, which means that although the affiliation for a number of developers is not defined, their contribution is small enough to be irrelevant for many purposes.

- Not all developers, nor even those with unique identifiers, have affiliation information. However, this information is available for developers contributing a very large fraction (certainly, more than 95%) of commits. This means that although the affiliation for a number of developers is not defined, their contribution is small enough to be irrelevant for many purposes.

- Although the tools used to retrieve the information have been extensively tested, are in use in many production environments, and the resulting data is continuously verified and validated with the help of the corresponding projects, it could contain errors due to bugs in the tools.

- The data in the repositories is not always completely reliable. For example, in the case of Gerrit in OpenStack, the information is not completely correct. For example, some code review process are shown to start well after the date of some comments linked to them. Of course that is not possible, and has been tracked to a bug in some (about 5,000) code review tickets, probably due to a data migration.

In summary, probably the main limitation of the information in the databases is the (comparatively) little affiliation and unique identities information for repositories other than git.

V. CONCLUSION

The collection presented in this paper provides detailed, extensive and accurate information from several repositories of two well known and relevant free, open source software
projects. Four different kinds of repositories are considered, and the information is organized in the same way in both projects, which eases comparative analysis. In addition, unique identities and affiliation information is provided for a large fraction of the developers of both projects. In addition, the information in the databases is being used to produce dashboards for both projects (Eclipse\textsuperscript{10}, OpenStack\textsuperscript{11}), which allows for visual inspection of some parameters. The information available in the databases is also comparable to that obtained from other projects, by running MetricsGrimoire tools on their repositories.

ACKNOWLEDGMENTS AND INVOLVEMENT

The authors thank Bitergia for letting them, with the consent of its customers, when applicable, use this database dumps for academic purposes. Gregorio Robles and Jesus M. Gonzalez-Barahona were involved in the design of the retrieval tools, and the database schemas. Daniel Izquierdo-Cortazar and Jesus M. Gonzalez-Barahona were involved in the retrieval of data. Other people in Bitergia (mainly Alvaro del Castillo) were involved in importing and checking affiliation information and in merging identities. The authors want to thank specially to the projects that, by making their software development repositories public, allow for the existence of these databases.

REFERENCES


\textsuperscript{10}http://dashboard.eclipse.org (March 29, 2015).
\textsuperscript{11}http://activity.openstack.org (March 29, 2015).