Identifying Companies in Open Source Software Ecosystems

J.P.A. Buis BSc

dr. S. Jansen

Technical Report UU-CS-2015-011
July 2015

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Identifying Companies in Open Source Software Ecosystems

Jakob Buis and Slinger Jansen

Dept. of Information and Computing Sciences
Utrecht University, The Netherlands
slinger.jansen@uu.nl

Abstract. Software companies with an interest in specific open source ecosystems can contribute by having employees contribute code to the software. It is of interest to the field of software ecosystems to determine the quantity of these contributions. This requires an effective and reliable way to determine as to at which company a contributor works. We propose a blend of various, fully-automated strategies and apply these in a case study of the software ecosystem around Eclipse. The results are validated using manual inspection and an e-mail survey. Using multiple strategies yields an identification of the place of employment in 41.4% of cases with 85.7%. Using these strategies, commercial contributors can be identified more accurately, enabling software ecosystem researchers to correctly identify commercial patronage.

Keywords: software ecosystems, open source, contributions, employment, company, identification

1 Introduction

Large software companies have a strategy for how they reuse and contribute to open source software [5]. These contributions vary in scope and in nature, ranging from simple monetary contributions to more complex agreements in which other company resources (expertise, development man-hours) are dedicated to the cause. While partner levels as studied by Aarnoutse et al. [6] can provide a rough approximation of a company's involvement in a particular software ecosystem, these other, *in natura* contributions are less measurable. These partner levels provide a contract, a minimum amount of contribution that companies commit to, but it does not specify the actual contributions made by the companies nor the impact of those contributions on the project as a whole.

A major way in which companies can contribute to open source software, is by dedicating developers to writing actual code for the product. The creator of Redis, a major NoSQL-database server, is employed by Redis Labs [7] to work full-time on the development of Redis. In the software ecosystem around Eclipse, IBM has estimated the value of its investment in R&D for Eclipse up to 2001 to be $40 million [8]. In 2011, the director of the Eclipse Foundation estimated the total investment in R&D by many companies in the Eclipse ecosystem to be "well over $800 million" [8]. These investments into R&D from companies are ultimately converted into more or improved code in the Eclipse project and software ecosystem.

Contributions in code are never tagged with company names for the convenience of researchers. Before we can draw any conclusions on the quantity of contributions in code by a company made to a particular ecosystem, we need to develop a method to associate

the contributions (commits) made by individual programmers to companies with which they are employed. Considering the huge numbers of commits made every year to large-scale open source projects such as Eclipse, we strive to develop a fully-automated solution and prove its accuracy. While this paper does not attempt to provide a complete, one-fits-all solution to this particular problem, we show that identifying the employment of single developers at companies is at least a feasible process.

To develop a method that is suited to more than one specific ecosystem, identification of the company must be based only on information that is widely available with code commits. A quick sample of widely used, multi-user version control systems such as VCS, Subversion and Git, concludes that we can effectively only rely on the name of the developer and an e-mail address.

2 Research question

The main research question for this research project is: Can the employer of a developer be reliably identified based on the name and e-mail address?

This research question provides an answer to only a small part of the larger research gap stated in the introduction. It is nevertheless a very important sub-problem that needs to be properly investigated. It is easier to determine which contributor has contributed a section of code. In nearly all Version Control Systems (VCS), which are to be considered essential for multi-user software development, changes are associated with a timestamp and the user whom contributed the change. Making the connection from a user account to a corporate sponsor is a lot more involved and has not been extensively studied. Creating this step enables researchers in the future to gain more insight into contributions by corporate sponsors to open source software ecosystems.

To answer this research question, we gather data from the Eclipse project on Github, and define and execute multiple search strategies which yield candidate names for the employer of the developer. The names are subsequently validated through manual inspection and a survey amongst the developers.

3 Research method

To assess the reliability, we test the method on the Eclipse Foundation organisation on Github[1]. An organisation account on Github consists of one or more git repositories, containing code written by users of Github and published on the account. Paying organisations can have private repositories that cannot be seen unless the viewer is a member of the organisation. The authors are aware that the complete Eclipse ecosystem consists of many more repositories hosted on various other platforms. Nevertheless, this initial set of repositories is a representative subset of the entire ecosystem and serves as a proper starting point for a preliminary investigation of the ecosystem around Eclipse. As these repositories are relatively easily accessible through the GitHub REST API, adding more repositories from

[1] `https://github.com/eclipse`

different services would be comparatively much more expensive. At this point, broadening our scope is not supportive of our research goal: the authors aim to prove that reliable identification of employment can be done, not to give an exhaustive overview of companies in the Eclipse ecosystem.

At this moment, the Eclipse Github-organisation has 312 repositories containing code that are publicly accessible. From these repositories, we gather all users (n = 5,493) that have committed contributed at least one commit to a repository. We have constructed a small set of scripts to retrieve the data set from the Github API.[2] The data on these users such as the repositories contributed to, names, e-mail addresses and commit counts is than stored in a relational database. Note that these 5,493 entries are not individual users: a user which has contributed to three separate repositories is included thrice.

Attempts to establish the identity of a user and its employer based on the full name of the user, yielded no results. Not only do developers often use their usernames in lieu of their full name, when they do use their full name,these names are simply not unique enough to correctly determine an employer with even negligible accuracy. These attempts were quickly abandoned. Determining a user's employer based on their e-mail address proved much more useful.

It is readily apparent that neither git nor Github enforces any constraints on the e-mail addresses given by users. Conducting a manual inspection of the data quickly yields a number of cases in which the data is simply invalid. Numerous entries used invalid email addresses at non-existent locations such as *adam@localhost*[3], *adam@192.168.1.2*[4] or *admin@finn-pc.none*. Moreover, a lot of users simply repeat their full name as their email address. We suspect either by mistake or to hide their true email address from automated harvesters. Eliminating all contributors that provided email addresses with an invalid syntax or address, removed 1,667 entries (30.3%) from the data set.

A number of commits are made by users using an address of a known email provider, yielding extensions such as `hotmail.com` or `gmail.com`. These e-mail addresses do not provide any clue as to the name of the employer, and all these entries were excluded from analysis. A list of known email providers [1] was used to filter these entries, resulting in the removal of an additional 849 entries (15.5%).

Finally, a large number of contributors have contributed very little code to repositories. For each contributor and repository, the number of commits is counted and plotted in figure 1. The histogram is cut-off for values larger than 50 to maintain readability. The largest number of commits is 7,228 contributions to a single repository by one user. Using this cut-off does not alter the overall shape of the histogram. From the histogram, it becomes readily apparent that a large portion of users in the dataset only contribute very few commits to a repository. A few cutoff points are suggested from the histogram. Users contributing 0

[2] These scripts are released as open-source code under a permissive license on Github. The code can be found on https://github.com/jakobbuis/womath.

[3] Localhost or the equivalent IP address 127.0.0.1 is a local loop-back address referring to *this computer*, dependent on which computer the address is referenced. This yields no further useful information.

[4] IP addresses in the 192.168.1-255.1-255 are only assigned and usable on the local network, probably referring to a PC on the users own home or business network.

commits to a repository are users which have been added as a contributor to the repository, giving them the right to directly add commit to it, but never wrote a single commit. It is obvious that they should be excluded from analysis. The shape of the histogram suggests to only include users which have contributed 2, 3, 6 or 10 commits or more. These cut-offs would result in the inclusion of 75.0%, 65.4%, 52.9% or 44.4% of the remaining entries respectively. The authors choose having at least 6 commits as a cutoff. Users which have contributed less than 6 commits to a repository are excluded from analysis. This excludes a further 1,402 entries (25.5%) from the result set.

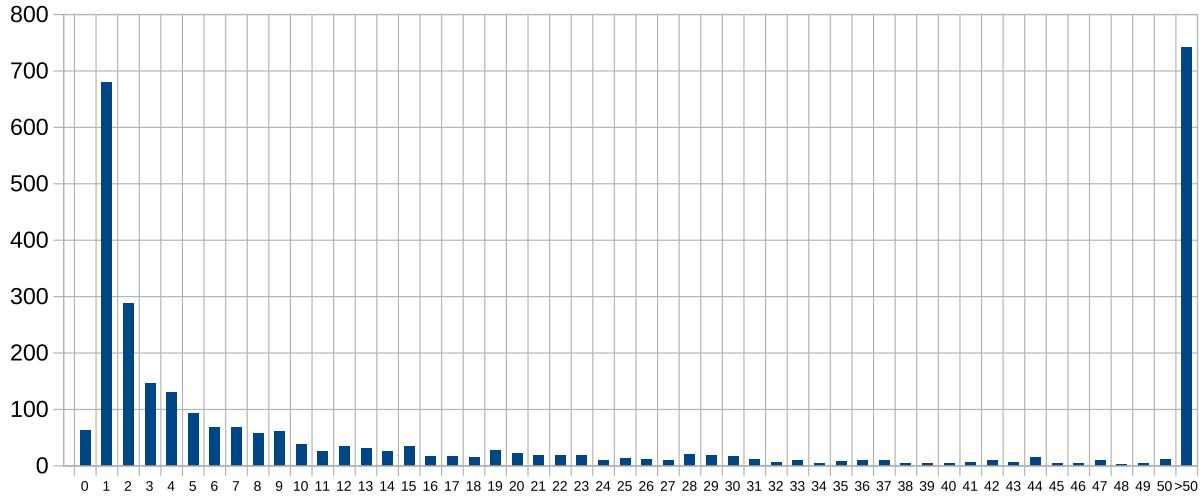

Fig. 1. Histogram of the number of commits per user-repository

The final data set of contributors to repositories consists of 1,575 entries (28.67%). For these entries, we attempt to automatically determine the name of the employer. Determining the name of a company is a two-step process: first to find potential names for the employer, and second to determine the most likely employer name from the candidate names. We will discuss these steps in turn.

3.1 Finding potential company names

Four separate strategies were used to attempt to find potential company names. Each strategy could return several names or none at all. We have included one fallback strategy that always generated exactly one answer. This ensures that every entry has at least one candidate name.

The first strategy is using the LinkedIn Company API to find companies by a domain name. Submitting the domain part of the e-mail address to this API yields a number of company names at which LinkedIn members work who have registered an email address at this specific domain. While this may seem perfect at first glance, the results are often overly specific or plainly incorrect. For example, inspecting the results for *@microsoft.com*

yields five company names as the result: Microsoft India, Microsoft France, Microsoft Ltd., Microsoft and IBM. The top result for *@ibm.com* addresses is Rational Software, a now defunct software company which was acquired by IBM in 2003 [2].

Secondly, WHOIS data was used to determine the origins of an email address. A WHOIS query is a technical query that returns various administrative records for a given domain name. These records include an administrative contact, a technical contact, full address records of the Internet Service Provider which registered the domain name on behalf of a client, and the date of registration. The administrative contact is usually the company itself, but it could also be a specific person or department inside of the company. Most fields are allowed to be left empty and obfuscating services are routinely used to protect the privacy of registrants.

As a third strategy the system requests the front page of the website at the given domain to analyse its contents. The strategy looks for consecutive capitalised words which may indicate a company name, in a number of common places on the page such as the title of the page, the first heading on the page, or the alternative text of the first image (often a logo) on the page. This step is much slower to execute than the previous two strategies and often yields too greedy results such as "ABC Company Homepage" or "XYZ Productions: Your Very Best Help At Anything!". If a particular subdomain yields no results, the first subdomain is stripped and the system retries to find a website one level up. For example, for the e-mail address *user@web.mail.abc.com*, this strategy will try to find a website containing a company name at *web.mail.abc.com*, *mail.abc.com* and *abc.com* in turn.

Finally, we use the domain name itself as a fallback strategy. The most-significant part of a domain name is determined by removing the common extensions such as *.com* and *.co.uk* and stripping off any subdomains such as *mail.xyz.com*. This part is subsequently capitalised and used as a measure of last resort.

All four strategies are executed for each entry and yield a set of candidate names for the employer. These sets are reduced to the single, most-probable answer in the next step.

3.2 Converging to an Answer

The system uses simple string operations to determine the most likely company name. If there is only one candidate available, it is chosen as the most likely answer straight away. Suppose the e-mail address *user@ibm.com* results in the following set of candidate names from all four strategies:

```
[IBM, IBM India, Sun, IBM, Microsoft, Macrosoft,  ]
```

Any unsound candidates such as the last, empty candidate in this case are removed. Identical names are than merged and each name is assigned a corresponding weight, resulting in the map:

```
{IBM => 2, IBM India => 1, Sun => 1, Microsoft => 1, Macrosoft => 1}
```

To remove locations such as "IBM India", the script merges all longer answers into the shortest possible substring, resulting in this case in:

```
{IBM => 3, Sun => 1, Microsoft => 1, Macrosoft => 1}
```

In the final processing step, spelling mistakes are corrected using the levenshtein distance[5]. All entries with a levenshtein distance lower than 3 are merged, with the highest-weighted label prevailing, or the first label if the weights are identical. In this example, the final set is

```
{IBM => 3, Sun => 1, Microsoft => 2}
```

The script than picks the candidate name with the largest weight as the final answer. If two or more candidate names have the largest weight, it picks one of them at random. This final answer is recorded in the database and used for validation.

3.3 Validation

The final answers were subsequently validated by e-mailing a short e-mail to each contributor. The e-mail includes a short background on the purpose of the survey, the final name of the employer and two buttons (yes/no) for the receiver to indicate whether these findings were correct. When clicked, these buttons reported the validation choice of the user to a server which ran a small PHP application that stored the results in the database[6].

Each individual user was e-mailed once, regardless of the number of repositories contributed to. De-duplicating the final 1,575 entries resulted in 689 individual users receiving an e-mail asking for their confirmation.

4 Results

The initial data set consists of 5,493 entries. Entries with invalid data, either invalid data or non-existent e-mail addresses amounted to 1,667 entries which were removed from the data set. A further 849 entries with e-mail addresses from common providers were disregarded. Entries which contributed only minor amounts of commits were excluded too, removing 1,402 entries from the set. The remaining 1,575 entries were analysed to determine the name of the employer. Afterwards all 689 users received an e-mail asking to confirm their employment status.

119 users (17.3%) confirmed their employment status through the e-mails, either by clicking the buttons or e-mailing back with detailed explanations of their situation. 102 users (85.7%) answered that the proposed employer was in fact correct, while 17 users (14.4%) answered that the proposed employer was wrong. Ultimately, for 41.2% of all valid e-mail addresses a potential employer name was identified. 85.7% of these employer names proved to be correct, based on a sample of 17.3%.

[5] The levenshtein distance (also called the edit distance) is the minimum number of operations that is needed to turn a string into another string, with the operations being insertion, deletion of substituion of a character. For example, the levenshtein distance of *bank* and *can* is 2 (bank → cank → can).

[6] The code of this service too is released as open-source code under a permissive license on Github. The code can be found on https://github.com/jakobbuis/womath_validator.

5 Limitations

A number of limitations and issues are identified which limit the reliability of the results and the application to other circumstances. These issues primarily arise from two sources: limitations that occur because of the approach taken and limitations related to the subject data. We discuss each set of limitations in turn.

5.1 Limitations of the approach

Whether a contribution to a repository is considered to be of significant size and impact, is measured by the number of commits. This method is time efficient but presumably only moderately accurate as a proxy for the impact by a user. A better measure would be to consider the total lines of code contributed by a user. Gathering this data from the Github API is considerably more involved than gathering counts of commits. In addition, this approach suffers from a new class of problems on its own. Counting lines of code does not consider the expressiveness per programming language, i.e. the fact that different programming languages need different number of lines to express the same concepts. Additionally, writing less code to reach the same effect is actually a sign of the better programmer. Finally, it requires significant additional data and deeper insights to account for interpretation errors. Suppose a contributor removes 100 lines of code and adds 80 new lines. If these 80 lines have the same functionality as the 100 removed lines, the 'contribution' of this user could be considered "20 unneeded lines removed" (by rewriting the code). If these 100 new lines of code are an entirely different, unrelated feature, and the 80 lines removed are a feature that is simply no longer useful, the contribution of this user should be considered "80 unneeded lines removed, added new feature of 100 lines", which has a very different impact on the project as a whole.

The approach taken in this research project to determine whether names are similar and relate to the same company results in shorter slugs being preferred over more complete names. For example, the set

```
[Sun, Sun Microsystems, Sun Microsystems Inc.]
```

is reduced to a single entry "Sun" with weight 3 and is thus subsequently selected as the final answer. The official name of the company is "Sun Microsystems Inc." and selecting that candidate name as the answer would have been preferred. One could improve upon this method by reversing the process after identifying the lowest common denominator and reworking the solution to reach a more complete version of the company name, though this process has limitations of its own.

The research method uses two separate lists: a list of 3,168 extensions of known email providers such as @gmail.com, @hotmail.com or @outlook.com, and the Mozilla Public Suffix List [4]. Though the lists are both recently updated, some issues remain. The source of the former list is unclear: it was discovered using a common search engine and while it appear to be quite complete, missing only a single entry (@xs4all.nl) which occured in the Eclipse data set, nothing is known on the process of including new entries and validating

the list. The latter list is properly sourced by Mozilla and uses formal, fully described updates but Mozilla aims to maintain a sufficiently concise list and explicitly disclaims to be exhaustive. These problems cast some doubt on the broader application of using these methods.

The levenshtein distance used to determine similarity between company names is a crude but effective measure. While comparing *Microsoft* and *Microsofd* is clearly a spelling mistake and should be considered one company, *IBM* and *IHC* are very different companies indeed. This could be improved by making the maximum acceptable levenshtein distance for merging two candidate names dependent on the length of the candidate names. The main issue in using a levenshtein distance is a lack of semantical interpretation of a name. This becomes especially important in a international context in which company names might be translated to local variants. A more powerful, more effective method may be substituted to improve results.

Finally, the script does not report on the confidence it has in its answer. Results could be considered more reliable if separate strategies generated identical or at least comparable candidates, and this fact could be used to determine a more likely answer.

5.2 Issues related to the subject data

A commit (which includes the name and email address used for study) is immutable in principle. After publication it is never updated or altered. Companies however are far from immutable objects and subject to changes which are not reflected in the data retrieved from prior commits. For example, a very small number of companies in our data set were since acquired by other companies, most notably Sun Microsystems Inc. which has been acquired by ORACLE in 2010 [3]. For the purposes of this study, these results were judged to be correct, though the authors recognize that this interpretation is tenuous at best. Additionally, a number of the domains appear to be no longer in use. It is unknown whether these domains were in use at the time of the commit, or were never valid to begin with. Some users responded to the survey with a more specific statement detailing that while they had made these contributions as an employee of the company named, they had now moved on or the company had gone out of business in the meantime. For the purpose of this study, these replies were considered to be a "correct" validation.

A number of results are not companies in the strict, incorporated sense of the word. Some contributors use websites, mostly blogs, to contribute to the Eclipse project mostly under made-up names such as "Paragraph Lost". For the purpose of this study, we have identified these results as correct as the given name correctly establishes the affiliation of the contributor. However, the contributor might be associated with a software development company in the same ecosystem, presenting some interesting challenges which we discuss in more depth in the section on Future Work.

In some cases the script proposes candidate names that are too specific, for example returning "Business Consulting", a specific study track at the University of Illinois instead of returning the full name of the university. The correctness of these cases is determined

on a case by case basis dependent on whether the parent company is clearly identifiable: "Microsoft France" is accepted, "Business Informatics" or "R&D" are not.

WHOIS records are not consistent. Content of the records vary with each top-level domain name and each country has different procedures and requirements. In practice the usage of the administrative and technical contact differ too. In some cases the administrative contact is the company and the technical contact the Internet Service Provider. In other cases the administrative contact lists only an employee who is (presumably) responsible for the domain and the IT department as its technical contact. These effects make identifying the correct company more difficult.

6 Conclusions

In a case study of the software ecosystem around Eclipse, it is possible to identify the employer of a contributor in 41.4% of cases with 85.7% accuracy, based on the e-mail address of the contributor. The main barriers to identification are users supplying invalid data such as non-existent or syntactically invalid e-mail addresses, and users using generic e-mail providers such as Gmail and Hotmail.

References

1. Jones, B. T. (2013). A list of free email provider domains. Retrieved November 5, 2014 from the Github Web site: https://gist.github.com/tbrianjones/5992856
2. Wikipedia (2014). Rational Software. Retrieved November 5, 2014 from the Wikipedia Web site: http://en.wikipedia.org/wiki/Rational_Software
3. Wikipedia (2014). Sun Microsystems. Retrieved November 5, 2014 from the Wikipedia Web site: http://en.wikipedia.org/wiki/Sun_Microsystems
4. Mozilla (2014). Public Suffix List. Retrieved November 5, 2014 from the Mozilla Web site: https://wiki.mozilla.org/Public_Suffix_List
5. Andersen-Gott, M., Ghinea, G., & Bygstad, B. (2012). Why do commercial companies contribute to open source software?. International Journal of Information Management, 32(2), 106-117.
6. Aarnoutse, F., Renes, C., Snijders, R., & Jansen, S. (2014, August). The Reality of an Associate Model: Comparing Partner Activity in the Eclipse Ecosystem. In Proceedings of the 2014 European Conference on Software Architecture Workshops (p. 8). ACM.
7. Sanfilippo, S. (July 15, 2015). Redis Creator, Salvatore Sanfilippo (Antirez), Joins Redis Labs. Retrieved July 17, 2015 from `https://redislabs.com/press-releases/redis-creator-salvatore-sanfilippo-antirez-joins-redis-labs`.
8. Milinkovich, M. (November 3, 2011). IBM and Eclipse: A Decade of Software Innovation. Retrieved July 17, 2015 from `http://asmarterplanet.com/blog/2011/11/ibm_and_eclipse_10_years.html`.