A Dataset of Enterprise-Driven Open Source Software: Extended Description

Diomidis Spinellis, Zoe Kotti, Konstantinos Kravvaritis, Georgios Theodorou, and Panos Louridas

Athens University of Economics and Business

April 21, 2020

Abstract

We present a dataset of open source software developed mainly by enterprises rather than volunteers. This can be used to address known generalizability concerns, and, also, to perform research on open source business software development. Based on the premise that an enterprise's employees are likely to contribute to a project developed by their organization using the email account provided by it, we mine domain names associated with enterprises from open data sources as well as through white- and blacklisting, and use them through three heuristics to identify 17 264 enterprise GitHub projects. We provide these as a dataset detailing their provenance and properties. A manual evaluation of a dataset sample shows an identification accuracy of 89%. Through an exploratory data analysis we found that projects are staffed by a plurality of enterprise insiders, who appear to be pulling more than their weight, and that in a small percentage of relatively large projects development happens exclusively through enterprise insiders.

Keywords: Software engineering economics, software ecosystems, open source software in business, Fortune Global 500, SEC 10-K, SEC 20-F, EDGAR, dataset

This is a technical note expanding reference [49], which should be cited in preference to this text.

1 Introduction

Despite the size and wealth of software product and process data available on GitHub, their use in software engineering research can be problematic [31, 10], raising issues regarding the generalizability of the corresponding findings [51]. In particular, the open source nature of accessible GitHub repositories means that projects developed by volunteers through open source software development processes [13, 46] are overrepresented, biasing results, especially those related to software architecture or communication and organization structures, through the application of Conway's Law [9, 26]. In addition, many researchers are investigating differences between open source and proprietary software products and processes [42, 37, 47, 3].

Here we present a dataset of open source software developed mainly by enterprises rather than volunteers. This can be used to address the identified generalizability concerns and, also, to perform research on the differences between volunteer and business software development. One might think that open source software development by enterprises is a niche phenomenon. As others have identified [44] and also as is evident from our dataset, this is far from true. A series of queries on GitHub PushEvents published during 2017 found that companies such as Microsoft and Google had hundreds of employees contributing to open source projects [28].

The goal of the dataset's construction is to create a set of GitHub projects that are most probably developed by an enterprise. For the purposes of this work, we define as an enterprise project, one that is likely to be mainly developed by financially compensated employees, working full time under an organization's management. This definition excludes volunteer efforts such as Linux, KDE projects, VLC, and GIMP (even though some companies pay their employees to contribute to them), but includes for-profit company and funded public-sector organization projects that accept volunteer contributions, such as Google's Trillian, Apple's Swift, CERN's ALICE, and Microsoft's Typescript. Our aim is to minimize the number of false positives in the dataset, but we are not interested in the number of false negatives. We do not aspire to create a comprehensive dataset of enterprise projects, but one that contains a number sufficient to conduct generalizable empirical studies.

In the following sections we present how we collected the data (Section 2) and evaluated them (Section 3), the data schema and availability (Section 4), as well as indicative findings (Section 5), related work (Section 6), and ideas for research and improvements (Section 7).

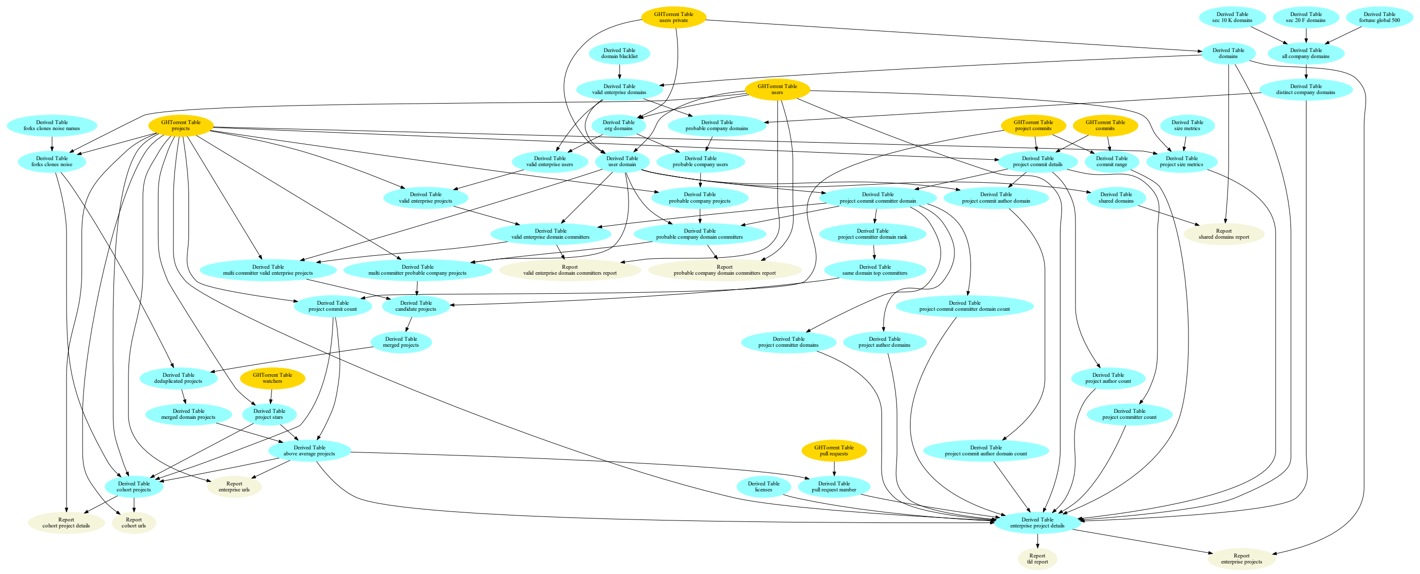
GHTorrent Table users private
Derived Table sec 10 K domains
Derived Table sec 20 F domains
Derived Table fortune global 500
Derived Table domain blacklist
Derived Table domains
Derived Table all company domains
Derived Table valid enterprise domains
GHTorrent Table users
Derived Table distinct company domains
Derived Table forks clones noise names
GHTorrent Table projects
Derived Table org domains
Derived Table probable company domains
GHTorrent Table project commits
GHTorrent Table commits
Derived Table size metrics
Derived Table forks clones noise
Derived Table valid enterprise users
Derived Table user domain
Derived Table probable company users
Derived Table project commit details
Derived Table commit range
Derived Table project size metrics
Derived Table valid enterprise projects
Derived Table probable company projects
Derived Table project commit committer domain
Derived Table project commit author domain
Derived Table shared domains
Derived Table valid enterprise domain committers
Derived Table probable company domain committers
Derived Table project committer domain rank
Report shared domains report
Derived Table multi committer valid enterprise projects
Derived Table multi committer probable company projects
Report valid enterprise domain committers report
Report probable company domain committers report
Derived Table same domain top committers
Derived Table project commit count
Derived Table candidate projects
Derived Table project commit committer domain count
Derived Table merged projects
Derived Table project committer domains
Derived Table project author domains
Derived Table deduplicated projects
GHTorrent Table watchers
Derived Table project author count
Derived Table merged domain projects
Derived Table project stars
Derived Table project committer count
Derived Table above average projects
GHTorrent Table pull requests
Derived Table project commit author domain count
Derived Table cohort projects
Report enterprise urls
Derived Table licenses
Derived Table pull request number
Report cohort project details
Report cohort urls
Derived Table enterprise project details
Report tld report
Report enterprise projects

Figure 1: Overview of the dataset creation process

2 Dataset Construction

An overview of the dataset's construction process is depicted in Figure 1. The projects were selected from GitHub by analyzing the GHTorrent [20, 19] dataset (release 2019-06-01) by means of the *simple-rolap* relational online analytical processing and *rdbunit* relational unit testing frameworks [21]. Following published recommendations [30], the code and primary data associated with this endeavor are openly available online.[1]

The basic premise for constructing the dataset is that an enterprise's employees are likely to contribute to a project developed by their organization using the email account provided by it. Furthermore, it is unlikely that pure volunteer projects will have contributors using emails from a single enterprise-related domain address. Based on this premise, we identified projects where a large number of commits were contributed through accounts linked to the same enterprise email domain address. To increase the dataset's quality we then removed project clones [50], and only retained projects having more than the identified dataset's average stars (14) and commits (29). Finally, we created one table with diverse details regarding each selected project and one with details regarding each associated enterprise domain. The following paragraphs detail each step, starting from the creation of two tables: *valid enterprise domains* and *probable company domains*.[2]

Valid enterprise domains This table (1 922 492 records, Listing 38) was created by filtering all email domains found in the users' email table (Table *domains*—3 899 753 records, Listing 8). We did this by examining frequently occurring email domains, and creating rules to retain only those associated with enterprise development. Specifically, we removed from the set of domains a blacklist (Table *domain blacklist*—146 records, Listing 7) containing those associated with:

- email providers (e.g. *gmail.com, qq.com, outlook.com, yandex.ru*);
- top and second level organization domains (e.g. *.org, .org.nz, .or.at*), and thereby the many associated with volunteer open source organizations (e.g. *apache.org, openssl.org*);
- open source hubs (e.g. *sourceforge.net*);

[1] `https://doi.org/10.5281/zenodo.3742973`

[2] In the interest of readability, this text replaces the underscores in the table names with spaces.

- top and second level educational domains (e.g. *.edu, .edu.au, .ac.uk*) and, explicitly, the domains of more than 20 hand-picked universities (e.g. *eurecom.fr, tu-dortmund.de*);
- individuals (e.g. *schildbach.de*).

We did not remove government organizations (e.g. *lexingtonky.gov*) and research centers (e.g. *cern.ch*) as these mainly operate as enterprises with professional developers. When in doubt, we looked up company emails in the RocketReach provider of company email format details.

Probable company domains This table (786 099 records, Listing 18) was created by identifying domains that are likely to belong to companies from publicly available data and domain heuristics. We obtained the domains associated with large companies in two ways. First, we screen-scrapped, downloaded, and filtered the data associated with the Fortune Global 500 companies: the largest corporations across the globe measured by revenue (Table *fortune global 500*—499 records, Listing 10). Second, we obtained the US Securities and Exchange Commission (SEC) yearly company filings that are made in machine readable form (in the XBRL—eXtensible Business Reporting Language—an application of XML) and extracted from them the company domains. Specifically, we obtained from EDGAR—the SEC's Electronic Data Gathering, Analysis, and Retrieval system—the XBRL files associated with two forms, namely a) Form 10-K, that gives a comprehensive summary of a company's financial performance (Table *sec 10 K domains*—5 597 records, Listing 33), and b) Form 20-F, that provides an annual report filing for foreign private issuers—non-U.S. and non-Canadian companies that have securities trading in the U.S. (Table *sec 20 F domains*—599 records, Listing 34).

We then extracted the internet domain (e.g. *intel.com*) associated with each company from the XBRL files. Although the SEC provides guidance for using a corporate web site to disseminate public information,[3] it does not appear to collect these sites in a structured manner, within e.g. the XBRL files. We obtained the company domains by looking at the XML name space used in the files, which in most cases contains the company's domain. We combined the three sources into the Table *distinct company domains* (6 191 records, Listing 6) and complemented it with the Table *valid enterprise domains* (1 922 492 records, Listing 38) filtered to include

[3]`https://www.sec.gov/rules/interp/2008/34-58288.pdf`

only records associated with top and second level commercial domains such as *.com*, *.co.uk, .com.au.*

From enterprise organizations to their projects As a next step we combined the two tables with another listing domains registered for GitHub organizations (Table *org domains*—169 281 records, Listing 16), to get tables with user domains linked to GitHub organizations—Tables *valid enterprise users* (110 116 records, Listing 40) and *probable company users* (62 427 records, Listing 20). The intuition here is that many companies developing software on GitHub will have configured a company organization under their domain name. Combining the two tables with the GHTorrent *Projects* table yielded the corresponding projects hosted under a GitHub organization: *valid enterprise projects* (1 332 891 records, Listing 39) and *probable company projects* (756 136 records, Listing 19).

These two tables were then linked with a table of each user's email domain (Table *user domain*—32 411 734 records, Listing 36) and one identifying each commit's committer (Table *project commit committer domain*—523 434 215 records, Listing 24), giving the number of committers in each project associated with the corresponding organization: *valid enterprise domain committers* (32 278 records, Listing 37) and *probable company domain committers* (18 804 records, Listing 17). This stage ended by selecting projects from organizations having a minimum number of committers appearing on GitHub with an email associated with the organization's domain giving the tables *multi committer valid enterprise projects* (132 886 records, Listing 15) and *multi committer probable company projects* (164 470 records, Listing 14). The employed floor values (ten and five correspondingly) were selected to exclude projects associated with individuals operating under a personal but commercial-looking domain (e.g. *johnsmith.com*).

Enterprise-dominated projects To cover enterprises that may not have GitHub organizations registered with emails under their domain, we also established in each project a rank of committers with valid enterprise email addresses according to their number of commits (Table *project committer domain rank*—20 953 718 records, Listing 28), and obtained those projects having committers from the same organizations as the topmost three (Table *same domain top committers*—99 368 records, Listing 32).

Final filtering and reporting For the three types of possible enterprise projects we then formed their union (Table *candidate projects*—396 724 records, Listing 2), combined their metrics (Table *merged projects*—293 172 records, Listing 13), removed duplicate projects (Table *deduplicated projects*—168 306 records, Listing 5), combined records referring to the same project (Table *merged domain projects*—167 958 records, Listing 12), and joined them with the number of their commits (Table *project commit count*—59 172 876 records, Listing 26) and their stars (Table *project stars*—10 317 662 records, Listing 31), to select those with above average such metrics (Table *above average projects*—17 673 records, Listing 1). For each one of the shortlisted projects, we `git-cloned` from GitHub the project's repository and calculated its basic size metrics in terms of files and text lines (Table *size metrics*—16 852 records, Listing 35). (Due to churn from the date the GHTorrent dataset was published, not all repositories could be retrieved for measuring project size.)

Finally, to provide context for each project, we combined this table with each project's earliest and most recent commit (Table *commit range*—100 366 312 records, Listing 4), number of commits (Table *project commit committer domain count*—17 246 572 records, Listing 25) and committers (Table *project committer domains*—17 246 572 records, Listing 29) for each committer domain, number of commits (Table *project commit author domain count*—23 866 053 records, Listing 23) and committers (Table *project author domains*—17 246 572 records, Listing 22) for each author domain, total number of committers (Table *project committer count*—59 172 876 records, Listing 27) and authors (Table *project author count*—59 172 876 records, Listing 21), size metrics (Table *project size metrics*—15 613 records, Listing 30), project license as provided by the GitHub API (Table *licenses*—16 850 records, Listing 11), as well as details about the derivation of the corresponding domain. This process created the table *enterprise project details* (17 264 records, Listing 9) and the corresponding report *enterprise projects*.

3 Evaluation

We manually evaluated a random sample of an earlier version of this dataset,[4] following the systematic review guidelines by Brereton et al. [7]. The sample size was calculated at around 378 using Cochran's sample size and correction formulas [8] (95% confidence, 5% precision). To keep the raters alert

[4] `https://doi.org/10.5281/zenodo.3653878` and `https://doi.org/10.5281/zenodo.3653888`. This was updated following the peer review suggestions, and differs by 64 projects (0.37%—26 removed, 38 added) from the currently supplied one.

we complemented the sample with 22 GitHub projects randomly selected from a set of projects with similar quality characteristics that were part of the dataset (Table *cohort projects*—311 223 records, Listing 3). The third and fourth authors were instructed to individually label the 400 projects as enterprise or not based on the definition in Section 1. To improve the labeling's reliability the two raters did not know the employed heuristics, and were also asked to complete the main reason the project was open source and write a few words to support their decision. Their ratings led to 78% inter-rater agreement and 29% reliability using Cohen's kappa statistic. The second author then resolved the conflicts by majority vote; after excluding the 22 irrelevant projects, 89% of the 378 projects were finally identified as enterprise. We used the bootstrap method [12] with 1000 iterations to establish a confidence interval (CI) for the percentage of enterprise projects in our sample; the 95% CI was calculated at [87–93]%. To generalize, 15 354 (CI: 15 009–16 044) projects of our dataset are expected to be truly enterprise-developed.

Regarding the dataset's external validity, note that although our evaluation addresses the dataset's precision, our method was not targeting a high recall and this was also not evaluated. Consequently, the dataset can be used to address empirical research generalizability concerns we identified in the introduction mainly by providing a set of enterprise-developed projects to be used in work employing stratified sampling, in cohort studies, or in case studies. Furthermore, the number of committers floor we employed in our selection means that the dataset excludes organizations that are small or have a tiny number of their employees committing on GitHub. Finally, the selection of above average projects in terms of stars and commits means that the dataset does not include stillborn or unpopular projects.

4 Dataset Overview

The dataset[5] is provided as a 17 264 record tab-separated file with the following 29 fields:

url the project's GitHub URL;

project_id the project's GHTorrent identifier;

sdtc true if selected using the same domain top committers heuristic (9 016 records);

[5]`https://doi.org/10.5281/zenodo.3742962`

mcpc true if selected using the multiple committers from a valid enterprise heuristic (8 314 records);

mcve true if selected using the multiple committers from a probable company heuristic (8 015 records);

star_number number of GitHub watchers;

commit_count number of commits;

files number of files in current main branch;

lines corresponding number of lines in text files;

pull_requests number of pull requests;

github_repo_creation time stamp of the GitHub repository creation;

earliest_commit time stamp of the earliest commit;

most_recent_commit time stamp of the most recent commit;

committer_count number of different committers;

author_count number of different authors;

dominant_domain the project's dominant email domain;

dominant_domain_committer_commits number of commits made by committers whose email matches the project's dominant domain;

dominant_domain_author_commits corresponding number for commit authors;

dominant_domain_committers number of committers whose email matches the project's dominant domain;

dominant_domain_authors corresponding number for commit authors;

cik SEC's EDGAR "central index key";

fg500 true if this is a Fortune Global 500 company (2 233 records);

sec10k true if the company files SEC 10-K forms (4 180 records);

sec20f true if the company files SEC 20-F forms (429 records);

project_name GitHub project name;

owner_login GitHub project's owner login;

company_name company name as derived from the SEC and Fortune 500 data;

owner_company GitHub project's owner company name;

license SPDX license identifier.

An additional file provides the full set of 311 223 cohort projects (not part of the enterprise dataset), selected as described in Section 3, with the following four fields:

url the project's GitHub URL;

project_id the project's GHTorrent identifier;

stars number of GitHub watchers;

commit_count number of commits.

5 Typology of Enterprise OSS

We performed a preliminary analysis of the details we collected to obtain a picture of how enterprise software is developed. Overall, we see that projects are staffed by a plurality of enterprise insiders, who appear to be pulling more than their weight. Regarding the distribution of contributors, across all identified projects in the dataset we found that 33% of the authors and 24% of the committers are associated with the project's dominant domain. Similarly, regarding the distribution of work, 45% of the commits are made by the enterprise's authors, and 41% of the commits are made by the corresponding committers.

The ten most popular out of the 110 top level domains associated with projects are: *com* (13 494 projects), *io* (763), *de* (383), *gov* (339), *net* (256), *ru* (142), *fr* (134), *cn* (120), *br* (118), and *uk* (111). Similarly, out of 5 097 owners, those associated with the highest number of GitHub projects are: *Microsoft* (855 projects), *Azure* (328), *google* (123), *twitter* (93), *18F* (90), *udacity* (82), *SAP* (79), *Netflix* (79), *hashicorp* (77), and *GoogleCloudPlatform* (77).

In very few projects does development appear to be exclusively controlled by the enterprise: we found 90 projects (0.5%) where all commits came

Table 1: Enterprise (E) and Reaper (R) Dataset Metrics

	Min		Max (k)		Avg		Stddev	
Metric	E	R	E	R	E	R	E	R
Stars	15	0	80	51	355	11	1661	221
Commits	30	0	304	383	1159	70	5323	1196
PRs	0	0	25	42	161	3	672	94
Authors	1	0	26	5	27	2	213	10
Committers	1	0	26	5	22	2	208	7

from an enterprise committer and 220 projects (1.3%) where all commits came from an enterprise author. We were expecting these projects to be small, but in fact they sport an average line count of 453k for projects with exclusively enterprise authors and 976k for projects with exclusively enterprise committers. Considerable development seems to happen through pull requests, with 95% of the projects having pull requests associated with them, with an average of 161 pull requests per project.

In total, according to their SPDX [17, 32] identifiers, the projects are licensed using 29 different open source licenses. The two most common licenses used are the MIT (4 340 projects) and Apache 2.0 (3 761 projects), with the GPL version 2 or 3 license used only by 780 projects. This finding indicates that few enterprise open source projects seem to follow a business model based on relicensing GPL code for proprietary development. Surprisingly, for 3 535 projects no license was found, and for 3 374 projects the license did not match one with an SPDX identifier.

We compared the earlier version of this dataset mentioned in Section 3 against the Reaper dataset of engineered software projects [40] in terms of stars, commits, pull requests (PRs), authors, and committers (see Table 1). Reaper initially contained 1 853 205 projects in the form *login-name/project-name*, from which 1 849 500 were successfully associated with a project ID of GHTorrent. Null values were substituted with zero in both datasets, thus metrics were calculated on the basis of the entire dataset sizes (17 252 for this, 1 849 500 for the Reaper). It appears that in all dimensions this dataset is considerably richer than the Reaper one. The difference most likely stems from this dataset's considerable selectivity, as it contains two orders of magnitude fewer projects than Reaper.

6 Related Work

While the relationship between academic or semi-academic institutions and open source software has been favorable [33], with large open source projects such as the Berkeley Software Distribution (BSD) [43] originating from them, this has not always been the case for business. The relationship between business and open source software was often tense in the past, with GPL-licensed software described as "an intellectual property destroyer", un-American, and "a cancer" [38]. Meanwhile, others asserted that open source was compatible with business [24], and researchers quickly identified several business models that are based on open source software [5, 1], as well as significant industrial adoption of open source software products [48]. In short, research associated with the involvement of enterprises in open source software can be divided into four areas [29]: a) company participation in open source development communities [6, 25]; b) business models with open source in commercial organizations [5, 22]; c) open source as part of component based software engineering [2, 34]; and d) usage of open source processes within a company [35, 14].

We consider our study part of the first area. According to Bonaccorsi et al. [6], companies participated in one third of the most active projects on SourceForge as project coordinators, collaborators in code development, or code providers. Hauge et al. [23] also identified the role of component integrator. By providing their proprietary software to the open source community, companies can benefit from reduced development costs, advanced performance, repositioning in the market, and additional profit from new services [29]. Still, the provided software should be accompanied by adequate documentation and information to help the community members engage in it [23].

Although companies marginally participated in open source projects in the past, the participation has recently increased, especially in the larger and more active projects, with a crucial part of the open source code being provided by commercial organizations, particularly small and medium-sized enterprises (SMEs) [36]. For instance, 6%–7% of the code in the Debian GNU/Linux distribution over the period 1998–2004 was contributed by corporations [45].

Bird et al. in their study regarding email social networks [4] faced the challenge of duplicate email aliases while matching identities of the email archives of Apache to identities of Concurrent Version System (CVS) repositories. The issue was resolved by extracting email headers that included the sender to produce a list of $< name, email >$ identifiers. The similarity of the identifier pairs of the list was then computed through a clustering algorithm, and the resulting clusters were manually evaluated.

Similarly, German and Mockus [15] linked identical contributors of CVS repositories with multiple names or emails of different spelling. Using their infrastructure they identified the top contributors of the Ximian Evolution project, and found that the top ten contributors were Ximian employees and consultants, and also that private companies such as RedHat, Ximian and Eazel, severely affected the development of the GNOME project [16], similarly to the way the Mozilla project was mainly developed by Netscape employees [39].

7 Research Ideas

The provided dataset can be employed in various ways. First, it can be used to study the involvement of enterprises in OSS development by examining whether they are mostly *takers* or *givers*, their roles within projects, and how they shape a project's evolution and success [6]. Second, it can be employed in studies regarding OSS business models, to investigate how their choice is affected by different enterprise characteristics such as the employees' education level, the enterprise's age, size, service variety, and whether it is family-owned or not [22]. Third, it can be used for research on the composition and structure of OSS supply chains and value chains, particularly to identify the added, deleted, and unchanged dependencies and their effect between releases for different types of packages such as build and test tools [11]. Furthermore, it can be employed in studies concerning enterprise-driven global software development, to measure benefits and tackle issues induced from the physical separation among project members such as strategic, cultural, communication, and knowledge management issues [27]. Another use involves identifying product or process differences between enterprise and volunteer-driven software development in terms of cost, service and support, innovation, security, usability, standards, availability, transparency, and reliability [41]. Finally, it can be used to study enterprise regulatory, compliance, and supply chain risks, to investigate the risk domains that enterprises face when engaging in OSS development, the available sources of risk mitigation, and the heuristics by which managers apply this understanding to manage such projects. From these insights, formalized risk mitigation instruments and project management processes can be developed [18].

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825328.

D. Spinellis created the dataset and its unit tests. Z. Kotti evaluated the rating results and researched related work. Both contributed equally to the paper's writing. K. Kravvaritis and G. Theodorou rated the dataset's repositories. P. Louridas implemented the bootstrap method.

References

[1] Stephanos Androutsellis-Theotokis, Diomidis Spinellis, Maria Kechagia, and Georgios Gousios. 2011. Open Source Software: A Survey from 10,000 Feet. *Foundations and Trends in Technology, Information and Operations Management* 4, 3–4 (2011), 187–347. https://doi.org/10.1561/0200000026

[2] Claudia Ayala, Øyvind Hauge, Reidar Conradi, Xavier Franch, Jingyue Li, and Ketil Velle. 2009. Challenges of the Open Source Component Marketplace in the Industry, Vol. 299. 213–224. https://doi.org/10.1007/978-3-642-02032-2_19

[3] Adrian Bachmann and Abraham Bernstein. 2009. Software Process Data Quality and Characteristics: A Historical View on Open and Closed Source Projects. In *Proceedings of the Joint International and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) Workshops (IWPSE-Evol '09)*. Association for Computing Machinery, New York, NY, USA, 119–128. https://doi.org/10.1145/1595808.1595830

[4] Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and Anand Swaminathan. 2006. Mining Email Social Networks. In *Proceedings of the 2006 International Workshop on Mining Software Repositories (MSR '06)*. ACM, New York, NY, USA, 137–143. https://doi.org/10.1145/1137983.1138016

[5] Andrea Bonaccorsi, Silvia Giannangeli, and Cristina Rossi. 2006. Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry. *Management science* 52, 7 (2006), 1085–1098.

[6] Andrea Bonaccorsi, Dario Lorenzi, Monica Merito, and Cristina Rossi. 2007. Business Firms' Engagement in Community Projects. Empirical Evidence and Further Developments of the Research. In *Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS '07)*. IEEE Computer Society, USA, 13. `https://doi.org/10.1109/FLOSS.2007.3`

[7] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. 2007. Lessons from Applying the Systematic Literature Review Process Within the Software Engineering Domain. *J. Syst. Softw.* 80, 4 (April 2007), 571–583. `https://doi.org/10.1016/j.jss.2006.07.009`

[8] William G. Cochran. 1977. *Sampling Techniques* (3rd ed.). John Wiley & Sons, Inc., USA.

[9] Melvin E Conway. 1968. How do Committees Invent? *Datamation* 14, 4 (1968), 28–31.

[10] V. Cosentino, J. L. C. Izquierdo, and J. Cabot. 2016. Findings from GitHub: Methods, Datasets and Limitations. In *MSR 2016: IEEE/ACM 13th Working Conference on Mining Software Repositories*. 137–141.

[11] Tapajit Dey and Audris Mockus. 2018. Are Software Dependency Supply Chain Metrics Useful in Predicting Change of Popularity of npm Packages?. In *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering*. 66–69.

[12] B. Efron. 1979. Bootstrap Methods: Another Look at the Jackknife. *Ann. Statist.* 7, 1 (Jan. 1979), 1–26. `https://doi.org/10.1214/aos/1176344552`

[13] Joseph Feller, Brian Fitzgerald, et al. 2002. *Understanding Open Source Software Development*. Addison-Wesley, London, UK.

[14] Gary Gaughan, Brian Fitzgerald, and Maha Shaikh. 2009. An Examination of the Use of Open Source Software Processes as a Global Software Development Solution for Commercial Software Engineering. In *Proceedings of the 2009 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA '09)*. IEEE Computer Society, USA, 20–27. `https://doi.org/10.1109/SEAA.2009.86`

[15] Daniel German and Audris Mockus. 2003. Automating the Measurement of Open Source Projects. *Proceedings of the 3rd Workshop on Open Source Software Engineering* (Jan. 2003).

[16] Daniel M. German. 2002. The Evolution of the GNOME Project. In *Proceedings of the 2nd Workshop on Open Source Software Engineering.* 4. `https://flosshub.org/sites/flosshub.org/files/German.pdf`

[17] Matt Germonprez, Julie E Kendall, Kenneth E Kendall, and Brett Young. 2014. Collectivism, Creativity, Competition, and Control in Open Source Software Development: Reflections on the Emergent Governance of the SPDX Working Group. *International Journal of Information Systems and Management* 1, 1-2 (2014), 125–145.

[18] Matt Germonprez, Brett Young, Lars Mathiassen, Julie E Kendall, Ken E Kendall, and Warner Brian. 2012. Risk Mitigation in Corporate Participation with Open Source Communities: Protection and Compliance in an Open Source Supply Chain. In *Proceedings of the 7th International Research Workshop on IT Project Management (IRWITPM '12).*

[19] Georgios Gousios. 2013. The GHTorrent dataset and tool suite. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR '13)*. IEEE Press, Piscataway, NJ, USA, 233–236. `https://doi.org/10.5555/2487085.2487132`

[20] Georgios Gousios and Diomidis Spinellis. 2012. GHTorrent: Github's Data from a Firehose. In *9th IEEE Working Conference on Mining Software Repositories (MSR)*, Michele Lanza, Massimiliano Di Penta, and Tao Xie (Eds.). IEEE, 12–21. `https://doi.org/10.1109/MSR.2012.6224294`

[21] Georgios Gousios and Diomidis Spinellis. 2017. Mining Software Engineering Data from GitHub. In *Proceedings of the 39th International Conference on Software Engineering Companion (ICSE-C '17)*. IEEE Press, Piscataway, NJ, USA, 501–502. `https://doi.org/10.1109/ICSE-C.2017.164` Technical Briefing.

[22] Elad Harison and Heli Koski. 2010. Applying open innovation in business strategies: Evidence from Finnish software firms. *Research Policy* 39, 3 (April 2010), 351–359. `https://doi.org/10.1016/j.respol.2010.01.008`

[23] Øyvind Hauge, Carl-Fredrik Sørensen, and Andreas Røsdal. 2007. Surveying Industrial Roles in Open Source Software Development. In *Open Source Development, Adoption and Innovation*, Joseph Feller, Brian Fitzgerald, Walt Scacchi, and Alberto Sillitti (Eds.). Springer US, Boston, MA, 259–264.

[24] Frank Hecker. 1999. Setting up Shop: The Business of Open-Source Software. *IEEE software* 16, 1 (1999), 45–51.

[25] Joachim Henkel. 2008. Champions of revealing—the role of open source developers in commercial firms. *Industrial and Corporate Change* 18, 3 (Dec. 2008), 435–471. `https://doi.org/10.1093/icc/dtn046` arXiv:https://academic.oup.com/icc/article-pdf/18/3/435/2415321/dtn046.pdf

[26] James D. Herbsleb and Rebecca E. Grinter. 1999. Splitting the organization and integrating the code: Conway's law revisited. In *ICSE '99: Proceedings of the 21st international conference on Software engineering.* IEEE Computer Society Press, Los Alamitos, CA, USA, 85–95.

[27] James D Herbsleb and Deependra Moitra. 2001. Global Software Development. *IEEE software* 18, 2 (2001), 16–20.

[28] Felipe Hoffa. 2017. Who contributed the most to open source in 2017 and 2018? Let's analyze GitHub's data and find out. Available online `https://medium.com/@hoffa/the-top-contributors-to-github-2017-be98ab854e87`. Accessed January 25th, 2020. Optional.

[29] Martin Höst and Alma Oručeviundefined-Alagiundefined. 2011. A Systematic Review of Research on Open Source Software in Commercial Software Product Development. *Inf. Softw. Technol.* 53, 6 (June 2011), 616–624. `https://doi.org/10.1016/j.infsof.2010.12.009`

[30] Darrel C Ince, Leslie Hatton, and John Graham-Cumming. 2012. The Case for Open Computer Programs. *Nature* 482, 7386 (2012), 485–488.

[31] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2016. An In-Depth Study of the Promises and Perils of mining GitHub. *Empirical Software Engineering* 21, 5 (01 Oct. 2016), 2035–2071. `https://doi.org/10.1007/s10664-015-9393-5`

[32] Georgia M Kapitsaki, Frederik Kramer, and Nikolaos D Tselikas. 2017. Automating the License Compatibility Process in Open Source Software with SPDX. *Journal of Systems and Software* 131 (2017), 386–401.

[33] Josh Lerner and Jean Tirole. 2001. The open source movement: Key research questions. *European Economic Review* 45, 4-6 (May 2001), 819–826. `https://ideas.repec.org/a/eee/eecrev/v45y2001i4-6p819-826.html`

[34] Jingyue Li, Reidar Conradi, Christian Bunse, Marco Torchiano, Odd Petter N. Slyngstad, and Maurizio Morisio. 2009. Development with Off-the-Shelf Components: 10 Facts. *IEEE Softw.* 26, 2 (March 2009), 80–87. `https://doi.org/10.1109/MS.2009.33`

[35] Juho Lindman, Matti Rossi, and Pentti Marttiin. 2008. Applying Open Source Development Practices Inside a Company. In *Open Source Development, Communities and Quality*, Barbara Russo, Ernesto Damiani, Scott Hissam, Björn Lundell, and Giancarlo Succi (Eds.). Springer US, Boston, MA, 381–387.

[36] Bjørn Lundell, Brian Lings, and Edvin Lindqvist. 2006. Perceptions and Uptake of Open Source in Swedish Organisations. In *Open Source Systems*, Ernesto Damiani, Brian Fitzgerald, Walt Scacchi, Marco Scotto, and Giancarlo Succi (Eds.). Springer US, Boston, MA, 155–163.

[37] Alan MacCormack, John Rusnak, and Carliss Y. Baldwin. 2006. Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code. *Management Science* 52, 7 (2006), 1015–1030. `https://doi.org/10.1287/mnsc.1060.0552`

[38] Joseph Scott Miller. 2002. Allchin's Folly: Exploring Some Myths About Open Source Software. *Cardozo Arts & Entertainment Law Journal* 20 (2002), 491.

[39] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. 2002. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 11, 3 (2002), 309–346. `https://doi.org/doi:10.1145/567793.567795`

[40] Nuthan Munaiah, Steven Kroh, Craig Cabrey, and Meiyappan Nagappan. 2017. Curating GitHub for Engineered Software Projects. *Em-*

pirical Software Engineering 22, 6 (01 Dec 2017), 3219–3253. `https://doi.org/10.1007/s10664-017-9512-6`

[41] N Pankaja and PK Mukund Raj. 2013. Proprietary Software versus Open Source Software for Education. *American Journal of Engineering Research* 2, 7 (2013), 124–130.

[42] James W. Paulson, Giancarlo Succi, and Armin Eberlein. 2004. An Empirical Study of Open-Source and Closed-Source Software Products. *IEEE Transactions on Software Engineering* 30, 4 (April 2004), 246–256.

[43] Eric S. Raymond. 1999. *The Cathedral and the Bazaar* (1st ed.). O'Reilly & Associates, Inc., USA.

[44] Carol A Robbins, Gizem Korkmaz, José Bayoán Santiago Calderón, Daniel Chen, Claire Kelling, Stephanie Shipp, and Sallie Keller. 2018. Open Source Software as Intangible Capital: Measuring the Cost and Impact of Free Digital Tools. In *Paper from 6th IMF Statistical Forum on Measuring Economic Welfare in the Digital Age: What and How*. 19–20.

[45] Gregorio Robles, Santiago Dueñas, and Jesus Gonzalez-Barahona. 2007. Corporate Involvement of Libre Software: Study of Presence in Debian Code over Time, Vol. 234. 121–132. `https://doi.org/10.1007/978-0-387-72486-7_10`

[46] Walt Scacchi, Joseph Feller, Brian Fitzgerald, Scott Hissam, and Karim Lakhani. 2006. Understanding Free/Open Source Software Development Processes. *Software Process: Improvement and Practice* 11, 2 (2006), 95–105. `https://doi.org/10.1002/spip.255`

[47] Diomidis Spinellis. 2008. A Tale of Four Kernels. In *ICSE '08: Proceedings of the 30th International Conference on Software Engineering*, Wilhelm Schäfer, Matthew B. Dwyer, and Volker Gruhn (Eds.). Association for Computing Machinery, New York, 381–390. `https://doi.org/10.1145/1368088.1368140`

[48] Diomidis Spinellis and Vaggelis Giannikas. 2012. Organizational Adoption of Open Source Software. *Journal of Systems and Software* 85, 3 (March 2012), 666–682. `https://doi.org/10.1016/j.jss.2011.09.037`

[49] Diomidis Spinellis, Zoe Kotti, Konstantinos Kravvaritis, Georgios Theodorou, and Panos Louridas. 2020. A Dataset of Enterprise-Driven Open Source Software. In *17th International Conference on Mining Software Repositories (MSR '20)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3379597.3387495

[50] Diomidis Spinellis, Zoe Kotti, and Audris Mockus. 2020. A Dataset for GitHub Repository Deduplication. In *17th International Conference on Mining Software Repositories (MSR '20)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3379597.3387496

[51] Hyrum K. Wright, Miryung Kim, and Dewayne E. Perry. 2010. Validity Concerns in Software Engineering Research. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research (FoSER '10)*. Association for Computing Machinery, New York, NY, USA, 411–414. https://doi.org/10.1145/1882362.1882446

A Appendix: Key SQL Queries

Listing 1: SQL query for deriving the table *above average projects*

```sql
-- Projects with above average number of stars and commits

CREATE TABLE indoss.above_average_projects as
  select merged_domain_projects.*,
      project_stars.stars as star_number,
      project_commit_count.commit_count
    from indoss.merged_domain_projects
    inner join indoss.project_commit_count
    on project_commit_count.id = merged_domain_projects.project_id
    inner join indoss.project_stars
    on project_stars.id = merged_domain_projects.project_id
    where
      project_stars.stars > (select avg(stars) from
          indoss.project_stars) and
      project_commit_count.commit_count > (
  select avg(commit_count) from indoss.project_commit_count);
```

```sql
create index on indoss.above_average_projects(project_id);
create index on indoss.above_average_projects(domain_id);
```

Listing 2: SQL query for deriving the table *candidate projects*

```sql
-- Candidate company projects from all heuristics

CREATE TABLE indoss.candidate_projects as
  select project_id,
  -- Same domain top committers
  a as sdtc_domain,
  -- Multi-committer valid enterprise
  b as mcve_domain,
  -- Multi-committer probable company
  c as mcpc_domain
  from (
    -- The null casts are needed due to BUG #5974: UNION construct
        type cast gives poor error message
    select project_id,
      domain_id as a, null::integer as b, null::integer as c
    from indoss.same_domain_top_committers
    union all
    select project_id,
      null::integer, domain_id as b, null::integer as c
    from indoss.multi_committer_valid_enterprise_projects
    union all
    select project_id,
      null::integer, null::integer as b, domain_id as c
    from indoss.multi_committer_probable_company_projects
  ) as t;

create index on indoss.candidate_projects(project_id);
```

Listing 3: SQL query for deriving the table *cohort projects*

```sql
-- Cohort projects matching the above average projects

CREATE TABLE indoss.cohort_projects as
  select projects.id as project_id,
    project_stars.stars,
    project_commit_count.commit_count
    from projects
    inner join indoss.project_commit_count
    on project_commit_count.id = projects.id
    inner join indoss.project_stars
    on project_stars.id = projects.id
    left join indoss.forks_clones_noise
    on projects.id = forks_clones_noise.id
    left join indoss.above_average_projects
    on projects.id = above_average_projects.project_id
    where
      -- Remove project clones
      forks_clones_noise.id is null and
      -- No common projects across cohorts
      above_average_projects.project_id is null and
      project_stars.stars > (select avg(stars) from
          indoss.project_stars) and
      project_commit_count.commit_count > (
  select avg(commit_count) from indoss.project_commit_count);

create index on indoss.cohort_projects(project_id);
```

Listing 4: SQL query for deriving the table *commit range*

```sql
-- The most recent commit for each project

CREATE TABLE indoss.commit_range AS
  select project_commits.project_id as project_id,
    min(created_at) as earliest,
    max(created_at) as most_recent
  from commits
  inner join project_commits
  on project_commits.commit_id = commits.id
  group by project_commits.project_id;

create unique index on indoss.commit_range(project_id);
```

Listing 5: SQL query for deriving the table *deduplicated projects*

```
-- Probable company projects deduplicated by mean metrics

CREATE TABLE indoss.deduplicated_projects as
  select merged_projects.* from indoss.merged_projects
    left join indoss.forks_clones_noise
    on merged_projects.project_id = forks_clones_noise.id
    where
    -- Remove project clones
    forks_clones_noise.id is null;

create index on indoss.deduplicated_projects(project_id);
```

Listing 6: SQL query for deriving the table *distinct company domains*

```
-- All collected domains

create table indoss.distinct_company_domains as
select domain, max(name) as name, max(cik) as cik,
  bool_or(fg500) as fg500, bool_or(sec10k) as sec10k,
      bool_or(sec20f) as sec20f
from indoss.all_company_domains
group by domain;

create index on indoss.distinct_company_domains(domain);
```

Listing 7: SQL query for deriving the table *domain blacklist*

```
create table indoss.domain_blacklist (domain varchar(255));

\copy indoss.domain_blacklist from
    'data/domain_blacklist_input.txt';

create index on indoss.domain_blacklist(domain);
```

Listing 8: SQL query for deriving the table *domains*

```
CREATE TABLE indoss.domains AS
  select (row_number() over())::int id, domain::text
  from (
    SELECT DISTINCT LOWER(SUBSTR(email, POSITION('@' in email) +
        1)) AS domain
    FROM users_private
  ) d;
```

```sql
create index on indoss.domains(id);
create index on indoss.domains(domain);
```

Listing 9: SQL query for deriving the table *enterprise project details*

```sql
-- Details of the identified company projects

CREATE TABLE indoss.enterprise_project_details as
  select
    concat('https://github.com/', substr(projects.url, 30)) as url,
    aap.*,
    project_size_metrics.files,
    project_size_metrics.lines,
    coalesce(pull_request_number.pull_requests, 0) as pull_requests,
    projects.created_at as github_repo_creation,
    commit_range.earliest as earliest_commit,
    commit_range.most_recent as most_recent_commit,
    project_committer_count.committer_count,
    project_author_count.author_count,
    domains.domain as dominant_domain,
    coalesce(project_commit_committer_domain_count.domain_count, 0)
      as dominant_domain_committer_commits,
    coalesce(project_commit_author_domain_count.domain_count, 0)
      as dominant_domain_author_commits,
    coalesce(project_committer_domains.domain_count, 0)
      as dominant_domain_committers,
    coalesce(project_author_domains.domain_count, 0)
      as dominant_domain_authors,
    distinct_company_domains.cik,
    distinct_company_domains.fg500,
    distinct_company_domains.sec10k,
    distinct_company_domains.sec20f,
    projects.name as project_name,
    users.login as owner_login,
    distinct_company_domains.name as company_name,
    users.company as owner_company,
    license

    from indoss.above_average_projects as aap

    inner join indoss.domains
    on domains.id = aap.domain_id

    left join indoss.distinct_company_domains
    on distinct_company_domains.domain = domains.domain

    inner join projects
    on projects.id = aap.project_id

    inner join users
    on projects.owner_id = users.id

    left join indoss.commit_range
    on commit_range.project_id = aap.project_id

    left join indoss.project_commit_author_domain_count
    on aap.domain_id = project_commit_author_domain_count.domain_id
        and
      aap.project_id = project_commit_author_domain_count.project_id

    left join indoss.project_commit_committer_domain_count
    on aap.domain_id =
        project_commit_committer_domain_count.domain_id and
      aap.project_id =
          project_commit_committer_domain_count.project_id

    left join indoss.project_author_domains
    on aap.domain_id = project_author_domains.domain_id and
      aap.project_id = project_author_domains.project_id

    left join indoss.project_committer_domains
    on aap.domain_id = project_committer_domains.domain_id and
      aap.project_id = project_committer_domains.project_id

    left join indoss.project_committer_count
    on aap.project_id = project_committer_count.id

    left join indoss.project_author_count
    on aap.project_id = project_author_count.id

    left join indoss.pull_request_number
    on aap.project_id = pull_request_number.project_id

    left join indoss.project_size_metrics
    on aap.project_id = project_size_metrics.project_id

    left join indoss.licenses
    on concat('https://github.com/', substr(projects.url, 30)) =
        licenses.url
    where not projects.deleted;

create index on indoss.enterprise_project_details(project_id);
create index on indoss.enterprise_project_details(domain_id);
```

Listing 10: SQL query for deriving the table *fortune global 500*

```
create table indoss.fortune_global_500 (name varchar(255), domain
    varchar(255));

\copy indoss.fortune_global_500 from
    'data/fortune-global-500/data.csv' CSV;

create index on indoss.fortune_global_500 (domain);
```

Listing 11: SQL query for deriving the table *licenses*

```
create table indoss.licenses (url varchar(1023) primary key,
  license varchar(255));

\copy indoss.licenses from 'data/licenses.txt' delimiter ' ' null
    'null';
```

Listing 12: SQL query for deriving the table *merged domain projects*

```
-- Projects with a single common coalesced domain
-- Removing projects with different domains excludes 3874 out of
    3303639
-- projects, which is insignificant.

CREATE TABLE indoss.merged_domain_projects as
  select project_id,
    (sdtc_domain is not null) as sdtc,
    (mcpc_domain is not null) as mcpc,
    (mcve_domain is not null) as mcve,
    coalesce(sdtc_domain, mcve_domain, mcpc_domain) as domain_id
    from indoss.deduplicated_projects
    where
    coalesce(mcve_domain, mcpc_domain, sdtc_domain) =
    coalesce(mcpc_domain, sdtc_domain, mcve_domain)
    and
    coalesce(mcpc_domain, sdtc_domain, mcve_domain) =
    coalesce(sdtc_domain, mcve_domain, mcpc_domain);
```

```
create index on indoss.merged_domain_projects(project_id);
create index on indoss.merged_domain_projects(domain_id);
```

Listing 13: SQL query for deriving the table *merged projects*

```
-- Candidate company projects merged across heuristics

CREATE TABLE indoss.merged_projects as
  select project_id,
    max(sdtc_domain) as sdtc_domain,
    max(mcve_domain) as mcve_domain,
    max(mcpc_domain) as mcpc_domain
    from indoss.candidate_projects
  group by project_id;

create index on indoss.merged_projects(project_id);
```

Listing 14: SQL query for deriving the table *multi committer probable company projects*

```
-- Projects by companies that have multiple committers using
-- the company's domain

CREATE TABLE indoss.multi_committer_probable_company_projects as
  select projects.id as project_id, user_domain.domain_id
  from indoss.probable_company_domain_committers
  inner join projects
  on probable_company_domain_committers.company_id =
      projects.owner_id
  inner join indoss.user_domain
  on projects.owner_id = user_domain.user_id
  -- Avoid individuals using e.g. .com domains; 5 is top decile
  where committer_count >= 5;

create index on
    indoss.multi_committer_probable_company_projects(project_id);
```

Listing 15: SQL query for deriving the table *multi committer valid enterprise projects*

```
-- Projects by companies that have multiple committers using
-- the company's domain

CREATE TABLE indoss.multi_committer_valid_enterprise_projects as
  select projects.id as project_id, user_domain.domain_id
  from indoss.valid_enterprise_domain_committers
  inner join projects
  on valid_enterprise_domain_committers.company_id =
      projects.owner_id
  inner join indoss.user_domain
  on projects.owner_id = user_domain.user_id
  -- Avoid individuals using e.g. .com domains
  where committer_count >= 10;

create index on
    indoss.multi_committer_valid_enterprise_projects(project_id);
```

Listing 16: SQL query for deriving the table *org domains*

```
-- Domain associated with each organization

CREATE TABLE indoss.org_domains as
  select users.id as id,
    substr(email, position('@' in email) + 1) as domain
    from users_private
    inner join users
    on users.login = users_private.login
    where type = 'ORG' and email is not null;

create unique index on indoss.org_domains(id);
create index on indoss.org_domains(domain);
```

Listing 17: SQL query for deriving the table *probable company domain committers*

```
-- Number of organization's committers using its domain

CREATE TABLE indoss.probable_company_domain_committers as
  select company_id, count(*) as committer_count from (
    select distinct company_id, committer_id
    from indoss.probable_company_projects
    inner join indoss.project_commit_committer_domain
    on probable_company_projects.project_id =
        project_commit_committer_domain.project_id
    inner join indoss.user_domain as company_domain
    on company_domain.user_id = probable_company_projects.company_id
    where company_domain.domain_id =
        project_commit_committer_domain.domain_id
  ) as all_company_domain_committers
  group by company_id;

create index on
    indoss.probable_company_domain_committers(company_id);
```

Listing 18: SQL query for deriving the table *probable company domains*

```
-- The valid company domains those that belong to known companies
-- or end in a company suffix top or second-level domain

CREATE TABLE indoss.probable_company_domains AS
  select valid_enterprise_domains.domain
  from indoss.valid_enterprise_domains
  left join indoss.distinct_company_domains
  on distinct_company_domains.domain =
      valid_enterprise_domains.domain
  where
  distinct_company_domains.domain is not null or
  right(valid_enterprise_domains.domain, 4) = '.com' or
  left(right(valid_enterprise_domains.domain, 7), 5) = '.com.' or
      -- e.g. .com.au
  left(right(valid_enterprise_domains.domain, 6), 4) = '.co.'; --
      e.g. .co.uk

create index on indoss.probable_company_domains(domain);
```

Listing 19: SQL query for deriving the table *probable company projects*

```
-- Projects under under probable company users

CREATE TABLE indoss.probable_company_projects as
  select projects.id as project_id,
    probable_company_users.user_id as company_id
  from indoss.probable_company_users inner join projects
  on probable_company_users.user_id = projects.owner_id;

create index on indoss.probable_company_projects(project_id);
create index on indoss.probable_company_projects(company_id);
```

Listing 20: SQL query for deriving the table *probable company users*

```sql
-- Users whose email matches that of a company

CREATE TABLE indoss.probable_company_users as
  select org_domains.id as user_id,
    indoss.probable_company_domains.*
    from indoss.org_domains
    inner join indoss.probable_company_domains
    on probable_company_domains.domain = org_domains.domain;

create index on indoss.probable_company_users(user_id);
```

Listing 21: SQL query for deriving the table *project author count*

```sql
-- Count of different authors per project

CREATE TABLE indoss.project_author_count AS
  select project_id as id, count(*) as author_count
  from (
    select distinct project_id, project_commit_details.author_id
    from indoss.project_commit_details
  ) as ac group by project_id;

create index on indoss.project_author_count(id);
```

Listing 22: SQL query for deriving the table *project author domains*

```sql
-- Count of different authors for each committer domain per project
-- Will be used to obtain the number of authors for the dominant
    domain

CREATE TABLE indoss.project_author_domains AS
  select project_id, domain_id, count(*) as domain_count from (
      select project_id, domain_id, author_id
      from indoss.project_commit_committer_domain
      group by project_id, domain_id, author_id) as a
  group by project_id, domain_id;

create index on indoss.project_author_domains (project_id);
create index on indoss.project_author_domains (domain_id);
create unique index on indoss.project_author_domains (project_id,
    domain_id);
```

Listing 23: SQL query for deriving the table *project commit author domain count*

```sql
-- Count of commits for each committer domain per project

CREATE TABLE indoss.project_commit_author_domain_count AS
  select project_id, domain_id, count(*) as domain_count
  from indoss.project_commit_author_domain
  group by project_id, domain_id;

create index on
    indoss.project_commit_author_domain_count(project_id);
create index on
    indoss.project_commit_author_domain_count(domain_id);
create unique index on
    indoss.project_commit_author_domain_count(project_id,
    domain_id);
```

Listing 24: SQL query for deriving the table *project commit committer domain*

```sql
-- Commits and their committer's domain

CREATE TABLE indoss.project_commit_committer_domain AS
  select project_commit_details.*, domain_id
  from indoss.project_commit_details
  left join indoss.user_domain
  on project_commit_details.committer_id = user_domain.user_id
  where domain_id is not null;

create index on indoss.project_commit_committer_domain(project_id);
create index on indoss.project_commit_committer_domain(domain_id);
```

Listing 25: SQL query for deriving the table *project commit committer domain count*

```sql
-- Count of commits for each committer domain per project

CREATE TABLE indoss.project_commit_committer_domain_count AS
  select project_id, domain_id, count(*) as domain_count
  from indoss.project_commit_committer_domain
  group by project_id, domain_id;

create index on
    indoss.project_commit_committer_domain_count(project_id);
create index on
```

```
    indoss.project_commit_committer_domain_count(domain_id);
create unique index on
    indoss.project_commit_committer_domain_count(project_id,
    domain_id);
```

Listing 26: SQL query for deriving the table *project commit count*

```
-- Number of commits per project

CREATE TABLE indoss.project_commit_count AS
  select project_id as id, count(*) as commit_count
  from project_commits
  left join projects
  on project_commits.project_id = projects.id
  where
    projects.forked_from is null
  group by project_id;

create index on indoss.project_commit_count(id);
```

Listing 27: SQL query for deriving the table *project committer count*

```
-- Count of different committers per project

CREATE TABLE indoss.project_committer_count AS
  select project_id as id, count(*) as committer_count
  from (
    select distinct project_id, project_commit_details.committer_id
    from indoss.project_commit_details
  ) as ac group by project_id;

create index on indoss.project_committer_count(id);
```

Listing 28: SQL query for deriving the table *project committer domain rank*

```
-- Project committers ranked by the number of their commits

CREATE TABLE indoss.project_committer_domain_rank AS
  select project_id, domain_id, committer_id, commit_number,
  rank() over (
    partition by project_id
    order by commit_number desc
  ) as committer_rank
  from (
    select project_id, domain_id, committer_id, count(*) as
        commit_number
    from indoss.project_commit_committer_domain
    group by project_id, domain_id, committer_id
  ) as cn;
```

Listing 29: SQL query for deriving the table *project committer domains*

```
-- Count of different committers for each committer domain per
    project
-- Will be used to obtain the number of committers for the dominant
    domain

CREATE TABLE indoss.project_committer_domains AS
  select project_id, domain_id, count(*) as domain_count from (
      select project_id, domain_id, committer_id
      from indoss.project_commit_committer_domain
      group by project_id, domain_id, committer_id) as a
  group by project_id, domain_id;

create index on indoss.project_committer_domains (project_id);
create index on indoss.project_committer_domains (domain_id);
create unique index on indoss.project_committer_domains
    (project_id, domain_id);
```

Listing 30: SQL query for deriving the table *project size metrics*

```
-- Files and lines per project

CREATE TABLE indoss.project_size_metrics AS
  select projects.id as project_id, files, lines
  from indoss.size_metrics
  inner join users
  on users.login = split_part(size_metrics.name, '/', 1)
  inner join projects
  on projects.name = split_part(size_metrics.name, '/', 2) and
  projects.owner_id = users.id;

create index on indoss.project_size_metrics(project_id);
```

Listing 31: SQL query for deriving the table *project stars*

```
-- Number of stars per project

create table indoss.project_stars AS
  select id, count(watchers.repo_id) as stars
  from watchers
  left join projects
  on watchers.repo_id = projects.id
  group by id;

create index on indoss.project_stars(id);
```

Listing 32: SQL query for deriving the table *same domain top committers*

```
-- Projects whose top N committers come from the same domain

CREATE TABLE indoss.same_domain_top_committers AS
  select distinct project_id, domain_id
  from indoss.project_committer_domain_rank
  where committer_rank <= 3
  group by project_id, domain_id
  having count(*) = 3;

create index on indoss.same_domain_top_committers(project_id);
create index on indoss.same_domain_top_committers(domain_id);
```

Listing 33: SQL query for deriving the table *sec 10-K domains*

```
create table indoss.sec_10_K_domains (cik INT, domain varchar(255),
    name varchar(255));

\copy indoss.sec_10_K_domains from 'data/sec/domains-10-K.txt' CSV;

create unique index on indoss.sec_10_K_domains(cik);
create index on indoss.sec_10_K_domains(domain);
```

Listing 34: SQL query for deriving the table *sec 20-F domains*

```
create table indoss.sec_20_F_domains (cik INT, domain varchar(255),
    name varchar(255));

\copy indoss.sec_20_F_domains from 'data/sec/domains-20-F.txt' CSV;

create index on indoss.sec_20_F_domains(domain);
create unique index on indoss.sec_20_F_domains(cik);
```

Listing 35: SQL query for deriving the table *size metrics*

```
create table indoss.size_metrics (name varchar(1023), files INT,
    lines INT);

\copy indoss.size_metrics from 'data/size_metrics.txt' delimiter '
    ';
```

Listing 36: SQL query for deriving the table *user domain*

```
-- The email domain associated with each user

CREATE TABLE indoss.user_domain AS
  select users.id AS user_id, valid_enterprise_domains.domain_id
  FROM users LEFT JOIN users_private ON
  users.login = users_private.login
  LEFT JOIN indoss.valid_enterprise_domains on
  LOWER(SUBSTR(email, POSITION('@' in email) + 1)) =
      valid_enterprise_domains.domain;

create unique index on indoss.user_domain(user_id);
create index on indoss.user_domain(domain_id);
```

Listing 37: SQL query for deriving the table *valid enterprise domain committers*

```
-- Number of organization's committers using its domain

CREATE TABLE indoss.valid_enterprise_domain_committers as
  select company_id, count(*) as committer_count from (
    select distinct company_id, committer_id
    from indoss.valid_enterprise_projects
    inner join indoss.project_commit_committer_domain
    on valid_enterprise_projects.project_id =
        project_commit_committer_domain.project_id
    inner join indoss.user_domain as company_domain
    on company_domain.user_id = valid_enterprise_projects.company_id
    where company_domain.domain_id =
        project_commit_committer_domain.domain_id
  ) as all_company_domain_committers
  group by company_id;

create index on
    indoss.valid_enterprise_domain_committers(company_id);
```

Listing 38: SQL query for deriving the table *valid enterprise domains*

```
-- Domains that are syntactically valid and unlikely to belong to a
    non-company
CREATE TABLE indoss.valid_enterprise_domains AS
  select domains.id as domain_id,
  domains.domain as domain
  from indoss.domains
  left join indoss.domain_blacklist
  on domain_blacklist.domain = domains.domain
  where
  domain_blacklist.domain is null and
  position(' ' in domains.domain) = 0 and
  position('@' in domains.domain) = 0 and
  position('(' in domains.domain) = 0 and
  position('.' in domains.domain) != 0 and
  right(domains.domain, 4) != '.org' and
  right(domains.domain, 6) != '.local' and
  position(right(domains.domain, 1) in '0123456789') = 0 and --
      e.g. 10.0.0.3
  left(right(domains.domain, 7), 5) != '.org.' and -- e.g. .org.nz
  left(right(domains.domain, 6), 4) != '.or.' and -- e.g. .or.at
  right(domains.domain, 4) != '.edu' and
  left(right(domains.domain, 7), 5) != '.edu.' and -- e.g. .edu.au
  left(right(domains.domain, 6), 4) != '.ac.'; -- e.g. .ac.uk

create index on indoss.valid_enterprise_domains(domain_id);
create index on indoss.valid_enterprise_domains(domain);
```

Listing 39: SQL query for deriving the table *valid enterprise projects*

```sql
-- Projects under under valid company users

CREATE TABLE indoss.valid_enterprise_projects as
  select projects.id as project_id,
    valid_enterprise_users.user_id as company_id
  from indoss.valid_enterprise_users inner join projects
  on valid_enterprise_users.user_id = projects.owner_id;

create index on indoss.valid_enterprise_projects(project_id);
create index on indoss.valid_enterprise_projects(company_id);
```

Listing 40: SQL query for deriving the table *valid enterprise users*

```sql
-- Users whose email matches that of a valid company domain

CREATE TABLE indoss.valid_enterprise_users as
  select org_domains.id as user_id,
    indoss.valid_enterprise_domains.*
    from indoss.org_domains
    inner join indoss.valid_enterprise_domains
    on valid_enterprise_domains.domain = org_domains.domain;

create index on indoss.valid_enterprise_users(user_id);
```