On the analysis of non-coding roles in open source development

An empirical study of NPM package projects

Javier Luis Cánovas Izquierdo[1] (ID) · Jordi Cabot[2]

Abstract

The role of non-coding contributors in Open Source Software (OSS) is poorly understood. Most of current research around OSS development focuses on the coding aspects of the project (e.g., commits, pull requests or code reviews) while ignoring the potential of other types of contributions. Often, due to the assumption that these other contributions are not significant in number and that, in any case, they are handled by the same people that are also part of the "coding team". This paper aims to investigate whether this is actually the case by analyzing the frequency and diversity of non-coding contributions in OSS development. As a sample of projects for our study we have taken the 100 most popular projects in the ecosystem of NPM, a package manager for JavaScript. Our results validate the importance of dedicated non-coding contributors in OSS and the diversity of OSS communities as, typically, a contributor specializes in a specific subset of roles. We foresee that projects adopting explicit policies to attract and onboard them could see a positive impact in their long-term sustainability providing they also put in place the right governance strategies to facilitate the migration and collaboration among the different roles. As part of this work, we also provide a replicability package to facilitate further quantitative role-based analysis by other researchers.

✉ Javier Luis Cánovas Izquierdo
jcanovasi@uoc.edu

Jordi Cabot
jordi.cabot@icrea.cat

[1] UOC – IN3, Barcelona, Spain

[2] ICREA – UOC, Barcelona, Spain

1 Introduction

Open Source Software (OSS) is the infrastructure on which our digital society relies (Eghbal 2016). Nevertheless, many critical Open Source projects suffer from grave sustainability issues[1] as many people use the software but very few contribute to it. Indeed, well-known problems of other types of "public goods" like the tragedy of the commons (Schweik and English 2007) and rich-club behaviour (Gasparini et al. 2020) also impact Open Source.

A large number of research works have studied how to optimize the collaboration of existing contributors and the onboarding process to attract new ones (e.g., Steinmacher et al. 2019; Casalnuovo et al. 2015). However, the vast majority focus on the study of user profiles aimed at contributing code and taking care of other technical tasks (e.g., review or merge code) for the project. Nevertheless, this is only a partial view of what actually should constitute (and make advances) an Open Source project, which generally builds upon a community of users with a rich variety of profiles. Everybody is invited to help even if they cannot write code, thus helping on the sustainability of OSS projects; not only to collaborate on marketing, promotion and design aspects but also to help writing documentation or participate in the discussions about the future evolution of the project (e.g., features to implement next).

While the importance of non-coding contributors[2] is more and more accepted (Rozas et al. 2021; Cheng and Guo 2019; Carillo et al. 2017; Lima et al. 2016; Trinkenreich et al. 2020) there is a still a lack of quantitative analyses focusing on the study of this type of contributors. For instance, is the number of non-coding contributors significant, or OSS projects are still very much "code-driven"? And, what about the contributor diversity?, i.e., are these contributors only collaborating to the project in these non-coding roles, or are they just the same group of core developers taking all types of roles depending on the needs of the project? Or, are these non-coding contributors migrating to coding roles?

In this paper we perform a quantitative analysis of contributions to address the previous questions. More concretely, we explore these issues in the GitHub platform, one the most relevant social coding platforms. In particular, we focused on repositories developing packages for the NPM[3] ecosystem. NPM is the package manager for the Node JavaScript platform[4] which provides a centralized solution to import and manage dependencies in Node projects. By focusing on these repositories, we ensure similar development practices and structure but also high variety on their purpose (there are more than 94K NPM packages). It is also important to note that GitHub is the default option to store and develop NPM packages.

For each project we have classified all project actions and members based on a precise definition of possible contribution roles in the GitHub platform, and compute several metrics related to role compositions, diversity and evolution. Among other results, our analysis reveals a high presence of actions related to non-coding activities (e.g., opening or commenting on issue requests, or reacting to other's contributions). A deeper analysis also shows that these activities are usually performed by people not involved in coding roles, uncovering the presence and importance of dedicated non-coding contributors in OSS. Furthermore, contributors with single non-coding roles prevail, even when migrating from one

---

[1] https://sustainoss.org/

[2] *non-coding* from the point of view of the contributions to a specific project. A person could be a professional developer but act as non-coding contributor in a certain Open Source project while being a core committer in another one.

[3] https://www.npmjs.com

[4] https://nodejs.org

role to the other, thus revealing a high community specialization and the need for projects to put in place proper migration or collaboration paths to ensure the proper communication and interaction among members playing different roles.

Besides the analysis itself, we also provide a replication package aimed to facilitate quantitative role-based analysis of Open Source communities, opening the door to further analysis in this area.

The rest of the paper is structured as follows. Section 2 presents the roles we consider in this study. Section 3 describes the research methodology and Section 4 the results, which are complemented with an additional discussion in Section 5. Section 6 describes the replicability package. Sections 7 and 8 presents the threats to validity and related work, respectively. Section 9 ends the paper and presents the future work.

2 Role Characterization in GitHub

The Open Source principles favor a rich variety of possible ways to contribute to the project development and evolution beyond code contributions. Due to its community-driven approach to development, users of the software can contribute feature requests and bug reports, comment on those made by others or vote to help in the prioritization of the project next steps, among others. Social coding platforms like GitHub provide the infrastructure to facilitate these types of interactions.

Obviously, the same person can play different roles on a project, e.g., she can submit a bug report (with a *user* hat on) while later committing some code that adds a new feature (with a *developer* hat on).

To better understand the different types of contributions and their relative importance to the overall project evolution, we defined a set of contributors' roles targeting the specificities of the project development process enabled by GitHub and other similar social coding platforms (see Section 8 for other previous role classification proposals). GitHub promotes a pull-based development process, where new contributions to the code base are submitted and reviewed via pull requests. This is specially true for external developers and occasional contributors, as members of the project can (and usually do) directly push their code to any branch in the repository. To facilitate the collaborative development of the project, GitHub also offers an issue-tracker, a wiki system and project's activity reports.

Based on this, we have identified six types of contributors' roles in GitHub:

DEVELOPER   The activity of this contributor' role is mainly focused on submitting commits and/or pull requests with code modifications. They may also comment on their pull requests.

REVIEWER   Code contributions can be revised by any GitHub user via pull request reviews. This is the role of the REVIEWER, which is focused on reviewing others' code (and commenting on these reviews).

MERGER   In GitHub, pull requests must be explicitly accepted and merged into the project's codebase to make the contribution effective. This is the role of the MERGER. This typically happens after the REVIEWERs have completed their job and DEVELOPERs have modified and resubmitted their code accordingly.

REPORTER   The activity of this contributor' role is devoted to contribute issues (and comment on such issues) to help raising concerns on the project, give ideas for its future evolution or influence its development.

COMMENTER   GitHub allows users to comment on any aspect of the project, in particular, on issues and/or pull requests. A COMMENTER is a person that enrich the project discussion by commenting on other people's opened issues. Comments on pull requests fall instead into the REVIEWER category above.

REACTOR   As any other social platform, GitHub allows users to react to contributions by others (i.e., issues, pull requests, reviews and comments) via emojis (e.g., thumbs-up, heart, etc.). This kind of reactions serves as a quick acknowledgment on a task (e.g., attaching a thumbs-up to a request in an issue comment) and can be considered a less thoughtful contribution than a comment as they do not enrich the discussion but express support (or disagreement) to a current line of thought. We call REACTORs to anybody that uses this reaction feature.

A more precise description of the set of actions that correspond to each role is described in Section 4.

It is worth noting that these six roles can be classified into two more generic categories: coding and non-coding roles, based on the level of expertise required to be involved in coding activities. Thus, DEVELOPER, REVIEWER and MERGER are considered coding roles as they require some expertise in software development (and generally in the GitHub platform). On the other hand, REPORTER, COMMENTER and REACTOR can be regarded as non-coding roles.

Note that coding and non-coding terms are also employed with a broader perspective in other works (Rozas et al. 2021; Cheng and Guo 2019; Carillo et al. 2017; Lima et al. 2016; Trinkenreich et al. 2020). However, in this work we define with precision such terms for the specific context of GitHub and other social coding platforms with similar features.

3 Research Method

In this section we discuss how our study has been set up. We first present our research questions (Section 3.1), then we report on the dataset construction process and its main descriptive statistics (Sections 3.2 and 3.3), and we end the section describing the statistical method followed (Section 3.4).

3.1 Research Questions

Our objective is to grasp a better understanding of OSS community composition in terms of the prominent types of roles involved in the projects and the number of users playing them. More specifically, we have identified the following research questions:

RQ1   What is the role-based activity distribution in OSS? This research question aims to cluster the actions around an OSS project using the role-based classification described in the previous section. We are interested in identifying the most predominant role/s and which would the average contributor profile be based on them.

RQ2   How specialized is the community around each role? In this research question we are interesting in analyzing and comparing the subcommunities around each role. In particular, we want to investigate whether, typically, contributors play a single role or, on the contrary, play several of them and therefore role members overlap. If the latter, we want to study whether there is a significant amount of project members mixing coding/non-coding roles, and the typical role changes among the lifespan of the project.

To answer each research question we propose to analyze OSS projects following a general-to-specific approach. First, we analyze the full collection of projects in our study, thus providing a general view. Second, we conduct a deeper analysis grouping the projects, thus allowing us to uncover pieces of evidence in projects of specific groups.

Based on our knowledge and participation in different forums related to Open Source, we propose two factors to analyze the projects as groups, namely: project type and community size. On the one hand, we distinguish between two project types: those owned by an individual and those owned by an organization. The former type of project lives in a user's GitHub profile, while the latter is developed within an organization account. Our hypothesis is that there will be differences between individual and organization projects, as the purposes of projects living in each account types are different.

On the other hand, we define three tiers for the project community size. To set the limits of these tiers, we rely on the descriptive statistics of the project' community sizes. Thus the Tier 1 contains those projects with a number of contributors lower than the minimum value of the interquartile, the tier 2 contains those projects with a number of contributors which falls into the interquartile, and the tier 3 contains those projects with a number of contributors higher than the maximum value of the interquartile. In this case, our hypothesis is that projects in Tier 1 will behave different from those in Tiers 2 and 3; as they may require more role overlapping among the contributors (the smaller the community size, the higher the presence of contributors playing more than one role to cover all the required project needs).

3.2 Dataset Construction

We built a dataset composed of the top 100 most starred GitHub repositories developing a NPM module. The Github's *star* mechanism (equivalent for a *like* in other social networks) is a common proxy for the popularity of GitHub projects (Borges et al. 2016), and is often used to create project datasets (e.g., works by Coelho et al. 2018 or Nakamaru et al. 2020) when the analysis to be done does not require selection of projects based on a specific dimension. Throughout this paper, we will use the terms repository and project indistinctly. The construction of the dataset involved three phases: (1) retrieval and cloning, (2) analysis, and (3) graph generation. In the following, we briefly describe how we conducted each phase.

Retrieval and Cloning This phase was in charge of obtaining a list the top most starred repositories in GitHub which are developing a NPM module. The star mechanism is offered by GitHub to allow users to mark their favorite repositories and can serve to measure the popularity of GitHub repositories. To this aim, we relied on the GitHub REST API to list repositories according to a set of constraints. In our case, we configured the API query to order the results according to the number of stars and include *npm-package* as topic of the repository.[5] Once we obtained the list (see full list in Table 5 in Appendix), we cloned the repositories to enable the next phase of the dataset construction process. This step was performed on September, 23rd 2020, and therefore our results evaluate the state of the repositories until this day.

Analysis Cloned repositories were analyzed using SOURCECRED,[6] a measurement tool for collaborative solutions such as GitHub repositories, among others. SOURCECRED is able to

---

[5]https://api.github.com/search/repositories?q=topic:npm-package&sort=stars&order=desc&per_page=100
[6]https://sourcecred.io/

analyze GitHub repositories and build a collaboration graph, where nodes represent assets of the repository (e.g., users, comments, issues, pull requests, etc.) and edges represent relationships among those (e.g., a user authors a commit, a comment belongs to an issue, etc.). An important issue to consider when retrieving information from git repositories is the special support required for user disambiguation (i.e., different commit metadata information can refer to the same committer). As SOURCECRED relies on the GitHub API, which uses unique identifiers for users in commits, it ensures a proper disambiguation. To generate the collaboration graph, SOURCECRED first needs to retrieve the metadata from the GitHub repository. Therefore, we launched the tool for each repository of our dataset.

Graph Generation Collaboration graphs generated by SOURCECRED follow a proprietary format and we needed to convert them into a standard graph representation format to facilitate their analysis in our study. This phase performed transformation processes and calculations to enable the rest of the study and address our research questions.

Our collaboration graphs include several node types apart from a main node representing the repository: (1) users, (2) bots, (3) comments, (4) commits, (5) issues, (6) pull requests and (7) reviews. On the other hand, the edges in our graphs can represent: (1) authoring (e.g., a commit is authored by a user), (2) parenting (e.g., a comment has an issue as a parent), (3) merging (e.g., when a commit is merged in other commit), (4) reactions (e.g., a user reacts with thumbs up to a comment), and (5) references (e.g., an issue references a different issue). Table 1 lists these elements. Figure 1 shows an example of a collaboration graph for a project of our dataset.

3.3 Dataset Descriptive Statistics

At the end of the dataset construction process, our dataset was composed of 100 GitHub repositories for which we obtained the collaboration graphs. We collected a total number of 28,468 users, 38,502 commits, 13,941 issues, 12,312 pull requests, 15,567 pull request reviews and 89,484 comments. Table 2 shows the main descriptive statistics for the dimensions of our dataset. Figure 2 shows the boxplots for these variables. For the sake of

Table 1 Node and edge type in collaboration graphs

Section	Element	Description
Nodes	*User*	A contributor to the GitHub repository
	Bot	A bot contributing to the GitHub repository
	Comment	A comment attached to an issue or pull request
	Commit	A commit on the Git of the repository
	Issue	An issue on the GitHub repository
	Pull	A pull request on the GitHub repository
	Review	A code review attached to a pull request
Edges	*[authors]*	A *User/Bot* authors a contribution (e.g., *Issue* or *Comment*)
	[hasParent]	A *Comment* may have an *Issue* (or *Pull*) as a parent
	[mergedAs]	A *Pull* is merged into the repository (a new *Commit* is created)
	[reacts]	A *User/Bot* reacts to a contribution (e.g., *Issue* or *Comment*)
	[references]	A contribution refers to other (e.g., and *Issue* refers to a *Pull*)
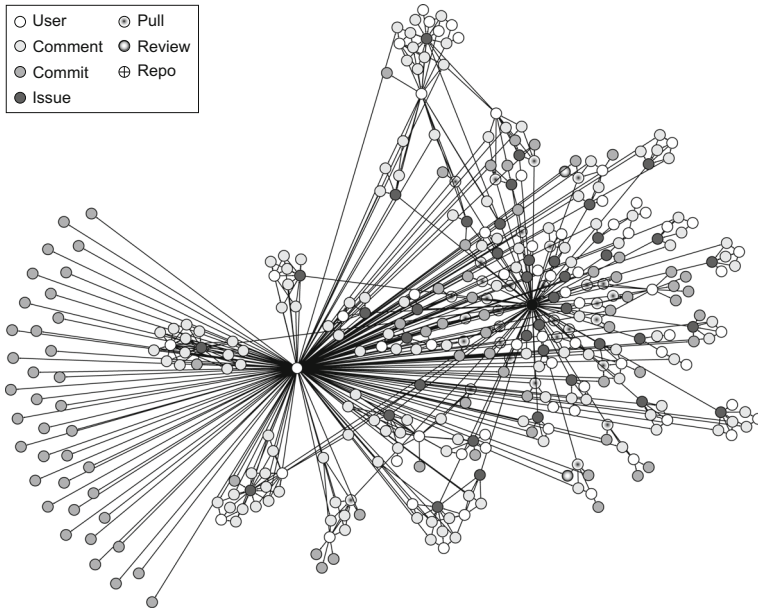
Fig. 1 Collaboration graph for `dalenguyen/firestore-backup-restore` GitHub project

readability, boxplots do not show the outliers for the variables (Fig. 7 in Appendix shows the boxplots including outliers).

Given this descriptive statistics, the tiers of the community size are the following. Tier 1 contains those projects with less than 31.5 contributors, Tier 2 contains those projects with a number of contributors between 31.5 and 196, and the Tier 3 contains those projects with a number of contributors higher than 196.

3.4 Statistical Method

We now describe the statistical method we apply when we study whether the distributions of a variable are different for a given factor.

When we use the project type as factor, which has two levels (i.e., organization and individual), we apply a two-sided Student's t-test, which tests the null hypothesis that there

Table 2 Descriptive statistics of the main dimensions of our dataset

Dimension	Min.	Median	Mean	Std. Dev.	Max.
Users	5	65.00	284.68	798.19	5,433
Commits	8	98.00	385.02	969.88	6,765
Issues	0	31.50	139.41	324.52	1,987
Pulls	0	23.00	123.12	330.37	2,349
Reviews	0	11.50	155.67	464.46	3,043
Comments	3	146.00	894.84	2,319.30	17,217
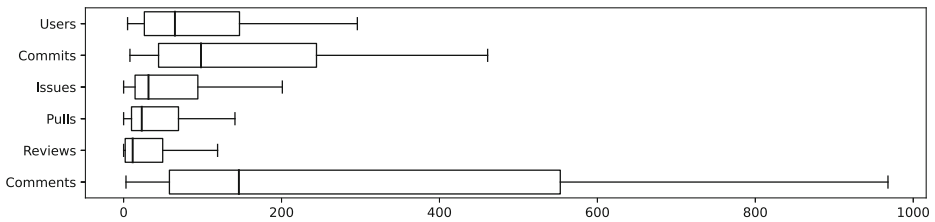
Fig. 2 Distribution of the main dimensions of our dataset (with no outliers)

are not significant differences in the distributions. For instance, for the DEVELOPER role, it would test the null hypothesis that there are not significant differences in the distribution of the number of developer actions with regard to the project type. Student's t-test assumptions require the data to both follow a normal distribution and have homogeneity of variances, which we check via the Shaphiro-Wilk normality test and the Barlett test, respectively. If only the Shaphiro-Wilk normality test is passed, we apply the Behrens-Fisher problem, which tests a global null hypothesis checking the difference between the means of two normally distributed populations when the variances of the two populations are not assumed to be equal. If only the Barlett test is passed, we apply Mann-Whitney-Wilcoxon test, which is a nonparametric test where the null hypothesis checks that the population distributions are identical without assuming them to follow the normal distribution. If no assumption is met, we apply variable transformation (e.g., log or sqtr) and repeat the process.

When we use the community size as factor, which has more than 2 levels, we apply either an ANOVA or Kruskal-Wallis test, depending on the assumptions the data met. The null hypothesis in this case is similar to the previous case, for instance, for the DEVELOPER role, we would test the null hypothesis that there are not significant differences in the distribution of the number of developer actions with regard to the community size. If the variable under study passes the normality and homogeneity of variances tests, we can trust on the ANOVA test to check whether the distributions of the variable are different for the given factor. If the variable only passes the homogeneity of variances tests, we apply the Kruskal-Wallis test. As before, if no assumption is met, we transform the variable and repeat the process.

In both cases, for variables not passing the normality and equality of variances assumptions even after applying variable transformations, we cannot trust on the results of the tests. We follow these procedures to rely first on parametric tests, as they are usually considered to have more statistical power than nonparametric tests and therefore it is more likely to detect a significant effect when one truly exists.

4 Results

4.1 RQ1. Role-based Activity Distribution

In this research question we want to study the activity distribution in OSS projects, grouping the activities according to a fixed set of roles for a better analysis of the main driving forces in OSS. To this aim, we need to first map the project activities appearing in the collaboration graphs (cf. Table 1) to one of the roles we have predefined (cf. Section 2).

Table 3 Role detection in collaboration graphs. Evidences are expressed as *sourceNode − [edge] → targetNode*. Nodes and edges have names following the format *name:type* when required. Refer to Table 1 for node and edge types

Role	Evidences
DEVELOPER	*User − [authors] → Commit*
	User − [authors] → Pull
	u1:User − [authors] → Comment − [hasParent] → Pull ← [authors] − u1:User
REVIEWER	*User − [authors] → Review*
	u1:User − [authors] → Comment − [hasParent] → Review ← [authors] − u1:User
MERGER	*User − [authors] → Commit ← [mergedAs] − Pull*
REPORTER	*User − [authors] → Issue*
	u1:User − [authors] → Comment − [hasParent] → Issue ← [authors] − u1:User
COMMENTER	*u1:User − [authors] → Comment − [hasParent] → Issue ← [authors] − u2:User*
REACTOR	*u1:User − [reacts] → Issue ← [authors] − u2:User*
	u1:User − [reacts] → Pull ← [authors] − u2:User
	u1:User − [reacts] → Review ← [authors] − u2:User
	u1:User − [reacts] → Comment ← [authors] − u2:User

4.1.1 Detecting Roles in Collaboration Graphs

Table 3 lists the proposed roles and the evidences that we use when mapping activities to roles in collaboration graphs. Evidences follow a Cypher[7]-like graph query format. While and reactions that are mapped differently depending on whether they are part of a contribution (e.g., an issue or comment) started by that same user or not. For instance, one only becomes reactor when reacts to contributions made by other users.

As we are focusing on community activities, evidences mainly refer to the *User* node type plus the *[authors]* and *[reacts]* edges of the graph. Note that evidences for a role are mutually exclusive. No activity will be ever assigned to two roles.[8]
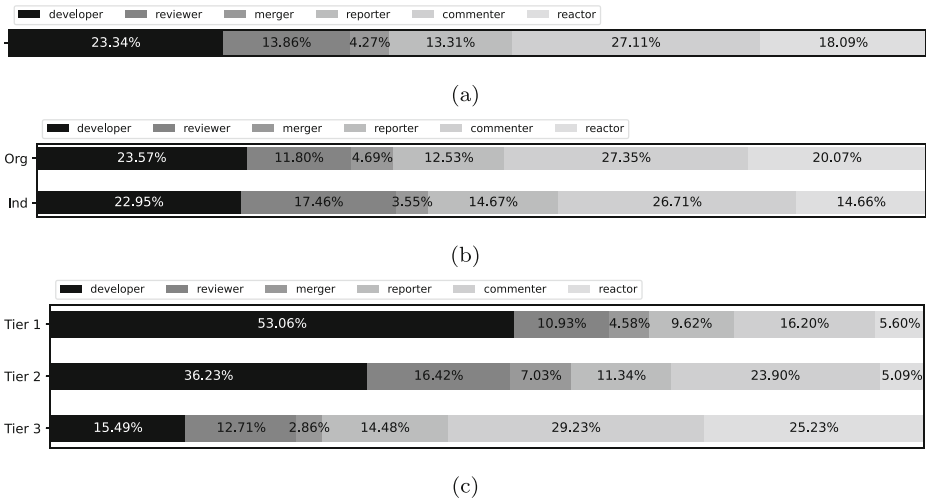
4.1.2 Activity Distribution Analysis

Based on the previous table, we analyze all the projects of the dataset as a whole to obtain the activity distribution of NPM projects.[9] Figure 3a shows the results of the analysis using a stacked bar. As can be seen, more than a half of the actions map to developer and commenter roles, being the merger and reviewer roles the ones with less presence. Nevertheless, we can observe how no single role is dominant and that, therefore, OSS is really a collective effort involving a significant number of all types of actions around the project. Note also that

---

[7]https://neo4j.com/developer/cypher

[8]This is not in contradiction with the possibility of users playing multiple roles. Each user activity will be assigned to a specific role but a user can perform actions that classify in different roles. For instance, a user may daily commit code but also frequently review others' contributions, thus classifying for DEVELOPER and REVIEWER roles.

[9]Figure 8 of the Appendix show the boxplot for each variable.

developer	reviewer	merger	reporter	commenter	reactor
23.34%	13.86%	4.27%	13.31%	27.11%	18.09%

(a)

	developer	reviewer	merger	reporter	commenter	reactor
Org	23.57%	11.80%	4.69%	12.53%	27.35%	20.07%
Ind	22.95%	17.46%	3.55%	14.67%	26.71%	14.66%

(b)

	developer	reviewer	merger	reporter	commenter	reactor
Tier 1	53.06%	10.93%	4.58%	9.62%	16.20%	5.60%
Tier 2	36.23%	16.42%	7.03%	11.34%	23.90%	5.09%
Tier 3	15.49%	12.71%	2.86%	14.48%	29.23%	25.23%

(c)

Fig. 3 Role-based action distribution of the analyzed projects (a) in the dataset, (b) grouped by project type, and (c) grouped by community size

as the commenter role definition specifically refers to those users commenting on others' issues, our results highlight the importance of collaboration among project's members.

If we group the projects in the dataset according to the project type (i.e., organization and individual), we obtain the results shown in Fig. 3b. Only the ratios of from reviewers' and reactors' actions show differences between organizations and users. The ratio of reviewers' actions is higher in individual projects than organization ones, and the contrary happens for the ratio of reactors' actions. Further statistical analysis tested the null hypothesis that there are not significant differences in the distribution of the number actions for each role when grouping by project type. The results rejected the null hypothesis and revealed that there are significant differences between the distributions for the roles of developer, merger, reporter, commenter and reactor actions with regard to the project type (cf. Table 6a in Appendix).

If we take into account the community size of a project to analyze the role-based action distribution we obtain the results shown in Fig. 3c. It is particularly interesting how the ratio of developers' actions decreases as the community size increases and, in compensation, the ratio of commenters' and reactors' actions grows. In particular, the ratio of commenters' and reactors' actions surpasses the 50% ratio in big projects (i.e., Tier 3). We also conducted further statistical analysis to test the null hypothesis that there are not significant differences in the distribution of the number of actions for each role but this time, when grouping by community size. The results revealed that there are significant differences in the distribution of the variables for these roles (cf. Table 6b in Appendix) when considering the community size.

4.1.3 Prototypical Contributor Profile

As an alternative representation of the importance of each role and to better understand the distributions described above, we now characterize the typical profile of an Open Source contributor.

The profile is built by depicting the expected number of actions per role of this prototypical contributor.

To calculate the expected number of actions for a role a_r, we measure the size of the r cluster (i.e., the number of actions classified as belonging to that role) and divide it by the total number of users.

Figure 4a shows the results as a radar plot. These are the overall values, so the radar is based on the actions and users across all projects in the dataset. Obviously, results are aligned with the stacked bar above, with commenters and developers (with approximately 2 expected actions) as main ones. However, this representation emphasizes even more how balanced are all the roles.

Figure 4b and c show the results of the expected number of actions considering the project type, resulting in similar profiles. Figure 4d, e and f show the results considering the community size, which reveal a peak on developer actions for projects in Tier 1 and 2, and a balance for projects in Tier 3 similar to the picture provided earlier in Fig. 4a.

> **Answer to RQ1:** There is a high presence of commenters' actions (higher than developers'), showing a high collaboration rate.
>
> Both reviewers' and reactors' actions seem to grow as the community does, potentially denoting an increasing structure on the development side and a broader participation of non-coding contributors that use reactions to other member actions as quick approach to contribute to the project.
>
> Overall, all roles have their importance highlighting the complexity of OSS development that cannot, by any means, be reduced to code-related, not even purely technically-related, activities. The fact that this situation is already found in GitHub (i.e., a platform more oriented towards supporting code-related activities, which pushes some projects to use other external tools, like forums, to facilitate the discussions) also demonstrates the importance of all roles in OSS.

4.2 RQ2. Role Diversity

This research question studies whether the prototypical contributor profile characterized in the previous section has, in fact, any resemblance to the reality of OSS contributors. More specifically, we want to study whether each role is mostly played by a specialized group of people or, on the contrary, there is a large overlapping between the subcommunities playing each role, especially including both coding and non-coding roles. If the former, projects may consider putting in place specific onboarding strategies and governance policies to target the users of each specific role so that they all feel part of the project.

To this aim, we first calculate the set of roles each user in our project dataset plays. We say a user u plays a role r iff u has performed actions classified as belong to r according to Table 3. As discussed before, u can play more than one role as long as the previous condition holds for several roles.

4.2.1 Role Distribution

Once we know the roles each user plays, we can then analyze the role distribution of each project in the dataset by calculating the ratio of members playing each role.[10] Figure 5a depicts this composition. Each bar states the percentage of members playing that role in

---

[10]Figure 9 of the Appendix show the boxplot for each variable.
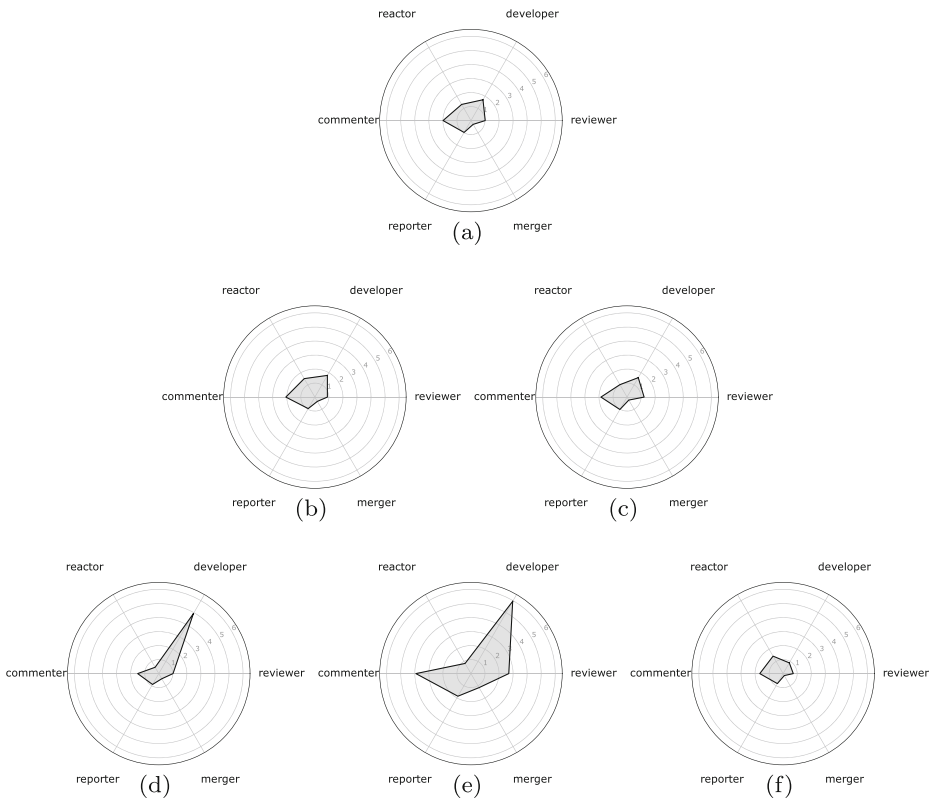
Fig. 4 Expected number of actions per user of the analyzed projects (a) in the dataset, grouped by project type (i.e., (b) Organization and (c) Individual), and grouped by community size (i.e., (d) Tier 1, (e) Tier 2 and (f) Tier 3)

the set of analyzed projects and the average numbers of project members that percentage corresponds to. Figure 5b and c shows this same analysis considering the project type and the community size, respectively.

As can be seen, the most relevant roles are reporters, commenters and reactors. Among these, except for Tiers 1 and 2 projects, the number of reactors is higher (double) than the number of reporters and commenters. This may reveal that, in general, in the development of the analyzed projects it is frequent to find users who socially engage and comment in others' contributions. Note that total percentages add up to more than one hundred per cent in all cases, so we have always a number of project members that play more than one role.

Globally, we can also observe that there is a high presence of reactors. This is somehow surprising as reactor actions were important but not the most dominant ones when we analyzed RQ1 (cf. Fig. 3a). This implies that while a large number of members of a project play the role of a reactor they only play it very occasionally, not amounting for a lot of reactor activity overall. The contrary happens for developers, as results show a relative low presence of developers but they amount to a large number of project actions.

When grouping by project type, we did not observe a different behavior as for all the NPM projects. This is confirmed by the statistical analysis, which did not allow us to reject the null hypothesis, which is that there are not significant differences in the distributions of
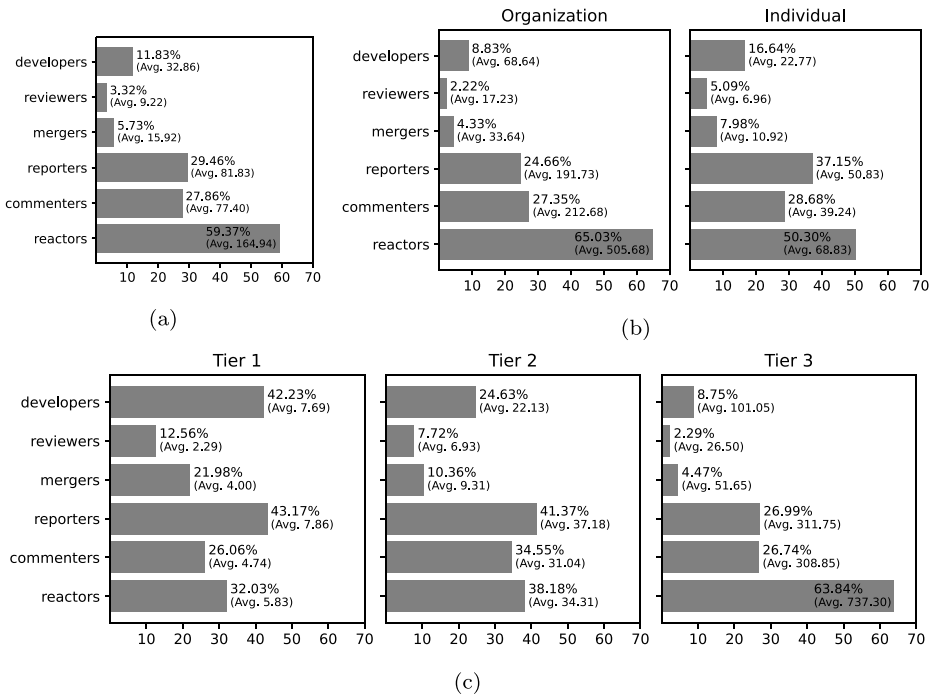
Fig. 5 Role distribution of the analyzed projects (a) in the dataset, (b) grouped by project type, and (c) grouped by community size

the number of contributors per role when grouping by project type. We therefore did not find any significant difference in the distribution of the community size of each role (cf. Table 7a in Appendix). If we group by community size, we observe a higher presence of developers, in particular, in Tiers 1 and 2. Results for Tier 3 seem to return back to the distribution obtained for NPM projects. In this case, we did rejected the null hypothesis when grouping by community sice, thus confirming that there are significant differences in the distribution of the community size for developer, reviewer, merger and commenter roles (cf. Table 7b in Appendix).

4.2.2 Most Common Role Configuration

From the previous analysis, it is clear that some project members play different roles. We now analyze what are the most common role configurations for those "multi-role" users. We believe knowing what roles are typically played together, especially to see if the most common configurations mix coding and non-coding roles, helps to understand the community composition of OSS projects.

Table 4 shows the top 10 most common user role configurations in the analyzed projects. For the sake of space, only the first three letters of the name of the roles are used. In particular, Table 4a, b and c show the results for all the projects in the dataset, projects grouped by project type, and projects grouped by community size, respectively. A role configuration is titled as a hyphen-separated string composed of the name of the roles it refers.

Table 4 Top 10 most common set of roles in (a) all projects, (b) projects according to their type, and (c) projects according to the community size. For the sake of space, only first three letter of role names are shown

(a)

Size	Group
11759	CHE
4988	REP
3565	COM
1900	COM-CHE
1093	REP-CHE
800	DEV
519	REP-COM-CHE
510	REP-COM
464	DEV-MER
184	DEV-REP-MER-REV-COM-CHE

(b)

Organization		Individual	
Size	Group	Size	Group
8408	CHE	3351	CHE
2497	REP	2491	REP
2148	COM	1417	COM
1259	COM-CHE	641	COM-CHE
571	REP-CHE	522	REP-CHE
366	DEV	434	DEV
328	REP-COM-CHE	252	DEV-MER
307	REP-COM	203	REP-COM
212	DEV-MER	191	REP-COM-CHE
99	DEV-REP-MER-REV-COM-CHE	97	DEV-REP

(c)

Tier 1		Tier 2		Tier 3	
Size	Group	Size	Group	Size	Group
163	REP	1049	REP	10914	CHE
73	DEV	783	CHE	3776	REP
67	COM	631	COM	2867	COM
62	CHE	276	DEV	1658	COM-CHE
51	DEV-MER	221	COM-CHE	871	REP-CHE
39	REP-CHE	183	REP-CHE	460	REP-COM-CHE
21	COM-CHE	123	DEV-MER	451	DEV
13	DEV-MER-CHE	81	REP-COM	423	REP-COM
11	DEV-REP	78	DEV-COM	290	DEV-MER
10	DEV-CHE	62	DEV-REP-MER-REV-COM-CHE	119	DEV-REP

The results reveal that one-role configurations CHE, REP and COM are the most common in all projects and groups, except for small projects, where the configuration DEV substitutes to CHE in the top three. This finding states that for many contributors, the first and only

way to contribute to a project is by performing one of these tasks. Note that, especially the reactor one, is also the easiest one (as it is just reacting to somebody else contribution) showing that these roles are a good way to detect new project members that could later (with the right onboarding strategies in place) migrate to more involving roles.

We also find interesting to remark that the developer role appears in sixth position when studying the projects as a whole or according to the project type; only when studying the projects according to the community size we see this role promotes to second position (in small projects) or fourth position (in medium projects). And it does it alone, not in combination with other roles. Configurations of developers with other roles only shows up down the list and, most often, starting with the developer-merger configuration, which is a purely technical one.

4.2.3 Role Migration Paths

As we have seen before, non-coding roles have a high presence in the analyzed projects. We now study the typical role migration paths followed by contributors[11] While there is a high number of one-role configurations (cf. previous section), we are interested in analyzing how roles evolve in multi-role configurations. We believe this information may help us to understand the contributor conversion rate, thus shedding some light on whether non-coding roles may eventually become code-related contributors.

To analyze the role migration paths, we count all the different paths for each project in the dataset. A role migration path is a list $r_1, r_2, ..., r_n$ where r_i is one of the six roles of our study. Each r_i represents one or more actions of such role performed by a contributor of the project, thus a path does not contain equal consecutive roles, i.e., $r_i <> r_{i+1}$. A path such as reactor→reactor→commenter is invalid. However, a path can include non-consecutive repeated roles, i.e., $r_i = r_j$ where $j > i + 1$, for instance, reactor→commenter→reactor.

Figure 6 shows the role migration for all analyzed projects with a Sankey diagram.[12] For the sake of clarity, we analyzed the first three actions of all the calculated paths (shown as stages in the Sankey diagram). As can be seen, the vast majority of contributors just performed one action (see how most flows go from FIRST ACTION stage directly to NO MORE DIFFERENT ACTION) stage, which confirms the results of the previous section. It is also important to note that contributors starting with a non-coding role mainly evolve towards other non-coding roles. Very few moved to a coding role in the second action (i.e., 5.93% from DEVELOPER, MERGER and REVIEWER roles in SECOND ACTION stage) and even fewer in the third action (i.e., 3.76% from DEVELOPER, MERGER and REVIEWER roles in THIRD ACTION stage).

We obtained similar results when grouping the projects according to their type and community size. For the sake of space, we do not show here the Sankey diagrams, but they can be found in the Appendix (cf. Figs. 10 and 11).

---

[11]Note that in every project there is always a large number of occasional or one-time contributors (Lee and Carver 2017; Pinto et al. 2016; Barcomb et al. 2019), which, as the name suggests, are not interested in having a continuous active participation in the project and therefore do not take part in any role migration path.

[12]A Sankey diagram is a directional flow chart in which the width of the streams is proportional to the quantity of flow, and where the flows can be combined, split and traced through a series of events or stages.
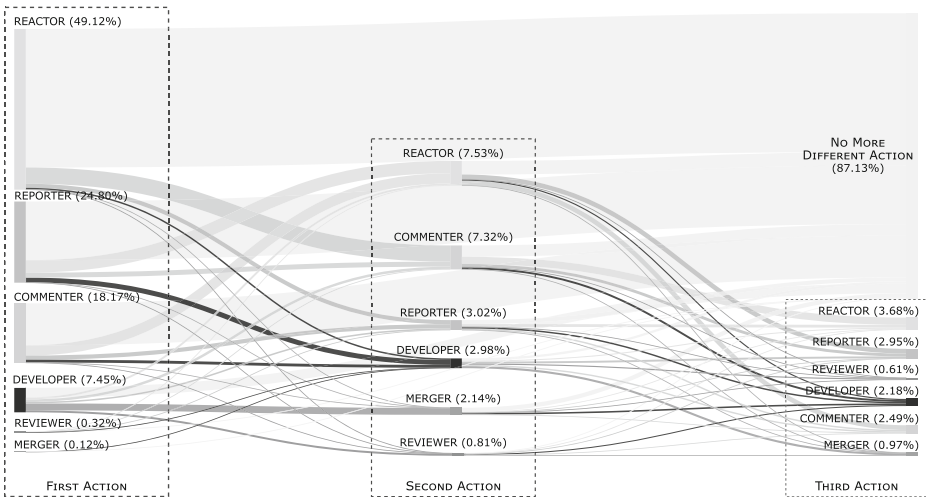
Fig. 6 Role migration paths for all projects (percentages are calculated with regard to the full number of users)

> **Answer to RQ2:** Analyzed projects are diverse, with high presence of reactors, commenters and reporters. This highlights the significant presence of non-coding roles. Moreover, these three roles most often appear in a one-role configuration which may confirm they are a common entry point for new people joining the project. However, a deeper analysis on the role migration paths confirmed that one-role configuration still persists or move to other non-coding roles, which may be denoting a low onboarding rate towards coding roles. We observe a lack of cross-role configurations combining coding and non-coding roles, thus showing the existence of members that specialize in non-coding tasks in the project. Overall, these results suggests that onboarding new contributors in non-coding roles is indeed useful (and easier) for the project as they end up also contributing to other non-coding tasks. Instead, projects interested in attracting more coding contributors should make an extra effort in helping some of these non-coding contributors becoming coding contributors as this is not a natural evolution.

5 Discussion

Beyond the main conclusions reported so far, we would like to highlight some additional insights derived from the results and the feedback we got when sharing these results with a few developers involved in Open Source projects, including the leaders of three of the projects analyzed in this study.

Improve Onboarding The importance of non-coding contributors is not recognized enough in many projects. For instance, in most OSS projects, efforts to attract and onboard new contributors are clearly targeting developers, as it can be easily seen when looking at the `contributing.md` files in the projects' repositories (Elazhary et al. 2019). As such, projects miss out on the opportunity to attract other types of profiles that, as we have just

seen, would indeed help in the advancement and long-term sustainability of the project. Some of these non-coding contributors could even, for instance, be incentivized to participate in coding activities for those projects that also suffer a lack of coding contributors (Hata et al. 2015). Moreover, onboarding mechanisms should address episodic contributors as well, aiming at convincing them to stay in the project (Barcomb et al. 2019).

Governance of Non-coding Contributors Given our results, we advocate for specific onboarding process for non-coding roles that include a clear definition of their tasks but also rights (e.g., participation in the project governance, typically dominated by people in the coding roles). Note that non-coding contributors actions are often less visible in the code hosting platforms (Rozas et al. 2021) and therefore is up to the project managers to make sure they are properly made visible in the community. Otherwise, their importance may be dismissed by coding contributors.[13]

Define and Promote Migration Paths the Project is Interested in When roles are clearly identified in OSS projects, it is easier to define and promote role migration paths. Information on the roles of the project and how each role is welcome (and can evolve) could be a positive signal for potential contributors (Qiu et al. 2019). A typical migration path goes from onboarding as a non-coding role to becoming a developer with permission to merge code. But this is not the only possible path. It is up to the project to identify its needs and then define migration paths that help bringing more people to the roles that fulfill those needs. With well-identified paths, contributors can clearly see and decide how to focus their "career" within the project.

Importance of Member Identification and Contribution Visualization Mechanisms The fact that there may exist overlap between the roles played by a contributor makes more difficult to identify the key leaders in each role and the overall top contributors to the project, which could be useful when putting a representative governance strategy in place. In this paper we have proposed to use radar and bar graphs to visualize the number of role actions and distribution of roles, respectively, but other visualizations could also be applied. Furthermore, we also provide these visualizations for each individual project (see Section 6) with the aim of helping to understand the role contributions. Our conversations with the developers confirmed that these visualizations help to understand their project's community. We hope to see social coding platforms integrating some of these visualizations as a way to help project owners to understand and manage the project community.

Need for Temporal Analysis Most empirical analysis, including our own, focus on a project snapshot. However, many of the community analysis could benefit from a temporal dimension that, as an example, helps to visualize whether new onboarding/governance strategies do have an effect on the role distribution and migration paths. A temporal dimension could also be used to cluster projects depending on their "maturity" to see if we can observe common patterns among them that help to predict and anticipate future challenges. Right now, this type of analysis is lacking also on the social coding platforms that limit themselves to basic activity graphs.

---

[13]This is easy to see when, for instance, suggesting governance models where non-coding roles have a saying in the project evolution causes a strong reaction from core developers that want to keep the decision power in their hands.

6 Replicability Package

To facilitate the replication of our study, we have prepared a GitHub repository[14] for researchers interested in repeating or complementing our evaluations. The repository includes the main elements of our dataset (i.e., graphs in different formats) together with the data used in the study.

7 Threats to Validity

Our work is subjected to a number of threats to validity, namely: (1) internal validity, which is related to the inferences we made; and (2) external validity, which discusses the generalization of our findings.

Regarding the internal validity, the collaboration graph generation process relies on the information provided by the GitHub API, which is queried by SOURCECRED to build the graph. Sometimes the data requested to the API is not available and may cause the generation of dangling edges in the graph (e.g., when the author of a commit does not exist in the platform anymore). Dangling edges are ignored in the study to avoid inconsistent results. The ratio of dangling edges in the collaboration graphs of the dataset is lower than 9%.

The quality of the data is another internal threat. The first concern is regarding the distinction between users and bots. To answer our research questions we aim at GitHub users representing actual developers and not bots. To address this issue, we relied on SOURCE-CRED to distinguish the two. SOURCECRED uses information available from the GitHub API to spot bot user accounts. However, this task is far from trivial (Golzadeh et al. 2020) and sometimes not all the information is available via de the API, thus SOURCECRED may have missed some bots.

Also related to the data quality, our study relies on the information provided by the collaboration graphs which do not include the textual content of the user's contribution (i.e., the text of the issue, the message of the commit or the content of the commit). This limits a more detailed analysis of issues and comments.

Another threat is related to our choice of statistical methods and techniques. To minimize this, we have carefully reported each step of our study and also provided a companion package to promote replicability (cf. Section 6).

As for the external validity, note that our dataset is based on the set of GitHub projects tagged with the *npm-package* label available as of September 23rd 2020, and therefore our results should not be directly generalized to other types of Open Source projects without proper comparison and validation.

8 Related Work

Open Source software development has been studied from a number of different perspectives (Cosentino et al. 2017; Crowston et al. 2012; Kalliamvakou et al. 2016). For instance, characterizing the size of projects and teams, the distribution of issues or the use of labels to annotate them. In this section, we review previous related work on social factors and role

---

[14]http://hdl.handle.net/20.500.12004/1/J/ESEM/2021/001

characterization in Open Source development to better compare our contributions with these previous works.

8.1 Social Factors in Open Source Development

Social factors are recognized as important in many aspects of the development process (Lima et al. 2016; Dias et al. 2021). As an example, they influence the acceptance of pull requests (Tsay et al. 2014a, b; Casalnuovo et al. 2015) depending, among other things, of the connection between the submitter and other core members. Internal community dynamics are also useful, for instance, to determine how committers efforts are distributed over the project files (Palazzi et al. 2019), to help in the discovery of implicit subteams/subsystems in the project (Ashraf et al. 2020), facilitate the onboarding of newcomers (Steinmacher et al. 2019) or its socialization (Carillo et al. 2017). Analysis of internal dynamics can also be used to identify the most active members (Gasparini et al. 2019) and leaders (Li et al. 2012) or as a way to predict the future contributions of project members (Decan et al. 2020). However, these works do not take into account the role dimension in their analysis and use a single global "activity" value as the key metric in the analysis of the community.

8.2 Role Classification in Open Source Development

A few authors have tried to analyze and classify Open Source projects taking a higher-level view of the organization and managerial aspects of the project (Capra and Wasserman 2008; Soto and Ciolkowski 2009; Samoladas et al. 2008; Adewumi et al. 2016; Vasilescu et al. 2015). While some of these works study governance and community aspects, very few classify the roles participating in the project and measure their importance in the project development.

Among them, Yamashita et al. (2015) and Onoue et al. (2016) identify two kinds of users, core and non-core developers (where the former are granted with write permission on the project while the latter are not). A more fine-grained classification of roles in Open Source has been proposed by Crowston and Howison (2006), Sack et al. (2006), Bhattacharya et al. (2014), and Wang and Perry (2015). In all these works, roles are defined mostly from a technical perspective (e.g., passive users, active users, co-developers, core developers and project leaders), based on the technical level of the participants though it is also accepted that there is some mixes and crossovers between the roles. A survey dataset by Robles et al. (2014) performs user profiling including dimensions which go beyond non-technical features, such as personal characteristics, education or level of English. Relationship between roles could either follow a "classical" onion-like structure or adopt a more core-periphery continuum structure (Christian and Vu 2020).

Relevant exceptions on this more technical-oriented classification are the works by Cheng and Guo (2019), which aims to derive roles from the observed activity of contributors, though it does not group roles in coding and non-coding roles; Barcellini et al. (2014), that distinguishes between implementation roles and discussion roles; and the work by Trinkenreich et al. (2020) that, through a series of interviews, validates the importance of these non-coding roles in individual projects. The importance of non-coding contributions has also been studied specifically for the Drupal project by Rozas et al. (2021).

A couple of authors have also covered role migration paths, e.g., the works by Jensen and Scacchi (2007) and Jergensen et al. (2011) validate the existence of role migration in three specific projects and the GNOME ecosystem respectively. Nevertheless, in both cases they

focus on role migration through a classical onion model therefore covering only a subset of the roles we have studied in this work.

8.3 Comparison with Related Work

Our paper also presents a role classification, grouping them into coding and non-coding roles in the context of GitHub, but goes one step further by providing a precise description of all roles (via a fine-grained definition of the tasks comprised in each role, especially in the context of social coding platforms such as GitHub). This enables an automatic computation of several role-related metrics, covering both coding and non-coding roles, something that was not possible with the more general and coarse-grained descriptions from previous works.

Moreover, we perform ourselves a data-based analysis of a dataset of GitHub projects to automatically (1) measure the contribution of each role in any given project, (2) identify and compare the community members occupying such roles, mostly to see if we can observe a diversity in the people taking each role and (3) study the role migration paths from one role to the other.

9 Conclusion and Future Work

This paper has analyzed the different roles participating in Open Source development by providing a precise role definition for a quantitative role-based analysis of Open Source projects. This opens the door to a number of other quantitative analysis of Open Source communities to complement existing qualitative studies. Among these analysis, we have focused on this paper in the study of non-coding roles visible in GitHub. For instance, our results show that non-coding roles (e.g., commenter or reactor) have a high presence in the analyzed projects and that those roles are often taken by people that specialize in contributing to the project only on non-coding activities, complementing the work of coding contributors that, on the contrary, have little involvement in non-coding tasks. This specialization highlights the importance of all types of roles in an OSS project, demystifying the topic of few core coders driving the full spectrum of actions in the project. But at the same time, the limited migration of members from non-coding to coding roles emphasizes the need of better onboarding and governance strategies that facilitate this role evolution or, at the very least, a better collaboration between the different roles.

We believe these results would be even more evident if we had analyzed other sources of project data outside of the GitHub ecosystem (e.g., mailing lists, forums, twitter discussions, etc) where other non-coding roles are also more visible (Trinkenreich et al. 2020). This is part of our future work, together with the replication of the study on other sets of projects. In this sense, we are especially interested in studying how these observations evolve when moving to project ecosystems (Blincoe et al. 2015; Mockus et al. 2020) instead of single projects.

Finally, we would like to enrich the analysis of some of the roles, for instance, studying their typical internal collaboration patterns. We think it would be interesting to analyze the subcommunity of contributors playing a certain role to see how they organize themselves. As an example, we could study the emergent communication and governance patterns (e.g., are some members acting as leaders and controlling the role activities?), and the quality of the discussions taking place among the role members. For the latter, we could cluster the

types of reactions (Sawant et al. 2019; Borges et al. 2019) and apply pretrained language models to detect toxic individuals[15] or developers' burning out (Sarker et al. 2019).

Appendix

Table 5 List of repositories collected in our study ordered by number of stars

| Rank	Stars	Owner	Name	Type
1	12575	netlify	netlify-cms	Organization
2	8302	sindresorhus	got	Individual
3	5819	sindresorhus	ky	Individual
4	5541	sindresorhus	np	Individual
5	5452	filamentgroup	tablesaw	Organization
6	5221	pastelsky	bundlephobia	Individual
7	5012	sindresorhus	query-string	Individual
8	4412	angular-translate	angular-translate	Organization
9	3718	mysticatea	npm-run-all	Individual
10	3671	ngx-translate	core	Organization
11	3422	sindresorhus	speed-test	Individual
12	3193	vuejs	eslint-plugin-vue	Organization
13	2966	niieani	hashids.js	Individual
14	2954	sindresorhus	ow	Individual
15	2843	sindresorhus	modern-normalize	Individual
16	2835	sindresorhus	type-fest	Individual
17	2666	ocombe	ocLazyLoad	Individual
18	2503	sindresorhus	electron-store	Individual
19	2041	sindresorhus	trash	Individual
20	1966	reactopt	reactopt	Organization
21	1926	sindresorhus	slugify	Individual
22	1899	sandoche	Darkmode.js	Individual
23	1880	sindresorhus	fast-cli	Individual
24	1454	yeoman	update-notifier	Organization
25	1445	susam	texme	Individual
26	1433	sindresorhus	on-change	Individual
27	1394	bokub	chalk-animation	Individual
28	1359	airtap	airtap	Organization
29	1335	teenyicons	teenyicons	Organization
30	1329	tulios	kafkajs	Individual
31	1318	sindresorhus	pageres-cli	Individual
32	1220	sindresorhus	p-queue	Individual
33	1096	sindresorhus	emittery	Individual

---

[15]There are a number of available language models that are fine-tuned for this type of classification tasks like sentiment analysis or toxicity detection, see for instance https://github.com/unitaryai/detoxify

Table 5 (continued)

Rank	Stars	Owner	Name	Type
34	1016	styfle	packagephobia	Individual
35	988	ankeetmaini	react-infinite-scroll-component	Individual
36	964	sindresorhus	is	Individual
37	947	sindresorhus	capture-website	Individual
38	903	afc163	fanyi	Individual
39	849	sierra-library	sierra	Organization
40	819	Annihil	github-spray	Individual
41	742	sindresorhus	electron-util	Individual
42	736	sindresorhus	terminal-image	Individual
43	690	mysticatea	eslint-plugin-node	Individual
44	653	sindresorhus	node-module-boilerplate	Individual
45	639	sindresorhus	react-extras	Individual
46	636	waud	waud	Organization
47	623	sindresorhus	conf	Individual
48	598	ziyasal	scientist.js	Individual
49	588	nastyox	Rando.js	Individual
50	580	nikhilk	node-tensorflow	Individual
51	549	sindresorhus	pretty-ms	Individual
52	546	madlabsinc	mevn-cli	Organization
53	509	sindresorhus	capture-website-cli	Individual
54	499	75lb	command-line-args	Individual
55	488	sindresorhus	crypto-hash	Individual
56	471	sindresorhus	strip-json-comments	Individual
58	464	IgniteUI	ignite-ui	Organization
59	460	imsnif	synp	Individual
60	441	ModelDepot	tfjs-yolo-tiny	Organization
61	436	canonical-web-and-design	vanilla-framework	Organization
62	435	sindresorhus	sindresorhus-cli	Individual
63	421	sindresorhus	electron-better-ipc	Individual
65	415	sindresorhus	negative-array	Individual
66	408	sindresorhus	electron-reloader	Individual
67	402	cloudinary	cloudinary_npm	Organization
68	378	Klemen1337	node-thermal-printer	Individual
69	369	mysticatea	cpx	Individual
70	364	sindresorhus	ky-universal	Individual
71	360	flexdinesh	npm-module-boilerplate	Individual
72	358	sindresorhus	eslint-formatter-pretty	Individual
73	357	sindresorhus	conduct	Individual
74	340	jkrup	meteor-now	Individual
75	339	kenshinji	yddict	Individual
76	337	nobrainr	typescript-webpack-starter	Organization
78	330	sindresorhus	terminal-link	Individual

Table 5   (continued)

Rank	Stars	Owner	Name	Type
79	327	lucagrulla ..	node-tail	Individual
81	320	shuiRong	vue-drag-tree	Individual
82	319	sindresorhus	react-router-util	Individual
83	316	sindresorhus	alfred-emoj	Individual
84	310	Rob--	memoryjs	Individual
85	307	squirrellyjs	squirrelly	Organization
86	298	aravindnc	A-to-Z-List-of-Useful-Node.js-Modules	Individual
87	295	sindresorhus	alfred-npms	Individual
88	291	sindresorhus	grunt-php	Individual
89	287	wcoder	highlightjs-line-numbers.js	Individual
90	284	wasmerio	wasmer-js	Organization
91	281	sindresorhus	class-names	Individual
92	279	sindresorhus	bitbar	Individual
93	278	sindresorhus	electron-serve	Individual
94	274	sindresorhus	term-img	Individual
95	272	tinify	tinify-nodejs	Organization
96	268	crewdevio	Trex	Organization
97	261	shelfio	jest-mongodb	Organization
98	252	kabirvirji ..	singlespotify	Individual
99	243	darlanrod	input-range-scss	Individual
100	242	dalenguyen ..	firestore-backup-restore	Individual
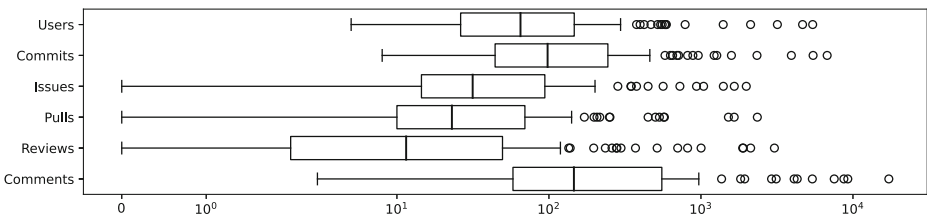
Fig. 7   Distribution of the main dimensions of our dataset (including outliers)
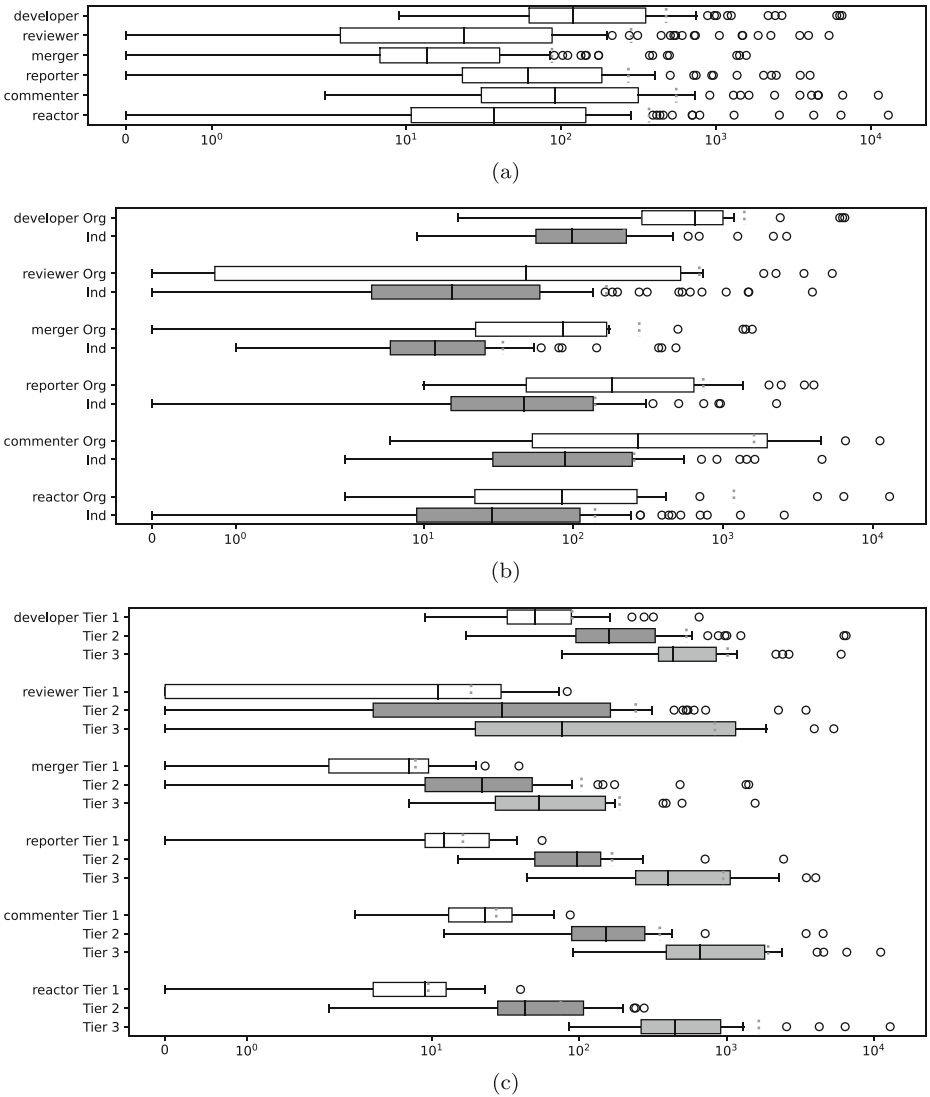
Fig. 8 Role-based action distribution of the analyzed projects (a) in the dataset, (b) grouped by project type, and (c) grouped by community size

Table 6 Activity distribution analysis, grouped by the role they are attributed to, according to (a) project type and (b) community size. TRANSF. indicates the transformation applied to the variable. NORM. TEST and HOMG. TEST report whether the variable passes the Saphiro-Wilk and Barlett and tests, which are the main assumptions of the following tests: T-TEST stands for Student's T test, BF-TEST stands for Behrens-Fisher problem and WX-TEST stands for Mann-Whitney-Wilcoxon test

(a)

VARIABLE	TRANSF.	NORM. TEST	HOMG. TEST	T-TEST	BF-TEST	WX-TEST
Developer	$log(x+1)$	✓	×	N.A.	✓**	N.A.
Reviewer	(not found)	–	–	N.A.	N.A.	N.A.
Merger	$log(x+1)$	×	×	N.A.	N.A.	✓***
Reporter	$log(x+1)$	✓	✓	✓**	N.A.	N.A.
Commenter	$log(x+1)$	✓	×	N.A.	✓*	N.A.
Reactor	$log(x+1)$	✓	✓	✓*	N.A.	N.A.

(b)

| VARIABLE	TRANSF.	NORM. TEST	HOMG. TEST	ANOVA	KW-TEST
Developer	$log(x+1)$	×	✓	N.A.	✓***
Reviewer	(not found)	–	–	–	–
Merger	(not found)	–	–	–	–
Reporter	$log(x+1)$	×	✓	N.A.	✓***
Commenter	$log(x+1)$	×	✓	N.A.	✓***
Reactor	$log(x+1)$	✓	✓	✓***	N.A.

✓ means that the test is passed, × means that the test is not passed

No superscript corresponds to p-value ≥ 0.05, * corresponds to $0.01 \leq$ p-value < 0.05

** corresponds to $0.001 \leq$ p-value < 0.01 and *** corresponds to p-value < 0.001

If T-TEST, BF-TEST or WX-TEST are passed, there are significant differences in the distribution

If ANOVA or KW-TEST are passed, there are significant differences in the distribution (in all combinations)

**Fig. 9** Role distribution of the analyzed projects (a) in the dataset, (b) grouped by project type, and (c) grouped by community size

Table 7 Role distribution analysis according to (a) project type and (b) community size. TRANSF. indicates the transformation applied to the variable. NORM. TEST and HOMG. TEST report whether the variable passes the Saphiro-Wilk and Barlett and tests, which are the main assumptions of the following tests: T-TEST stands for Student's T test, BF-TEST stands for Behrens-Fisher problem and WX-TEST stands for Mann-Whitney-Wilcoxon test

(a)

VARIABLE	TRANSF.	NORM. TEST	HOMG. TEST	T-TEST	BF-TEST	WX-TEST
Developer	$log(x+1)$	✓	×	N.A.	×	N.A.
Reviewer	*(not found)*	–	–	–	–	–
Merger	*(not found)*	–	–	–	–	–
Reporter	$log(x+1)$	×	✓	N.A.	N.A.	×
Commenter	$log(x+1)$	✓	×	N.A.	×	N.A.
Reactor	$log(x+1)$	×	✓	N.A.	N.A.	×

(b)

VARIABLE	TRANSF.	NORM. TEST	HOMG. TEST	ANOVA	KW-TEST
Developer	$sqrt(log(x+1))$	×	✓	N.A.	✓***
Reviewer	$sqrt(log(x+1))$	×	✓	N.A.	✓***
Merger	$sqrt(log(x+1))$	×	✓	N.A.	✓***
Reporter	*(not found)*	–	–	–	–
Commenter	$sqrt(log(x+1))$	×	✓	N.A.	✓***
Reactor	*(not found)*	–	–	–	–

✓ means that the test is passed, × means that the test is not passed

No superscript corresponds to p-value ≥ 0.05, * corresponds to $0.01 \leq$ p-value < 0.05

** corresponds to $0.001 \leq$ p-value < 0.01 and *** corresponds to p-value < 0.001

If T-TEST, BF-TEST or WX-TEST are passed, there are significant differences in the distribution

If ANOVA or KW-TEST are passed, there are significant differences in the distribution (in all combinations)
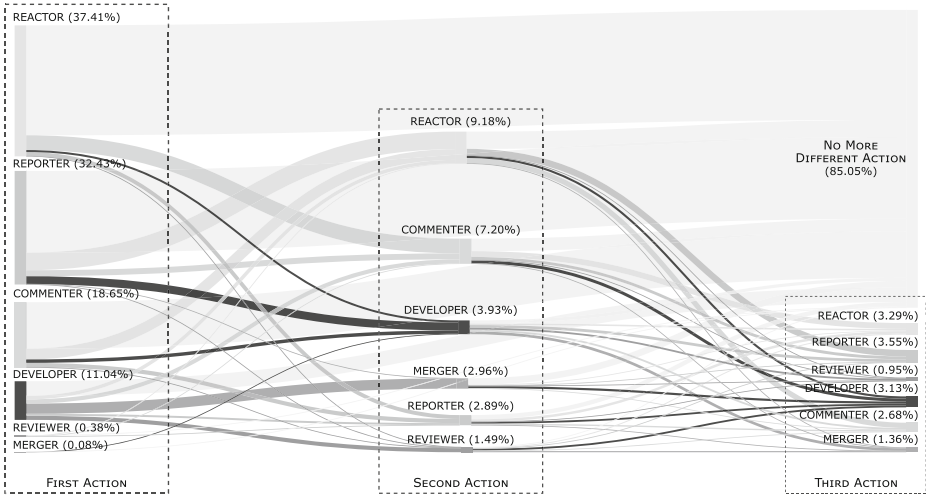
(a)
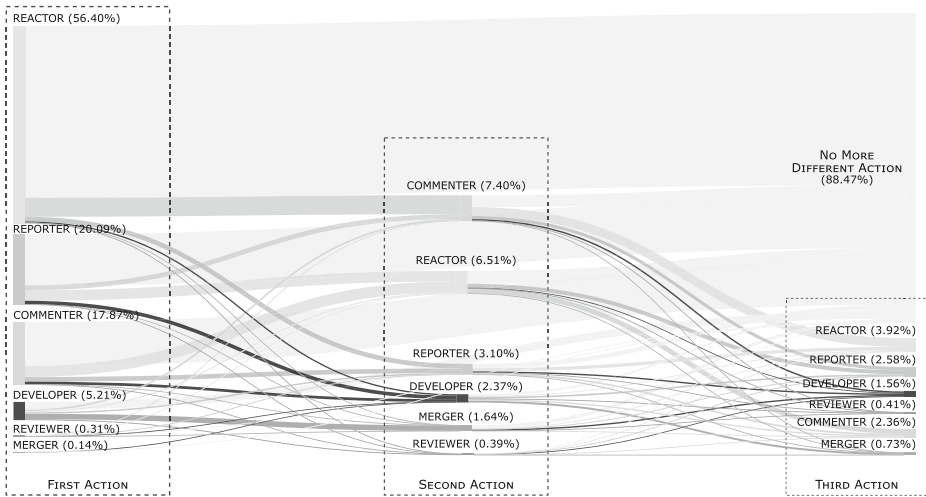
(b)

Fig. 10 Role migration paths for (a) individual projects and (b) organization projects
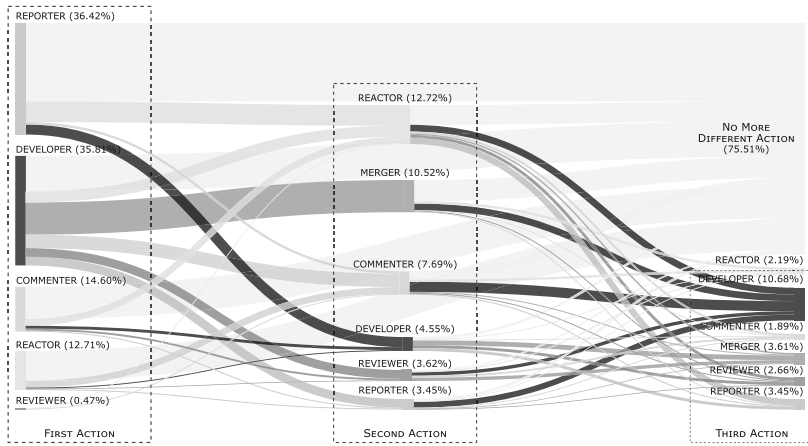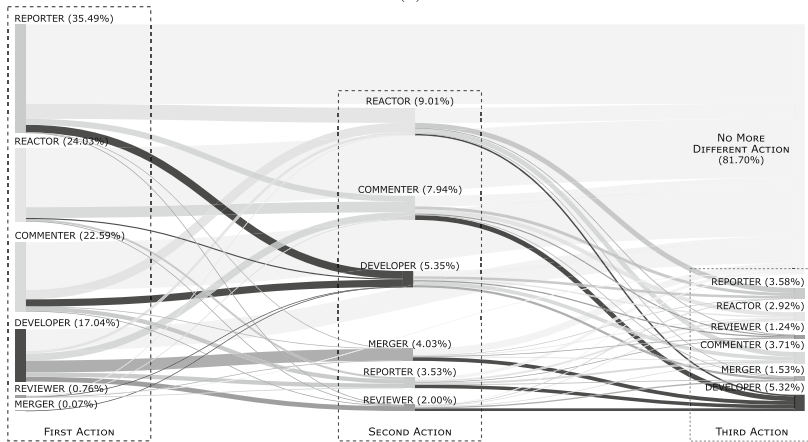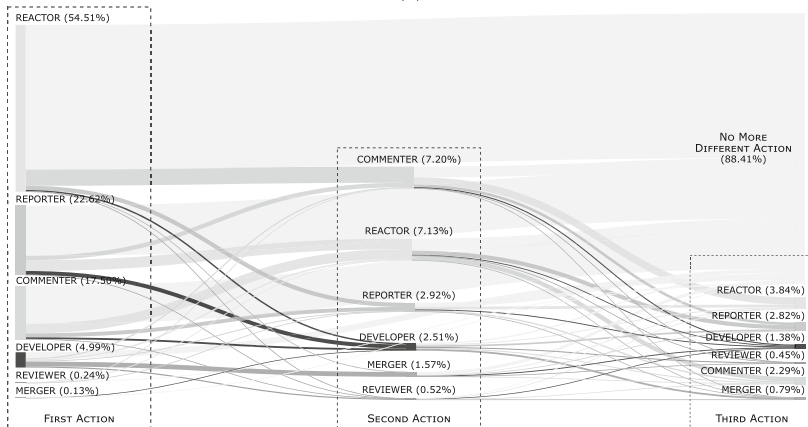
Fig. 11 Role migration paths for (a) tier 1 community size projects, (b) tier 2 community size projects, (c) tier 3 community size projects

References

Adewumi A, Misra S, Omoregbe N, Crawford B, Soto R (2016) A systematic literature review of open source software quality assessment models. SpringerPlus 5(1):1936

Ashraf U, Mayr-dorn C, Egyed A, Panichella S (2020) A mixed graph-relational dataset of socio-technical interactions in open source systems. In: International conference on mining software repositories, pp 538–542

Barcellini F, Détienne F, Burkhardt J (2014) A situated approach of roles and participation in open source software communities. Hum Comput Interact 29(3):205–255

Barcomb A, Stol K, Riehle D, Fitzgerald B (2019) Why do episodic volunteers stay in FLOSS communities? In: International conference on software engineering, pp 948–954

Bhattacharya P, Neamtiu I, Faloutsos M (2014) Determining developers' expertise and role: a graph hierarchy-based approach. In: International conference on software maintenance and evolution, pp 11–20

Blincoe K, Harrison F, Damian DE (2015) Ecosystems in github and a method for ecosystem identification using reference coupling. In: Working conference on mining software repositories, pp 202–211

Borges H, Hora AC, Valente MT (2016) Understanding the factors that impact the popularity of github repositories. In: International conference on software maintenance and evolution, pp 334–344

Borges H, Brito R, Valente MT (2019) Beyond textual issues: understanding the usage and impact of github reactions. In: Brazilian symposium on software engineering, pp 397–406

Capra E, Wasserman AI (2008) A framework for evaluating managerial styles in open source projects. In: Open source development, communities and quality, IFIP 20th world computer congress, working group 2.3 on open source software, IFIP, vol 275, pp 1–14

Carillo K, Huff S, Chawner B (2017) What makes a good contributor? Understanding contributor behavior within large free/open source software projects—a socialization perspective. J Strateg Inf Syst 26(4):322–359

Casalnuovo C, Vasilescu B, Devanbu P, Filkov V (2015) Developer onboarding in github: the role of prior social links and language experience. In: International symposium on foundations of software engineering, pp 817–828

Cheng J, Guo JLC (2019) Activity-based analysis of open source software contributors: roles and dynamics. In: International workshop on cooperative and human aspects of software engineering, pp 11–18

Christian J, Vu AN (2020) Task-based structures in open source software: revisiting the onion model. R&D Management

Coelho J, Valente MT, Silva LL, Shihab E (2018) Identifying unmaintained projects in github. In: International symposium on empirical software engineering and measurement, pp 15:1–15:10

Cosentino V, Cánovas Izquierdo J, Cabot J (2017) A systematic mapping study of software development with github. IEEE Access 5:7173–7192

Crowston K, Howison J (2006) Assessing the health of open source communities. Computer 39(5):89–91

Crowston K, Wei K, Howison J, Wiggins A (2012) Free/libre open-source software development: what we know and what we do not know. ACM Comput Surv 44(2):7:1–7:35

Decan A, Constantinou E, Mens T, Rocha H (2020) Gap: forecasting commit activity in git projects. J Syst Softw 165:110573

Dias E, Meirelles P, Castor F, Steinmacher I, Wiese I (2021) Pinto g what makes a great maintainer of open source projects? In: International conference on software engineering, pp 982–994

Eghbal N (2016) Roads and bridges. The unseen labor behind our digital infrastructure

Elazhary O, Storey MD, Ernst NA, Zaidman A (2019) Do as i do, not as i say: do contribution guidelines match the Github contribution process? In: International conference on software maintenance and evolution, pp 286–290

Gasparini M, Cánovas Izquierdo J, Clarisó R, Brambilla M, Cabot J (2019) Analyzing rich-club behavior in open source projects. In: International symposium on open collaboration, pp 6:1–6:9

Gasparini M, Clarisó R, Brambilla M, Cabot J (2020) Participation inequality and the 90-9-1 principle in open source. In: International symposium on open collaboration, pp 6:1–6:7

Golzadeh M, Legay D, Decan A, Mens T (2020) Bot or not?: detecting bots in Github pull request activity based on comment similarity. In: International conference on software engineering. ACM, pp 31–35

Hata H, Todo T, Onoue S, Matsumoto K (2015) Characteristics of sustainable OSS projects: a theoretical and empirical study. In: International workshop on cooperative and human aspects of software engineering, pp 15–21

Jensen C, Scacchi W (2007) Role migration and advancement processes in ossd projects: a comparative case study. In: International conference on software engineering, pp 364–374

Jergensen C, Sarma A, Wagstrom P (2011) The onion patch: migration in open source ecosystems. In: Symposium on the foundations of software engineering, pp 70–80

Kalliamvakou E, Gousios G, Blincoe K, Singer L, Germán DM, Damian DE (2016) An in-depth study of the promises and perils of mining Github. Empir Softw Eng 21(5):2035–2071

Lee A, Carver JC (2017) Are one-time contributors different? A comparison to core and periphery developers in floss repositories. In: International symposium on empirical software engineering and measurement, pp 1–10

Li Y, Tan CH, Teo HH (2012) Leadership characteristics and developers' motivation in open source software development. Inf Manag 49(5):257–267

Lima T, dos Santos RP, Oliveira J, Werner C (2016) The importance of socio-technical resources for software ecosystems management. J Innov Digit Ecosyst 3(2):98–113

Mockus A, Spinellis D, Kotti Z, Dusing GJ (2020) A complete set of related git repositories identified via community detection approaches based on shared commits. In: International conference on mining software repositories

Nakamaru T, Matsunaga T, Yamazaki T, Akiyama S, Chiba S (2020) An empirical study of method chaining in java. In: International conference on mining software repositories, pp 93–102

Onoue S, Hata H, Monden A, Matsumoto K (2016) Investigating and projecting population structures in open source software projects: a case study of projects in github. IEICE Trans Inf Syst 99-D(5):1304–1315

Palazzi MJ, Cabot J, Cánovas Izquierdo J, Solé-Ribalta A, Borge-Holthoefer J (2019) Online division of labour: emergent structures in open source software. Sci Rep 9(1):1–11

Pinto G, Steinmacher I, Gerosa MA (2016) More common than you think: an in-depth study of casual contributors. In: International conference on software analysis, evolution, and reengineering, pp 112–123

Qiu HS, Li YL, Padala S, Sarma A, Vasilescu B (2019) The signals that potential contributors look for when choosing open-source projects. ACM Hum-Comput Interact 3:1–29

Robles G, Arjona Reina L, Serebrenik A, Vasilescu B, González-Barahona JM (2014) FLOSS 2013: a survey dataset about free software contributors: challenges for curating, sharing, and combining. In: Working conference on mining software repositories, pp 396–399

Rozas D, Gilbert N, Hodkinson P, Hassan S (2021) Talk is silver, code is gold? Beyond traditional notions of contribution in peer production: the case of drupal. Front Hum Dyn 3:12

Sack W, Détienne F, Ducheneaut N, Burkhardt JM, Mahendran D, Barcellini F (2006) A methodological framework for socio-cognitive analyses of collaborative design of open source software. Comp Sup Coop Work 15(2–3):229–250

Samoladas I, Gousios G, Spinellis D, Stamelos I (2008) The Sqo-oss quality model: measurement based open source software evaluation. In: Open source development, communities and quality, IFIP 20th world computer congress, working group 2.3 on open source software, vol 275, pp 237–248

Sarker F, Vasilescu B, Blincoe K, Filkov V (2019) Socio-technical work-rate increase associates with changes in work patterns in online projects. In: International conference on software engineering, pp 936–947

Sawant AA, Robbes R, Bacchelli A (2019) To react, or not to react: patterns of reaction to Api deprecation. Empir Softw Eng 24(6):3824–3870

Schweik CM, English R (2007) Tragedy of the foss commons? Investigating the institutional designs of free/libre and open source software projects. First Monday 12(2)

Soto M, Ciolkowski M (2009) The qualoss open source assessment model measuring the performance of open source communities. In: International symposium on empirical software engineering and measurement, pp 498–501

Steinmacher I, Treude C, Gerosa MA (2019) Let me in: guidelines for the successful onboarding of newcomers to open source projects. IEEE Softw 36(4):41–49

Trinkenreich B, Guizani M, Wiese I, Sarma A, Steinmacher I (2020) Hidden figures: roles and pathways of successful oss contributors. Proc ACM on Hum-Comput Interact 4(CSCW2):1–22

Tsay J, Dabbish L, Herbsleb J (2014a) Influence of social and technical factors for evaluating contribution in github. In: International conference on software engineering, pp 356–366

Tsay J, Dabbish L, Herbsleb J (2014b) Let's talk about it: evaluating contributions through discussion in github. In: International symposium on foundations of software engineering, pp 144–154

Vasilescu B, Filkov V, Serebrenik A (2015) Perceptions of diversity on github: a user survey. In: International workshop on cooperative and human aspects of software engineering, pp 50–56

Wang Z, Perry DE (2015) Role distribution and transformation in open source software project teams. In: Asia-pacific software engineering conference, pp 119–126

Yamashita K, McIntosh S, Kamei Y, Hassan AE, Ubayashi N (2015) Revisiting the applicability of the pareto principle to core development teams in open source software projects. In: International workshop on principles of software evolution. ACM, pp 46–55

Javier Luis Cánovas Izquierdo received the B.Sc. and Ph.D. degrees in computer science from the University of Murcia. He was a Post-Doctoral Fellow in the AtlanMod team, at the Ecole des Mines de Nantes, France. He is currently an associate professor at the IT, Multimedia and Telecommunications Department of Universitat Oberta de Catalunya (UOC). He is also a member of the SOM Research Lab within the Internet Interdisciplinary Institute (IN3-UOC). His research interests fall into the areas of software engineering, web engineering and socio-technical analysis of software systems. You can contact him at jcanovasi@uoc.edu or visit https://jlcanovas.es/.

Jordi Cabot received the B.Sc. and Ph.D. degrees in computer science from the Technical University of Catalonia. He was a Leader of an INRIA and LINA Research Group at Ecole des Mines de Nantes, France, a Post-Doctoral Fellow with the University of Toronto, a Senior Lecturer with the Open University of Catalonia, and a Visiting Scholar with the Politecnico di Milano. He is currently an ICREA Research Professor at Internet Interdisciplinary Institute. His research interests include software and systems modeling, formal verification and the role AI can play in software development (and vice versa). He has published over 200 peer-reviewed conference and journal papers on these topics. Apart from his scientific publications, he writes and blogs about all these topics in several sites like modeling-languages.com and livablesoftware.com. He is also the co-founder and CEO of Xatkit, an open-source chatbot development framework.