Identification of User Activity Types using Issue Tracker Events

Inna Kurnosova[1], Dmitrii Timofeev[2,3][a] and Alexander Samochadin[2,3]
[1]*NeuroTech Lab, Institute of Applied Mathematics and Mechanics, Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia*
[2]*Mobile Device Management Lab, Institute of Computer Science and Technology, Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia*
[3]*Higher School of Software Engineering, Institute of Computer Science and Technology, Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia*

Keywords: Repository Mining, User Activity, User Modeling, Software Engineering.

Abstract: The paper studies the roles users play when contributing to open-source projects using modern code hosting and issue tracking platforms like GITHUB. Role identification has been performed using cluster analysis of the feature vectors generated from the events corresponding to user activity. The method was applied to three open-source projects of different sizes. The roles of maintainers and developers (core team), casual contributors, and watchers were identified, as well as the differences in work organization in these projects.

1 INTRODUCTION

The processes of software development highly vary from team to team and often lack proper documentation. This situation makes it difficult for new members to adapt to the way the team works. It also complicates the estimation of time and cost for customers: they often cannot reliably transfer the previous experience to a new company. Software industry developed a set of practices to fight the problem. In commercial setting, many companies use variations of more-or-less standard agile processes, and developers may successfully apply their soft skills in a new setting. Customers may use the Capability Maturity Model (CMM) framework to estimate risks.

At the same time, many teams don't use certified processes, and their approaches to software development may vary a lot because of geographic distribution of participants, the level of their involvement into the project, or the developers background. This informal approach to the software development process is typical for community-developed open-source projects where participants work in their free time and do not formally report to managers. Knowledge about the properties of the project management in an open-source project (e.g., the estimated bug report reaction time) is beneficial for planning participation or integration of the code into a commercial project.

Even if exact processes the team implements are unknown, it is still possible to make predictions based on the team's public activity. Most developers use some version control system, with Git being the most popular (StackOverflow, 2018). Git hostings like GITHUB, BITBUCKET, and GITLAB, besides the GIT service by itself, provide issue management, bug tracking, time tracking, pull requests, wiki, release management, continuous integration, and similar services. Each service assumes a specific base process (e.g., a set of states that forms the life cycle of an issue). This base process influences the procedures performed by the developers who use the service. Moreover, services record all users' actions and often provide APIs to access them.

We propose to view project management services not only as code repositories but as repositories of process data as well. We can use this data to get insight into the team and practices. In this paper, we utilize this approach to infer the roles of GITHUB users in projects they participate. For that, we have selected three active GITHUB repositories of different sizes and aggregated user activity for one month. Using this data, we try to answer two questions: what roles do users play in the project, and how do they organize the work.

[a] https://orcid.org/0000-0003-2409-8412

Kurnosova, I., Timofeev, D. and Samochadin, A.
Identification of User Activity Types using Issue Tracker Events.
DOI: 10.5220/0008383604050411
In *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019)*, pages 405-411
ISBN: 978-989-758-382-7

2 RELATED WORK

Mining of code repositories is a widely employed research method in the domain of software engineering. The rich project management functionality of modern code hosting platforms allows researchers to study not only the source code changes, but also the more complex aspects, including project life cycle, user attention, and communications between developers. A short review of topics presented in GITHUB mining papers can be found in (Cosentino et al., 2016).

There are three main research directions in the area of repository-based user modeling:

- project ecosystem, demography, and diversity,
- team structure and interaction between users,
- user activity and their impact on the project.

Ecosystem studies deal with the impact of the developers' personalities and team structure on the project development process. Liao et al. (2019) measure qualitative metrics of software projects such as numbers of forks, watchers, issues, pull requests, pull request comments, and core developers. They define the notion of a "healthy ecosystem" and do a quantitative study of several large open-source projects from this point of view. A series of papers by Vasilescu et al. (2015a,b) studies the diversity of open-source software teams, especially from the point of view of gender. Bird and Nagappan (2012) describe the result of developers' geographical locations and affiliations analysis. Robbes and Röthlisberger (2013) studied the impact of the developer expertise on the development time. The expertise measure uses the number of commits and their age as features. Xavier et al. (2014) measure the popularity of GitHub users, which they define as the number of followers. They describe each user as a feature vector using the user role (either reporter or assignee), the time since registration, the number of commits, and the number of collaborating projects as features.

The team demography and the motivation of developers directly influence the team structure and activity. Vasilescu et al. (2015a) describe four kinds of teams:

- fluid teams where participants are volunteers that start and stop contributing at wish;
- teams working on commercial projects using GITHUB infrastructure and following industrial project management practices, including transferring developers between projects;
- academic teams involving students and researchers whose participation is shaped by academic term boundaries, course deadlines, and available funding;
- stable and usually small teams that do not change much with time.

An essential feature of open-source projects is the involvement of external contributors. These contributors do not belong to the core team but participate in the development process in several ways. In particular, they create and discuss issues and contribute patches to the code and documentation. While core developers have direct write access to the repository, external committers depend on the core team members to accept their pull requests into the project. The core team may use different practices when working with outsiders, e.g., by differently processing the issues created by team members and external contributors (Grammel et al., 2010). Some developers never manage to get their pull requests accepted into the project (Padhye et al., 2014), that is probably related to their failure to comply with the project rules or code quality standards.

More centralized work distribution, when most issues are assigned to members of a small core team, increases the issue closure rate while increasing the number of project members decreases the long-term issue closure rate (Jarczyk et al., 2018).

A systematic review of the online community research is presented in (Malinen, 2015).

From the user activity point of view, there are two significant aspects.

- Work area: in what parts of the project the user is interested?
- Work roles: what operations does the user perform?

The task of user interests and work area detection has been actively researched, especially from the side of recommendation algorithms that predict, for example, which issues or pull requests may be relevant to the user (Jiang et al., 2017; Yu et al., 2016; Soares et al., 2018). Another approach the detection of user interests involves the analysis of the social graph corresponding to users and their interactions (Bidoki and Sukthankar, 2019; Buntain and Golbeck, 2014).

The roles that users play in software projects are less researched. Yu and Ramaswamy (2007) proposed to use hierarchical clustering of feature vectors corresponding to users activities. They identified two roles, *core members* and *associate members*, and analyzed the interaction between these groups. Commit history was used in (German, 2006) to determine the roles of specific participants of the PostgreSQL project. Gousios et al. (2008) introduced a metric for measuring developers contributions using detailed action statistics. User roles based on social interaction may be detected using social graph mining techniques (Buntain and Golbeck, 2014).

3 RESEARCH QUESTIONS

As we can see, user activity in software repositories and issue tracking systems, such as GITHUB, is an active research topic, but it is insufficiently developed in the aspect of developer roles. In this paper, we address the following questions.

1. What roles do developers play in the development process?
2. Do developers differ in their working schedule?

We try to answer these questions by analyzing the GITHUB event logs for several open-source software projects, and define the roles as frequency distributions on the set of action types that correspond to the user activities.

4 DATA DESCRIPTION

We studied events which occurred during approximately a month from three GITHUB repositories:

- *acl-anthology*[1] (R1): a research community initiative of maintaining an open publication archive,
- *coc.nvim*[2] (R2): an auto-completion engine for Vim editor, selected as a random active open-source project,
- *react*[3] (R3): a large-scale open-source project driven by a corporate team.

Table 1 shows the statistics of repository activity.

In this study, we try to classify users concerning their activity in the project. We analyze the stream of events users produce by interacting with GITHUB services:

- watching and forking the repository;
- opening, closing, reopening, and commenting issues and pull requests;
- pushing commits to the repository.

All these events have been obtained using GITHUB API. Although users may participate in several projects, we restrict our analysis to studying user activity in a single project, and we study each of the three projects independently.

As we want to identify what role each user plays in the project, we use the project association type as a feature. GITHUB Insights service supports the following association types.

[1]https://github.com/acl-org/acl-anthology
[2]https://github.com/neoclide/coc.nvim
[3]https://github.com/facebook/react

Table 1: Repository activity statistics.

	R1	R2	R3
User count	34	577	2317
Label count	6	8	29
Event count	700	1081	3431
Comments created	384	206	717
Issues			
Opened	37	49	74
Closed	23	48	45
Reopened	0	4	2
Issue lifetime			
Min	68m	167s	99s
Max	191d	33d	528d
Average	34d	3d	23d
Median	5d	8h	11h
Pull requests			
Opened	33	20	79
Closed	32	20	94
Reopened	0	0	0
Pull request lifetime			
Min	83m	22s	65s
Max	22d	5d	537d
Average	2d	11h	48d
Median	107m	29m	14h

- *Collaborator*: the user has been invited to collaborate on the repository.
- *Contributor*: the user has previously committed to the repository.
- *First-time Contributor*: the user has not previously committed to the repository.
- *First-timer*: the user has never committed to GITHUB.
- *Member*: the user is a member of the organization that owns the repository.
- *Owner*: the user owns the repository.
- *None*: the user is not associated with the repository.

The users of the selected projects are collaborator, contributor, member, or have no association with the corresponding project.

We suppose that the people associated with a repository fall into two main categories: project developers and project users. Developers usually write code, fix bugs, and provide user support. Users mostly report bugs, but they may fork repositories and create pull requests as well.

5 USER ACTIVITY ANALYSIS

We identify user activity types using the following method.

1. Convert the sequence of user-related events into a feature vector.
2. Cluster the set of vectors to detect similar users.
3. Analyze the vectors of each cluster to describe the corresponding activity.

The activity of each user can be described in several ways. We chose four feature sets. All these features use the notion of the activity period, i.e., the time span when the user produces events (activity phase), following by the period without events (inactivity phase). In the common commercial development setting, the inactivity phase corresponds to the developer's free time. We can expect that voluntary development efforts would be less regular, as developers often work in their spare time.

1. Operational features: minimum and maximum number of operations for a period of activity, total operations count, operation frequency per period. These features are measured for each operation type.
2. Activity features: the length and number of activity periods and the relative length of activity and inactivity phases.
3. Work focus features: the distribution of events by issue labels, e.g., *bug* or *enhancement*.
4. Role model features:
   - the project association type;
   - the number of created issues, closed issues, own closed issues, reopened issues;
   - the number of comments, comments to own issues;
   - the number of issues assigned,
   - the number of issues where the user has been a reviewer,
   - the number of commits;
   - the number of wiki pages created or updated;
   - the number of release operations;
   - the number of push operations;
   - the number of created or deleted branches and tags.

In order to analyze the data, we run k-means clustering on each of the four sets of vectors. The number of clusters was estimated using the average distance from each point to the centroid of the corresponding cluster. We have chosen to use 4 or 5 clusters depending on the repository and the feature set.

As an example, Figure 1 represents the results of cluster analysis obtained for the R1 repository. In order to make the image, vectors have been translated into two-dimensional space using the PCA transform. We can see that most users belong to a single cluster, while other clusters are small groups of size 1 to 3. This is indeed a common case that holds for all the projects we studied.

For large clusters, the points have been clustered one more time to detect more subtle differences between users. We use cluster centroids as an "activity portrait" of the typical user of each class.

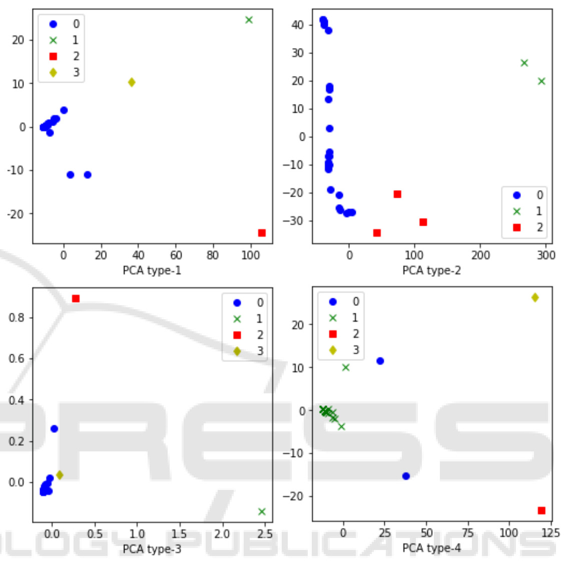

Figure 1: Example of clusters for the R1 repository.

The three projects we have selected may differ not only in size but also in the level of volunteered vs. full-time work. To classify main contributors from this point of view, we computed the time distribution of the commits during a day. The developers are geographically distributed, so we have to know the time zone corresponding to the location of each contributor. GITHUB API only provides UTC timestamps for events, so in this part of the study we restricted ourselves to a subset of all events that can be obtained from commit logs, as Git stores timestamps with time zone information. As a consequence, this part of the study deals with maintainers and active developers who directly push changes to the repository.

6 RESULTS

The R1 repository is a small project. Its domain allows users to produce focused contributions, i.e., to fix a single bibliography record, or implement a script for a specific task.

This observation was confirmed by clusters discovered in operational features. The largest cluster of size 31 corresponds to the majority of contributors, and each of them makes small contributions by commenting issues or, to a lesser extent, by creating pull requests. The three other users form a core team and can be viewed as maintainers. Their activity is different enough to put each vector in its own cluster: one user mostly deals with issues while the other two users are more active in modifying the repository content. These users also differ in the number of operations.

The distinction between core developers and casual contributors is also reflected in activity features. The activity of core developers is more regular and has periodic patterns. The work focus clusters demonstrated that most users in the R1 project do not use labels or assign only *correction* label to their issues, while core developers classify tasks in a much more detailed way. Clustering of role model features showed that some casual contributors perform substantial work, but the tasks are only assigned to the core team members.

As a result, R1 seems to be a project with a homogeneous community and a small core team. External contributions are actively accepted.

R2 is a typical open-source project. Its community is by order of magnitude larger, than R1's one.

Operational features analysis identified two individual users whose activity patterns significantly differs from the rest of the community. One of these users, the repository owner, leads the project. This user closes issues, applies push requests, and makes releases. During the study time span, no other user pushed code and made release except the repository owner. The second special user is the Greenkeeper bot[4] that automates the update of dependencies.

Other users contribute to the project in several ways, that were identified by further clustering:

- mostly create and discuss issues;
- mostly contribute code and create pull requests;
- maintain the wiki.

Although R2 has a much larger community, it is managed in a more centralized way with the lesser delegation of responsibilities to other community members. At the same time, the community actively contributes to the project.

R3 is a large and widely used software project. During this study time span, 2317 users have contributed.

The following are the most interesting clusters.

[4]https://github.com/apps/greenkeeper

- Watchers: users that started watching or forked the repository. Some users in this group have also created or commented issues but did not make any pull requests during the study period.
- Casual contributors: users that participated in issue discussions and created pull requests.
- Active contributors: users whose activity pattern is similar to casual contributors, but who are much more productive (i.e., the pull request frequency in this group is 10x larger).
- Maintainers: users that actively work on the project, manage branches, and apply pull requests contributed by others.

We may suppose that maintainers and active contributors form the small core team (in our study clusters consisted of 4 persons each). About 25% of all users are casual contributors, and the rest do not actively participate in the project.

The study of these three projects of different size and popularity allows making several remarks about the structure of open-source projects.

Users related to the project subdivide into three groups.

- The core team that does most of the work. It is further split into maintainers, who are responsible for the entire project, and developers who solve specific tasks.
- Casual contributors that provide code and feedback.
- Watchers who sometimes participate in discussions but are not otherwise involved. It seems that more well-known projects should have more watchers.

These roles can be identified by analyzing user activity. The feature clustering approach allows to group users that implement similar processes. We can also detect subtler differences by further splitting the clusters. These features, especially operational features, may be used to classify users.

The time distribution of the user activity for each project is presented in Figure 2. The majority of contributions to the R3 project have been done during the regular working hours, that agrees well with the role of Facebook in the development of React. Some maintainers of the R2 projects actively work in the morning and evening, so we may suggest that they contribute to the project during their spare time. The time distribution of commits to the R1 projects is bimodal: a significant number of commits have been produced in the evening. The more detailed analysis of personal contributors shows that at least three R1 contributors use several Git identities for commits, that probably correspond to them making commits using different computers (e.g., office and home), or pushing code using the Git program and GITHUB user interface. This approach to the project maintenance seems to agree with the academic nature of the project itself.

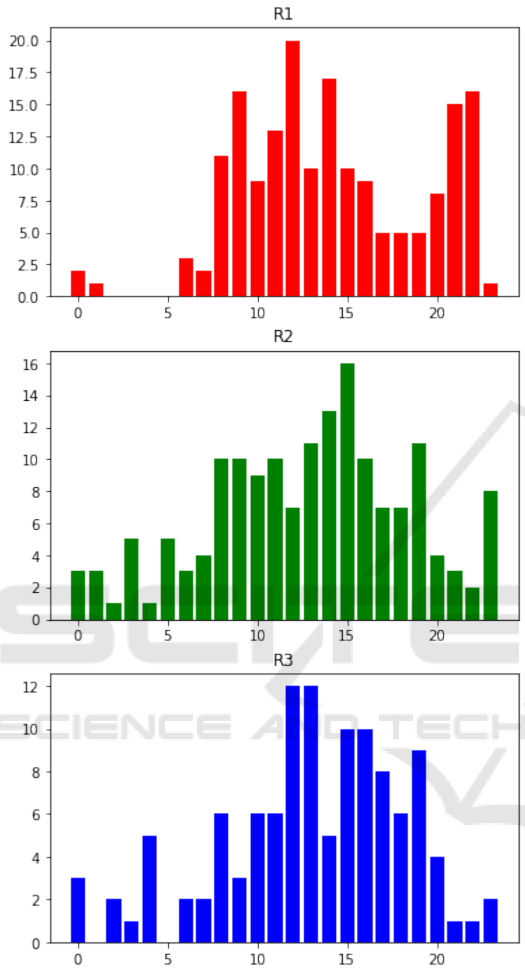

Figure 2: Time distribution of commits.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we addressed the understudied problem of modeling the user roles in the context of modern code hosting and issue tracking platforms like GITHUB.

Using the clusters of feature vectors extracted from user activity log, we identified three primary user roles: core team (that further splits into maintainers and developers), casual (or external) contributors and watchers. Although these classes are rather predictable, their discovery from activity logs shows that GITHUB provides enough event data to distinguish between classes of working processes implemented by users. The time distribution of commits performed by individual project contributors during a day, as well as overall time distribution of commits, is a valuable feature that provides information about the level of involvement and the working schedule of developers.

There are three research directions we are now pursuing based on this study.

What are the processes associated with the user roles? GITHUB implements a typical contribution process: a user wanting to contribute forks the project, makes changes and creates the pull request that may be accepted by the project team. At the same time, many aspects may differ from project to project, i.e., how are pull requests discussed, who decides to accept the pull request, and so on. Other aspects of the project management, like the issue life cycle, are defined by the team itself. The understanding of the processes would lead to better planning and more efficient project management.

How do the user roles evolve? This study and other published research deal with the static view of user roles. Nevertheless, users change their roles at least in two ways. First, an active casual contributor may enter the core team, or a maintainer may retire. A better understanding of factors that enable or disallow the role change may help projects to acquire new active participants. The other interesting aspect is the participation of a user in several projects. Do the user play the same role in all of them, or there are different roles for each project? This research topic is connected to the study of project management processes, as the role change may be influenced by the change of process.

ACKNOWLEDGMENTS

This research is a part of the joint project by Intelin LLC (Moscow, Russia) and Peter the Great St. Petersburg Polytechnic University (St. Petersburg, Russia). This work is financially supported by the Ministry of Education and Science of the Russian Federation (state contract 03.G25.31.0247 from 28.04.2017).

REFERENCES

Bidoki, N. H. and Sukthankar, G. (2019). Network semantic segmentation with application to github. *arXiv preprint arXiv:1902.05220*.

Bird, C. and Nagappan, N. (2012). Who? where? what? examining distributed development in two large open source projects. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, pages 237–246. IEEE.

Buntain, C. and Golbeck, J. (2014). Identifying social roles in reddit using network structure. In *Proceedings of the 23rd international conference on world wide web*, pages 615–620. ACM.

Cosentino, V., Izquierdo, J. L. C., and Cabot, J. (2016). Findings from github: methods, datasets and limitations. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 137–141. IEEE.

German, D. M. (2006). A study of the contributors of postgresql. In *MSR*, volume 6, pages 163–164.

Gousios, G., Kalliamvakou, E., and Spinellis, D. (2008). Measuring developer contribution from software repository data. In *Proceedings of the 2008 international working conference on Mining software repositories*, pages 129–132. ACM.

Grammel, L., Schackmann, H., Schröter, A., Treude, C., and Storey, M.-A. (2010). Attracting the community's many eyes: an exploration of user involvement in issue tracking. In *Human Aspects of Software Engineering*, page 3. ACM.

Jarczyk, O., Jaroszewicz, S., Wierzbicki, A., Pawlak, K., and Jankowski-Lorek, M. (2018). Surgical teams on github: Modeling performance of github project development processes. *Information and Software Technology*, 100:32–46.

Jiang, J., Yang, Y., He, J., Blanc, X., and Zhang, L. (2017). Who should comment on this pull request? analyzing attributes for more accurate commenter recommendation in pull-based development. *Information and Software Technology*, 84:48–62.

Liao, Z., Yi, M., Wang, Y., Liu, S., Liu, H., Zhang, Y., and Zhou, Y. (2019). Healthy or not: A way to predict ecosystem health in github. *Symmetry*, 11(2):144.

Malinen, S. (2015). Understanding user participation in online communities: A systematic literature review of empirical studies. *Computers in human behavior*, 46:228–238.

Padhye, R., Mani, S., and Sinha, V. S. (2014). A study of external community contribution to open-source projects on github. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 332–335. ACM.

Robbes, R. and Röthlisberger, D. (2013). Using developer interaction data to compare expertise metrics. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 297–300. IEEE Press.

Soares, D. M., de Lima Júnior, M. L., Plastino, A., and Murta, L. (2018). What factors influence the reviewer assignment to pull requests? *Information and Software Technology*, 98:32–43.

StackOverflow (2018). Stack Overflow developer survey results 2018. https://insights.stackoverflow.com/survey/2018.

Vasilescu, B., Filkov, V., and Serebrenik, A. (2015a). Perceptions of diversity on github: A user survey. In *Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 50–56. IEEE Press.

Vasilescu, B., Posnett, D., Ray, B., van den Brand, M. G., Serebrenik, A., Devanbu, P., and Filkov, V. (2015b). Gender and tenure diversity in github teams. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pages 3789–3798. ACM.

Xavier, J., Macedo, A., and de Almeida Maia, M. (2014). Understanding the popularity of reporters and assignees in the github. In *SEKE*, pages 484–489.

Yu, L. and Ramaswamy, S. (2007). Mining cvs repositories to understand open-source project developer roles. In *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*, pages 8–8. IEEE.

Yu, Y., Wang, H., Yin, G., and Wang, T. (2016). Reviewer recommendation for pull-requests in github: What can we learn from code review and bug assignment? *Information and Software Technology*, 74:204–218.