A Study of the Characteristics of Developers' Activities in GitHub

Saya Onoue, Hideaki Hata, and Ken-ichi Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology, Nara, Japan
{onoue.saya.og0, hata, matumoto}@is.naist.jp

***Abstract*—What types of developers do active software projects have? This paper presents a study of the characteristics of developers' activities in open source software development. GitHub, a widely-used hosting service for software development projects, provides APIs for collecting various kinds of GitHub data. To clarify the characteristics of developers' activities, we used these APIs to investigate GitHub events generated by each developer. Using this information, we categorized developers based on measures such as whether they prefer communication by coding or comments, or whether they are specialists or generalists. Our study indicates that active software projects have various kinds of developers characterized by different types of development activities.**

I. Introduction

When trying to identify good developers and understand how developers set goals in life, the different types of developers are an interesting topic for investigation. For example, the blog article "Are You a Good Programmer?[1]," 1 describes four types of good programmers.

This article suggests that the four types of programmers approach code in different ways based on their motivation. **The Philosopher**, driven by a need for safety and security, writes tightly controlled code. **The Inventor**, driven to explore, creates quirky and unique code. **The Conqueror**, driven to compete, looks for harder challenges. Finally, **The Problem Solver**, motivated to create value, tries to deliver the desired outcome.

Similarly, the article "The 6 Types of Software Engineers: Identification, Care and Feeding[2]," introduces the following *six* types of software engineers: **The Veteran**, **The Hotshot** (smart and young engineer), **The Great One** (always delivers on schedule, writes solid code, and so on), **The Teflon-gineer** (will do anything to reduce his work), **Offshore**, **The Maverick** (smart, creative, dependable, does not want to build on or maintain the existing codebase). Then the article "10 Types of Programmers You'll Encounter in the Field[3]:" presents 10 types: **Gandalf** (as adept at working magic as Gandalf), **The Martyr** (a workaholic), **Fanboy**, **Vince Neil**, **The Ninja** (the team ' s MVP, and no one knows it), **The Theoretician**, **The Code Cowboy**, **The Paratrooper**, **Mediocre Man**, and **The Evangelist**.

These lists of developer types are neither complete nor comprehensive, nor do developers have to fit these categories. The point is that there are many different types of developers, with many different ways of making valuable contributions. This research studied types of developers based on actual developers' activities. We did not investigate the social structure in software projects, nor did we analyze the roles of developers in such projects. Instead, we tried to characterize developers based on their observed activities.

As case studies for this research, we chose two active software projects, node and jQuery. Both of these use GitHub[4], which is a widely-used hosting service for software development projects that used the Git revision control system. GitHub provides "social coding" services for developers to collaborate with each other. GitHub provides APIs (GitHub API v3[5]) which allowed us to collect developers' activity data.

Our study collected data from the two projects, which we then analyzed to investigate the characteristics of developers' activities. We analyzed the frequencies of types of activities, such as code-related, comments-related, and issue handling. We also investigated the specialization of the projects and the rates of the activities. Our study showed that the top contributors in these projects include different kinds of developers with different characteristics of development activities.

II. GitHub Activities

Figure 1 provides an overview of a developer's activities in GitHub. A project is identified by the owner and the name of the project (`owner/project_name`). In GitHub, a developer can work on various projects. Also, in our study, we consider (a) and (b) projects as target projects and (c) and (d) as other projects. Therefore, we classify developer's activities in terms of target projects and other projects. This means that for a developer `User` who is a contributor to one specific project `Project` , there are four categories of working projects:

(a) Target project `TargetProject` belonging to the developer `User`. This code repository may have been forked from another user's repository.
(b) Target project `TargetProject` belonging to another developer `OtherUser`, who may be the owner of the project.
(c) Non-target projects `OtherProject` belonging to the developer `User`. These code repositories have been forked from the original repositories.

[1] http://techiferous.com/2011/08/are-you-a-good-programmer/
[2] http://crankypm.com/2008/08/the-6-types-of-software-engineers-identification-care-and-feeding/
[3] http://www.techrepublic.com/blog/10-things/10-types-of-programmers-youll-encounter-in-the-field/
[4] https://github.com/
[5] http://developer.github.com/

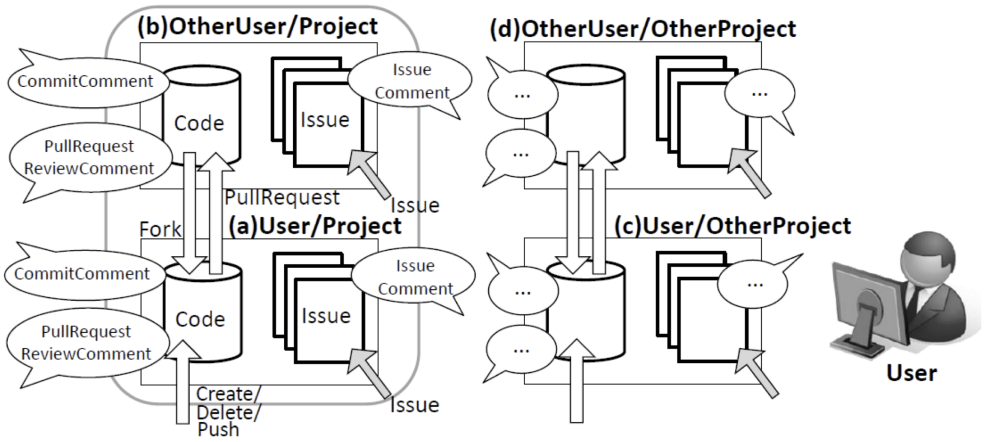


Fig. 1. Developer's Activities in GitHub.

TABLE I. STATISTICS OF TARGET PROJECTS (AUG. 15, 2013)

Project	Language	Commits	Stars	Forks	Contributors
node	JavaScript	8,974	23,984	4,572	447
jQuery	JavaScript	5,270	22,305	4,587	168

(d) Non-target projects `OtherProject` belonging to another developer `OtherUser`.

Each project has a code repository and an issue repository. Although a developer can directly access their own code repositories, they cannot directly push their changes to code repositories belonging to another developers' account. To work with code from another developer's code repositories, a developer can make cloned repositories in their own account. This operation is called `fork`ing. Similarly, a developer can `create` or `delete` branches and tags in their own repositories. A developer can also `push` changes from their local code repositories to the GitHub code repositories. As shown in Figure 1, when a developer wants to apply changes from their own code repositories ((a) or (c)) to another developer's code repositories ((b) or (d)), a `PullRequest` is sent to the other developer's projects.

Issue repositories contain reports of issues including bugs and feature requests. As shown in Figure 1, a developer can `open`, `close`, or `reopen` any issues in her issue repositories for (a), (b), (c) or (d). Developers can also make comments on commits (`CommitComment`), pull requests (`PullRequestReviewComment`) and issues (`IssueComment`).

In our study, we consider (a) and (b) projects as **target projects** and (c) and (d) as **other projects**. Therefore, we classify developer's activities in terms of target projects and other projects.

III. EMPIRICAL STUDY

A. Target Projects

To investigate various developers' activities, in this study we selected two active projects, `node`[6] (joyent/node. Description: evented I/O for v8 javascript http://nodejs.org/) and `jQuery`[7] (jquery/jquery. Description: jQuery JavaScript Library http://jquery.com/). We chose these projects because they have many stars and forks, and they appear in the trending repositories. Table I presents statistics for these two projects. These projects also have many contributors who have participated in many GitHub activities, as explained in Section II.

TABLE II. CHARACTERISTICS OF DEVELOPERS' ACTIVITIES. PARTIAL NAMES ARE USED, BECAUSE OF PRIVACY CONCERNS.

	node		jQuery	
1	*ry*	2,941	*er*	1,591
2	*sa*	1,413	*me*	436
3	*no*	1,208	*za*	318
4	*is*	502	*wl*	297
5	*nd*	288	*im*	267
6	*oo*	177	*au*	266
7	*oi*	157	*ra*	248
8	*el*	119	*le*	200
9	*re*	114	*ib*	145
10	—		*rk*	117

Table II shows the top contributors with more than 100 commits in these two projects, and how many commits each has made. Each project webpage shows such information. We only show partial names of developers because of privacy concerns. In this study, we selected data about such top contributors from the two target projects for our analysis of developers' activities. While GitHub identifies the contributors to projects based on the cumulative number of commits, we intended our study to clarify the differences in contributions between developers.

B. Data Collection

Using the GitHub APIs, we collected data about the developers' activities. There are various APIs for different kinds of GitHub data, such as Git revision control data, issues, repositories, and users. All API access is over HTTPS, and all data is received as JSON data structures. We collected data on Aug. 14, 2013 in two phases, first identifying contributors and second extracting activities.

[6]https://github.com/joyent/node

[7]https://github.com/jquery/jquery

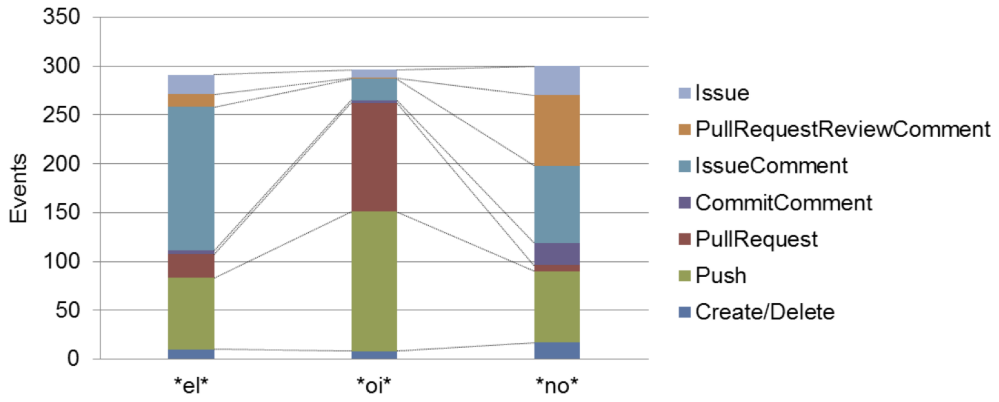


Fig. 2. Distributions of developer's activities for three typical developers in the node project.

Phase 1: Identifying Contributors. When we identified a repository, such as joyent/node or jquery/jquery, the repository-related API provided a list of contributors[8] to that repository. This list included the developers' names and number of contributions, which is the number of activities related to the project. For our study, we limited the contributors to developers who have made more than 100 contributions because we wanted to investigate active developers. In this phase, we identified 9 contributors for the node project and 10 contributors for the jQuery project. Table II shows all the contributors for both projects.

Phase 2: Extracting Activities. Using the names of the developers, the activity-related API provided a list of each developers' activity events[9]. GitHub has 18 different types of activity events. However, because some types of events seldom occurred in our data collection, we ignored them. In our study, we investigated these eight activity events: `CreateEvent`, `DeleteEvent`, `PushEvent`, `PullRequestEvent`, `CommitCommentEvent`, `IssueCommentEvent`, `PullRequestReviewComment`, and `IssueEvent`. Section II discussed the relationships between the events and the projects. For each event, the event lists include the date and touched repositories. The activity-related API limits the number of events to the most recent 300.

C. Threats to Validity

Construct Validity. The major threat to the construct validity is the limitation of available event data to the most recent 300 events in the activity-related GitHub API. In our study, we found that 300 events was not large enough for some developers because these developers produce 300 events or more within a few weeks, so the most recent 300 may not be representative activities for these developers. As a result, our study may have only clarified the characteristics of activities in a short-term window. For more representative analysis, data obtained over a longer term is desirable.

External Validity. Our study is limited to two projects written in JavaScript using the GitHub environment, so our results cannot be generalized to other projects and development languages. In addition, these were open source software projects, and developers' activities in industry software development may be different. Clarifying the characteristics of developers and their activities in various different projects and environments will be future work.

[8]http://developer.github.com/v3/repos/#list-contributors
[9]http://developer.github.com/v3/activity/events/

IV. Results

We investigated developers' activities in terms of four areas, the type of activity, the specialization of contributions, contributions relative to the day of the week, and the frequency of activity.

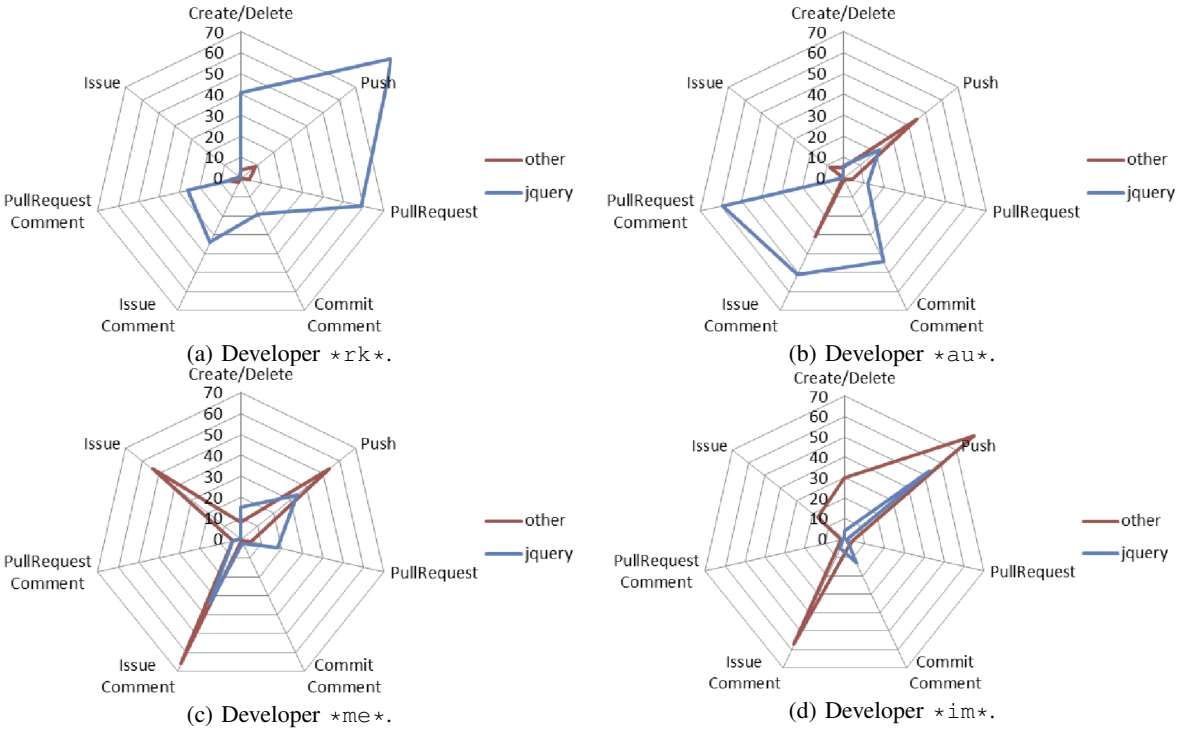
A. Coding, Commenting, and Issue Handling

First, we investigated the frequencies of the extracted eight events in the 300 events for each developer to identify their major focus of activities. For convenience, the `CreateEvent` and `DeleteEvent` counts were combined. Figure 2 presents a bar chart of the characteristic activities of three typical developers from the node project. These three developers were chosen because they represent a variety of approaches. In the bar chart, large areas of each bar represent high numbers of those types of activity events.

The events `Create`, `Delete`, `Push`, and `PullRequest` are related to coding, while the events `CommitComment`, `IssueComment`, and `PullRequestReviewComment` are related to commenting. As can be seen in Figure 2, the first developer, `*el*` has many `IssueComment` and `Push` events. Similarly, the second developer, `*oi*`, has many `Push` and `PullRequest` events. The third developer, `*no*`, has a relatively balanced mix of activities. Based on this categorization of events, we can identify the majority of the first developer's activities as related to commenting, the second developer's activities as related to coding, while the third developer has a balance of activities including both coding and commenting activities. Similarly, for every developer on both projects, we identified the majority focus of their activities. Table III summarizes this and other results.

B. Specialty of Contribution

Second, we investigated how much developers contribute to their own target projects and other projects. As explained in Section II, we classified projects into target projects and other projects. Figure 3 shows the radar charts of the activity events



(a) Developer `*rk*`. (b) Developer `*au*`.

(c) Developer `*me*`. (d) Developer `*im*`.

Fig. 3. Radar Charts of Developer's Activities on the jQuery Project

for four typical developers on the jQuery project. In these radar charts, we separately plotted the developers' activities on their target projects and other projects. The blue lines represent the activities for the target project, the jQuery project, while the red lines represent activities for other projects. Each axis shows the number of events for a specific activity. These charts allow us to easily see the differences in characteristics of developers' activities between their target project and other projects.

For example, in Figure 3 (a), the developer contributes mostly to their target project, particularly with code-related activities such as creates, deletes, pushes, and pull requests. In Figure 3 (b), the developer also makes most contributions to their target project, but with comment-related activities such as commit comments, issue comments, and pull request comments. This developer does not make so many code-related activities such as create, delete, or push. For other projects, this developer also has some push and issue comment activities.

In Figure 3 (c), while the developer makes code-related and issue comment contributions to their target project, most of their activities are on other projects with pushes, issues, and issue comments. In Figure 3 (d), the developer shows similar characteristics, but contributions to their target project are less than other projects, and most of the activities are either pushes or issue comments.

This analysis reveals that top contributors contribute differently to target and other projects. Some developers contribute most of their development activities to their target projects, while others contribute most of their activities to other projects. In our analysis, the rest of the developers showed similar patterns of activities.

C. Workdays

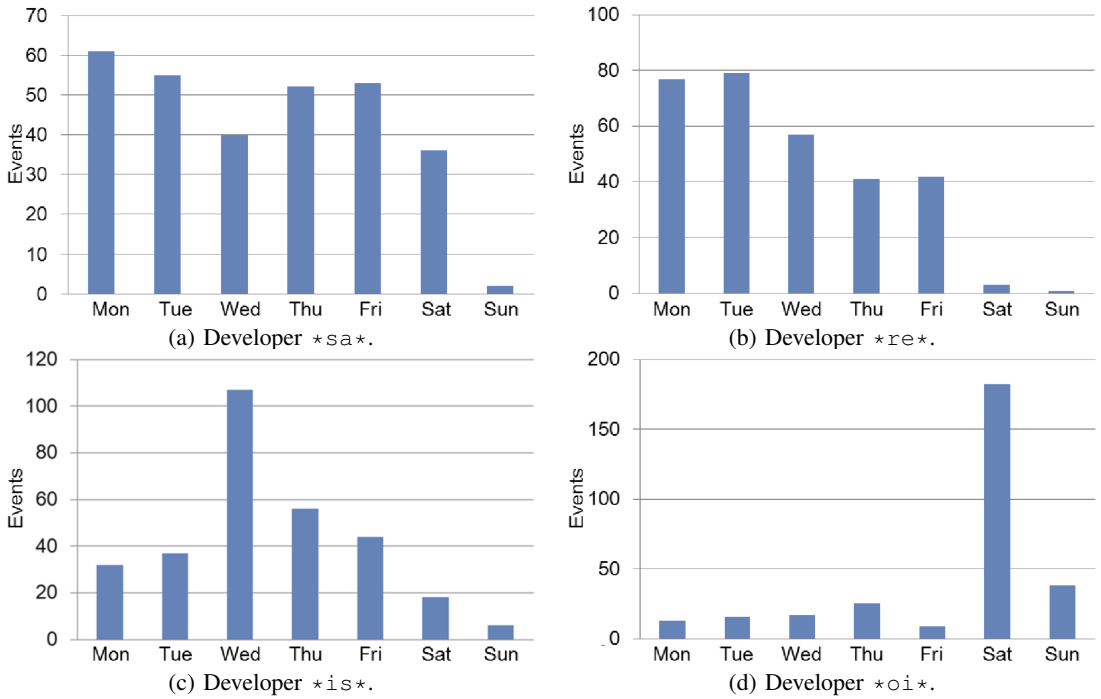
Figure 4 shows the number of developers' activities divided across the days of the week. Charts of four typical developers in the node project are shown. In Figure 4 (a), the developer works Monday through Saturday. In Figure 4 (b), the developer works Monday through Friday and does not work on weekends (Saturday and Sunday). In Figure 4 (c), the developer mainly works on Wednesday, while in Figure 4 (d), the developer works mostly on Saturday. The rest of the developers have similar distributions. We found that some developers work mostly on weekdays, while others work mainly on weekends. This analysis may allow us to distinguish professional developers from volunteer developers.

D. Frequencies of Activities

Figure 5 shows the frequencies of activity events over a range of months for four typical developers. Because the activity-related GitHub API limits the number of events per developer to 300, the date of the first recorded event for each developer differed. For example, in this figure, the developer `*no*` worked frequently in GitHub, with over 200 activities in a week. For comparison, the developer `*oi*` worked only a little bit in GitHub each week, so that their 300 activities covers over a year. We found several patterns of work among the developers, with some developers producing 300 events in a few weeks, while others took months for 300 events. Table III summarizes these results in the Activities column.

E. Summary

Table III summarizes the analysis results in three key areas, type of activities, specialization, and frequency of activities for the activities of the top contributors with more than



(a) Developer `*sa*`. (b) Developer `*re*`. (c) Developer `*is*`. (d) Developer `*oi*`.

Fig. 4. Frequency of Activities during Days of the Week in the Node Project

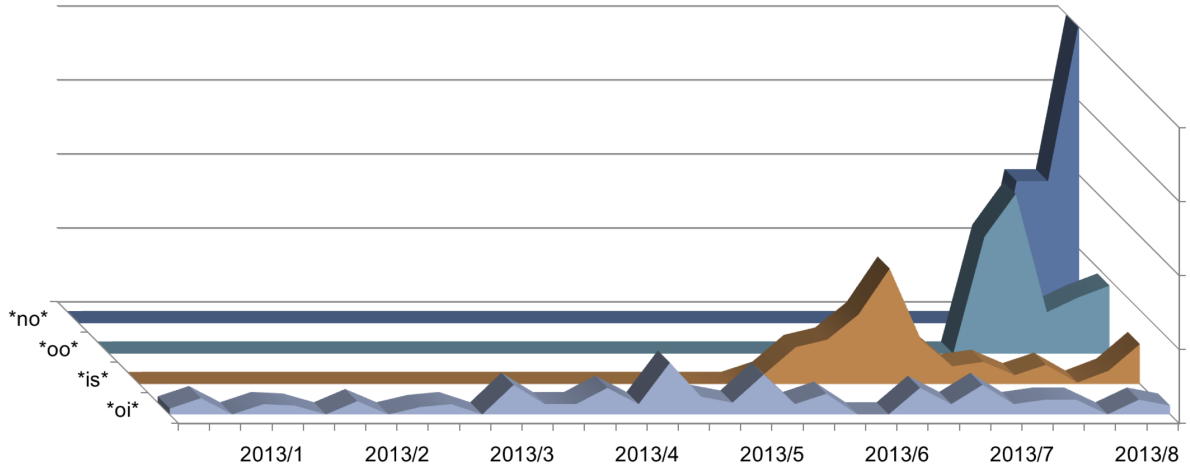


Fig. 5. Distributions of 300 Activities During Weeks for Four Typical Developers in the node Project

100 contributions to the target projects. The majority column shows the distribution of activities discussed in Section IV-A. If developers have a mixture of coding, commenting, and issue handling activities, we identified that as *balanced*. He specialization column shows the contributions to the target and other projects as discussed in Section IV-B. We identify developers as *specialists* if they work mainly on their target project, *others* if they work mainly on other projects, *both* if they work on both the target and other projects, and no contribution when the developer has not contributed to their target project within their 300 activities. The frequency column shows the periods of time needed for each of the developers to produce 300 activity events as described in Section IV-D.

As compared to Table II, which only identified the top contributors and numbers of cumulative comments, Table III shows the various characteristics of contributions for different developers. In this table, we can see that some developers concentrate on coding and/or commenting, while others contribute with a mix of activities including coding, commenting, and issue handling. Top contributors may mainly work on their target projects, some mainly work on other projects, and some have not contributed to their target projects recently. In addition, the activity rates of developers are very different, with some very active while others are much less active. In each of the two target projects, different developers have different characteristics of development activities.

V. Related Work

Social Network Analysis is a related field of study to this research. For example, Bird et al. Reported that developers play a significant social role in email lists [1]. Similarly, Bird et al analyzed email addresses in open source software

TABLE III. HEADINGS: CHANGE SPECIALTY TO SPECIALIZATION, AND ACTIVITIES TO FREQUENCY.

	node					jQuery				
	Developer	Commits	Majority	Specialty	Activities	Developer	Commits	Majority	Specialty	Activities
1	*ry*	2,941	Balanced	Specialist	Years	*er*	1,591	Code/comments	No contribution	Months
2	*sa*	1,413	Code/comments	Both	Weeks	*me*	436	Balanced	Others	Months
3	*no*	1,208	Balanced	Specialist	Days	*za*	318	Comments	No contribution	Months
4	*is*	502	Code/comments	Others	Months	*wl*	297	Balanced	No contribution	Days
5	*nd*	288	Code/comments	Both	Weeks	*im*	267	Code	Others	Weeks
6	*oo*	177	Code/comments	Others	Months	*au*	266	Comments	Specialist	Months
7	*oi*	157	Code	Both	A year	*ra*	248	Code	No contribution	Weeks
8	*el*	119	Comments	No contribution	Months	*le*	200	Balanced	No contribution	A year
9	*re*	114	Code	Specialist	Weeks	*ib*	145	Code/comments	Specialist	Months
10			—			*rk*	117	Code	Specialist	Months

projects to examine the community structure among developers [2]. Although our study collected data from different kinds of development archives, specifically the developers' activity events in GitHub, these results also indicated contributors have many different roles.

Personality Analysis is another related work. For example, using the Myers-Briggs type indicator (MBTI), Sfetsos et al investigated the impact of personality types on pair programming. Their results showed that pairs with heterogeneous and temperaments had better communication, pair performance, and pair collaboration-viability [3]. Salleh et al studied the affects of personality on pair programming using the five-factor model, which characterizes personality in terms of five broad personality traits, openness to experience, conscientiousness, extraversion, agreeableness, and neuroticism [4]. Compared to these personality-based analyses which look at how differences in individuals are reflected in their activities, our study investigated how developers could be characterized based on differences in the patterns of activities extracted from the project activity logs.

VI. Conclusion and Future Work

This study examined the characteristics of developers' activities by collecting and analyzing data from two active software projects in GitHub. This included code-related, comment-related, and issue handling activities. While GitHub provides a list of top contributors in each project based on the number of commits, we found that various contributors had different characteristics in their development activities.

In both projects in this study, the top contributors had a mixture of characteristics. For example, some developers were balanced in doing coding, commenting, and issue handling, while others focused more on code or comments. Further, some developers were specialists, doing most of their work on their own target project, some did most of their work on other projects, and others mixed target and other project contributions. Finally, activity rates varied from days to a year between developers.

While these results are specific to these projects and GitHub as a development environment, they suggest that active software projects need a mixture of developers' characteristics, including generalists and specialists in activities such as coding, commenting, and issue handling, as well as developers who focus on one project and transients who look in on multiple projects. Even the differences in activity levels, while complicating attempts at project management, are part of the open source software environment and should be expected.

Based on this study, there is some desirable future work, including:

- As discussed in section III-C, Threats to Validity, the limitation of the number of available activity events to 300 is problematic. By accessing the API on different days, we might be able to obtain more data and generalize our results. In addition, we should study more projects.
- In this study, if projects were not the target projects, they were grouped as *other projects*. More detailed analyses of which other projects are involved, the patterns of activities in these projects, and studies of the migration of developers between projects may also be useful.
- Based on studies such as this of the classifications of developers and the effects of different types of developers on projects, we may be able to recommend ways to improve management of software development and participation and measure the effects of such changes.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 25880015.

References

[1] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *Proc. of 3rd Int. Workshop on Mining Softw. Repositories*, ser. MSR '06. New York, NY, USA: ACM, 2006, pp. 137–143. [Online]. Available: http://doi.acm.org/10.1145/1137983.1138016

[2] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," in *Proc. of 16th ACM SIGSOFT Int. Symp. on Found. of Softw. Eng.*, ser. SIGSOFT '08/FSE-16. New York, NY, USA: ACM, 2008, pp. 24–35. [Online]. Available: http://doi.acm.org/10.1145/1453101.1453107

[3] P. Sfetsos, I. Stamelos, L. Angelis, and I. Deligiannis, "An experimental investigation of personality types impact on pair effectiveness in pair programming," *Empirical Softw. Eng.*, vol. 14, no. 2, pp. 187–226, Apr. 2009. [Online]. Available: http://dx.doi.org/10.1007/s10664-008-9093-5

[4] N. Salleh, E. Mendes, J. Grundy, and G. S. J. Burch, "An empirical study of the effects of conscientiousness in pair programming using the five-factor personality model," in *Proc. of 32nd Int. Conf. on Softw. Eng.*, ser. ICSE '10. New York, NY, USA: ACM, 2010, pp. 577–586. [Online]. Available: http://doi.acm.org/10.1145/1806799.1806883