Software Team Member Configurations: A Study of Team Effectiveness in Moodle

Noppadol Assavamkamhaenghan*, Morakot Choetkiertikul*, Suppawong Tuarob*, Raula Gaikovina Kula†, Hideaki Hata†, Chaiyong Ragkhitwetsagul*, Thanwadee Sunetnanta*, and Kenichi Matsumoto†
*Faculty of Information and Communication Technology (ICT), Mahidol University
†Nara Institute of Science and Technology (NAIST)
Email: noppadol.assava@gmail.com, {morakot.cho, suppawong.tua, chaiyong.rag, thanwadee.sun}@mahidol.ac.th
{raula-k, hata, matumoto}@is.naist.jp

***Abstract*—Many open source projects organize teams to collaboratively manage their software development activities (i.e. issue resolution processes). Therefore good configurations of software development teams can be an important factor, as effective allocation and completion of tasks may result in a more effective activity (i.e. changing configurations after an issue is reopened). To validate this assumption, we present an exploratory study on software team member configuration when resolving issues. We mine the JIRA issue tracking system to assess whether different team member configurations are quicker to resolve issues after being reopened. In a case study of Moodle, our results confirm that the combinations of team members in different roles correlate with reopened issues and the changing of team members is found to resolve those reopened issues. Moreover, the study shows that the issue type is an important factor when assigning team members.**

***Index Terms*—Team Member Configuration, Issue Tracking Systems, Reopened Issues**

I. Introduction

Software development involves much non-trivial tasks that requires a good collaboration between team members [1]. With the rise of social coding platforms like GitHub, to collaborative management systems, developers now are able to work in teams to develop and maintain code. Software development teams also demand different sets of skills from different roles to deliver a software [1], [2]. An example is that during the issue resolution process, there are specific roles that are designated to members on any issue (e.g. developer, reviewer, tester and so on).

Software quality is heavily based on software development team performance. Sharp and Ryan [3] refer to the software team member configuration as a pattern or combination among team members involved in a specific task. The team member configuration's capability can be thus reflected by the quality of the delivered products [4], [5]. Currently, teams usually work together via an issue tracking system (e.g. JIRA Software) where their progress and activities can be tracked and monitored.

In this paper, we study the correlation between different team member configurations and their performance in resolving issues. The study is empirically performed on over 80,000 issue reports collected from Moodle. An issue report can present different characteristics based on its type such as bug, new feature request, and task. Once an issue has been resolved, it is reopened if the resolution does not meet the expectations (e.g. poor quality). In particular, our study aim to mine and explore whether the the quality of delivered products in forms of issue resolving is affected by the different combinations of team members (i.e. team member configuration). The following research questions are developed to frame our study.

RQ1: Can effective team member configurations be identified?

answer: *Yes, the teams which can resolve issues without reopening can be identified.*

RQ2: Does the team member configuration be changed when issues require additional work (reopen)?

answer: *Yes, the team size is changed when the prior resolution does not meet expectations.*

RQ3: Does the issue type affect the combinations of team members?

answer: *Yes, different types of issues involve with different sets of team members.*

II. Background

A. Team Member Configurations in Software Development

A software development team consists of different roles e.g. designers, developers, and testers. They work together to deliver high quality software. Especially, there are four main roles usually involved in issue resolving (e.g. fixing bugs, implementing new features) which are developers, testers, reviewers, and integrators [2]. We briefly discuss these four roles as follows:

1) Developers: are responsible for writing programming code to resolve an issue either adding new code or editing the existing code. Technical skills is not only the essential component required for developers, but also experiences in a project and expertise in particular project components.

2) Testers: are responsible for testing software components to ensure that it works properly, meets expectations, and does not cause any error in other modules.

3) Reviewers: are responsible for inspecting the implemented code to assure the code quality in several aspects (e.g., understandability, maintainability). This role needs highly expertise in both technical skills and experience in a project.

4) Integrators: are responsible for integrating new code changes to the current code version. Rathan focusing on a particular component of software, integrators need to consider effects of the code changes on all related components.

An effective team configurations is essential to the success of software projects. Especially, in a large software project, software components are split to different teams where different team member configurations have different expertise (e.g., experience in a project) [6]. By considering this, it is important to have a team which suitable to specific tasks. The study from Demirors et al. [7] confirms that the effective team

Dev - Developer,Rev - Reviewer, Int - Integrator, Tes - Tester

Fig. 1. MDL-47270's Overall Process

increases their productivity over the capability of individual team member.

B. Issue Tracking Systems

Issue tracking systems (e.g. JIRA[1], Bugzilla[2]) are used to manage software development project. These systems also aim to support communication and collaboration among team members such as discussion and task assignment. Our study thus makes use of the data collected from the JIRA issue tracking system. JIRA is a popular issue tracking system that hosts a number of well-known open source projects such as Apache, Moodle, and JBoss. This platform represents software-related artifacts as an *issue* including bug report, new feature request, and development task. The characteristics of an issue are reflected by issue's attributes: *Status* shows the current development state of an issue (e.g., open, in-progress), *Component/s* shows which part of the projects the issue involved with, *Summary/Description* is a textual description of an issue, *Priority* shows how important an issue is (e.g., high, medium, and low), and *Resolution* shows how an issue is resolved (e.g., fix, duplicate).

In addition, an issue tracking system also enhances team's collaboration and motivates people to work as a team. For example, an issue can be assigned to different team members that performed different roles. Thus, the progress of each role can be monitored by the status of an issue such as *in-progress*, *resolved*, and *reopened*. The team members who resolve issues are also recorded.

C. Motivating Example

Since our study focuses on team member configurations in issue resolving, in this section we thus discuss scenarios showing how team members use an issue tracking system.

MDL-47270[3] is an example of how a software team use issue tracking system to work together. The overall process of the issue is described in Figure 1. This issue is a bug issue created by *Dev A*. *Dev A* is a developer who fixes this bug, and uploads to github. *Dev A* then changes the issue status to "Waiting for peer review". After that, *Rev B* reviews the code and then sets the status of the issue to "Peer review in progress". Once *Rev B* finishes reviewing, *Rev B* sets the status of issue to "Waiting for integration review". Then *Int C* integrates the code and changes the status according to the process that *Int C* does. After the code is integrated *Tes D* performs a testing and also changes the issue status accordingly. Upon completion of the testing process, the issue is closed with "Fixed" resolution.

In this study, we would wonder if teams affect the quality of issues resolving or not. Beaver and Schiavone [4] reported that "development team skill is found to be a significant factor in the adequacy of the design and implementation, and inexperienced software developers are tasked with responsibilities ill-suited to their skill level, and thus have a significant adverse effect on the quality of the software product." The quality of team members significantly associates with the performance of resolving issues (e.g. resolving time, quality of work products) [5]. These support an idea that the selection of software teams affect the quality of resolving issue.

D. Quality of Issue Resolving

In this research, we use the status of *reopened issue* to indicate the quality of issue resolving. Resolved issues usually are reopened if they were not resolved correctly or the solution did not pass the quality assessment process such as code reviewing. For example, in MDL-50508[4], *Dev A* is a developer who was trying to solve this issue. After *Dev A* finishes the coding part, uploads the code to Github, and changes the issue status to "Waiting for peer review", *Rev B* reviews the code and suggests *Dev A* what should be changed in the code. After *Dev A* finishes editing the code and *Rev B* approves the edited code, *Dev A* then asks for integration by setting the issue's status to "Waiting for integration review". *Int C* continues with integration review and *Int C* asks *Dev A* for the explanation of *Dev A*'s work since *Int C* finds some code that is not necessary. *Int C* then reopens the issue. Regardless, *Dev A* remain unresponsive to *Int C*'s inquiry, consequently, the issue remains reopened and unresolved. This example shows that "Reopened" status of issue can be used to indicate the quality of issue resolution.

III. Analysis Approach

In this section, we briefly discuss the techniques that we use in our study: Association Rule Mining, Jaccard Coefficient, and the Kruskal-Wallis statistical testing.

A. Association Rule Mining

Association Rule Mining is a technique to find patterns in data. The patterns are in the forms of association rules [8] that can be defined as follows. Let $I = \{I_1, I_2, ..., I_n\}$ be a set of items. An association rule is an implication of the form $X \rightarrow Y$, where $X \subset I$,$Y \subset I$ and $X \cap Y = \varnothing$. Tuples $T = \{T_1, T_2, ..., T_n\}$ occur in the dataset where $T \subset I$. There are two important concepts related to the use of the association rule analysis technique: Rule Support and Rule Confidence. Rule Support is denoted by $Support(X \rightarrow Y)$ = (the number

[1]https://www.atlassian.com/software/jira
[2]https://www.bugzilla.org/
[3]https://tracker.moodle.org/browse/MDL-47270
[4]https://tracker.moodle.org/browse/MDL-50508

of tuples containing both X and Y) / (total number of tuples). Rule Confidence is denoted by $Confidence(X \rightarrow Y)$ = (the number of tuples containing both X and Y) / (the number of tuples contain X). Therefore, support shows how frequently the rule appear and confidences show how often the rule appear to be true. The threshold of support and confidence is selected to filter out the association rules with low support or confidence. For example, there are 3 issue in the dataset which are resolved by $\{DevA, RevB, TesC\}$, $\{DevA, RevB\}$ and $\{DevA, RevD, TesC\}$ consequently. The Support of $\{RevB\} \rightarrow \{DevA\} = 2/3$ and its Confidence $= 2/2$. In this research we use the Apriori algorithm [9] implemented in R to mine association patterns.

B. Jaccard Coefficient

Jaccard coefficient is a measurement of the similarity between two sets of items. The formula to calculate the Jaccard coefficient between set A and B is $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$. We use this to measure the differences between two team member configurations.

C. The Kruskal-Wallis statistical test

The Kruskal-Wallis test [10] is a non-parametric statistical test to evaluate the statistically significant differences between two or more groups of samples. Being non-parametric, the test does not make any assumptions about the distribution of the data. We use significance level (α) of 0.05 to conclude whether the differences are significant. In this paper, we used R implementation of the Kruskal-Wallis test [11].

IV. Dataset

In this section, we discuss about our dataset including data collection, preprocessing, and the dataset statistics. We perform our study on the Moodle project due to its large and well-known community with hundreds of contributors.

A. Collecting Data

We used REST APIs provided by the JIRA platform to collect 80,000 issues created between April 2002 to May 2019 from all sub-projects hosted by Moodle. Since our study emphasizes on completed issues, we filtered out incomplete, non-fixed (bug reports), and currently open issues from our dataset. Furthermore, we found that there were some issues which do not have any people in the key roles we are interested in; therefore, we need to filter out these issues. After the aforementioned process, there are 25,910 issues with the total of 710 participants from different roles.

B. Extracting Teams from Issues

Our study focus on four roles (developer, reviewer, tester, and integrator) which mainly involve in issue resolution. We briefly discuss how we process the issue reports to extract team members and their role involved in issue resolving.

- **Developer**: are defined as the users who perform code commit in the Moodle's Github repository. The code commit are recorded in the development field of issue reports where the information related to the development activities can be found.
- **Reviewer, Tester, and Integrator**: As these roles are explicitly recorded under an issue's attributes (i.e. peer reviewer, tester, and integrator field of an issue report), they can be directly extracted.

Regarding extracting of team member configuration, there are different ways to do it in different types of issues. In this paper, we categorized issues into two types as followed:

- **Issues that have not been reopened before.** This kind of issue is very straightforward as they have no indication of reopening. We assume that the team members do not change during the issue resolution. Therefore, the developers, reviewers, integrators, and testers who are participated in these issues are considered as the set of people in a same team member configuration as shown in Figure 2. To extract the team member configuration, we looked into the change log of the issue to identify all of the reviewers, integrators, and testers in the team. Then, we looked at the commit history of issue to identify all of the developers.
- **Issues that have been reopened.** This kind of issue is more complex than the other as the issue was reopened at least once. We partitioned the timeline of this kind of issue into many periods by the time that the issue is reopened as in Figure 2. In each period, we treat the developers, reviewers, integrators, and testers who participated in solving the issue in the period as a team member configuration. Therefore, there can be more than one team member configuration in this kind of issue. To extract team member configurations from these issues, we look at the change log to find the date and time that the issue was reopened, and we separated them into many team member configuration. We then added the reviewers, integrators, and testers who was labeled as their role in the team member configuration according to the time they were labeled. After that, we added the developers to the team member configuration at the time they commit on the Github. In this paper, we call the team member configuration at the initial of issue as "initial team member configuration", and the team member configuration at the time issue is resolved as "resolved issue team member configuration."

After the aforementioned processes, we had the team member configurations of both the issues that were reopened and the issues that have not been reopened. The statistics of the dataset are described in Table I. From this table, the issue type which has the highest average of participants in both issues which have never been reopened and issues which were reopened is *Epic*. This is rational because, an Epic usually refers to a huge chunk of work which can be broken down into many sub tasks. Therefore, more people are typically required to work on an Epic. In addition, we observe that the average of participants in reopened issues is higher than one of issues which have never been reopened. We then perform further analysis and visualization on the dataset to find the answer of our research questions.

V. Research Questions and Discussions

In this section, we described the motivation, approach, and finding for each of our research questions which will show whether the team member configuration affects the ability to solve an issue or not.

A. *RQ1: Can effective team member configurations be identified?*

Motivation. To understand the team member configuration of overall issues, we first need to know whether there is

TABLE I
DESCRIPTIVE STATISTICS OF THE ISSUE REPORTS IN OUR DATASET

Issue Types	#Issues	#Reopen	No. of team members involved in resolved issues				No. of team members involved in reopened issues			
			Min	Mean	Max	S.D.	Min	Mean	Max	S.D.
Bug	16,126	1,735	1	2.9	8	1.45	1	3.48	10	1.69
Epic	27	8	1	3.37	11	2.6	1	5.63	11	3
Improvement	4,059	649	1	3.16	11	1.46	1	4.14	10	1.74
New Feature	934	139	1	3.1	14	1.71	1	4.19	9	1.96
Sub-task	3,436	495	1	2.16	8	1.27	1	3.22	9	1.65
Task	1,328	147	1	2.92	10	1.49	1	3.62	9	1.78
Total	**25,910**	**3,173**	**1**	**2.85**	**14**	**1.47**	**1**	**3.62**	**11**	**1.75**

Team members include developer, tester, reviewer, and integrator.

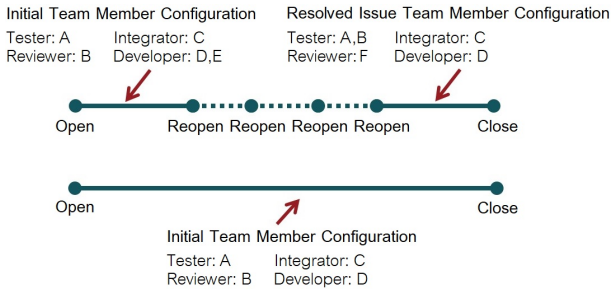

Fig. 2. Example of Team Member Configuration

sufficient evidence that could indicate effective team configurations.

Approach. As we separate issues into issues which were reopened and issues that have not been reopened, our approach for this research question will be analyzing team member configurations of the the two types of issues and compare them. For an issue that was reopened, there are two team member configurations: initial team member configuration and resolved issue team member configuration. For the issue that has not been reopened, there is only one team member configuration. We removed all empty team member configurations. Then, we apply the Apriori algorithm on each team member configuration. The usernames of the issue's participants were used as items and the team member configurations of issues were used as transactions. We set the support threshold to 0.0005 because there will be 0 rule in all team member configurations if we use the default value which is 0.1. This is because there are a huge number of combinations of teams which do not occur frequently, therefore, we need to lower the threshold.

The result is in Figure 3. It shows the number of association rules in each team member configuration. The number of rules of initial team member configuration of reopened issues is lower than resolved issue team member configuration of reopened issues. There are fewer rules than those of the resolved issue team member configuration of issues which have not been reopened. Since the support of association rule mining is related to the frequency that the combination of people occur, and we decreased only support threshold to find the association rules, we can conclude that the team member configuration which has higher number of rules is the team member configuration that is more common, i.e., the team that happens more frequent. In addition, the initial team member configuration of a reopened issue is the team that could not solve the issue, while the other two team member configurations are the team member configurations that could solve the issue. Therefore, the team member configuration

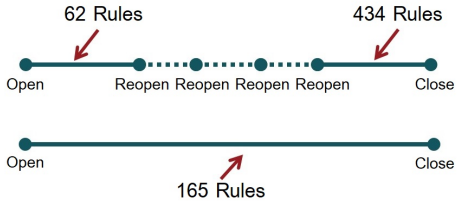

Fig. 3. Number of Association Rule in Each Team Member Configuration

which could solve the issue are more common than the team member configuration which could not solve the issue. This is the evidence of effective team member configuration.

There are differences between the team member configuration which could solve and could not solve the issue.

B. *RQ2: Does the team configuration be changed when issues require additional work (reopen)?*

Motivation. Based on the finding of the previous research question, we know that there exist both effective and ineffective team member configurations. In this research question, we investigated whether there is a modification of team member configuration after an issue is reopened to increase its effectiveness or not.

Approach. We counted the number of people in both initial team member configurations and resolved issue team member configurations of issues that were reopened. A visualization using bar-graph is presented to illustrate whether there is a change in team member configurations or not. To know how much the change is, we use Jaccard Coefficient, which can tell the difference between two sets to compare the initial team member configurations and resolved issue team member configurations. After that, we visualize the Jaccard Coefficient of every issue using a violin plot.

Figure 4 is the visualization of the number of people in a team in initial team member configuration and resolved issue team member configuration. We also observed that 54.82% of the team member configuration become bigger, 30.67% remain the same, and 14.51% become smaller. From this, we know that the team tend to get bigger after an issue is reopened. We also visualize the Jaccard coefficients of issues as in Figure 5. The Jaccard coefficients between initial team member configuration and resolved issue team member configuration tend to be very low in the graph. Therefore, there are huge differences between these two team member configurations.

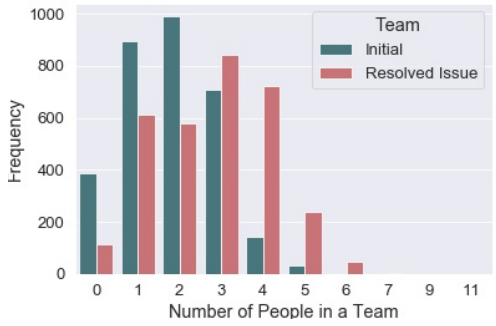

Fig. 4. The Frequency of Size of Teams of Reopened Issue

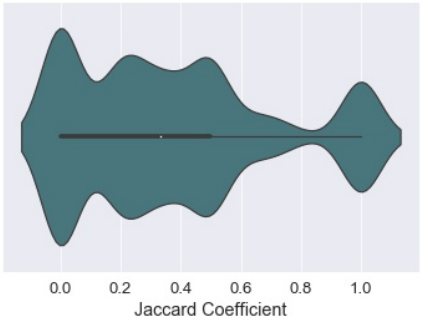

Fig. 5. Distribution of Jaccard Coefficient between Initial and Resolved Issue Team Member Configuration

There are huge differences between Initial and Resolved Issue Team Member Configuration

C. *RQ3: Does the issue type affect the combinations of team members?*

Motivation. Based on the previous research question, we know that there is a change in team member configuration after an issue is reopened. It is also interesting to know whether the changes in team member configuration are different in each type of issue or not.

Approach. To get the overall picture of the changing in team member configurations in each type of issue, we first calculate Jaccard coefficient of team member configurations in each type of issues and visualize them using a box plot. After that, we use KruskalWallis test on the distributions of Jaccard coefficient of types of issues to examine the significance of difference in changes in team member configurations among the two types of issues.

The Kruskal-Wallis test returned the p-value $= 7.783 \times 10^{-10}$, concluding that the change in team member configurations among types of issues are significantly different (p-value< 0.05). In addition, the Figure 6 shows the distribution of Jaccard coefficient of the changes in team member configurations when issues were reopened. As the Jaccard coefficient of *Bug* and *New Feature* distribution is higher than the other types of issues, we can say that *Bug* and *Feature* team member configurations are less likely to change after an issue is reopened.

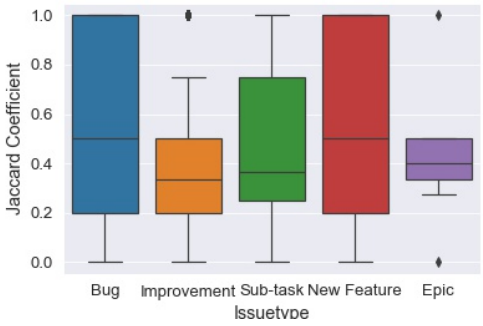

Fig. 6. Boxplots of Jaccard Coefficient of Each Issue Type

There are significant differences between the changes of team member configurations among the types of issues.

VI. Threats to Validity

In this section, we discuss the factors which could affect the validity of our study consisting of threat to construct validity, threat to internal validity, and threat to external validity.

Our threat to construct validity is that we did not take other hidden roles which may be participated in issues into our experiments. There might be other roles which are not explicitly labeled in the dataset; therefore, this could affect the results since our research focuses on the team member configuration, the changing of team member configuration, and size of teams.

Another important threat to validity of our research is the threat to internal validity. Since, we consider the team member configuration between the initial team member configuration and the team member configuration that revolves an issue the changing of the team configuration during the resolving was not considered. In addition, the difficulty of issues was also not considered. We thus mitigate this threat by removing incompleted issues and non-resolved issues from our dataset.

The threat to external validity of our research is generalizability. Since we perform the study on an open source project setting, we thus need to expand our study to other setting such as a commercial software development project.

VII. Related Work

In this section, we discuss the previous work on the team structure and issue reopening.

A. Team Structure

Ortu et al. [12] studied communities-structure of developers in JIRA by analyzing 7 popular projects. They built their dataset by collecting issue from the Apache Software Foundation Issue Tracking system from 2002 to December 2013. They proposed a developer network where the nodes represent the developers and and edge $A \rightarrow B$ represents that the developer A comments on developer B's issue. They found that there is a community of developers in open source project hosted by JIRA. This work used the same data set as our work; however, we perform further analysis on the team structure in JIRA.

Mezouar et al. [13] studied the team structures formed by the developers within the projects on GitHub. They investigated the 7,850 most popular projects and developed a pull-based network which is a network that links two developers together

when one integrates a pull request submitted by each other. They found that only low percentage of projects witness a change of team structure; however, the improvement in team structure shows strong association with the improvement of performance of managing the pull requests. They also found that the projects, characterized with a well-connected, centralized team around core contributors, are associated to higher response, processing and closing of the pull requests. This work is most related to our work since they also investigated the changing of team structure. In addition, our finding that the team member configuration changes over time to resolve an issue is similar to their findings.

B. Reopening of Issues

Caglayan et al. [14] investigated the potential factors of issue reopening. They analysed issues activities of a large release of an enterprise software product. They considered that the followings are the potential cause of issue reopening: developer activity; issue proximity network; static code metrics of the source code changed to fix an issue; issue reports and fixes. They then built logistic regression models to identify the key factors that trigger issue reopening. In addition, they conducted a survey regarding the aforementioned factors with the QA Team of the product. The study reveal that issue complexity and developers workload affect issue reopening. This work investigated on reopened issues, which are similar to ours; however, we perform further analysis on the effects of issue reopening on team member configuration.

Mi and Keung [15] performed empirical analysis on the reopening of bugs. They used four of Eclipse's open source projects (e.g. CDT, JDT, PDE, and Platform). The projects used an Bugzilla as the bug tracking system. They reported the statistic on the bug reopening proportion, impacts, and bug fixing time distribution. They then investigate the causes of bug reopening by looking at developer discussions recorded in Eclipse Bugzilla. They found that 6%-10% of bugs were reopened, 93% of reopened bugs cause serious effect on the normal operation of the system. In addition, they found several key factors of bug reopening (e.g. poor communication, further improvement and Unsuccessful fix). They proposed that providing an effective and efficient communication among bug reporters and developers could reduce the bug reopening rate. This work studied several factors related to reopened issues but not the combinations of the team members.

VIII. Conclusion and future work

In this study, we performed empirical analysis on Moodle Tracker, one of the projects which uses JIRA Software to track their development process. We collected 8,865 issues from Moodle Tracker to analyze the team member configuration in the issues. We found that there are differences between team member configurations which can solve issues and team member configurations which cannot solve issues. We performed further analysis the change in team member configuration between the initial team member configuration and resolved issue team member configuration. We found that 54.82% of teams become larger to solve an issue. We also calculate the Jaccard coefficient between the two team member configurations and found there are many team member configurations which have low Jaccard coefficientd. This mean that there are huge differences between Initial and Resolved Issue Team Member Configurations. Furthermore, we calculated the Jaccard coefficient for each type of issue, and we used Kruskal-Wallis test on the distributions of Jaccard coefficientd of types of issues. We got p-value $= 7.783 \times 10^{-10}$; therefore, we concluded that there are significant differences between the changes of team member configurations among the types of issues. Our findings in this research have established the importance of having an effective team member configuration. As our future direction, we will investigate the possibility develop an algorithm for recommending effective software teams for JIRA software projects. Furthermore, we would like to perform extensive experiments on other projects on JIRA as well.

Acknowledgment

This research project was partially supported by Faculty of Information and Communication Technology, Mahidol University, and JSPS KAKENHI (Grant Numbers 16H05857, 17H00731, 18H04094).

References

[1] Y. Dubinsky and O. Hazzan, "Roles in agile software development teams," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3092, pp. 157–165, 2004.

[2] H. Zhu, M. C. Zhou, and P. Seguin, "Supporting software development with roles," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 36, no. 6, pp. 1110–1123, 2006.

[3] J. Sharp and S. Ryan, "Global agile team configuration," *Journal of Strategic Innovation and Sustainability*, vol. 7, no. 1, p. 120, 2011.

[4] J. M. Beaver and G. A. Schiavone, "The effects of development team skill on software product quality," *ACM SIGSOFT Software Engineering Notes*, vol. 31, no. 3, p. 1, 2006.

[5] M. Hoegl and H. G. Gemuenden, "Teamwork Quality and the Success of Innovative Projects: A Theoretical Concept and Empirical Evidence," *Organization Science*, vol. 12, no. 4, pp. 435–449, 2001.

[6] D. Walz and B. Curtis, "Inside a software design team: Knowledge acquisition, sharing, and integration," *Commun. ACM*, vol. 36, pp. 63–77, 10 1993.

[7] E. Demirors, G. Sarmasik, and O. Demirors, "The role of teamwork in software development: Microsoft case study," in *EUROMICRO 97. Proceedings of the 23rd EUROMICRO Conference: New Frontiers of Information Technology (Cat. No.97TB100167)*, Sep. 1997, pp. 129–133.

[8] P. Prasad and L. Malik, "Using association rule mining for extracting product sales patterns in retail store transactions," *International Journal of Computer Science and Engineering, Volume 3, Issue 5*, vol. 3, 05 2011.

[9] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules (expanded version). Research Report IBM RJ 9839," *Proc. of 20th Intl. Conf. on VLDB*, pp. 487—-499, 1994.

[10] A. Kruskal, William H. Wallis, "Use of Ranks in One-Criterion Variance Analysis Author (s): William H . Kruskal and W . Allen Wallis Published by : American Statistical Association Stable URL : http://www.jstor.org/stable/2280779," *Journal of the American Statistical Association*, vol. 47, no. 260, pp. 583–621, 1952. [Online]. Available: http://www.jstor.org/stable/2280779

[11] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2019. [Online]. Available: https://www.R-project.org/

[12] M. Ortu, G. Destefanis, M. Kassab, and M. Marchesi, "Measuring and understanding the effectiveness of JIRA developers communities," *International Workshop on Emerging Trends in Software Metrics, WETSoM*, vol. 2015-Augus, pp. 3–10, 2015.

[13] M. El Mezouar, F. Zhang, and Y. Zou, "An empirical study on the teams structures in social coding using GitHub projects," *Empirical Software Engineering*, may 2019.

[14] B. Caglayan, A. Misirli, A. Miranskyy, B. Turhan, and A. Bener, "Factors characterizing reopened issues: A case study," in *PROMISE 2012 - 8th International Conference on Predictive Models in Software Engineering, Co-located with ESEM 2012*, 10 2012, pp. 1–10.

[15] Q. Mi and J. Keung, "An empirical analysis of reopened bugs based on open source projects," in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '16, 2016, pp. 37:1–37:10.