Measure Students' Contribution in Web Programming Projects by Exploring Source Code Repository

BAO-AN NGUYEN
Department of Information Engineering and Computer Science
Feng Chia University
Taichung, 407, Taiwan
annb@mail.fcu.edu.tw

KUAN-YU HO
Department of Information Engineering and Computer Science
Feng Chia University
Taichung, 407, Taiwan
M0721053@o365.fcu.edu.tw

HSI-MIN CHEN
Department of Information Engineering and Computer Science
Feng Chia University
Taichung, 407, Taiwan
hsiminc@fcu.edu.tw

2020 International Computer Symposium (ICS) | 978-1-7281-9255-0/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICS51289.2020.00099

***Abstract*—Large scale software systems are usually developed by multiple teams. Communication and collaboration skills are required as essential skills of qualified developers in the modern environment, in addition to the technical hard skills of software engineering. Therefore, group projects are included in almost courses to train students' professional skills and teamwork abilities. However, in their group projects, most of the students are only required to accomplish functional requirements with limited awareness about the quality of the software. This issue seriously affects the development speed, since it is difficult for team members to read others' code during the development process. Hence, we propose a quality-driven assessment system for group projects with the capability of checking the coding style after each submission and providing immediate feedback to help student teams maintain the code quality for their group projects. Besides, to objectively assess the contribution of all students in group projects, we propose a set of repository mining metrics including two major categories of productivity and quality. A case study about the usage of the system and the proposed contribution evaluation metrics are introduced to illustrate the feasibility of our approach.**

Keywords—Group Project, Continuous Integration, Automation, Code Quality

I. INTRODUCTION

Today's large-scale software systems are usually developed by groups [1], and the ability of teamwork is very important for software engineers [2][3]. Therefore, in addition to bringing up students' abilities in program design in program education, students should also practice their teamwork skills. In this regard, the most commonly used method is to let students develop group projects in the course [4][5].

However, evaluating the contribution of students to a group project is a difficult task. In the past, the contribution of each group member could only be judged by the students' mutual evaluation [6] or the teacher's subjective judgment. Nevertheless, both of these methods of assessment results contain too many subjective factors, which may cause the assessment results to be inconsistent with the facts. Therefore, an objective method must be used to evaluate the degree of contribution to make the assessment results more accurate, for example: analyze the contribution of each group member to the group project through the submission record in the code repository [7], but there are two problems:

1) The existing learning management system is not suitable for the delivery of group projects.
2) There is always no consensus among relevant researchers on how to evaluate the contributions of individuals in group projects[8].

In programming education, the importance of code quality[9] is often overlooked. In that case, if the group project is included in the course, students will not pay attention to the code quality when they develop group projects collaboratively. The code is difficult to read if the students do not pay attention to code quality, which leads to the delay of development progress and difficulty in maintaining. Therefore, it is suggested that the quality of the code should be carefully considered when developing a group project.

Based on the discussions above, we propose a group project assessment system that integrates version control systems [10], continuous integration tools [11], and software quality testing, called ProgEdu4Group. This system uses continuous integration technology [12-15] with multiple advantages as an iterative automated program assessment system. After the students delivered the code of the group project, the system uses the static checking tool to check the coding style. If the submission does not meet the coding style, the project status will be marked as a coding style failure (CSF). The test results are presented in the system for students and teachers to review immediately. Besides, this research will propose a set of metrics based on exploring source code repository to measure individual contributions in group projects and analyze the contributions, then present them in the system as visual charts for students and teachers to look over. With these metrics and by the use of the system, we aim to address the following research questions:

RQ1: Do students with excellent final grade modify more code?

RQ2: Do students with excellent final grades submit more code?

RQ3: Do students with excellent final grade fix more errors?

RQ4: Do students with excellent final grade introduce fewer errors?

RQ5: Can ProgEdu4Group effectively help most student groups maintain the code quality?

Authorized licensed use limited to: Norges Teknisk-Naturvitenskapelige Universitet. Downloaded on March 20,2022 at 16:45:02 UTC from IEEE Xplore. Restrictions apply.

RQ6: Does the final grade well reflex students' contribution to the group projects?

II. Related work

A. Automated Program Evaluation Systems

Wu et al. applied the automated program evaluation system to the teaching of C language [16]. Teachers can upload practice questions in the system at any time, the system will automatically check whether the code submitted by the students is correct. It not only reduces the workload of teachers, improves the process of supervising students handing in homework, but also makes scores more objective, improves the quality of teaching, and discovers the following benefits: train students' thinking, get immediate feedback, and inspire interest in learning.

Chen et al. developed an iterative automated program evaluation system called ProgEdu that supports both Java and web programming [17]. In addition to checking whether the code submitted by students is correct, the system also checks whether the code follows the coding style so that students can pay attention to the quality of the code.

In these systems, the teacher must set up questions and write test cases, and then use the test cases to check the code submitted by the students. However, group projects usually let students decide what to do by themself, so it is difficult for teachers to write test cases to check whether the students' code is correct. Therefore, we propose a quality-driven assessment system to check the coding style and code quality and security of the project. To replace the dynamic inspection using test cases for inspection and supports multiple group members to deliver the same project.

B. Contribution Analysis

Jalerson Lima et al. mentioned that the productivity brought about by the contribution of a single developer is an important part of software development companies to maintain their market competitiveness [7]. However, how to measure the productivity or contribution of a single developer remains an open question. Therefore, they proposed a set of contribution metrics based on the mining code repository and issue tracking system based on the experience of the project leader and group leader. It includes code contribution, average complexity of each function, introduced errors, and fixed errors.

Parizi et al. stated that many software engineering related courses include project development and teamwork, though, to fairly evaluate student performance, it is necessary to quantify the workload of each student in a group project [18]. In the past, the most common way was for teachers to subjectively judge the contributions of students through observation, but this is not a truly fair evaluation method and it is very time-consuming. Hence, they proposed a set of Git-driven contribution metrics, including the number of commits, the number of merges, the number of files, the number of code lines.

Bassi et al. mentioned that the use of object-oriented software development methods has many advantages, such as reusability, therefore, to ensure code quality, object-oriented metrics must be considered [19]. They then proposed metrics based on code quality, which includes complexity, inheritance, size, coupling.

From the mentioned related work, we can conclude that in the projects developed in different contexts such as internal projects, open-source projects, and student group projects, the metrics in different contexts are not the same.

III. System architecture

The system architecture of ProgEdu4Group is shown in Figure 1. Using the system, the teachers can add, modify, and delete groups through group management features, as well as assign students to created groups. When the group is established, the system will create a code repository for the group project on the version control server (GitLab [20]). At the same time, the CI server Jenkins creates a job for a group project. This job will include all inspection processes, including code quality checker, security checker (SonarQube [21]), and coding style checker (by HTMLHint [22] and stylelint [23]). The log extractor is responsible for analyzing the log generated by the code analyzer and judging the status of the project after the submission. The judged status of the project after submission is stored in the database. Contribution Analyzer will analyze the contribution of group members in the group project according to the contribution metric proposed by this research, then stores the results in the database.

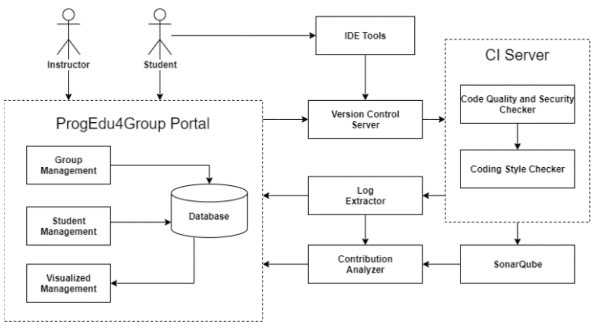

Fig. 1. System architecture diagram

IV. Metrics

We divide the metrics into two types: individual performance and group contribution. As shown in Table I, the individual performance metric aims to present the actual productivity of the individual in the group project. They can be used not only to compare with other members of the group but also to compare with other groups' members. They also can be used to evaluate the student's performance in the class.

Besides the code writing style, we also imported SonarQube to detect bugs, vulnerabilities, and code smells. This part is to analyze whether students have spontaneously improved the quality of the code in addition to complying with the coding style we set. Therefore, students can only get feedback on the coding style test results from ProgEdu4Group. A summary of possible resulted status after each resubmission is shown in Table II. These statuses can be divided into five types, according to the relationship between the status of the project before and after submission.

Authorized licensed use limited to: Norges Teknisk-Naturvitenskapelige Universitet. Downloaded on March 20,2022 at 16:45:02 UTC from IEEE Xplore. Restrictions apply.

TABLE I. INDIVIDUAL PERFORMANCE METRICS

Category	Metrics	Notation
Productivity	Line of Additions	LOCAdd
	Line of Deletions	LOCDel
	Line of Modification	LOCMod
	Number of Submissions	SubTotal
	Frequency of Submissions	SubFreq
Quality	Number of Conform Coding Style (Build Success)	BuildSucc
	Number of HTML CSF	HtmlCsf
	Number CSS CSF	CssCsf
	Number of Introduced Errors	IntroErr
	Number of Unfixed Errors	UnfixErr
	Number of Keep Right	KeepRight
	Number of Partial Errors Fixed	PartFix
	Number of All Error Fixed	AllFix
	Total number of fixed errors	TotalFix
Quality (SonarQube)	Number of Introduced Errors	SonarIntroErr
	Number of Fixed Errors	SonarFixErr

TABLE II. RELATION OF PROJECT STATUS BEFORE AND AFTER SUBMISSION

Project Status (Before Submission)	Project Status (After Submission)	Relation
Initialization	CSS CSF	IntroErr
Initialization	HTML CSF	IntroErr
Build Success	CSS CSF	IntroErr
Build Success	HTML CSF	IntroErr
CSS CSF	HTML CSF	IntroErr
HTML CSF	CSS CSF	PartFix
HTML CSF	Build Success	AllFix
CSS CSF	Build Success	AllFix
HTML CSF	HTML CSF	UnfixErr
CSS CSF	CSS CSF	UnfixErr
Initialization	Build Success	KeepRight
Build Success	Build Success	KeepRight

With group contribution, we measure the percentage of contribution of each student to his/her group. The contribution percentage of a student i in a group G on the metric M is computed by:

$$Pc.M_i = \frac{M_i}{\sum_{i \in G} M_i}$$

As a result, we can compute the contribution percentage of all individual metrics in Table I, except the SubFreq. We use these two groups of metrics for analyzing the association between students' contributions and the final grade. The analysis results are shown in the following section.

V. EXPERIMENTAL RESULTS

We apply ProgEdu4Group to the elective course (Web Programming) offered by the Department of Information Engineering and Computer Science of Feng Chia University, which is for sophomore. In this course, students will learn the knowledge and technology of web application development, of which the front-end technology is the focus, including HTML, CSS, JavaScript. In addition to the professional skills mentioned above, we also believe that the ability of teamwork and communication is very important, even as important as professional skills. Therefore, in this course, students are required to form a group to develop a group project and publish it at the end of the semester, to cultivate students' collaborative ability.

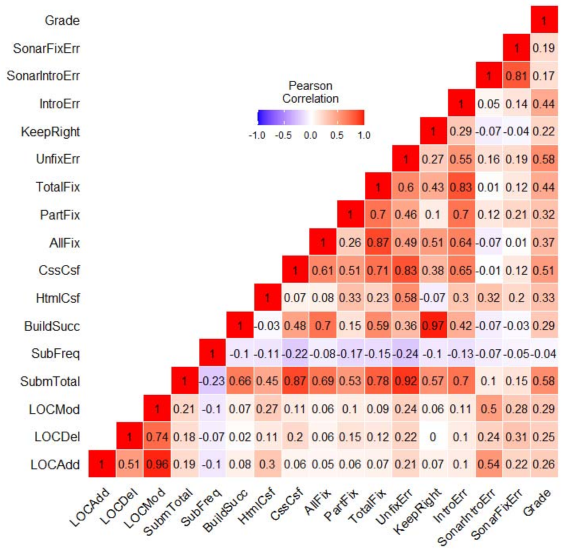

Fig. 2. Correlations matrix between individual features.

There are 41 groups in this course, with a total of 146 students participating in this experiment. The student group must use the knowledge and technology learned in this course to complete a static web page and use ProgEdu4Group to deliver the code during the development process. Whenever the code is delivered, ProgEdu4Group will check whether the group project conforms to the coding style and give feedback immediately so that the student group can maintain the code quality of the project.

We collect the submission records of the students, and analyze the contribution of their submission behavior, and analyze the correlation between the measurement results and the final grade (45% for group projects, 30% for mid-term and final exams, and 25% for homework) to verify the research questions. The Pearson correlation coefficients between individual features are shown in the heatmap in Fig. 2.

A. *Individual Performance*

In terms of code modification, as shown in Figure 2, there is a large gap between the number of LOCs added and deleted. We add *LOCAdd* and *LOCDel* to be *LOCMod* (Line of Modifications) and analyze the correlation with the final grade. The correlation

Authorized licensed use limited to: Norges Teknisk-Naturvitenskapelige Universitet. Downloaded on March 20,2022 at 16:45:02 UTC from IEEE Xplore. Restrictions apply.

coefficient is 0.29, which is a modestly positive correlation. We find that the *LOCMod* is not highly correlated with the final grade. According to the experimental results, the first research question (RQ1: Do students with excellent final grade modify more code?) is disconfirmed in an individual context. Students with excellent grades in the semester do not necessarily modify more code

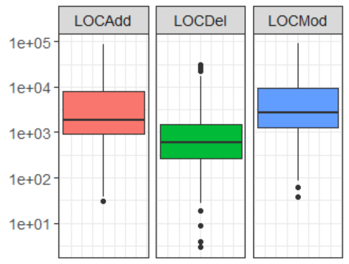

Fig. 3. Boxplots indicate code modifications (LOCs)

In terms of the number of submissions, as shown in Figure 2, the correlation coefficient between the *SubmTotal* and the final grade is 0.58, which is a moderately positive correlation. Therefore, according to the experimental results, the second research question (*RQ2: Do students with excellent final grade submit more code?*) is confirmed. The students with excellent grades make more submissions of code.

In terms of the submission frequent, the time interval between two consecutive submissions is used as an estimate of their development time. As shown in Figure 3, the median frequency of code submitting of students is every 30 hours, even though the first quartile (Q1) reached 14.8 hours. This far exceeds the time it takes to complete a function development normally. We believe this is because, in university education, students usually take multiple courses at the same time, which makes us unable to estimate the time it takes to develop it. Therefore, this metric is not suitable for use in university education.

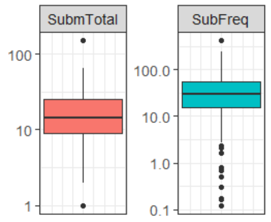

Fig. 4. Boxplots indicate the total of submissions and submission frequency *(in hours)*

In terms of the coding style, as shown in Figure 3, each student has an average of 18.4 submission records. Among returned results, there are 2.95 times of *BuildSucc*, 5.13 times *HtmlCsf*, and 10.32 times *CssCsf.* The results indicate that most students rarely check whether the submitted code conforms to the coding style before submitting it. Therefore, the project status in most of the submission records does not conform to the coding style.

In terms of the project status before and after submission, as shown in Fig 2, the correlation coefficient between the total number of fixed errors and the final grade is 0.437674436, which is a moderately positive correlation. Therefore, according to the experimental results, the third research question *(RQ3: Do students with excellent final grade fix more errors?)* is confirmed. The students with excellent final grades do usually fix more errors. In terms of introduced errors, the correlation coefficient between the number of *IntroErr* and final grade is 0.436174923, which is a moderately positive correlation. Hence, according to the experimental results, the fourth research question *(RQ4: Do students with excellent final grade introduce fewer errors?)* is disconfirmed. The students with excellent final grades do usually introduce more errors. The main reason for this situation is that students with excellent final grades usually have more submissions. This also means they have more opportunities to introduce errors.

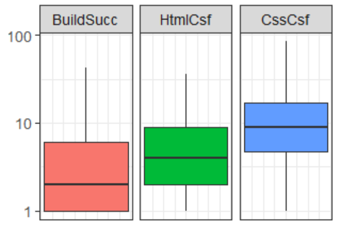

Fig. 5. Boxplots indicate submission built results

According to the experimental results, each student has an average of 18.4 submission records. Among given status after resubmissions, there are 1.52 times of *IntroErr*, 13.16 times of *UnfixErr*, 2.2 times of *KeepRight*, 0.67 times of *PartFix*, 074 times of *AllFix*. This means that most students will not immediately fix the errors after receiving feedback on errors in coding style. However, 53.6% of the group projects have a final build status that conforms to the coding style. This means that many groups immediately fix them even if they don't encounter coding style errors, they will fix the errors before the final presentation. According to the experimental results, we verify the fifth research question (*RQ5: Can ProgEdu4Group effectively help most student groups maintain the code quality?)* is confirmed. This result is consistent with the findings of prior researches on the ancestor of ProgEdu4Group, ProgEdu[22][23] . However, there are still a small number of student groups that do not follow the coding style, so we recommend to add restrictions on this to force the student group to maintain the code quality and let students understand the importance of code quality.

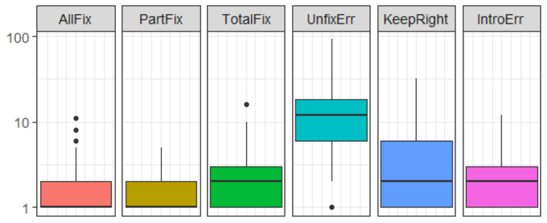

Fig. 6. Boxplots indicate results of resubmissions

Authorized licensed use limited to: Norges Teknisk-Naturvitenskapelige Universitet. Downloaded on March 20,2022 at 16:45:02 UTC from IEEE Xplore. Restrictions apply.

In terms of errors measured by *SonarQube*, as shown in Figure 6, the average of the *SonarIntroErr* is 93.7 times, while the average of *FixSonarErr* is only 37.5 times, although only about 40% of errors are fixed. Because these errors are spontaneously fixed by the students, we believe this is a good result, which means that in addition to the coding style we established, students will also improve other code quality.

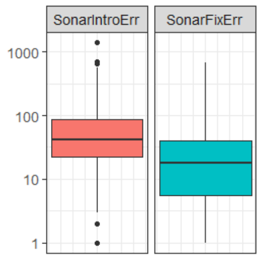

Fig. 7. Boxplots indicate SonarCube Errors

B. Group Contribution

We use the contribution percentages to analyze the association between the project contribution of students and the final grade. The analysis is conducted to answer the sixth research question. First, we rank the students in each group by their grade sorted in descending order and assign one of four letters {A, B, C, D} according to their rank in the group, respectively. Since there are three or four students in a group, the students with the highest grade in a group will be assigned A, followed by the students assigned B, and the students with the lowest grade will be assigned C or D. The Table III shows the mean and the standard deviation of contribution percentages of students in group projects. We employed pair-wise t-test to all groups of students. If the p-value is less than 0.05, we can conclude that there is a significant difference between students in the group in the considered metric. A visualization of this analysis is shown in Fig. 7.

TABLE III. DIFFERENCES IN STUDENTS' CONTRIBUTION TO GROUP PROJECT (MEAN (SD))

	Stud. A	**Stud. B**	**Stud. C**	**Stud. D**	**p-value**
Grade	82.20 (5.73)	77.66 (6.00)	70.56 (7.67)	61.91 (9.16)	< 0.001
SubFreq	37.86 (41.27)	39.06 (31.93)	59.27 (78.03)	34.47 (36.97)	0.674
Percentage of project contribution (%)					
LOCAdd	38 (28)	32 (27)	21 (21)	16 (16)	0.001
LOCDel	38 (22)	34 (24)	20 (22)	15 (16)	< 0.001
LOCMod	39 (26)	32 (26)	20 (21)	15 (15)	< 0.001
SubmTotal	38 (14)	33 (10)	19 (12)	17 (08)	< 0.001
BuildSucc	39 (33)	28 (31)	17 (26)	07 (14)	< 0.001
HtmlCsf	34 (24)	25 (20)	20 (23)	20 (24)	0.017
CssCsf	38 (20)	33 (19)	18 (15)	15 (14)	< 0.001
AllFix	44 (44)	15 (25)	16 (30)	01 (05)	< 0.001
PartFix	40 (39)	27 (34)	15 (26)	14 (26)	0.003
TotalFix	45 (30)	28 (29)	17 (22)	11 (17)	0.369
UnfixErr	39 (17)	31 (13)	19 (13)	16 (08)	< 0.001
KeepRight	25 (32)	29 (35)	13 (24)	11 (26)	0.061
IntroErr	41 (30)	27 (27)	25 (26)	08 (17)	< 0.001
SonarIntroErr	32 (25)	37 (25)	19 (19)	21 (19)	< 0.001
SonarFixErr	31 (25)	37 (26)	24 (23)	16 (20)	0.002

Table III shown that there are significant differences in the contributions of students to group projects in almost metrics. Groups of students A and B contribute the majority of works in the project and resulted in the higher final grade. The most significant contribution of high-performance students in the group is that they fixed more errors in the counterpart. 45% of fixed errors were done by students in group A. The students in group B seem to have comparable contributions with students in group A in some metrics, especially in *SonarQube* related errors. Thus we can assume that the second rank students in groups are encounter more errors in code than the first rank ones.

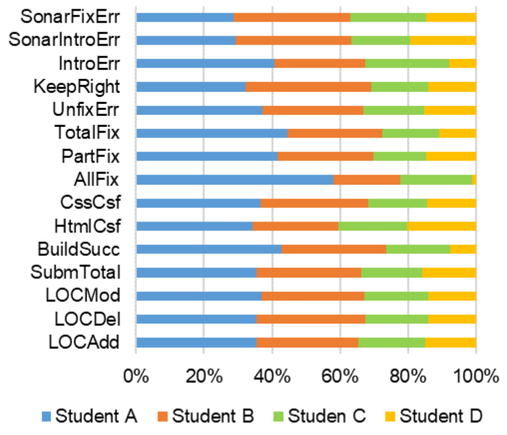

Fig. 8. Contribution percentages of groups of students.

All the above analysis supports the confirmation of the sixth research questions *(RQ6: Does the final grade well reflex student's contribution to the group project?)*. The differences in contribution percentage are well reflexed by differences in final grades. In other words, our systems and proposed metrics can be a good framework for capturing and measuring the project contribution of students.

VI. CONCLUSION AND FUTURE WORK

We propose a quality-driven assessment system for group projects, which can accept the delivery of group projects, ProgEdu4Group. The system can accept the delivery of group projects, and after assessing the coding style, it immediately gives students feedback regarding coding style assessment results. Through the feedback given by the system, students can quickly fix parts of group projects that do not conform to the coding style and help the student group maintain its code quality.

Besides, we proposed a set of metrics based on mining code repositories to evaluate the contributions of students in group projects. Analysis based on these metrics and final grades helps us to verify six research questions with results summarized as follows: i) There is a weak association between modified *LOCs* and final grade; ii) Students with better grade submitted more code; iii)

Authorized licensed use limited to: Norges Teknisk-Naturvitenskapelige Universitet. Downloaded on March 20,2022 at 16:45:02 UTC from IEEE Xplore. Restrictions apply.

Students with better grade fixed more errors; iv) Students with better grade do not introduce fewer errors, yet more errors instead; v) ProgEdu4Group effectively help more than a half of student groups maintain the code quality; vi) The final grade well reflexes the contribution of students' contribution to the group projects.

In the future, we aim to use the data set collected in this study for further analysis to better understand the contribution behavior of students to group projects. Clustering might be applied to detect latent groups of students based on their contributions. A predictive model to predict students' final grades can be built later on to provide an early warning system for students with lower predicted grades. For example, since it is confirmed that students who submit more frequently usually have higher final grades, so if the number of submissions of a group/individual to group project is too small, a warning message will be provided to students to remind the student to be more involved in the group project.

REFERENCES

[1] O. Macek and M. Kom´rek, "Evaluation of Student Teamwork," In *Proceedings of 2012 IEEE 25th Conference on Software Engineering Education and Training*, pp. 130-133, 2012.

[2] G. Matturro, F. Raschetti and C. Fontán," In *Proceedings of Soft Skills in Software Development Teams: A Survey of the Points of View of Team Leaders and Team Members," 2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 101-104, 2015.

[3] D. Damian and A. Borici, "Teamwork, coordination and customer relationship management skills: As important as technical skills in preparing our SE graduates," In *Proceedings of 2012 First International Workshop on Software Engineering Education Based on Real-World Experiences (EduRex)*, pp. 37-40, 2012.

[4] J. Straub, S. Kerlin and D. Whalen, "Teaching software project management using project based learning (PBL) and group projects," In *Proceedings of 2017 IEEE International Conference on Electro Information Technology (EIT)*, pp. 016-021, 2017.

[5] L. Collingbourne and W. K. G. Seah, "Teaching project management using a real-world group project," In *Proceedings of 2015 IEEE Frontiers in Education Conference (FIE)*, pp. 1-8, 2015.

[6] S. Battur et al., "Enhancing the Students Project with Team Based Learning Approach: A Case Study," In *Proceedings of 2016 IEEE 4th International Conference on MOOCs, Innovation and Technology in Education (MITE)*, pp. 275-280, 2016.

[7] J. Lima, C. Treude, F. F. Filho and U. Kulesza, "Assessing developer contribution with repository mining-based metrics," In *Proceedings of 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 536-540, 2015.

[8] G. S. d. A. Junior and S. R. d. L. Meira, "Towards Effective Productivity Measurement in Software Projects," In *Proceedings of 2009 Fourth International Conference on Software Engineering Advances*, pp. 241-249, 2009.

[9] X. Fang, "Using a coding standard to improve program quality," In *Proceedings of Proceedings Second Asia-Pacific Conference on Quality Software*, pp. 73-78, 2001.

[10] D. Spinellis, "Git," in IEEE Software, vol. 29, no. 3, pp. 100-101, May-June 2012.

[11] N. Seth and R. Khare, "ACI (automated Continuous Integration) using Jenkins: Key for successful embedded Software development," In *Proceedings of Proceedings 2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS)*, pp. 1-6, 2015.

[12] A. Agarwal, S. Gupta and T. Choudhury, "Continuous and Integrated Software Development using DevOps," In *Proceedings of* 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE), pp. 290-293, 2018.

[13] P. Agrawal and N. Rawat, "Devops, A New Approach To Cloud Development & Testing," In *Proceedings of 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, pp. 1-4, 2019.

[14] P. Perera, R. Silva and I. Perera, "Improve software quality through practicing DevOps," In *Proceedings of 2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, pp. 1-6, 2017.

[15] P. Perera, M. Bandara and I. Perera, "Evaluating the impact of DevOps practice in Sri Lankan software development organizations," In *Proceedings of* 2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer), pp. 281-287, 2016.

[16] H. Wu, Y. Liu, L. Qiu and Y. Liu, "Online Judge System and Its Applications in C Language Teaching," In *Proceedings of 2016 International Symposium on Educational Technology (ISET)*, 2016, pp. 57-60, 2016.

[17] H. M. Chen, W. H. Chen, C. C. Lee," *An Automated Assessment System for Analysis of Coding Convention Violations in Java Programming Assignments," Journal of Information Science and Engineering*, Vol. 34, No. 5, 2018.

[18] R. M. Parizi, P. Spoletini and A. Singh, "Measuring Team Members' Contributions in Software Engineering Projects using Git-driven Technology," In *Proceedings of 2018 IEEE Frontiers in Education Conference (FIE)*, pp. 1-5, 2018.

[19] P. R. de Bassi, G. M. P. Wanderley, P. H. Banali and E. C. Paraiso, "Measuring Developers' Contribution in Source Code using Quality Metrics," In *Proceedings of 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 39-44, 2018.

[20] J.M. Hethey, "GitLab Repository Management," Packt Publishing Ltd, 2013.

[21] D. Marcilio, R. Bonifácio, E. Monteiro, E. Canedo, W. Luz and G. Pinto, "Are Static Analysis Violations Really Fixed? A Closer Look at Realistic Usage of SonarQube," In *Proceedings of 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC), Montreal*, pp. 209-219, 2019.

[22] Y.-X. Yan, J.-P. Wu, B.-A. Nguyen, and H.-M. Chen, "The Impact of Iterative Assessment System on Programming Learning Behavior," in *Proceedings of the 2020 9th International Conference on Educational and Information Technology*, 2020, pp. 89–94.

[23] H.-M. Chen, B.-A. Nguyen, Y.-X. Yan, and C.-R. Dow, "Analysis of Learning Behavior in an Automated Programming Assessment Environment: A Code Quality Perspective," *IEEE Access*, 2020.

Authorized licensed use limited to: Norges Teknisk-Naturvitenskapelige Universitet. Downloaded on March 20,2022 at 16:45:02 UTC from IEEE Xplore. Restrictions apply.