Poster: A Multi-Year Analysis of Students' Build Errors in Agile Software Development Educational Projects

Erina Makihara
Nara Institute of Science and Technology
Nara, Japan
makihara.erina.lx0@is.naist.jp

Hiroshi Igaki
Osaka Institute of Technology
Osaka, Japan
hiroshi.igaki@oit.ac.jp

Norihiro Yoshida
Nagoya University
Aichi, Japan
yoshida@ertl.jp

Kenji Fujiwara
National Institute of Technology, Toyota College
Aichi, Japan
fujiwara@toyota-ct.ac.jp

Naoki Kawashima
Nara Institute of Science and Technology
Nara, Japan
kawashima.naoki.kd5@is.naist.jp

Hajimu Iida
Nara Institute of Science and Technology
Nara, Japan
iida@itc.naist.jp

ABSTRACT

Team exercises for software development project-based learning (SDPBL) adopting an agile development model have become popular for training and education worldwide. In the agile development model, an essential part is the build process. In this study, we investigated students' build errors in agile SDPBL projects by monitoring and collecting logs of the build process from 2013 to 2016. From 2013 to 2015, we categorized the build errors and then discussed the resolutions for each types of build errors. In 2016, the instructors modified the SDPBL project the build error types and corresponding cause and resolution. As the result, in 2016, the number of build errors and the time required to solve the build errors decreased compared to previous years.

CCS CONCEPTS

• **Software and its engineering → Agile software development**; *Software creation and management*; *Programming teams*;

1 INTRODUCTION

For software development exercises using project-based learning (referred to here as SDPBL) adopting agile development model, it is important to mentor students not only about programming, but also about the build process. According to many previous studies, building is regarded as a core step in software development[1, 2]. In this study, we collected the logs of build behaviors in an SDPBL project which is an educational project for master course students. We analyzed the build logs collected from 2013 to 2015 to answer the following research questions:

RQ1 What types of build errors occurred?
RQ2 How often does each type of build error occur?
RQ3 How long does it take to fix each type of build error?

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195064

Table 1: Statistics of Teams for Each Year

Year	2013	2014	2015	2016
# of Students	49	54	46	54
# of Teams	9	9	8	9

In this investigation, we collected two types of build logs, the local one and the remote one. The local build logs are the records of builds performed manually by each student in their local programming environment. On the other hand, the remote build logs are the records of builds performed automatically whenever any team member commits her/his source code to the team repository.

The knowledge that we gained from our investigation of the logs collected from 2013 to 2015 helped us to improve the educational methods used in 2016. As the result, in 2016, the number of build errors that occurred in remote builds are decreased and the time required to solve build errors was shorter than in the past three years. Based on these results, investigating build behaviors and providing corrections and feedback facilitated students' comprehension of the build errors.

2 INVESTIGATION TARGET AND RESEARCH QUESTIONS

In this study, we investigated the build errors occurring by students in a SDPBL exercise conducted as part of CloudSpiral, one of the educational projects in Japan for the students in master course. The SDPBL exercise is conducted as a five-day hackathon in the middle of every August. During these five days, students are required to develop a web application in a team consisting of five or six students using an agile development method. The numbers of students and teams are shown in Table1.

This study aimed at identifying the characteristics of build errors and the pitfalls of learning the build process through SDPBL. Based on knowledge gained from our investigation, we aimed to develop educational guidelines which should be considered by instructors when they mentor students who perform development in teams as in SDPBL. To achieve our research goal, we defined the following research questions:

Table 2: Statistics of Build Errors for Each Category and Success Builds

	LOCAL				REMOTE			
Classification	2013	2014	2015	2016	2013	2014	2015	2016
C1: Dependency	34	14	18	34	86	118	64	18
C2: Syntax	12	6	12	7	0	10	0	0
C3: TypeMismatch	2	3	1	16	28	27	18	2
C4: Semantic	6	16	2	10	19	5	10	2
C5: Annotation	6	22	11	15	16	59	2	8
C6: Environment	280	81	253	249	0	0	0	0
Total # of C1 to C5	60	61	44	82	149	219	94	30
Total # of Success Builds	2395	1506	1670	2655	2309	1610	1764	2065

RQ1 What types of build errors occurred?
RQ2 How often does each type of build error occur?
RQ3 How long does it take to fix each type of build error?

To answer RQ1, we categorized the build errors collected from the SDPBL logs using categories based on the study by Seo et al. at Google [2]. This allows us to identify the build errors which students commonly faced in this development process. The build error categories are based on Seo et al. [2]. To answer RQ2, we analyzed the frequency of each type of build errors in both the local and remote build logs. This allowed us to compare the frequencies of the different types of build errors to identify those that are most common in local builds, and those that commonly occur in remote builds. To answer RQ3, we measured how long it took to fix each type of build error identified in RQ1. We assume that the results of RQ3 reveal which types of build errors should be given priority in instruction, as well as supported by additional lecture material or educational tools.

3 RESULT FROM 2013 TO 2015

RQ1: What types of build errors occurred?: As shown in Table 2, we categorized all the build errors from the failure builds in Table 2 into 6 categories. During our investigation of RQ1, we added two new types of build errors, *Annotation* and *Environment*, to the error categories described by Seo et al. [2]. However, *Environment errors* are mostly caused by the development environment or tools, which are mostly prepared by the instructors. Therefore, we omitted the environment data from the following results because we do not regard the *Environment errors* as a target of our investigation.
RQ2: How often does each type of build error occur?: Table 2 shows the statistics of build errors in each categories. This indicates that both students and expert developers tend to make *Dependency errors*[2]. However, as can be seen in Table 2, more *Dependency errors* occurred in remote builds than in local builds. Based on this, we considered *Dependency errors* as one of the characteristics of team development. When different students separately implement the dependency files (e.g. test file and target file), frequently the one file is committed to the team repository before another dependency file is committed to the repository. Therefore, we consider it important that all the team members understand the details of each member's tasks and progress before each commit.

As the other finding, all the teams which had *Syntax errors* in their remote builds did not encounter any syntax errors in their local builds. Therefore, we consider that students can prevent *Syntax errors* from occurring in the team repository by thoroughly checking *Syntax errors* in their local builds.
RQ3: How long does it take to fix each type of build error?: Except for *Syntax*, *TypeMismatch*, and *Semantic errors* which contain a few build errors, the resolution times for *Dependency*, *Annotation errors* in remote builds was longer than for local builds. Furthermore, when multiple types of build errors are occurred by one remote build, it is the longest time to be solved. We assume that educators should teach students the risk of mixing multiple types of errors for preventing the prolongation of resolution time.

4 IMPROVEMENT IN 2016 AND CONCLUSION

Based on the results of our investigation from 2013 to 2015, we improved the project in the year of 2016 as follows:

(1) In the beginning of SDPBL in 2016, the instructors explained the main causes of each type of build error, as well as the importance of local builds. The instructors also explained that the spread of bugs and mixed errors occurs if students ignore fixing multiple errors in remote builds, which makes the resolution times become longer.
(2) After an iteration of development, the instructors required students to investigate the cause of each remote build error that occurred during the development for each team.
(3) After the investigation by the students, the instructors determined whether the error was caused by the lack of local builds or the lack of communication among the team members, and gave feedback on it to each team.

Not only the highest number of local but also the lowest number of remote build errors occurred in 2016 as shown in Table 2. According to these results, most students in each of the teams successfully remembered to perform local builds before each commit.

The resolution times for both local and remote build errors were shorter in 2016 than from 2013 to 2015. In addition, the number of multiple errors occurred by one remote build was reduced. We believe that it is helpful for students to teach the threat of mixing multiple types of errors in remote build and the spread of bugs in order to decrease the remote build errors and reduce the resolution time.

Based on these results, we found that the investigation of build errors and its feedback facilitates students' comprehension of the errors. Also, it decreased the number of the remote build errors.

ACKNOWLEDGMENTS

The CloudSpiral project, which is the target of this study, has been conducted as a part of the enPiT (Education Network for Practical Information Technologies) project. I would like to thanks to participants and instructors in CloudSpiral project. This study was supported by JSPS KAKENHI Grant Number 17K00500.

REFERENCES

[1] Ahmed E. Hassan and Ken Zhang. 2006. Using Decision Trees to Predict the Certification Result of a Build. In *Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering*. 189–198.
[2] Hyunmin Seo, Caitlin Sadowski, Sebastian Elbaum, Edward Aftandilian, and Robert Bowdidge. 2014. Programmers' Build Errors: A Case Study (at Google). In *Proceedings of the 36th International Conference on Software Engineering*. 724–734.