Code Review in Computer Science Courses: Take One

Zuzana Kubincová and Martin Homola

Comenius University in Bratislava,
Mlynská dolina, 842 48 Bratislava, Slovakia
{kubincova,homola}@fmph.uniba.sk

Abstract. Code review is an important tool regularly employed in information systems development practice. In this preliminary study we tested employment of code reviewing in educational setting in two Computer Science university-level courses: (a) in the first course we collected code reviews using an existing Learning Management System (LMS) that we regularly use to administer peer reviews; (b) in the other one we collected them using GitLab, an industrial strength source code management system. We compare results obtained from both experiments. Compared to our existing LMS, GitLab is readily equipped with more elaborate features for reviewing, commenting, and discussing code submitted by others, and the process is similar to industrial practice. However, as we also learned, the code reviewing workflow in GitLab is too open, which may be limiting in educational practice (e.g., the comments are immediately visible to everyone, hence it may happen that very active students leave little space for the remaining reviewers). This shows interesting directions for future research: one, to develop specific code-reviewing tool tailored to educational practice; but also to come up with novel methodologies which would enable to use real industrial tools while overcoming the issues pointed out in our study.

Keywords: Code review, peer review, education, tools

1 Introduction

Code review is an activity when a computer program code is reviewed by peers with the aim to find and fix its defects and thus improve the program quality. During this activity a reviewer reads the code (without compiling or executing it) and tries to understand its behavior and to detect and correct its defects by herself [2, 16]. This activity is essential especially in development large software applications since it can reveal 50-70 percent of defects [20]. To improve the process of code reviewing as well as its results new methods are continuously developed and studied [16, 10]. There is also research dealing with the influence of the personality on the person's performance on a code reviewer task [4, 5]. However, as pointed out in the study of Bacchelli and Bird [1], nowadays, code reviewing is less formal and less strict than e.g. 30 years ago: "The reviews are less about defects and instead provide additional benefits such as knowledge transfer, increased team awareness, and creation of alternative solutions to problems."

As already found out by researchers [13, 18] code review can serve as an activity suitable for educational purposes, where it can have various benefits on the learners. The other argument for employment of code review in our teaching practice, is that

since it is widely viewed as a well-established best practice in software engineering, graduates of computer science, software engineering, and related university programs should gain experience in this area, and should be equipped with basic code reviewing skills.

In this paper, we report on a preliminary qualitative study, that we conducted in order to prepare a wider application of code review in our university Applied Informatics curriculum. In the study we have focused on programming project assignments that students develop independently, and we have evaluated two systems to deliver the code reviews: our own previously developed LMS system with basic peer review capabilities [9]; and Gitlab, a widely used open-source code management platform with good code review capabilities.

The aims of our preliminary study were: (a) to test and evaluate both systems for code-reviewing associated with a real programming project assignment; (b) to better understand the requirements for delivering code reviews in educational settings, and to help us decide whether we want to continue using our own LMS system (with necessary enhancements) or it is better to switch to an established code-review platform with code-review features such as Gitlab or Github; (c) to evaluate also students' and teachers' perception and acceptance of the code reviewing activity.

Our small sample of students, and the teachers involved in our experiments, perceived the activity positively, which encourages us to employ code-reviews more widely in our educational practice. Our evalation of the systems shows, that while many useful features are offered by Gitlab, some requirements inherent to educational setting are not so easily met. We therefore plan to improve our own LMS system in the future, extending it with selected code-reviewing features.

2 Related Work

There are many possible ways how to integrate code reviewing into educational activities. Trytten [15] has developed a new design of code review activity for her introductory programming course. This activity comprised individual as well as team code reviews. Students who were divided into small groups performed the code review activity in three phases: at first individually, then within their group and afterwards also between groups. All of them reviewed the same code and answered a set of questions prepared by teachers. The activity ended in a whole-class discussion about the answers. Code reviewing was accepted by students quite positively. They gained some experience in reading code, have seen alternative designs for their own code and have learned about the necessity of well-developed communication skills in teamwork. Hundhausen et al. adapted the formal code review process for use in computer science courses and implemented 'pedagogical code review' [11]. Students in groups of 3–4 people were assigned roles of author, reader, inspector and recorder and in several rounds (according to the number of group members) read each other's program code, evaluated it against a predefined list of best coding practices, discussed all found issues and recorded them. All these steps were performed under the supervision of an experienced moderator. An analysis of this activity confirmed its positive influence on the improvement of the students' code quality, stimulation of more sophisticated discussions about programming

issues and practices and raise of the sense of community. There are also studies reporting the effort of the researchers to refine the definitions of roles and documents related to peer reviewing process with the aim to standardize the process and also the research in code review area and to simplify the communication among the researchers in this field [19].

Additionally, activities of this kind can also make a contribution to the student's final grade in the course. Wang et al. used this approach in two different programming courses for freshmen using an online assessment system EduPCR developed directly for this purpose. As they presented in their report [18], they observed significant improvements of student learning in various aspects, such as student programming skills, collaborative learning competence, compliance with coding standards, time management capability and competence of giving and accepting criticism. Li [12] also used marks assigned to students by their peers during code review process in final grading, although he did not employ any supportive tool for code review. Similarly to the previous study he concluded that this activity motivated students to learn coding standards and facilitated the communication among them.

GitLab and GitHub are web-based source code management services usualy used in software development. Although they are starting to emerge as collaborative platforms for education [17, 21, 7, 8, 3], studies introducing educational use of these services are not focused on code reviewing. On the other hand, several studies can be found in the literature, presenting new tools developed to facilitate the employment of code review in educational settings [18, 14, 22]. Since these tools are usually not available, we used GitLab and our own LMS with basic peer review functionality in our courses.

3 Experiments

To evaluate the potential of code review in computer science university courses, we conducted two preliminary experiments. Each experiment was with a different course at a different study level, and also used a different platform to administer the code review process.

3.1 Experiment 1: Undergraduate Course

The first course was an optional undergraduate-level (Bachelor) course on NoSQL database systems. There were 10 students enrolled, who took part in a practical project assignment, which was individual. The task was to implement an application on top of a non-relational database. The actual choice of the application and the database was free. While the project assignment was compulsory to pass the course, the code-review part was not (however, students who did not participate in this part were penalized by one level of overall grading for the course). All 10 students submitted the project, but only 7 participated in reviewing.

The students used the Gitlab platform to create a git repository, and to share it with their colleagues for code-review and comments. We followed the methodology similar to those mentioned in our previous works [9, 6] where after the first initial deadline the students were assigned to review each other's work (2 reviewers per project, randomly

assigned). They were given 10 days for review and 7 additional days to process the feedback. Then they submitted a new version for teacher's evaluation. This makes the review process immediately useful for the students, and, in case of code reviews it also simulates the real world application of this type of reviewing to some extent.

The choice of the Gitlab platform had some influence on the review process: the reviewing was not blind in any sense – the reviewers knew the identity of the code author and vice versa. Also, the comments appeared in the system at the moment they were submitted and were immediately visible to the author but also to the other reviewers (even if they did not submit their comments yet).

The students were given predefined criteria on which to focus their reviews: (a) choice of database system for the problem, (b) suitable database model, (c) optimal usage of the database system and its API, (e) code structure, modularity, and quality, (f) code organization and comments.

Gitlab permits to add comments to every line of code (of the merge request created by the students during submission) and in addition to add an overall comment at the bottom. The students were instructed to comment the lines of code as much as possible when useful, and to write the rest to the overall comment.

3.2 Experiment 2: Master's course

The second course was an optional Master's course on Web programming. There were 34 students enrolled. In our experiment we focused on a project assignment which was one of the activities in the course. The goal of the project was to implement a web application – an educational game. Students had to choose the game topic from a list provided by the teacher.

The project assignment was not compulsory, and did not contribute to the evaluation. It was part of a set of alternate assignments students could undertake in order to qualify to the exam. In the end, 11 submissions were delivered, 7 from 2-member teams, and 4 by individuals.

Submissions, and code reviews administration were handled by our own LMS system. Students had 3 weeks to develop and submit the project. Consequently, each submission author (a team or an individual) was randomly assigned 3 other submissions for the code review. Since this code review activity was not evaluated by the teacher, it was entirely voluntary. Students were encouraged to participate to help their colleagues to better understand the shortcomings of their projects. In the end the authors of 6 submissions took part in it.

Our LMS was designed based on our own requirements and was built from-scratch. In addition to common usage as LMS, it has been used to administer peer review based activities so far. It does not offer all functionalities usually employed for code reviewing however, it is equipped with configurable peer-review forms. The reviewers downloaded the submissions assigned for review, examined the code and afterwards they filled in the reviewing form prepared by the teacher. The form consisted of 10 questions corresponding to the same criteria, used in evaluation by the teacher. The criteria focused especially on required features and capabilities, employment of required technology, difficulty and technical quality of the implementation, and code documentation.

The students rated each criterion by 1–5 points and also by verbal justification of their rating. The reviewing process was blind – each reviewer was aware of the project author, but the authors did not know who reviewed their project.

As mentioned above, the system we used was not originally developed for code reviewing, thus it misses some functionality useful for this activity. However, it features other supportive functions that are helpful when working in teams, e.g., team formation, random assignment of projects for code reviewing, or peer reviewing of the team-mates' contribution to the join work. Also, some of them were even familiar with the system as they used it before in other courses.

4 Results

4.1 Participation in the Activity

In the first experiment, 10 students delivered the submissions, and 7 of them submitted it in time in order to get reviewers assigned. Each was assigned 2 reviewers, and received on average 16.43 comments from both reviewers combined. The students also responded to these comments averaging 6.43 replies per students (while 3 students did not respond at all).

In the second experiment the assignment was a team one. Here, 11 teams delivered the submission. Each team was assigned 3 other assignments to review. Only 6 teams participated in the reviews, 5 of them filled in all three assigned reviews and the remaining team missed out on one review. There was no way how to reply, but it was possible rate the review. Only 5 reviews were rated altogether, i.e., on average only 0.84 reviews were rated out of the 3 assigned to each submission.

We can see that a lower percentage of students participated in the feedback phase in the second experiment. Partly this could be because of the lengthy structured review form they had to complete at once, but this observation is not highly conclusive, due to large differences in the settings of the two experiments and also between the student groups.

4.2 Students' View

After the experiment, we conducted a survey focused on students perception of code review activity. Students from both courses were asked to fill in a questionnaire consisted from 15–18 questions. The core of both questionnaires was the same, however some questions were different according to the scenario used in code review process in a particular course.

The aim of the survey was to get insight into the opinions and attitudes of the students both towards the code reviewing activity and towards the different tools that we tested. The questionnaire was delivered online, and 4 students from each course responded to it.

Several questions of the questionnaire concentrated on the students' perception of the usefulness of the activity: students were asked how they profited from comments received from their peers but also from commenting the program code of the others.

Students in both groups mostly appreciated that the received feedback helped them to detect shortcomings of their project they have not noticed before submitting (Fig. 1). When asked if reviewing the code submitted by others was somehow useful, they predominantly indicated the answers "It was useful to realize a variety of errors occurring in computer programs," and "It was useful to gain experience in code reviewing" (Fig. 2). They largely concluded that this skill will be useful in their future job (more so, the group that used Gitlab), as depicted in Fig. 3.

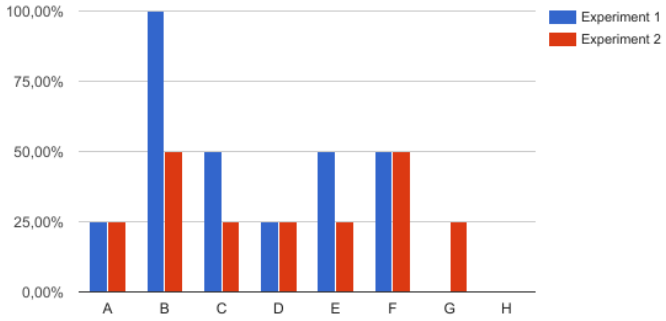

Fig. 1. What was the benefit of the comments you received from your colleagues? A – It helped me better understand the assignment of the project; B – It helped me to fix the shortcomings I did not notice before submitting; C – I learned how to review the code; D – It helped to improve my programming skills; E – I learned to accept constructive criticism; F – I learned to learn in cooperation with others; G – Nothing; H – Else.

We also tried to find out the students' opinion about what kind of feedback they received, as well as about the code review process in general. We asked if the feedback was more on the fundamental issues or minor improvements in programming and code standards. The students from Experiment 1 leaned slightly towards the latter, while those of Experiment 2 leaned more towards the former (Fig. 4).

Both groups agreed on that the received comments were mostly sufficiently specified. Regarding the form of feedback they would prefer to receive, the opinions of students differed according to the group they belong to. While the students of Experiment 1 stressed the importance of commenting particular lines of the program and giving the feedback also in the form of the corrected program code; the students of Experiment 2 indicated that it is enough to assess the project as a whole and it is sufficient to point out the defects. We observe that the preference of particular workflow matches the one which the students experienced in the experiment. This is quite natural to expect since they did not have a chance to try the other workflow. In the future it would be interesting to compare these two workflows more thoroughly.

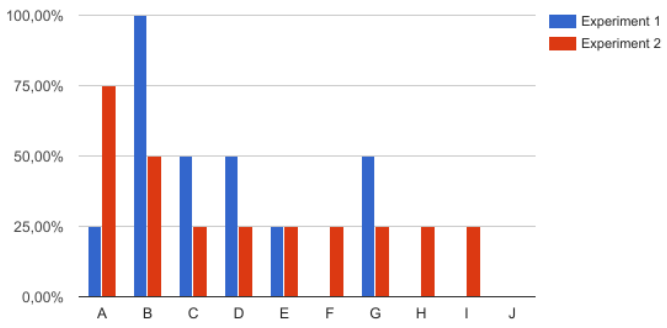

Fig. 2. What benefits you gained by commenting on colleagues' projects? A – I gained experience with code reviewing; B – I realized a variety of errors occuring in the program code; C – When looking at my colleagues' projects, I realized the shortcomings in my project; D – I improved my programming skills; E – I improved my communication skills; F – I learned to learn in cooperation with others; G – I learned to give constructive criticism; H – I earned extra points; I – I have never commented on projects; J – Else.

(a) Experiment 1. (b) Experiment 2.

Fig. 3. Do you think that the experience you gained in assessing the work of others will be useful for you in future programming practice? A – Certainly yes; B – Rather yes; C – Rather yes; D – Certainly no.

The commenting in Experiment 1 was non-anonymous (this is inherent to Gitlab and similar tools), while the reviewing in Experiment 2 was blind. Both groups were mostly neutral to this setting, it did not bother anybody. Both group expressed their preference towards randomly selected reviewers. The students in Experiment 1 were able to reply to the comments, which some found beneficial. As mentioned above, in Experiment 2 there was no such option.

When asked about their experience with the provided system that supported the activity, the Gitlab system in Experiment 1 was viewed as non-problematic, while our own

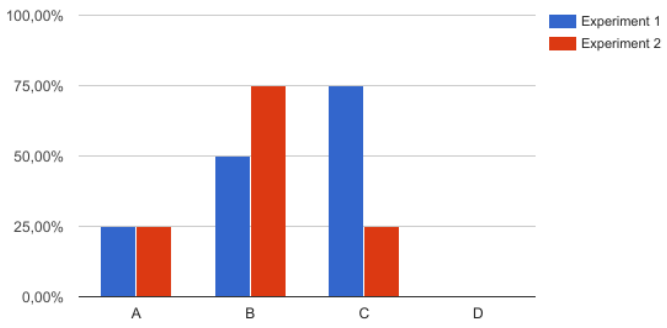

Fig. 4. What kind of feedback did you get? A – Fundamental to the proposal; B – Fundamental to the implementation; C – How to improve the code; D – I have not received any comments.

LMS system in Experiment 2 was reported to have some navigation issues (difficulty to find the reviews).

As for the evaluation of the whole code review activity, the students perceived it positively, even very positively in Experiment 1. This difference may be related to the different scenarios, e.g., the possibility to improve the project according to comments available only in Experiment 1, but also perhaps to the fact that the students in Experiment 1 were more satisfied with the provided tool.

4.3 Teacher's View

We also conducted an interview with the teachers responsible for both courses. The scope of the interview was broader, however we summarize the observations related to their perception of the activity and its usefulness.

In general, the teachers were aware of the code review feedback, and they also perceived its usefulness to certain degree. Their impression was that most of the feedback was on minor coding issues, however, there were notable exceptions. One such exception was when in Experiment 1, i.e., the database course, the students we able to communicate via comments the fact that the data model details, which were not submitted with the code of the submission is in fact needed for evaluation. Some of the students were even able to agree on passing additional files via comments which improved their reviewing experience as well as their feedback on the submission. This indicates that the discussion between the submission's author and the reviewers may be very useful.

Students also reflected on the other projects they reviewed during assessment. It happened that some of them felt the other projects were better then their own, they suggested they should be graded lower.

Another observation was on the commenting workflow inherent to Gitlab (Experiment 1), where the comments are immediately accessible to everyone as soon as they

are posted. This may cause problems when there were 2 or more reviewers assigned on a submission, and one of them overly active and quickly comments on most issues. The other reviewers may find that they have nothing to add. Such a problem does not appear in our own LMS, where the reviews are only visible once the deadline for reviewing expires.

5 Conclusions

In this study we tried two distinct supportive tools for implementing a code-review activity in university courses: our own LMS system with peer-review functionality, and Gitlab code management system. Regarding our aim (a) to test and evaluate both systems we conclude that, feature-wise, Gitlab and similar environments may offer a number of useful features for code review administration, and their workflow, which are not part of basic LMS system. But on the other hand these code management platforms lack other features which make their administration in the educational setting easier for the teachers: e.g., submissions overview, randomized reviewer assignment, postponed review delivery based on a deadline, anonymization, etc. Students working with Gitlab and similar systems also need some basic experience, and face some "industrial" overhead like working with git repository, creating merge requests, etc.

This experience also helped us with our aim (b) to better understand the requirements for delivering code-review activities in educational settings. While some of these requirements are met by Gitlab, some other important requirements related to anonymisation, and postponing review delivery until all students submit their comments cannot be achieved easily in such systems. This suggests the idea to extend our existing LMS system with advanced code-reviewing features such as commenting upon arbitrary line of code, etc., while keeping the new features in line with these requirements.

In regard to our aim (c) to evaluate students' and teachers' perception and acceptance of the activity, we may conclude that the activity was useful, and it was viewed predominantly positively by our small sample of surveyed students, and also by the teachers. This encourages us to extent our study in the future, to a larger, statistically significant sample.

Acknowledgment. The authors would like to thank to Roman Hrušecký, Ján Kľuka, Alexander Šimko, and Jozef Šiška – the lecturers who cooperated on this study.

References

1. Bacchelli, A., Bird, C.: Expectations, outcomes, and challenges of modern code review. In: Proceedings of the 2013 international conference on software engineering. pp. 712–721. IEEE Press (2013)
2. Boehm, B.W., et al.: Software engineering economics, vol. 197. Prentice-hall Englewood Cliffs (NJ) (1981)
3. Bonakdarian, E.: Pushing git & github in undergraduate computer science classes. Journal of Computing Sciences in Colleges 32(3), 119–125 (2017)

4. Da Cunha, A.D., Greathead, D.: Code review and personality: is performance linked to MBTI type? Tech. Rep. CS-TR-837, University of Newcastle upon Tyne (2004)
5. Da Cunha, A.D., Greathead, D.: Does personality matter?: an analysis of code-review ability. Communications of the ACM 50(5), 109–112 (2007)
6. Dropčová, V., Homola, M., Kubincová, Z.: Students' acceptance of peer review. In: International Conference on E-Learning, E-Education, and Online Training. pp. 52–59. Springer (2015)
7. Feliciano, J.: Towards a collaborative learning platform: The use of github in computer science and software engineering courses. Ph.D. thesis, University of Victoria (2015)
8. Feliciano, J., Storey, M.A., Zagalsky, A.: Student experiences using github in software engineering courses: a case study. In: Proceedings of the 38th International Conference on Software Engineering Companion. pp. 422–431. ACM (2016)
9. Homola, M., Kubincová, Z., Čulík, J., Trungel, T.: Peer review support in a virtual learning environment. In: State-of-the-Art and Future Directions of Smart Learning, pp. 351–355. Springer (2016)
10. Höst, M., Johansson, C.: Evaluation of code review methods through interviews and experimentation. Journal of Systems and Software 52(2), 113–120 (2000)
11. Hundhausen, C., Agrawal, A., Fairbrother, D., Trevisan, M.: Integrating pedagogical code reviews into a CS 1 course: An empirical study. ACM SIGCSE Bulletin 41(1), 291–295 (2009)
12. Li, X.: Incorporating a code review process into the assessment. In: 20th Annual Conf. of the National Advisory Committee on Computing Qualifications (NACCQ). pp. 125–131. Hamilton, New Zealand (2007)
13. Li, X., Prasad, C.: Effectively teaching coding standards in programming. In: Proceedings of the 6th conference on Information technology education. pp. 239–244. ACM (2005)
14. Tang, M.: Caesar: A social code review tool for programming education. Ph.D. thesis, Massachusetts Institute of Technology (2011)
15. Trytten, D.A.: A design for team peer code review. ACM SIGCSE Bulletin 37(1), 455–459 (2005)
16. Uwano, H., Nakamura, M., Monden, A., Matsumoto, K.i.: Analyzing individual performance of source code review using reviewers' eye movement. In: Proceedings of the 2006 symposium on Eye tracking research & applications. pp. 133–140. ACM (2006)
17. Villarrubia, A., Kim, H.: Building a community system to teach collaborative software development. In: Computer Science & Education (ICCSE), 2015 10th International Conference on. pp. 829–833. IEEE (2015)
18. Wang, Y., Li, H., Feng, Y., Jiang, Y., Liu, Y.: Assessment of programming language learning based on peer code review model: Implementation and experience report. Computers & Education 59(2), 412–422 (2012)
19. Wang, Y., Yijun, L., Collins, M., Liu, P.: Process improvement of peer code review and behavior analysis of its participants. ACM SIGCSE Bulletin 40(1), 107–111 (2008)
20. Wiegers, K.E.: Peer reviews in software: A practical guide. Addison-Wesley Boston (2002)
21. Zagalsky, A., Feliciano, J., Storey, M.A., Zhao, Y., Wang, W.: The emergence of github as a collaborative platform for education. In: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing. pp. 1906–1917. ACM (2015)
22. Zeller, A.: Making students read and review code. ACM SIGCSE Bulletin 32(3), 89–92 (2000)