Methods for Double-Blind Peer Review and Grade Prediction of Student Software Projects

Irfan Prazina and Vensada Okanović
* Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina
iprazina1@etf.unsa.ba, vokanovic@etf.unsa.ba

Abstract - In software engineering courses with many students, it is often impossible for a professor to give a detailed review of the software project for each student. One solution is to give students a task to make reviews for a few projects. One review consists of a comment on a project and a score. The score represents an ordinal number of the project when student sorts all projects in order by quality. The comment can be used as feedback to their colleagues and the score can be used in the final grade prediction. Double-blind peer review is used to prevent favoritism and unfairness. In the paper methods for random projects distribution, reviews gathering and analysis are presented. Methods have been used on over 100 projects and data gathered from them show moderate to high correlation with final grades. Project rank and project points have correlation factors from $r=-0.6$ to $r=-0.66$ and significance $p<0.001$. Projects have been independently graded with automatic tests.

Keywords - peer review; software projects analysis; software in education

I. INTRODUCTION

In the field of software engineering practice of code review is often used to improve code quality where other team members give feedback on the code to their colleagues. In the classroom environment often only a teacher is giving feedback to a project. When the number of students is too large that feedback is reduced to a few remarks. Importance of a code review is described in many papers [1], [2], [3] and [4].

A. Background

During one semester course on web technologies, students had one project divided into four iterations. Each iteration has multiple tasks. Iterations are two weeks long, and after an iteration students need to push their solution to a private repository. Private repositories are used so that students can not see each others solutions.

Each iteration is tested using an automatic test. Automatic tests are designed so that they detect most common errors but they often can not say why an error is present. In the paper [5] authors say that beginner students often find difficult to detect where in the code error appears when they get a failed test. They propose a solution which gives automatic feedback where is an error in the code and how to fix it, but it has limitations. The project description has to be very detailed and errors need to be predictable. We can not meet this requirement, because not all task can be given that way. Some of the tasks in the course have a higher degree of freedom of implementation. The course is a final year course and students need to demonstrate some independence in solving their tasks. If tasks are too strictly formulated then it is hard to write a task definition where a part of the solution or a method is not revealed.

There is also a problem with too strict automatic tests. If tests are too strict or coarse-grained then students who have a partially correct solution will get 0 points. Tests need to be fine-grained so that they can detect multiple errors that are present, otherwise if just one big error is reported students cannot learn very much from it.

Another solution to feedback is via peer evaluation. In this case, students who worked on tasks evaluate other solutions to the same tasks from their colleagues. In this case, the evaluation is more subjective than with automatic tests.

B. Literature overview

There are many papers who describe using automatic grading of programming assignments. In the paper [6] authors describe grading method where students write tests and code and grading system validates their testing performance. Testing performance is assessed in three ways: tests’ validity, tests’ completeness and code quality.

In the paper [7] authors propose a method where students review other students solutions by writing tests and running them. Reviewers get compiled programs not a code of the solution.

In the paper [8] authors give systematic literature review of assessment tools for programming assignments. They identify three types of approaches: instructor-centered, student-centered and hybrid approaches. In their research authors found that most of the tools support Java or C++ programming languages, they did not mention tools support for web-oriented languages (JavaScript, HTML and CSS).

There are also many papers who describe peer feedback and evaluation in teaching.

In the paper [9] authors propose a solution for peer feedback. This solution is used to rate discussion questions which assist students in their learning process.

In the paper [10] authors describe a few guidelines for the peer assessment task. Assessment groups should not be too large and assessors should not have many dimensions to assess.
Paper [11] concludes that peer evaluation should be used not only for students grading but also to improve the learning process and critical reflection.

By combining all the benefits of the code review, objective automatic tests, and peer feedback we tried to help students to recognize and correct their mistakes in code. For this purpose, we made one software solution and few methods which allow students to review each other work and rank their peers. This ranking showed moderate to high correlation with points gained by automatic tests.

In chapter II we described methods for a project assignment. In subchapter II.A structure of tasks is described. In subchapter II.B the software for the review process is presented. Analysis of the review results is also discussed. In the chapter III conclusion is given.

II. REVIEW METHODS AND SOFTWARE

A. Student’s Tasks

During a course on web technologies, students (N ~ 130) had one project divided into four iterations. All students had same project to implement. One iteration represents portion of the project students need to implement and it has its deadline. After each deadline, all students got 5 projects to evaluate and rank. They evaluate only current iteration.

Rules for evaluation, listed by importance, are iteration completeness, number of bugs and code quality. For each project, a student writes comment and rank projects by quality, where the first project is best and the last project is worst.

Students have a few days to analyze, evaluate and rank projects. After the deadline for a review ends reviews are analyzed and all students get comments from their colleagues and the average rank of their project.

Students keep their projects as private repositories on BitBucket. Only teacher and student have access to the repository. Project iterations are on individual branches. Teacher runs a node.js script which gets a list of all student’s repositories with correct branch using BitBucket web API.

After the list is composed teacher uses bash script which clones all repositories from the list. Each repository is then randomly assigned to five different students so that all get five different projects.

When projects assignment is done students get projects for review as a zip file on the branch of the current iteration (Figure 1). Readme files and any files which contain information about the identity of the student are removed from projects. This is done to reduce potential favoritism or conflicts in a peer evaluation process.

New software solution has been made for project review.

B. The Software for review

This software is made using opensource technologies. Electron.js [12] is used as desktop fronted solution written in javascript and Hapi.js [13] framework with PostgreSQL database for web API development (backend). The frontend is built for Windows, Linux and MacOS. The backend is deployed on a free virtual machine on Heroku cloud platform.

It is designed as a service-oriented solution. Communication between the desktop application and the backend is made using asynchronous HTTP requests.

Students use the desktop application to login to the system. After login, they can view all their reviews and when review deadline ends, all reviews from their colleagues for their project. The user interface is simple with only few relevant information is displayed.
Students rank five projects by quality from best to worst. Projects are named as letters A to E. For each project there is a text area for comment and there are five drop-down lists with project names representing the rank of a project (Figure 2).

Projects are distributed on BitBucket [15] repositories. In this way traffic to the backend is reduced and this system can be hosted on a free virtual machine instance on Heroku [16]. The number of tables in the database is also reduced to fit limits of free deployment. In the database, data about students, reviews, review results, project distribution and review deadlines are stored.

C. Review result analysis

When an iteration ends projects are tested by automatic tests. Tests detect errors in a project and the project is marked in a way that a project with no failed tests (no errors) gets all the points. If there are detected errors, percent of failed tests is deducted from the points. Each iteration is marked by 10 points where 1 point students can get for the review. Half of the point they can get for their review and half of the point is calculated using average rank their project is ranked.

The average rank of a project is calculated as the average of all ranks project got in the current iteration. One project is ranked by five different students. The average rank and final grade are then compared for each iteration. A moderate to strong negative correlation is observed ($r \sim -0.6$). Pearson correlation factors for all iterations are given in Table 1. Significance for all correlations is $p<0.001$.

| TABLE I. PEARSON CORRELATION BETWEEN RANK AND FINAL POINTS ($p=0.001$) |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| Iteration 1     | Iteration 2     | Iteration 3     | Iteration 4     |
| $r$             | -0.603          | -0.623          | -0.597          | -0.657          |

These results show that the rank can be used as a predictor of the final grade. In Figure 3 graphs of ranks and final points are presented.

Students error in ranking projects is also observed and no correlation between students points and error could be found. Error is measured using the largest difference between ranks and average positions, this value is then multiplied by the variance of average positions of all projects student reviewed in the current iteration. The variance of average positions is used so that error for students who get projects of different quality is larger than the error of students who get a very similar projects.

This error is compared to the points students get on the iteration. The observed correlation is low but this result does not have high statistical significance, correlation factor $r=0.07$, and $p=0.108$.

III. CONCLUSION

Evaluation of student’s projects can take a lot of time, especially if classes have a large number of students. Feedback from evaluation needs to be adequate to support the learning process. Peer evaluation can be used in software engineering courses complementary with an automatic test to give feedback to the students. More research needs to be done to evaluate does peer feedback help students to understand why and how their programs failed on automatic tests.

Further research needs to be done which will analyze the effects of peer evaluation in software engineering courses on code quality. A detailed analysis could be done on how students view peer feedback as a complementary tool to automatic tests.

REFERENCES


