Method for Social Programming and Code Review

Michal TOMLEIN*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
michal.tomlein@gmail.com

Code quality in software projects can generally be achieved in a number of different ways. In order to create high quality, reliable and reusable code, incorporating code review into the development process is among the most effective of the available options. The use of code review is, however, not limited to the domain of software development. In programming courses, peer review has the potential to be an effective driving force behind the learning process.

However, due to the significant amount of time reviews take, whether in software development or a programming course, they cannot be and are not done thoroughly in practice. Specifically in programming courses, reviewing code requires the selection of suitable reviewers, as pedagogues and graduate students are often not available in sufficient numbers to be able to help all of the students.

For this and other reasons, peer reviews are used to complement normal reviews. In many cases, peer reviews have been found to be adequate substitutes for pedagogical reviews. Peer reviews do, however, present a few problems in their realisation. Students' ability to review code and provide useful feedback varies to the degree that it becomes very important to select suitable reviewers for the individual problems the students have.

Reviewer selection is a non-trivial problem. The relevance and quality of the resulting review is highly dependent on reviewer selection, as different reviewers may have a different amount of experience with the particular problem they are assigned to review. Research has shown that reviewers with insufficient knowledge and experience in the problem area only contribute to user confusion and do not provide the necessary motivation [1]. Because of this, there is a need for a solution, which takes such reviewer attributes, as well as the reviewers' availability, into consideration when selecting the most suitable reviewer for a given problem.

Currently, there are a number of solutions aimed at reviewing code in the form of commits in a version control system. These solutions offer a varying degree of

* Supervisor: Jozef Tvarožek, Institute of Informatics and Software Engineering

Spring 2012 PeWe Workshop, April 1, 2012, pp. 47–48.

automation of this process, ranging from simple solutions to systems such as Gerrit, which can interoperate with continuous integration systems such as Jenkins. Some solutions, such as CodeCollaborator, offer integration with development environments such as Eclipse or Visual Studio. The use of these solutions is, by their nature, generally limited to larger projects and/or requires a certain discipline to be effective.

Previous research activity in the field of code review has also focused on code review as part of programming education. For example, the Caesar system was designed with the students' lack of experience with version control in mind, which is a major obstacle in the use of common code review solutions in programming courses [2]. Caesar attempts to overcome this problem through the division of code into smaller partitions, as opposed to other systems based on incremental code review.

Reviewer selection is generally done manually or using systems with defined rules, priorities and responsibilities, which divide work evenly among the available reviewers. Social approaches have so far not been applied to reviewer selection.

We propose a solution, which integrates social approaches and collaboration into the process of code review. The social approaches are based on user and reviewer modelling. We intend to model the experience and knowledge of reviewers primarily using review assessment by users. This explicit feedback will be aggregated in problem areas. The resulting model will be used to select reviewers for each new case. The user model can also be used in reviewer selection to match reviewers who have received positive feedback from users with a similar level of domain knowledge.

To evaluate the proposed solution, we intend to create a prototype of an extension for an integrated development environment used in programming education. This extension will serve to enable students to request reviews of their code and ask for help. In addition to this, a prototype of a web application will be created for reviewers to get access to pending requests for review. This application will provide the tools necessary to review the submitted code and provide feedback and help to the users.

In our work, we aim to make programming more effective through social interaction and peer reviews. We believe it is very important to reduce friction in the process, from asking for a review to getting feedback and communicating with a reviewer. To do so, we intend to integrate our solution with widely used development environments. We believe that by making it possible for students and software developers to collaborate more tightly and easily, we can speed up the development process and achieve higher quality overall.

Acknowledgement. This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant No. VG1/0971/11.

References

[1] Hundhausen, Ch.D., Agarwal, P., and Trevisan, M. (2011): *Online vs. face-to-face pedagogical code reviews: an empirical comparison.* Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE'11), ACM, New York, NY, USA, pp. 117 – 122.

[2] Tang M. (2011): *Caesar: A Social Code Review Tool for Programming Education.* Master Thesis at Massachusetts Institute of Technology.