Project proposals

Vladimir Kovalenko
Delft University of Technology
The Netherlands
v.v.kovalenko@tudelft.nl

January 30, 2019

Code Reviewer Recommendation

Code review is a software development practice consisting in manual peer review of code changes. The benefits of the process include higher code quality, increased awareness of technical changes among team members, and smoother onboarding of newcomers. [1]

These days code reviews in industry are performed with dedicated tools, such as Gerrit, Phabricator, Crucible, Upsource, and GitHub. These tools feature interfaces to view and discuss changes, and offer integration with external instruments, such as issue trackers and static analysis tools.

A prominent direction in evolution of code review tools is utilization of records of development history for improvement of user experience. Most of innovation in this area originates in academic research – techniques such as defect prediction [9] are motivated partly by the need to improve the efficiency of code review.

One of such techniques is *reviewer recommendation.* Over the last 5-6 years, this problem had formed an established area in the field of software engineering research. The general idea of reviewer recommendation is to use records of prior code changes and reviews to recommend optimal reviewers for a given change. Exact mechanics vary among implementations. [11, 7, 2, 8, 4, 10] Over the last 3 years, reviewer recommendation algorithms have been introduced in industrial code review tools, such as GitHub, Gerrit and Upsource.

The two projects described below aim at improving the state of the art in reviewer recommendation. A successful study on each of the proposed problems would result in a submission to a major software engineering research venue.

Candidates are expected to have decent programming skills, a good command of English, and basic knowledge of statistics. Experience in academic or technical writing is not necessary but appreciated.

Project 1: dependency information for reviewer recommendation

Information about code dependencies is an important point that developers consider during selection of code reviewers. [5] However, no existing algorithms for reviewer recommendation make use of this information.

The goal of this project is to investigate whether dependency information is a valuable source of input data for reviewer recommendation algorithms. We will use one or several of the state-of-the-art reviewer recommendation algorithms as a baseline, and evaluate the effect of adding dependency information on recommendation accuracy.
Preliminary research questions:

- Does additional information about code dependencies improve accuracy of reviewer recommendations?
- What is the best way to aggregate dependency information for usage in reviewer recommendation algorithms?

Project 2: optimizing RR for metrics beyond accuracy

Existing reviewer recommendation algorithms are technically evaluated as prediction models: the target metric is similarity of algorithms' output to actual historical records of reviewers. While such evaluation is a common technique in the broader field of recommender systems, focus on accuracy poses a potential threat to maintainbility of the underlying codebase: due to an effect similar to the filter bubble [6], existing reviewer recommendation approaches endorse deep division of code ownership among team members, which has a negative effect on maintainability of code (see *bus factor* [3]).

The goal of this project is to build a reviewer recommendation system that would promote collective ownership of code in teams, rather than "predict" reviewers. We will evaluate the system through modeling the reviewer selection process and estimating its potential impact on choices of reviewers made by users of the recommender system.
Preliminary research questions:

- Is it possible to endorse collective code ownership via reviewer recommendations?
- How closely should the users follow the recommendations for such system to function in the long run?

References

[1] Bacchelli, A., and Bird, C. Expectations, outcomes, and challenges of modern code review. In *Proceedings of the 2013 international conference on software engineering* (2013), IEEE Press, pp. 712–721.

[2] Balachandran, V. Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. In *Proceedings of the 2013 International Conference on Software Engineering* (2013), IEEE Press, pp. 931–940.

[3] Cosentino, V., Izquierdo, J. L. C., and Cabot, J. Assessing the bus factor of git repositories. In *Software Analysis, Evolution and Reengineering (SANER), 2015 IEEE 22nd International Conference on* (2015), IEEE, pp. 499–503.

[4] Jeong, G., Kim, S., Zimmermann, T., and Yi, K. Improving code review by predicting reviewers and acceptance of patches. *Research on software analysis for error-free computing center Tech-Memo (ROSAEC MEMO 2009-006)* (2009), 1–18.

[5] Kovalenko, V., Tintarev, N., Pasynkov, E., Bird, C., and Bacchelli, A. Does reviewer recommendation help developers? *IEEE Transactions on Software Engineering* (2018).

[6] Nguyen, T. T., Hui, P.-M., Harper, F. M., Terveen, L., and Konstan, J. A. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web* (2014), ACM, pp. 677–686.

[7] Ouni, A., Kula, R. G., and Inoue, K. Search-based peer reviewers recommendation in modern code review. In *Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on* (2016), IEEE, pp. 367–377.

[8] Thongtanunam, P., Kula, R. G., Cruz, A. E. C., Yoshida, N., and Iida, H. Improving code review effectiveness through reviewer recommendations. In *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering* (2014), ACM, pp. 119–122.

[9] Wahono, R. S. A systematic literature review of software defect prediction: research trends, datasets, methods and frameworks. *Journal of Software Engineering 1*, 1 (2015), 1–16.

[10] Yu, Y., Wang, H., Yin, G., and Wang, T. Reviewer recommendation for pull-requests in github: What can we learn from code review and bug assignment? *Information and Software Technology 74* (2016), 204–218.

[11] Zanjani, M. B., Kagdi, H., and Bird, C. Automatically recommending peer reviewers in modern code review. *IEEE Transactions on Software Engineering 42*, 6 (2016), 530–543.