Project: Predicting refactoring effects using machine learning

Martin Steinhauer

March 23, 2020

1 Overview

Lehman's laws for software evolution [7] described already in the 1970s, that software systems are always changing. These changes are often caused by new features and requirements. As a consequence, the size and complexity is increasing and it becomes more and more difficult for developers to keep track of all changes that happened within the source code and to sustain a high software quality. A common way to deal with changing requirements and to improve the software quality is by doing refactoring. These can be composed as a set of one or many operations that are applied to the source code. A well defined list methods is described by Fowler [5] which are used in the daily business of a software developer. But with increasing size and complexity it is getting harder to estimate the quality of results and to avoid unwanted side effects that may cause a lower software quality than intended. Therefore, it would be great to know the quality and impact of the changes before the developer carries out the often time-consuming refactoring process.

2 Related work

The most important paper [3] builds the basis for this research and tries to predict the software quality after an executed refactoring method. The developed tool is called RIPE and can predict the quality impact by measuring 12 different software metrics by 11 applied refactoring methods. They developed multiple (static) predicting functions and evaluated them on manually and automatically selected refactoring. Since the approach is based on static functions, the prediction results are good for manually introduced refactoring but are bad when applied to real world commits fetched from open source projects.

Another paper [6] tries to predict refactoring only on method level and uses five open source Java projects, but is more focused on the evaluation and comparison of machine learning models. This paper may also give a good overview of selected source code metrics and also provides an overview of related work papers [2, 4, 8] which are also a good orientation for conducting this research project.

3 Goals and constraints

The goal of this project is to predict the effects of a refactoring before it is done to support the developer to decide which refactoring method is the most suitable for the given code. Unlike Chaparro et al., this projects aims to find the prediction values not based on a static mathematical functions but instead by using a machine learning based approach based on the selection of multiple open source projects. Otherwise than Chaparro et al., we focus on mining the history data of existing code refactoring in a version control systems. In general, the following questions should be answered:

1. Which features of a commit-refactoring combination are suitable to predict the effect of the refactoring method?
2. How well does the trained machine learning model perform?

To answer these questions and to provide a precise prediction, some preliminary work and steps are required which are presented in the following section. Some constraints are determined beforehand: To keep the complexity of the analysis low, only repositories which contain Java code are considered as a data source for the dataset.

4 Methodology

The first step of this project is the generation and mining of valid cases based on open source projects. So a careful selection of capable open source projects which can be found on GitHub is important. The projects must provide a source code base which is big enough and also have an adequate amount of refactoring operations that can be identified. According to Chaparro et al., a good way to reduce noise and only identify operations that are also done within the same commit but are not part of the actual refactoring. The operations can be found by using the Markos Project tool [1]. Another challenge is to setup the tool and run an initial analysis because the tool is not up to date and may cause problems due to it's complex setup process.

After that, a suitable dataset has to be generated which may contain several code metrics assigned to each refactoring operation. A refactoring operation must consist of the metric values before and after the actual refactoring is done. Also a suitable set of software metrics has to be selected, a good start may be the proposed metrics in Chaparro et al., but the values have to normalized. That is because different metrics have a different range and scale, and some metrics are considered to be optimal when they have a low value and others when they have a high value.

The next step requires some data analysis of the previously generated dataset. All collected features that were extracted from the Git history have to evaluated if they are a good indicator for prediction. For example, since there are multiple variables, no correlating variables should be selected as an input for the machine learning model. The relation between the variables has to be discovered within this research. After selecting a suitable set of variables, a good machine learning model has to be selected. Since the generated dataset will provide several metric results before and after the refactoring, an initial guess would be to use some kind of multivariate regression to predict the resulting software quality after the commit.

Finally, the machine learning model has to be trained by splitting the repository dataset in a training and a testing dataset to evaluate the performance of the model.

References

[1] Markos project. https://sourceforge.net/projects/markosproject/. Accessed: 2020-03-23.

[2] An experimental investigation on the innate relationship between quality and refactoring. *J. Syst. Softw.*, 107(C):1–14, September 2015.

[3] O. Chaparro, G. Bavota, A. Marcus, and M. D. Penta. On the impact of refactoring operations on code quality metrics. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 456–460, Sep. 2014.

[4] Alexander Chávez, Isabella Ferreira, Eduardo Fernandes, Diego Cedrim, and Alessandro Garcia. How does refactoring affect internal quality attributes? a multi-project study. In *Proceedings of the 31st Brazilian Symposium on Software Engineering*, SBES'17, page 74–83, New York, NY, USA, 2017. Association for Computing Machinery.

[5] Martin Fowler. *Refactoring: improving the design of existing code.* Addison-Wesley Professional, 2018.

[6] Lov Kumar, Shashank Mouli Satapathy, and Lalita Bhanu Murthy. Method level refactoring prediction on five open source java projects using machine learning techniques. In *Proceedings of the 12th Innovations on Software Engineering Conference (Formerly Known as India Software Engineering Conference)*, ISEC'19, New York, NY, USA, 2019. Association for Computing Machinery.

[7] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski. Metrics and laws of software evolution-the nineties view. In *Proceedings Fourth International Software Metrics Symposium*, pages 20–32, Nov 1997.

[8] Jacek Ratzinger, Thomas Sigmund, Peter Vorburger, and Harald Gall. Mining software evolution to predict refactoring. pages 354–363, 10 2007.