An Automatic Way to Calculate Software Process Metrics of GitHub Projects With the Aid of Graph Databases

Péter Gyimesi

Bug prediction is a popular research area nowadays. There are many great studies about how to characterize software bugs and how to recognize bug prone code parts. The characterization [1] of such bugs can be done with classic product metrics, with software process metrics or with some metrics of a different nature like textual similarity. Product metrics are extracted from the structure of the source code, for example: lines of code, cyclomatic complexity, etc. Software process metrics are computed from developer activities. The most common ones are based on the number of previous modifications, number of different contributors, number of modified lines and the time of the modifications.

All of these metrics can be computed on different granularity levels (file, class, method). Product metrics are mainly calculated on file and class level, but with the continuously improving technology the method level is becoming more frequent. There are many great tools – some of them are free – that can produce these metrics for projects of different programming languages. Previous researches [2] have shown that software process metrics are generally better bug predictors than product metrics, however, tools that can compute these metrics are not widespread. It can be due to the difficulties of storing and processing of the historical information. Another problem with process metrics is that to compute them on class or method level every version of the source code has to be analyzed and the modified methods and classes have to be extracted, which is a challenging task.

In the last few years, the popularity of graph databases increased due to the improving technologies behind them. The historical data of software can also be represented as a graph, so studies [3] started to examine the use of these graph databases in software quality too, especially in the calculation of software process metrics.

Our goal is to present an automated way of calculating software process metrics with the use of graph databases. We chose GitHub as data source, because it has an open API to access the history of many open-source Java projects. For the database we selected Neo4j and to analyze the source code and extract the source code elements we used the SourceMeter static analyzer tool.

For this research, we collected the process metrics from the literature and we reproduced the calculation of these metrics in cypher query language. With this language process metrics are easy to formulate and these queries can be processed by Neo4j. For a given GitHub project version it can compute the process metrics on file, class and method level. The list of these metrics is easy to extend due to the modularity of the created framework.

With this tool we analyzed almost 18 000 versions of 5 Java projects from GitHub. These projects have a total of 28 release versions that are selected with six-months-long intervals. We computed the process metrics of these projects and we built a bug database with the use of our previous researches. For each release version it contains the source code element at file, class and method level with the corresponding process metrics and bug numbers.

References

[1] Radjenović, Danijel, et al. "Software fault prediction metrics: A systematic literature review.", *Information and Software Technology* 55.8 (2013): 1397-1418.

[2] Rahman, Foyzur, and Premkumar Devanbu. "How, and why, process metrics are better.", *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.

[3] Bhattacharya, Pamela, et al. "Graph-based analysis and prediction for software evolution.", *Proceedings of the 34th International Conference on Software Engineering*. IEEE Press, 2012.