A Longitudinal Study of Static Analysis Warning Evolution and the Effects of PMD on Software Quality in Apache Open Source Projects (Summary)

Alexander Trautsch[1], Steffen Herbold[2], Jens Grabowski[3]

Abstract: This article summarizes our work originally published in the journal Empirical Software Engineering [THG20].

Keywords: Static code analysis; Quality evolution; Software metrics; Software quality

Software engineering best practices have included the use of static analysis tools for years. These tools can help developers spot common coding mistakes and maintainability problems. Static analysis tools work by analyzing source code or byte code and perform pattern matching to find problematic lines of code. While they are seen by developers as quality improving there are also problems with false positives.

While some studies are investigating static analysis tools, none were focused on the evolution of warnings over the complete development history for a general purpose static analysis tool. In our study we use PMD as the static analysis tool. It contains a broad set of warnings, works directly on source code and has been under development for many years. Therefore, it is able to provide us with a comprehensive history of static analysis warnings in our study subjects.

We investigate 54 open source Java projects under the umbrella of the Apache Software Foundation. We collect up to 17 years of development history of our study subjects and plot the evolution and trends of static analysis warnings. Overall, we collect static analysis warnings and the number of logical lines of code for 112,266 commits of our study subjects. We also collect all reported bugs and complete build information including information from Maven Central for all study subjects. This data collection is facilitated by SmartSHARK [Tr17, Tr20] and a local HPC system.

As we do not want to rely on a heuristic to find removed warnings we include all warnings in every commit and plot the warning density, i.e., the sum of all warnings divided by the number of logical lines of code. To further restrict noise we only investigate production

[1] Georg-August-Universität Göttingen, Institut für Informatik, Goldschmidtstrasse 7, 37077 Göttingen, Deutschland alexander.trautsch@cs.uni-goettingen.de
[2] Karlsruher Institut für Technologie, AIFB, Kaiserstr. 89, 76133 Karlsruhe, Deutschland steffen.herbold@kit.edu
[3] Georg-August-Universität Göttingen, Institut für Informatik, Goldschmidtstrasse 7, 37077 Göttingen, Deutschland grabowski@cs.uni-goettingen.de

doi:10.18420/SE2021_41

code, excluding tests, documentation and example code. As we are interested in trends we restrict the commit graph of our study subjects to a single path to remove noise due to release branches.

Our study explores two main research questions. How are static analysis warnings evolving over time and what is the impact of using PMD. We want to know if "code gets better", i.e., are static analysis warnings removed over the observed development history. We find that while the sum of warnings usually increases the warning density decreases in most projects. The types of warnings that drive this positive trend are mostly related to coding best practices, e.g., naming, brace and design warnings. On average, every study subject removes 3.5 warnings per 1000 logical lines of code per year.

Using PMD as indicated in the build process of the study subjects has a positive impact on the number of warnings, however this is not the case in all of our study subjects. We find that the use of a specialized configuration of rules for PMD has a negligible impact on the removal of warnings. If we calculate trends of warning density we find that the instances where PMD was used are not statistically significantly better than those without PMD. However, if we use the sum of static analysis warnings there is a statistically significant, albeit small difference.

When we use defect density as a proxy metric for external software quality we find that years in which PMD is part of the build process perform slightly better. However, this part of the study is limited by the available data and will be investigated in more detail in a follow-up study.

Bibliography

[THG20] Trautsch, Alexander; Herbold, Steffen; Grabowski, Jens: A Longitudinal Study of Static Analysis Warning Evolution and the Effects of PMD on Software Quality in Apache Open Source Projects. Empirical Software Engineering, 2020.

[Tr17] Trautsch, Fabian; Herbold, Steffen; Makedonski, Philip; Grabowski, Jens: Addressing problems with replicability and validity of repository mining studies through a smart data platform. Empirical Software Engineering, August 2017.

[Tr20] Trautsch, Alexander; Trautsch, Fabian; Herbold, Steffen; Ledel, Benjamin; Grabowski, Jens: The SmartSHARK Ecosystem for Software Repository Mining. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings. ICSE '20, Association for Computing Machinery, New York, NY, USA, p. 25–28, 2020.