The Smell of Stars – A small study on the correlation between GitHub popularity and code quality.

GitHub is an online platform for the management of source code and provides collaborative tools for joint development. As a result, it has become the central management and storage location for a large number of open source projects. A key element of the social platform is the Star function, which allows users to favorite open projects.

A study by Hudson Borges and Marco Tulio Valente[1] suggests that the awarding of stars has a distinct social media character. The star function is used in particular to bookmark interesting projects or to express appreciation. It was also statistically determined that the number of stars correlates moderately with an increased number of contributors.

This favors the plausible assumption that a high number of stars can be an indicator of high-quality software solutions. This small study therefore aims to systematically investigate a possible correlation between the popularity of open source projects – measured by the number of stars – and the quality of the source code. Perhaps the German proverb Too many cooks spoil the broth applies after all.

Study Design

Software quality can be examined on a variety of dimensions. Two of these that are easy to measure are the number of *bad smells* and the *cognitive complexity*.[2] Bad smells indicate problematic code passages that can indicate structural defects or high-maintenance areas. Cognitive complexity, on the other hand, measures how challenging it is for developers to understand and maintain the code based on the complexity of control structures and logical branches. Both metrics can be acquired by static code analysis using the Community Edition of the SonarQube software tool.[3] Since both values increase with the size of projects, these can be further normalized based on the *non-comment lines of code* (NCLOC). This allows the comparison of projects of different sizes.

In order to prevent the results from being influenced by the choice of programming language, repositories written in Ruby, PHP, JavaScript (JS) and Python were examined. Because these scripting languages do not require a compilation step, they are much easier to utilize SonarQube. Although 960 repositories were originally planned, the automatic analysis of particularly large repositories led to timeouts and problems with limited memory, so that the final sample comprises 899 repositories. A more detailed overview of the crawled repositories can be found in Table 1. These projects were crawled using the GitHub API,[4] where the selection was not random but based on the best match pattern. This approach selected the best matching repositories for the applied search parameters.

[1] Hudson Borges and Marco Tulio Valente. "What's in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform". In: *arXiv preprint arXiv:1811.07643v1* (2018). Preprint submitted to Elsevier, November 20, 2018. URL: https://arxiv.org/pdf/1811.07643.

[2] Justus Bogner and Manuel Merkel. *To Type or Not to Type? A Systematic Comparison of the Software Quality of JavaScript and TypeScript Applications on GitHub*. 2022. arXiv: 2203.11115 [cs.SE]. URL: https://arxiv.org/pdf/2203.11115v1.pdf.

[3] *SonarQube*. https://www.sonarsource.com/products/sonarqube/. 2025.

[4] *GitHub API Documentation*. https://docs.github.com/en. 2025.

Category	**Stars**	**JS**	**PHP**	**Python**	**Ruby**	**Total**
Popular	$40 < x < 400$	110	110	108	114	457
Usual	$4000 < x$	116	112	113	116	442

Table 1: Overview of the automatically crawled projects.

In accordance with the research question, it was further necessary to decide which repositories should be categorized as popular or usual. In this study, repositories with more than 4000 stars were categorized as popular, while repositories with less than 400 stars were considered to be common. Early analyses revealed that without a lower limit, several empty repositories were included in the dataset. A minimum number of 40 stars was introduced as an additional criterion to prevent this. The disjoint groupings combined with the normalized results allow the application of the Mann-Whitney U-test and Cohans-D to statistically evaluate the differences between the crawled probesets.

The initial assumption of this study is that popular repositories indeed have a higher code quality compared to ordinary repositories. Considering the previous categorization, the following null and alternative hypotheses can be formulated:

Null Hypotheses	**Alternative Hypotheses**
$\mathbf{A_0}$: There is no significant difference in the number of code smells between popular repositories and usual repositories.	$\mathbf{A_1}$: Popular repositories tend to exhibit fewer code smells compared to usual repositories.
$\mathbf{B_0}$: There is no significant difference in the cognitive complexity between popular repositories and usual repositories.	$\mathbf{B_1}$: Popular repositories tend to have a lower cognitive complexity compared to usual repositories.

The assessments in this study were carried out using two Python scripts, one responsible for crawling and the other for analysis. Both scripts are available in the GitHub repository associated with this report at https://github.com/jnloos/The-Smell-of-Stars. In addition to the required Python libraries, a locally accessible instance of the SonarQube Community Server (e.g. as a Docker container) is required to execute the code. The SonarQube CLI must also be installed and stored in the path variables.

Evaluation

- Grundsätzlich kein signifikanter Unterschied an Smells und Komplexität
- Von den Sprachen abhängig? Python und PHP vs. Ruby und JS

Conclusion