Does Functional Programming Improve Software Quality? An Empirical Analysis of Open Source Projects on GitHub

Bachelor Thesis

# Organization

|  |  |
| --- | --- |
| Examiner: | Prof. Dr. Stefan Wagner |
| Supervisors: | Dr. Justus Bogner |
| Student: |  |
| Timeframe: | 2020-11-07 – 2021-05-07 |

# Context & Motivation

In 1989, John Hughes – who would later get involved with the Haskell programming language [1] – wrote an influential paper on “Why Functional Programming Matters” [2], claiming that it would substantially improve software modularity with higher-order functions and lazy evaluation. Today, over 30 years later, several functional languages like Clojure [3], Scala [4], or Elixir [5] are indeed rising in popularity. However, there is still a lack of sufficient empirical evidence for the claim that using functional programming has significant impact on software quality in comparison to other paradigms like procedural or object-oriented programming [6], especially since many other language prejudices could not be supported by studies in the past [7]. While Ray et al. [8] found some indication that functional languages may be less prone to defects than procedural or scripting languages, more research with an explicit focus on functional programming is needed to draw conclusions.

# Objectives

The goal of this study is therefore to empirically analyze projects using functional programming languages and to compare them to projects with non-functional languages. The comparison should provide insights into a potential influence of the functional programming paradigm on software qualities like functional correctness or maintainability. The concrete quality aspects to be analyzed as well as more detailed research questions should be defined by the student.

# Methods

The research will be conducted as a mining software repository (MSR) study [9][10] using a large number of open-source projects on GitHub [11]. Data collection should fully rely on tool support (e.g. static analysis tools) and automation to increase reproducibility. For the analysis, suitable techniques could be hypothesis testing, correlation, or regression. The detailed study design should be created by the student.

# References

[1] <https://www.haskell.org>

[2] J. Hughes, “Why Functional Programming Matters,” Comput. J., vol. 32, no. 2, pp. 98–107, Feb. 1989.

[3] <https://clojure.org>

[4] <https://scala-lang.org>

[5] <https://elixir-lang.org>

[6] I. Figueroa and R. Robbes, “Is functional programming better for modularity?,” in Proceedings of the 6th Workshop on Evaluation and Usability of Programming Languages and Tools - PLATEAU 2015, 2015, pp. 49–52.

[7] T. Roehm, D. Veihelmann, S. Wagner, and E. Juergens, “Evaluating Maintainability Prejudices with a Large-Scale Study of Open-Source Projects,” in Software Quality: The Complexity and Challenges of Software Engineering and Software Quality in the Cloud. SWQD 2019. Lecture Notes in Business Information Processing, vol 338, Springer, Cham, 2019, pp. 151–171.

[8] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, “A large scale study of programming languages and code quality in github,” in Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014, 2014, pp. 155–165.

[9] A. E. Hassan, “The road ahead for Mining Software Repositories,” in 2008 Frontiers of Software Maintenance, 2008, pp. 48–57.

[10] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, “The promises and perils of mining GitHub,” in Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014, 2014, pp. 92–101.

[11] <https://github.com>