Evaluating Haskell Metrics

Looking for correlations between bug occurrences and code metrics

1. Introduction

- Haskell's Lifecycle Gap:
  - Lacks understanding compared to Java.
  - Few studies on Haskell's lifecycle.
- Prior Research:
  - Ryder and Thompson's paper: limited case studies.
  - Lack of type-based metric exploration.
- Key Questions:
  - Metric Trends: Typical changes in mean metric values across lifecycles.
  - Bug Correlation: Identify metrics correlating with bug rates.

2. Objective

Find correlations between metrics and bug occurrences in Haskell projects

3. Methodology

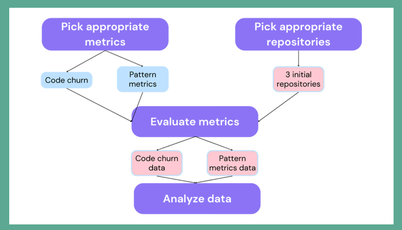
The metrics:

- Code churn - measures the amount of code modified over a small period, often used to assess the stability and maintainability of software projects
- PSIZ - quantifies the number of pattern cases used in a function that employs pattern-matching

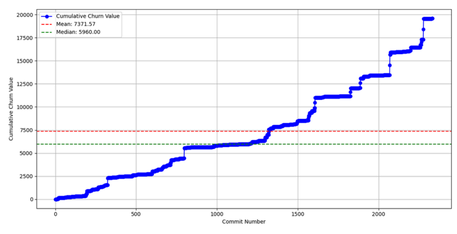
The repositories:

Repository Name	Number of Commits	Github Stars
quickcheck	1174	700
hackage-server	2350	410
lens	4332	2002

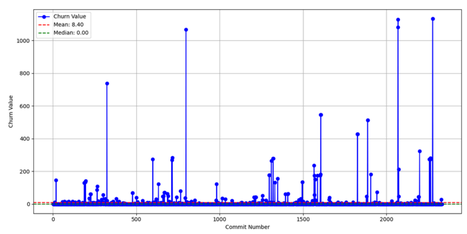
The plan:



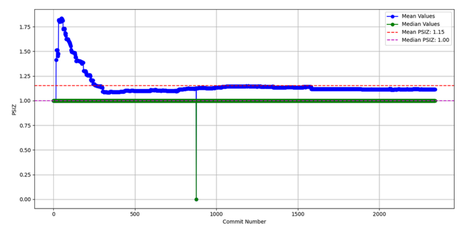
4. Results



hackage-server cumulative code churn



hackage-server non-cumulative code churn



hackage-server PSIZ values

Project	Mean	Median	File	File's Mean	File's Median
quickcheck	2.76	0	Property	19.38	1
			Gen	16.50	0
			Arbitrary	6.62	0
hackage-server	8.40	0	Html	18.18	1
			UserDetails	101.33	61
lens	15.68	0	FieldTH	60.29	4.50
			At	96.06	6.50
			TH	75.10	0
			Cons	175.83	28.50

Mean and Median Code Churn Values for Projects and Buggy Files

Project	Mean	Median	File	File's Mean	File's Median
quickcheck	1.36	1	Property	0.64	1
			Gen	1.23	1
			Arbitrary	1.00	1
hackage-server	1.15	1	Html	0	0
			UserDetails	1	1
lens	1.13	1	FieldTH	1.10	1
			At	0.69	1
			TH	0.44	0
			Cons	0.89	1

Mean and Median PSIZ Values for Projects and Buggy Files

5. Discussion

Code Churn Analysis:

- Buggy files have significantly higher mean code churn values compared to the overall project means, indicating frequent changes in files with documented bugs.
- Median code churn for repositories is 0, suggesting most commits do not introduce churn, while buggy files have a higher median, implying frequent changes.
- Larger projects with more commits tend to have higher average code churn.

Pattern Size (PSIZ) Analysis:

- PSIZ metric results are not as indicative of bug occurrences as code churn.
- Both project and buggy files have similar median PSIZ values, mostly around 1, indicating limited pattern complexity in functions.
- Differences in PSIZ means between buggy files and projects are not significant enough to draw reliable conclusions about bug correlation.

Takeaways:

- Code churn appears to be a more reliable metric for identifying files prone to bugs in Haskell projects compared to PSIZ.
- PSIZ may not effectively capture the complexity related to bugs, as many files and functions do not utilize extensive pattern matching.

6. Conclusion

In this research, the effectiveness of Code Churn and Pattern Size (PSIZ) metrics in predicting bug-prone areas in Haskell projects was investigated. The study found that Code Churn, which measures the frequency and extent of code changes, is a significant indicator of potential bugs, with buggy files exhibiting higher mean and median churn values. In contrast, the PSIZ metric, which measures pattern complexity in functions, did not show a strong correlation with bug occurrences, as both project-wide and buggy file medians were similar and close to 1. Future work could expand the analysis to more diverse projects and explore additional metrics to improve predictive accuracy and code quality insights.

Author Nikola Dzhunov

Responsible Professor Jesper Cockx

Supervisor Leonhard Applis

Examiner Koen Langendoen

Affiliation Delft University of Technology