The impact of branching and merging strategies on KPIs in open-source software

Author: Serban Ungureanu
ungureanu-2@student.tudelft.nl

Responsible professor: Sebastian Proksch
Supervisor: Shujun Huang

01. Introduction

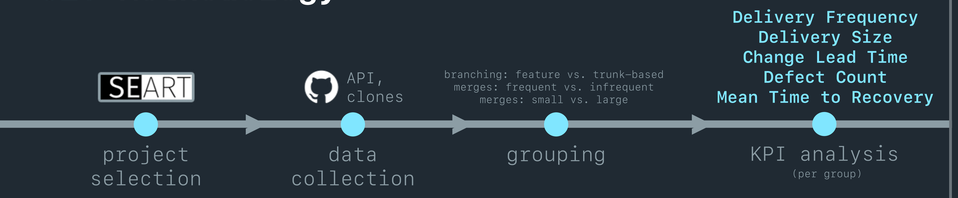
- Continuous Integration (CI) is a core part of modern open-source software (OSS)
- Prior work highlights general CI benefits [1, 2], but **often treats all workflows the same**
- Key practices like **branching strategy** and **merge behavior** may influence CI performance
- However, their impact on CI performance **remains unexplored** in large-scale empirical studies
- This study analyzes 565 GitHub repositories to compare the effects of workflow practices
- Our goal: **uncover how workflow affects delivery, recovery, and reliability**:

RQ1: Does **feature branch development** show a clear improvement in CI KPIs compared to **trunk-based development**?

RQ2: Do **frequent, small merges** correlate with better CI KPIs compared to **infrequent, large merges**?

RQ3: How do code management strategies **evolve over the lifetime** of a project?

02. Methodology



We analyzed **565** GitHub repositories using a combination of API mining and local Git inspection. Projects were grouped by:

Branching model: feature-based vs. trunk-based

Merge style: small vs. large, frequent vs. infrequent

To classify branching models, we extended existing methods [3] by scanning non-default branches and using PR and commit metadata.

Five Key Performance Indicators **(KPIs)** were calculated per group over monthly time buckets: **Delivery Frequency**, **Delivery Size**, **Change Lead Time**, **Defect Count** and **Mean Time to Recovery**. Groups were then compared against each other.

We also performed longitudinal analysis on a subset of mature projects to track **workflow evolution over time**.

03. Findings

Table 1: average KPI values per branching model group.

KPI	Feature	Trunk
Delivery Frequency	2.152	2.042
Delivery Size (LOC)	7824.42	12795.63
Change Lead Time (h)	157.229	169.581
Defect Count	20.268	26.152
Mean Time to Recovery (h)	336.938	284.167

RQ1: Feature-based workflows show improvements in Delivery Frequency and Defect Count, while trunk-based workflows lead in the other KPIs.

However, few projects utilize a trunk-based workflow (3% of our dataset), and it is possible that our findings can be attributed to other (possibly external) factors.

Table 2: average KPI values per merge group (S = small, L = large, F = frequent, I = infrequent).

KPI	SF	SI	LF	LI
DF	3.509	2.385	4.736	1.709
DS	10526	3523	30950	6491
DC	20.052	17.312	41.729	12.667
CLT	137.909	153.282	123.977	164.763
MTTR	409.573	347.660	326.010	312.267

RQ2: Projects with frequent merges generally show improved CI performance, with a higher Delivery Frequency and a lower Change Lead Time. Defect Count and Mean Time to Recovery show less clear patterns, however.

Frequent merges show general improvements, while merge size appears to not have as strong of an effect.

Figure 1: Branching model over time

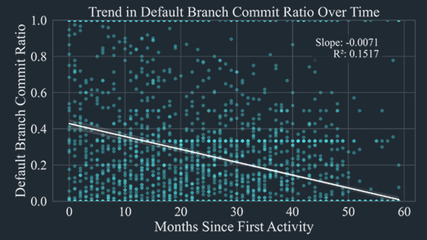


Figure 2: Trend in the ratio of commits made directly to the default branch

RQ3: We find that, as projects mature, they tend to adopt feature-based workflows, as observed in Figures 1 and 2.

However, merge size and frequency appears to stay consistent over a project's lifetime. The average merge size shows a near flat slope of *-8.97* lines of code per month decrease, with an R^2 of just *0.0006*. Similarly, the average number of days between merges decreases by *0.015* per month (R^2 = *0.0039*). These results are not significant enough to highlight a trend.

04. Conclusion

Our findings suggest that **development teams should choose a workflow that suits their projects' needs.** Feature-based workflows offer modularity and review infrastructure, but may introduce delays unless tightly managed. Trunk-based development, while rare, may remain viable in smaller teams or low-collaboration environments. However, these findings may stem from the underrepresentation of trunk-based workflows in our dataset.

Merge frequency emerges as a consistent predictor of CI success, while merge size has less of an observable effect. As such, teams should prioritize frequent, testable merges where possible.

Additionally, we find that **projects trend away from trunk-based development over time**, and that **merge size and frequency stay relatively constant**.

Future Work

Longitudinal repository tracking: tracking GitHub repositories in real-time can provide data that is otherwise inaccessible in our snapshot-based methodology. Such data includes how long lived feature branches tend to be, better detection of rebasing, when collaborators leave or join, and how they interact with workflows.
Developer-centric data: these insights could be complemented by developer surveys or interviews. These could fill in missing context about developer satisfaction and external tool usage. This could also be correlated with longitudinal tracking to identify trends in developer attraction to projects over time.
The measurability gap in trunk-based development: contrary to the patterns in our dataset, trunk-based development is extremely common [4]. However, many such repositories are either small, or personal projects, making it difficult to evaluate with our current methodology. Future work should explore new methods of evaluating such projects.

Limitations

- Several KPIs rely on GitHub Releases, which not all projects use.
- The dataset may include mirrors or archives, distorting KPI measurements.
- Our method for detecting rebases is only an approximation, and as such branching model misclassifications are possible.

References

[1] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov. Quality and productivity outcomes relating to continuous integration in github. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pages 805–816, 8 2015

[2] Yangyang Zhao, Alexander Serebrenik, Yuming Zhou, Vladimir Filkov, and Bogdan Vasilescu. The impact of continuous integration on other software development practices: a large-scale empirical study. In Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE '17, pages 60–71. IEEE Press, 2017.

[3] Yash Gupta, Yusaira Khan, Keheliya Gallaba, and Shane McIntosh. The impact of the adoption of continuous integration on developer attraction and retention. In 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), pages 491–494, 2017.

[4] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. An in-depth study of the promises and perils of mining GitHub. Empirical Software Engineering, 21:2035–2071, 2016. Publisher: Springer.