MINING REPRODUCIBLE DEPENDENCY UPDATES ACROSS ECOSYSTEMS

What changes are made to dependency update pull requests before they are accepted?

Paras Khan

Supervisor: Catherine Paulsen
Responsible Professor: Sebastian Proksch

INTRODUCTION

PROBLEM

Dependency management is a critical task in software engineering, but reliable automation remains difficult. Existing datasets of dependency update pull requests log build outcomes or high-level metadata, but not the causal code changes behind PR acceptance. A dataset that captures *why* PRs are accepted — not just whether — is needed to build dependency management tools that developers can trust.

RELATED WORK

Rebatchi et al. (2024) provides a large dataset of dependency update PRs across ecosystems, however the focus is only on security updates and high-level metadata. Reyes et al. (2024) provides a dataset of **reproducible** dependency update PRs, but focus is only on Java ecosystem, build outcomes and breaking changes.

OUR OBJECTIVE

We take a first step toward causal dependency-update datasets by categorising the code changes within accepted dependency update PRs at the commit level. This narrows down which changes plausibly caused a given build outcome, laying groundwork for future tools that combine this categorisation with build-outcome data, code analysis, and PR metadata.

RESEARCH QUESTIONS

- RQ1: **How can we create a categorisation model that describes dependency update PRs that get accepted?**
- RQ2: **To what extent can we automatically categorise dependency update PRs that get accepted?**

RESEARCH

METHODOLOGY

RQ1: We apply Nickerson et al.'s iterative taxonomy-development method to a partition of the Rebatchi et al. (2024) dataset (PRs mined Jan–Sep 2023). Starting from a meta-characteristic — "causes of dependency update PR acceptances" — we iteratively define and refine dimensions and characteristics until ending conditions are met, arriving at a model that represents PRs as ordered lists of commit-level label sets.
RQ2: We build a regex-based classifier implementing the RQ1 model, chosen over an ML approach for transparency and because no labelled training data exists for this task. We test it in two stages: first on 50 quota-sampled PRs to find weaknesses, then — after fixing the classifier's dependency-update detection logic — on 50 randomly sampled PRs for a final accuracy score.

RQ1: CATEGORISATION MODEL

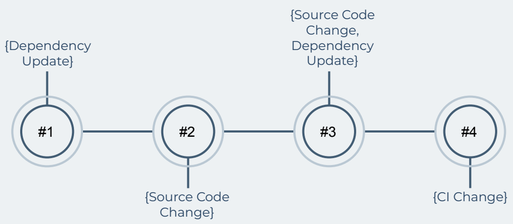

Figure 1: Example of a categorised PR

We model each PR as an ordered list of sets of labels: each set represents one commit, and the list represents commit order. A commit's label set captures every type of code change it contains — Source Code, Dependency Update, CI, Test, or Documentation — chosen because each independently affects (or does not affect) code output or build outcomes. Bot PRs (≈92% of the dataset) are almost always single dependency-update commits and are reproducible by default; for the remaining PRs, this structure lets researchers isolate which commit(s) plausibly drove a build outcome.

RQ2: CATEGORISATION TOOL

After quota-sampling 50 PRs, we found our classifier's dependency-update detection was too strict, misclassifying valid updates as "Other." We relaxed the matching logic to tolerate incidental changes (e.g. commit SHAs, timestamps) alongside the semver change. Testing the revised tool on 50 randomly sampled PRs gave 86% overall accuracy — statistically significant above a 70% reference threshold ($p = 0.007$), with a 95% Wilson confidence interval of 73–93%.

PR Type	Correct Labelling Ratio	Percentage
Dependency Update Only	17/20	85%
Source Code Only	3/3	100%
Other Only	2/2	100%
Test Only	0/0	N/A
CI Only	5/5	100%
Mixed	16/20	80%
Overall	43/50	86%

CONCLUSION

DISCUSSION

The model's high accuracy partly reflects that its categories are orthogonal and easy to differentiate once defined, rather than purely the classifier's design. Our taxonomy distinguishes between "reproducible" PRs (each commit isolates one change type) and "non-reproducible" PRs (commits mix change types, obscuring which change drove the outcome) — informing where deeper investigation is needed.

RECOMMENDATIONS

- Extend the taxonomy with finer-grained source code categories and deeper code analysis.
- Pair categorisation with build-outcome reproduction, PR metadata, and call-dependency graphs for a fuller causal picture.

CONCLUSION

Dependency update PRs can be described using ordered commit categorisation, and automatic categorisation is effective (86% accuracy on randomly sampled dependency update PRs).

REFERENCES

Rebatchi, H., Bissyandé, T. F., & Moha, N. (2024). Dependabot and security pull requests: Large empirical study. Empirical Software Engineering, 29 (5). https://doi.org/10.1007/s10664-024-10523-y

Reyes, F., Gamage, Y., Skoglund, G., Baudry, B., & Monperrus, M. (2024). Bump: A benchmark of reproducible breaking dependency updates. 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 159–170. https://doi.org/10.1109/saner60148.2024.00024

Robert C Nickerson, Upkar Varshney, and Jan Muntermann. "A method for taxonomy development and its application in information systems". In: European Journal of Information Systems 22.3 (May 2013), pp. 336–359. issn: 1476-9344. doi: 10.1057/ejis.2012.26. url: http://dx.doi.org/10.1057/ejis.2012.26