HOW SOFTWARE ENHANCEMENT PROPOSALS EVOLVE ALONGSIDE THEIR TARGET PROJECTS

Author: Kajus Kuncaitis (k.kuncaitis@student.tudelft.nl)
Responsible professor: Diomidis Spinellis (d.spinellis@tudelft.nl)

TU Delft

1. BACKGROUND

- Software Enhancement Proposal (**SEP**) - a document used to propose **new software features**, collect community views on issues, and record design decisions.
- Lehman's laws of software evolution try to describe how software projects evolve:
  - "**Increasing Complexity**" law states that active codebases become more complex unless specific work is done to reduce it.
  - "**Continuing Growth**" law states that new functionality must be continuously added to satisfy users.
- Many research papers focusing on mining open source code repositories and their version control histories to better understand software evolution, but one of pre-implementation phases - proposals themselves - are relatively **under-studied**.

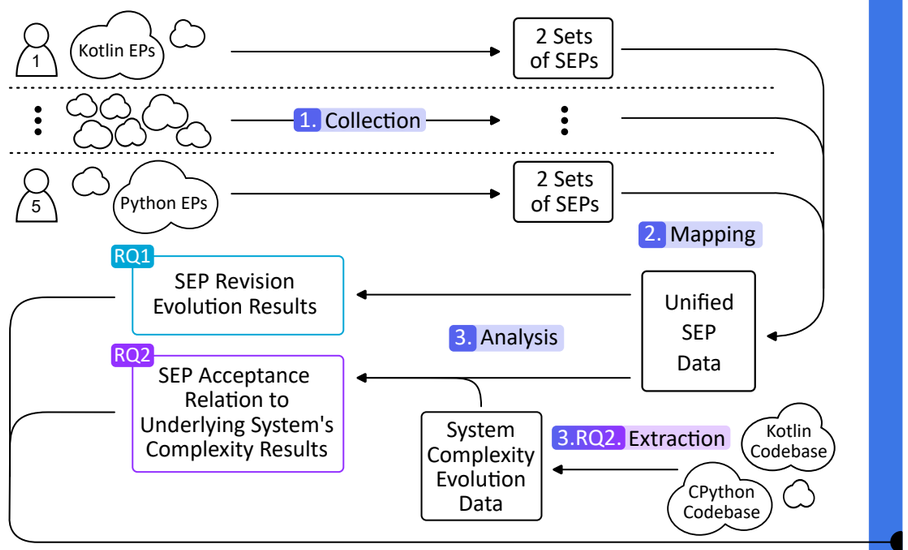
2. RESEARCH QUESTIONS

RQ1 *Which parts of proposals are frequently revised?*

&

RQ2 *How do proposal governance outcomes relate to the structural complexity of the underlying codebase?*

3. METHODOLOGY



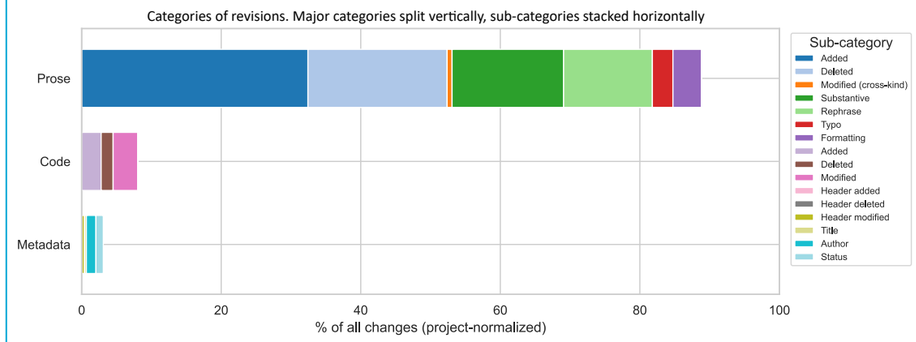
1. Collect proposal data from 10 different software projects: Python, Pandas, NumPy, JavaScript, C++, Rust, OpenJDK, Swift, Kubernetes, Kotlin (2 for each team member).
2. Each project's proposals follow different formats and conventions. Thus, each set of SEPs is individually mapped to a common schema and aggregated in an SQLite database.
3. From the common dataset, analysis was done:
   - RQ1. Identify types of changes between revisions, classify them with heuristic rules, and iteratively refine those rules.
   - RQ2. Collect underlying project codebase complexity metrics (cyclomatic complexity and total source lines of code) at semi-annual snapshots to get the complexity timelines. Then, perform statistical tests using Kendall's rank correlation coefficient on the following four hypotheses:

higher cyclomatic complexity (Ha) / larger codebase size (Hb)
at the time a proposal is introduced is associated with
longer time to acceptance (H1) / lower probability of acceptance (H2)

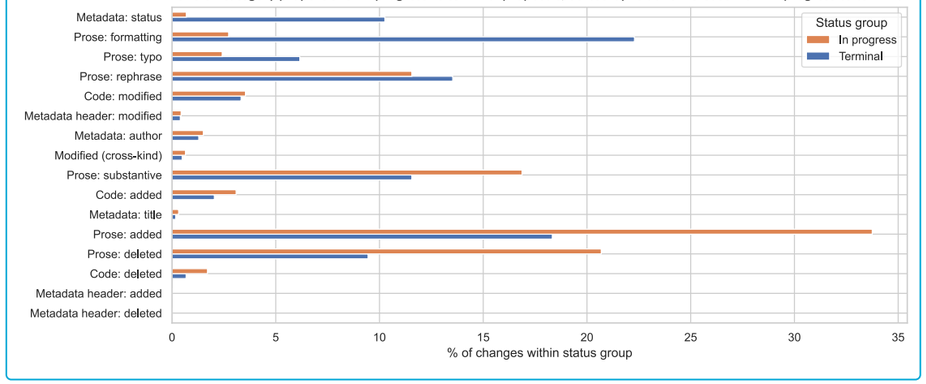
4. RESULTS

RQ1

12,714 proposals with **74,477** pairwise revisions analyzed.
Prose dominates with **88.9%** of all changes; large changes constitute the majority within prose (general text).

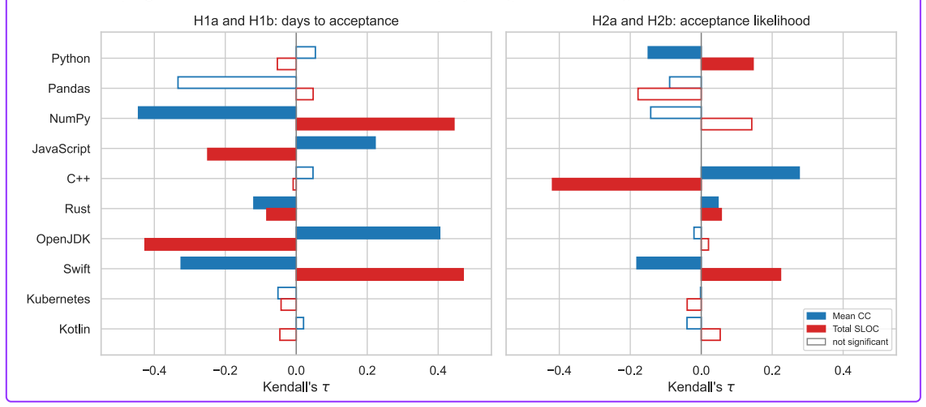


Majority of revision activity occurs while a proposal is still in progress (**~94.2%**).
In-progress revisions are dominated by **large-scale** content additions, deletions and changes.
Revisions made once a proposal has become terminal (accepted, rejected, etc.) shift toward **refinement** and bookkeeping-oriented changes (status transitions, formatting corrections, typo fixes, and rephrasings).



RQ2

Statistically significant associations exist in several projects (p-value < 0.05), but their direction **varies**.



5. CONCLUSIONS

RQ1 *Proposals undergo many major changes, especially while in progress. Terminal proposals are still revised, albeit more often in smaller scale fixes.*

RQ2 *None of the hypotheses are consistently supported across the ten projects.*

6. LIMITATIONS

Only a small sample of 10 projects was investigated, so the results might not generalize to other projects.

No inter-rater reliability test was done for classifying types of revisions.

The two code complexity measuring proxies used might not capture the full nuance of a system's complexity.

Other independent variables (e.g., proposal topic, author characteristics, etc.) were not accounted for.