Control Variables' Interactions with ForkEntropy for Response Variables

December 16th, 2024

ForkEntropy & NumFiles: Figure 1 shows the relationship between NumFiles and NumIntegratedCommits at different ForkEntropy levels (Low, Mean, and High). As seen, for any given value of NumFiles, the number of merged commits increases with higher ForkEntropy. Similarly, for any fixed level of ForkEntropy, the number of merged commits correlates positively with the number of modified files. Moreover, as ForkEntropy increases from low to high, the relationship between NumFiles and NumIntegratedCommits becomes steeper. Given the common practice in OSS projects where core developers regularly integrate recent file modifications, we speculate that when file modifications become more diverse (i.e., when ForkEntropy is high), commits to the master repo are more likely to be made by core members. As a result, this period typically corresponds to a higher number of integrated commits.

ForkEntropy & NumStars: The interaction between ForkEntropy and NumStars is divided into two phases by the dashed line: on the left side of the line, with a low star count, changes in ForkEntropy do not significantly affect the project's contribution reception. In all three ForkEntropy states, the difference in merged commits is less than 0.1. This convergence likely due to the project receiving limited attention in its early stages. As the project gains more attention, the relationship between ForkEntropy and merged commits becomes more pronounced. On the right side of the dashed line, as NumStars increases, the number of mergeable commits rises significantly with higher ForkEntropy. Besides, when ForkEntropy is low, there is a negative correlation between stars and NumIntegratedCommits. We speculate that occurs because more popular projects, which focus on core feature development, tend to be more cautious in accepting external contributions. This is consistent with general observations in open-source software development.

ForkEntropy & RatioOldVolunteers: The relationship between ForkEntropy and RatioOldVolunteers is straightforward. For any stage of an OSS project, experienced developers are always valued. As shown in the figure, regardless of the ForkEntropy state, the number of experienced developers is positively correlated with the number of accepted commits. Similarly, when the proportion of old contributors is fixed, as the project transitions from concentrated development to distributed file modification, the OSS maintainers take greater emphasis on contributions from experienced developers.

In Figure 2, we further explore the interaction between ForkEntropy and other control variables (NumForks, NumFiles, ProjectAge, and RatioVolunteers) when RatioMergedPrs is the response variable.

ForkEntropy & NumForks: At all levels of ForkEntropy, the number of forks is negatively correlated with the ratio of Merged PRs during a given period. When NumForks is fixed, as the project's file modification from focused to decentralized development, the proportion of PRs received from forks increases. This reflects the principle "more eyes, fewer bugs" [1] in the maintenance phase of OSS projects. However, the negative correlation between fork numbers and the proportion of Merged PRs also reveals that, regardless of the project state (concentrated or decentralized), truly valuable direct code contributions always make up a minority of the total forks. As the number of forks increases, this proportion becomes further diluted.

ForkEntropy & NumFiles: The relationship between the number of file modifications and the proportion of Merged PRs varies at different levels of ForkEntropy. When NumFiles is fixed, the probability of a PR being merged increases as ForkEntropy rises, which is consistent with the interaction with NumForks. We speculate this is because when the project is in a decentralized file modification, it has a higher acceptance rate for contributions from external contributors, like 'Bug Report', 'Feature Request'. When ForkEntropy is low or medium, PRs involving more file modifications tend to have a higher acceptance rate. We speculate that in the concentrated file modification (low or medium ForkEntropy), core developers collaborate via PRs rather than direct commits, making large-scale modifications. In contrast, when ForkEntropy is high, indicating a decentralized file modification, more file modifications may come from external contributors, leading to a mitigate decrease in the RatioMergedPrs.

ForkEntropy & ProjectAge: The interaction between ForkEntropy and ProjectAge shows a consistent pattern. Regardless of ForkEntropy levels, project age is positively correlated with the likelihood of accepting PRs. Similarly, for a given project age, higher ForkEntropy also increases PR acceptance. We hypothesize that as the project matures, its community becomes more organized, allowing it to consistently receive more valuable contributions.

ForkEntropy & RatioOldVolunteers: The interaction between ForkEntropy and experienced contributors follows a consistent pattern. At all ForkEntropy levels, a higher proportion of experienced contributors submitting PRs is associated with a higher acceptance rate. Similarly, when the number of experienced contributors is fixed, PR acceptance increases as the project shifts from decentralized to concentrated file modification. This aligns with the

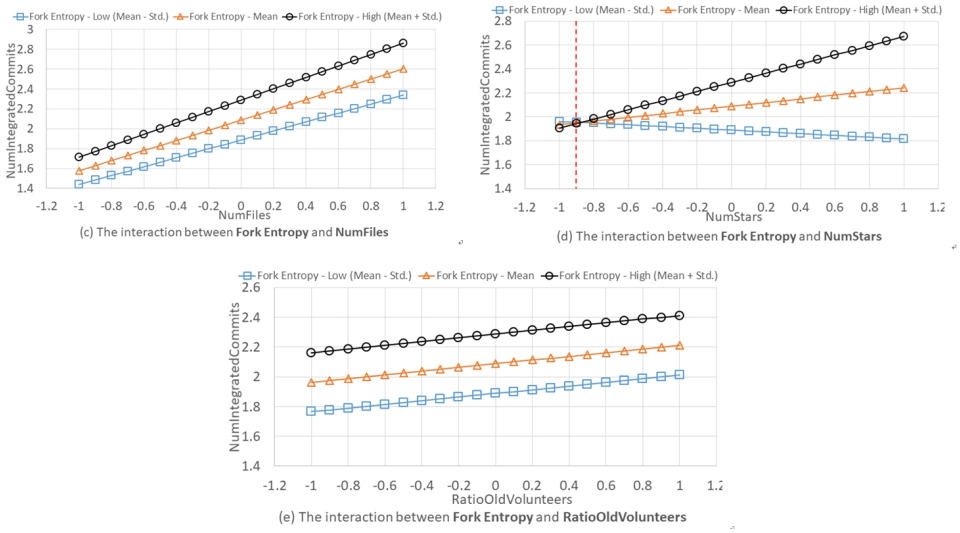

Figure 1: The Interactions over forkEntropy in the external productivity model

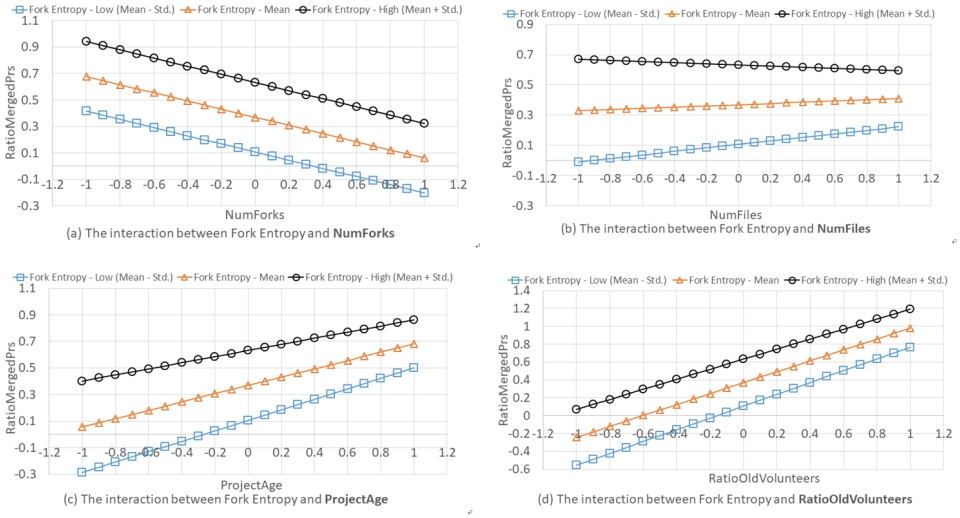

Figure 2: The Interactions over forkEntropy in the acceptance rate model

widely recognized value of experienced contributors in OSS projects.

In Figure 3, we have added the interaction between ForkEntropy and other control variables (NumFiles, ProjectAge, and NumStars), with NumReportedBugs as the response variable.

ForkEntropy & NumFiles: When NumReportedBugs is the response variable, the interaction between ForkEntropy and the number of file modifications varies across different ForkEntropy levels. At low ForkEntropy levels, the number of reported bugs decreases as file modifications increase. We speculate that during concentrated file modification phases, large-scale external contributions are less likely to be made just to report bugs. Additionally, since low ForkEntropy corresponds to concentrated modifications of a few files, bugs are often introduced in this phase, leading to more bug reports for fixed numbers of file modifications [2]. In contrast, at higher ForkEntropy levels, the number of reported bugs increases with more file modifications. We speculate that at this stage, external contributions tend to focus on simple fixes to completed features, which leads to an increase in both file modifications and reported bugs.

ForkEntropy & ProjectAge: The interaction between ForkEntropy and ProjectAge shows a consistent pattern. Regardless of the level of ForkEntropy, the project's development time is positively correlated with the number of bugs reported. When ForkEntropy is fixed, the number of reported bugs tends to increase as the project matures. We speculate that as the project develops, its community becomes more established, leading to an increase in bug reports from external contributors. Similar to the relationship with NumFiles, when the project is in its concentrated file modification phase, it tends to receive more bug reports.

ForkEntropy & NumStars: Similar to the relationship with ProjectAge, the interaction between ForkEntropy and NumStars follows a consistent pattern. Regardless of ForkEntropy level, the number of stars a project receives is positively correlated with the number of bugs reported. We speculate that as an OSS project gains more attention,

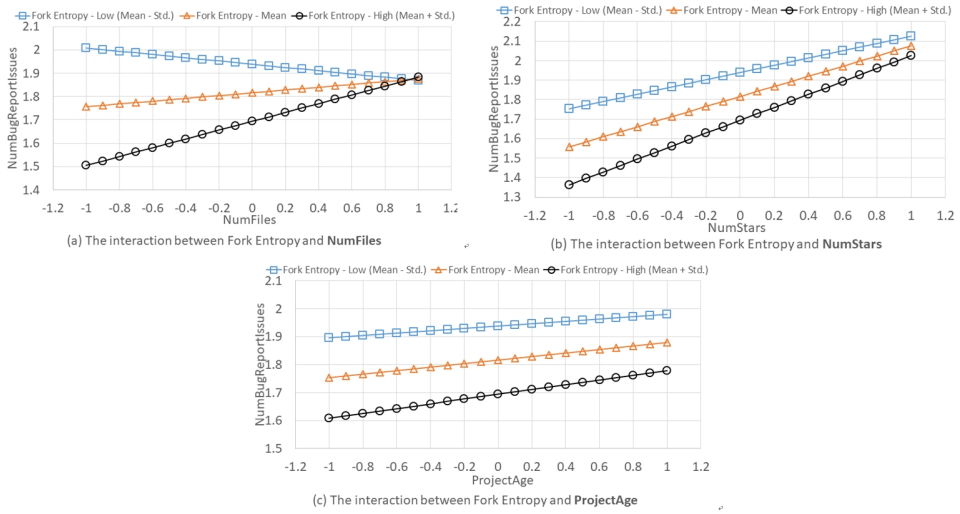

Figure 3: The Interactions over forkEntropy in the number of reported bugs mode

it also attracts more contributions, including both improvements and bug reports. As in previous cases, during the concentrated file modification phase, the project tends to receive more bug reports, regardless of its star count.

References

[1] E. Raymond. The cathedral and the bazaar. *Knowledge, Technology & Policy*, 12(3):23–49, 1999.

[2] H. Wu, L. Shi, C. Chen, Q. Wang, and B. Boehm. Maintenance effort estimation for open source software: A systematic literature review. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 32–43, 2016.