Reuse and Maintenance Practices among Variant Forks in Three Software Ecosystems

John Businge,* Moses Openja,† Sarah Nadi,† and Thorsten Berger§
*University of Antwerp, Belgium
†SWAT Lab., École Polytechnique de Montréal Montréal,Canada
‡University of Alberta, Edmonton, Canada
§Ruhr University Bochum, Germany

I. Presentation Abstract

With the rise of social coding platforms that rely on distributed version control systems, software reuse is also on the rise. Many software developers leverage this reuse by creating variants through forking, to account for different customer needs, markets, or environments. Forked variants then form a so-called software family; they share a common code base and are maintained in parallel by same or different developers. As such, software families can easily arise within software ecosystems, which are large collections of interdependent software components maintained by communities of collaborating contributors. However, little is known about the existence and characteristics of such families within ecosystems, especially about their maintenance practices. Improving our empirical understanding of such families will help build better tools for maintaining and evolving such families.

Many studies on forking exist, often focusing on the reasons and outcomes or on the community dynamics as influenced by forking. The community typically distinguishes between two kinds of forks: *social forks* that are created for isolated development with the goal of contributing back to the mainline and *variant forks* that are created for splitting off a new development branch, often to steer the development into another direction without intending to contribute back, while leveraging the mainline project that defines or adheres to some standards. Variant forks are more relevant for supporting large-scale software reuse—the focus of this study.

When studying code propagation techniques, existing studies do not consider the intricacies of git to identify the possible types of code propagation (e.g., offline git rebasing), but mainly focus only on pull requests. Furthermore, the few that have focused both techniques of Git and GitHub, have ignored propagation techniques that do not preserve commit IDs (e.g., pull request rebase / squash and Git cherry-picking). To address these intricacies, in this presentation abstract we present our where we designed a technique that identifies the majority of code propagation techniques on Git and GitHub by leveraging all commit meta data [1]. We also empirically explored maintenance practices in such fork-based software families within ecosystems of open-source software. Our focus is on three of the largest software ecosystems existence today: Android, .NET, and JavaScript. We investigated two main research questions: *RQ1–What are the characteristics of app families in our ecosystems?* and *RQ2–How are software families maintained and co-evolved in our ecosystems?*

The results for RQ1 show that families in fact exist in our three software ecosystems. We collected 38, 526, and 8,837 families from the Android, .NET and JavaScript ecosystems, respectively. While both the mainlines and forks have multiple releases, the number of releases is significantly higher than those of the forks. Still it indicates that the latter are usually not one-shot releases; with some having even more than their mainlines. We also discovered that the majority of the mainline–fork variant pairs for the three ecosystems are owned by different developers (91 % for Android variants, 95 % of JavaScript variants and 92 % of .NET variants). This implies that the majority of forked variants in our datasets are started and maintained by developers different from those maintaining the mainline counterparts.

The results for RQ2 show that in all the studied mainline–fork variant pairs of the three ecosystems, there are infrequent code propagation, regardless of the type propagation mechanism or direction. The most used code propagation technique is `git merge/rebase`, which is used in 33 % of Android mainline-fork pairs, 18 % of .NET pairs, and 11 % of JavaScript pairs. Integration using `git cherry-pick` integration technique is rarely used contributing to 7 % of Android mainline-fork pairs, 3 % of .NET pairs, and 1 % of JavaScript pairs. For integration using pull requests, developers often integrate code in the direction of fork→mainline compared to those in the direction of mainline→fork, in all the mainline–fork variants. The code integration in the direction of mainline→fork is often done using the `merge` pull request option or `git merge/rebase` outside GitHub. Moreover, the `squash` and `rebase` pull request options are less frequently used in mainline–fork variant pairs, although the `squash` pull request option is more used than the `rebase` pull request option.

Overall, we hope to raise awareness about the existence of software families within larger ecosystems of software, calling for further research and better tools support to effectively maintain and evolve them.

References

[1] John Businge, Moses Openja, Sarah Nadi, and Thorsten Berger. Reuse and maintenance practices among divergent forks in three software ecosystems. *Journal of Empirical Software Engineering*, 2021. *Accepted.*