Variant Forks – Motivations and Impediments

John Businge,* Ahmed Zerouali,‡ Alexandre Decan,† Tom Mens,† Serge Demeyer,* and Coen De Roover,‡
*University of Antwerp, Antwerp, Belgium
{ john.businge | serge.demeyer }@uantwerpen.be
†University of Mons, Mons, Belgium
{ alexandre.decan | tom.mens }@umons.ac.be
‡ Vrije Universiteit Brussels, Brussels, Belgium
{ ahmed.zerouali | coen.de.roover }@vub.be

***Abstract*—Social coding platforms centred around git provide explicit facilities to share code between projects: forks, pull requests, cherry-picking to name but a few. Variant forks are an interesting phenomenon in that respect, as they permit for different projects to peacefully co-exist, yet explicitly acknowledge the common ancestry. Several researchers analysed forking practices on open source platforms and observed that variant forks get created frequently. However, little is known on the motivations for launching such a variant fork. Is it mainly technical (e.g., diverging features), governance (e.g., diverging interests), legal (e.g., diverging licences), or do other factors come into play? We report the results of an exploratory qualitative analysis on the motivations behind creating and maintaining variant forks. We surveyed 105 maintainers of different active open source variant projects hosted on GitHub. Our study extends previous findings, identifying a number of fine-grained common motivations for launching a variant fork and listing concrete impediments for maintaining the co-existing projects.**

***Index Terms*—Mainlines, Variants, GitHub, Software ecosystems, Maintenance, Variability**

I. Introduction

The collaborative nature of open source software (OSS) development has led to the advent of social coding platforms centred around the git version control system, such as GitHub, BitBucket, and GitLab. These platforms bring the collaborative nature and code reuse of OSS development to another level, via facilities like forking, pull requests and cherry-picking. Developers may fork a *mainline repository* into a new *forked repository* and take governance over the latter while preserving the full revision history of the former. Before the advent of social coding platforms, forking was rare and was typically intended to compete with the original project [1]–[6].

With the rise of pull-based development [7], forking has become more common and the community typically characterises forks by their purpose [8]. *Social forks* are created for isolated development with the goal of contributing back to the mainline. In contract, *variant forks* are created by splitting off a new development branch to steer development into a new direction, while leveraging the code of the mainline project [9].

Several studies have investigated the motivations behind variant forks in the context of OSS projects [1]–[6]. However, most have been conducted before the rise of social coding platforms and it is known that GitHub has significantly changed the perception and practices of forking [8]. In this social coding era, variant projects often evolve out of social forks rather than being planned deliberately [8]. To this end, social coding platforms often enable mainlines and variants to peacefully co-exist rather than compete. Little is known on the motivations for creating variants in the social coding era, making it worthwhile to revisit the motivation for creating variant forks (*why?*).

Social coding platforms offer many facilities for code sharing (e.g., pull requests and cherry-picking). So if projects co-exist, one would expect variant forks to take advantage of this common ancestry, and frequently exchange interesting updates (e.g., patches) on the common artefacts. Despite advanced code-sharing facilities, Businge et al. observed very limited code integration, using the *git* and GitHub facilities, between the mainline and its variant projects [10]. This suggests that code sharing facilities in themselves are not enough for graceful co-evolution, making it worthwhile to investigate impediments for co-evolution (*how?*).

We therefore explore two research questions:

RQ1: Why do developers create and maintain variants on GitHub? The literature pre-dating *git* and social coding platforms identified four categories of motivations for creating variant forks: technical (e.g., diverging features), governance (e.g., diverging interests), legal (e.g., diverging licences), and personal (e.g., diverging principles). $RQ1$ aims to investigate whether those motivations for variant forks are still the same, or whether new factors have come into play.

RQ2: How do variant projects evolve with respect to the mainline? If, despite advanced code sharing facilities, there is limited code integration between the mainline and the variant projects, a possible cause could be related to how the teams working on the variants and the mainline are structured. Therefore, $RQ2$ investigates the overlap between the teams maintaining the mainline and variant forks, and how these teams interact. As such we hope to identify impediments for co-evolution.

The investigations are based on an online survey conducted with 105 maintainers involved in different active variant forks hosted on GitHub.

Our contributions are manifold: we identify new reasons for creating and maintaining variant forks; we identify and categorize different code reuse and change propagation practices between a variant and its mainline; we confirm that little code integration occurs between a variant and its mainline, and uncover concrete reasons for this phenomenon. We discuss

the implications of these findings and how tools can help to achieve an efficient code integration and collaboration between mainlines and diverging variant forks. Our replication package can be found [1].

II. Related Work

Previous research has focused on (A) motivations for creating or maintaining variant forks; and (B) interaction between variant forks and their mainline.

A. Motivations for creating or maintaining variant forks

Several studies have investigated motivations for creating and maintaining variant forks. However, most of these studies were carried out on SourceForge, pre-dating the advent of social coding platforms like GitHub [1]–[5], [11]. Several of those early studies report perceived controversy around variant forks [5], [12]–[17]. Jiang et al. [18] state that, although forking may have been controversial in the OSS community, it is now encouraged as a built-in feature on GitHub. They further report that developers create *social forks* of repositories to submit pull requests, fix bugs, and add new features. Zhou et al. [8] conclude that most variant forks started as social forks and that perceptions of forks have changed with the advent of GitHub. Robles and González-Barahona [2] carried out a comprehensive pre-GitHub study on a carefully filtered list of 220 potential forks referenced on Wikipedia. They report motivations and outcomes for forking on these 220 projects.

The literature has uncovered a number of motivations for creating variants. Below, we present those where both the mainline and variant co-evolve together. The motivation of *reviving an abandoned project* is not considered in this study since it does involve co-evolution if the variants.

- *Technical (addition of functionality).* Sometimes developers want to include new functionality into the project, but the main developer(s) do not accept the contribution. An example is Poppler, a fork of xpdf relying on the poppler library [2].
- *Governance disputes.* Some contributors from the community create a variant project because they feel that their feedback is not heard, or because the maintainers of the mainline are unresponsive or too slow at accepting their patches. A well-known example is a fork of GNU Emacs (originally Lucid) which was created as a result of the significant delays in bringing out a new version to support the Energize C++ IDE [19].
- *Legal issues.* This includes disagreements on the license and trademarks, and changes to conform to rules and regulations. An example is X.Org, which originated from XFree86 [2], [19]. XFree86 was originally MIT/X open source license that is GPL-compatible and then was changed to one that was not GPL-compatible. This caused many practical problems and a serious uproar in the community, resulting in the project fork X.Org.
- *Personal reasons.* In some situations, the developer team disagrees on fundamental issues (beyond mere technical matters) related to the software development process and the project. An example is the OpenBSD fork from NetBSD. One of the developers of NetBSD had a disagreement with the rest of the core developers and decided fork and focus his efforts on OpenBSD [20].

Focusing on variant forks in the Android ecosystem, Businge et al. [21] found that re-branding, simple customizations, feature extension, and implementation of different but related features are the main motivations to create forks of Android apps. Zhou et al. [8] interviewed 18 developers of hard forks on GitHub to understand reasons for forking in social coding environments that explicitly support forking. The motivations they observed align with the findings of the aforementioned studies.

Sung et al. [9] investigated variant forks in an industrial case study to uncover the implications of frequent merges from the mainline and the resulting merge conflicts in the variant forks. They implemented a tool that can automatically resolve up to 40% of 8 types of mainline-induced build breaks.

While the pre-GitHub studies reported perceived controversy around variant forks, Zhou et al. [8] report that this controversy has reduced with the advent of GitHub. Jiang et al. [18] report that, while forking is considered controversial in traditional OSS communities, it is actually embraced as a built-in feature in GitHub. Our study builds on these previous studies to identify whether the motivations for variant forks are still the same or whether new factors have come into play.

B. Interaction between variant forks and their mainline

We have only encountered two studies that investigated the interaction between variant forks and mainlines [8], [10]. Zhou et al. [8] conducted 18 semi-structured developer interviews. Many respondents indicated being interested in coordination across repositories, either for eventually merging changes back into the mainline, or to monitor activity in the mainline repository and select and integrate interesting updates into their variant project. Businge et al. [10] also investigated the interaction between mainline and variants. The authors quantitatively investigated code propagation among variants and their mainline in three software ecosystems. They found that only about 11% of the 10,979 mainline–variant pairs had integrated code between them. Since the mainlines and variants share a common code base, and with the collaborative maintenance facilities of *git* and the pull-based development model, one would expect more interactions between the mainline and its variants. We hypothesise that there are some impediments to enable such interactions. Since the two aforementioned studies do not report any such impediments, we decided to carry an exploratory qualitative survey with variant maintainers to identify possible impediments.

III. Study Design

To understand the motivations behind the creation and maintenance of variant forks we conducted an online survey with maintainers of variant forks. In this section, we explain

[1] 10.5281/zenodo.5855808

how we (i) designed the survey protocol; (ii) collected mainline–variant pairs and extracted the maintainers of the variant forks; and (iii) recruited the survey participants.

A. Survey Protocol Design

We designed a 12-question survey that would last at most 15 minutes. Since we aimed to learn from a large number of projects, we used an online survey as this data collection approach is known to scale well [22]. The survey can be found here[2]. The questions were designed to cover our two main research questions. 8 of the 12 questions were close-ended and respondents could answer them either via multiple choice or Likert scales. An optional free-text form was provided for 3 of the 8 close-ended questions to allow respondents to share additional thoughts and feedback. The 4 remaining questions are open-ended. All questions were carefully formulated so as not to bias respondents towards a specific answer. We validated them by subjecting them to the critical eye of 7 colleagues and by conducting trial runs of the survey with the same 7 participants.

B. Identifying variant projects and participants

Given the scope of the survey, we target respondents involved in the creation and maintenance of variant projects. Therefore, we first needed to identify such variants. To this end, we relied on two data sources: Libraries.io and GitHub.

Libraries.io contains metadata about projects distributed through various package registries. We collected the metadata for all projects of some of the largest package registries (npm, Go, Maven, PyPI and Packagist). We relied on this metadata to identify those projects that are variants of another one, following the variant identification method proposed by Businge et al. [10], [23]. We only considered variants that are actively maintained in parallel with their mainline counterparts. We extracted variants for which the mainline–variant pair was created before 2019-04-01 and updated at least once after 2020-04-01 (i.e., active projects). This process yielded *227 mainline–variant project pairs*.

We collected additional mainline-variant pairs from GitHub directly. To do so, we searched for mainline projects using the GitHub search endpoint. We looked for *popular* (> 50 stars and forks), *long-lived* (created before 2018) and *active* (still updated in 2020) repositories. We focused on software development repositories whose main language is among the top 17 of most popular languages used in GitHub (e.g., JavaScript, Java, Go, Python, Ruby, C, etc). For all the mainline projects we found, we tried to identify and collect variant forks. This process is subject to a known threat to validity since previous studies revealed that the majority of forks on GitHub are inactive [24], [25] or are social forks [21]. To reduce this threat, we filtered forks based on the following heuristics: ≥ 10 stars, ≥ 10 commits ahead of the mainline, ≥ 5 closed pull requests, diverging README files. We manually verified these remaining forks to ensure they corresponded to variants of the corresponding mainline. This process yielded *264 additional mainline-variant project pairs*, leading to **a total of 491 collected mainline–variant pairs**.

C. Participant Recruitment

Based on this collection of mainline-variant pairs, we identified contributors that had integrated at least one pull request into the variant. We retrieved their public-facing emails (if available) using the GitHub API, while ensuring to respect the GitHub Privacy Statement.[3] We individually contacted a total of 762 variant maintainers from the 491 variant projects, and received **a total of 105 responses (response rate 14%), representing a total of 105 variant forks (21%)**. All participants were requires to read and accept an informed consent form before taking part in the survey.

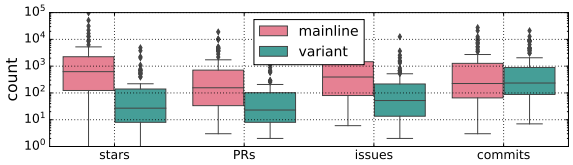


Fig. 1: Distribution of selected metrics. PRs, issues and commits are counted after the fork date for both mainline and variant.

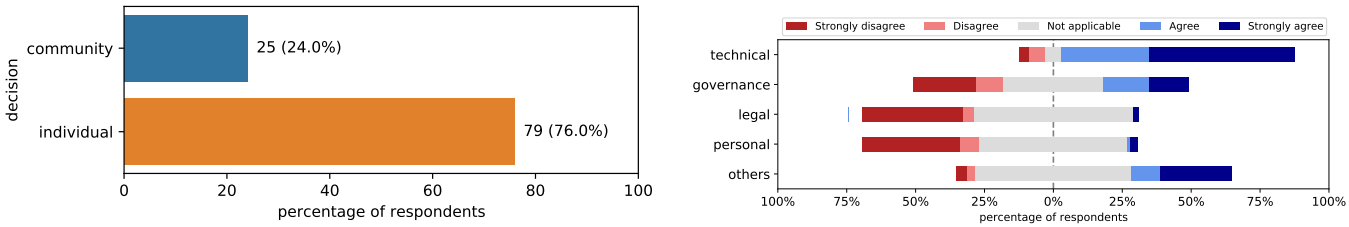
We wanted to compare the popularity of the 105 mainline–variant pairs since the fork date. To this end, we collected metrics of stars, pull requests, issues and commits from the projects. For the variant projects, all these metrics are calculated from the fork date. For the mainline projects, we calculate the metrics of pull requests, issues and commits from the fork date. The number of stars in the mainline are calculated throughout the lifetime of the project. The boxplots in Fig. 1 show the distributions of stars, pull requests, issues and commits for the selected mainline-variant pairs. While it is not surprising that the counts for mainline metrics are always higher than those of the variants, it is interesting that most variants are also popular in stars, pull requests and issues counts. This gives us confidence that we are studying real variants as opposed to social forks.

D. Analysis

We used open card sorting [26], on the 3 open-ended questions to identify common responses reported by the participants. In the analysis, we grouped similar responses from the open-ended questions into themes. We did not start with any pre-defined themes in mind, but instead derived the themes from the open-ended answers, iterating as many times as needed until reaching a saturation point. The first iteration of coding themes was performed by the first author of the paper, and any responses the first author was unsure of were decided by discussion with the second author. Once the first two authors agreed on the themes, a virtual meeting was set with all six authors to discuss the resulting themes and come to a negotiated agreement [27]. This allowed us to remove duplicates and, in some cases, to generalize or specialize themes.

[2] 10.5281/zenodo.5855808

[3] https://docs.github.com/en/github/site-policy/github-privacy-statement



(a) Was the motivation for creating the variant an individual decision or a community decision?

(b) What was the motivation for creating the variant of the mainline project?

Fig. 2: *RQ1:* Why do developers create and maintain variants on GitHub?

IV. *RQ1:* Why do developers create and maintain variants on GitHub?

$RQ1$ aims to investigate whether new motivations for creating variant forks have changed since the advent of social coding platforms. To do so, we asked the survey participants the following questions:

SQ^1_a: *Was the motivation for creating the variant an individual or a community decision?*

SQ^1_b: *What was the motivation for creating the variant of the mainline project?*

SQ^1_c: *What are the motivation details relating to the motivation in* SQ^1_b*?*

For SQ^1_a, we presented a multiple choice question. SQ^1_b presented Likert-scale answer options, while SQ^1_c was an optional open-ended question. For the latter, we coded the responses into themes and categorised common themes. When quoting the survey respondents, we refer to them using [RN] notation, where N is the respondent's ID. The respondents' answers that include the selection on the multiple choice answers as well as the themes resulting from coding open-ended answers are underlined. The open-ended responses are presented in *italics*. Where applicable, we integrate and compare our findings with related research findings.

A. Results

Fig. 2 summarises the responses for SQ^1_a and SQ^1_b. Fig. 2(a) shows that the majority of the participants responded that the decision was individual. Fig. 2(b) shows that the majority ranked highly the technical motivation for creating variants. We also see quite a number of highly ranked motivations of governance and others.

While previous studies have investigated the motivations for creating variants, no study has investigated the details of those motivations (SQ^1_c) . To identify these details, two optional open-ended questions allowed respondents to provide details on their Likert-scale answer to SQ^1_b. The two questions were (1) *Kindly provide details for your selected answer(s) on the motivation;* and (2) *If there are any links that are documented relating to your choice of answers on motivation detail, kindly point us there*.

100 of the 105 survey respondents answered the optional open-ended question SQ^1_c. Luckily, during the coding process (cf. Section III-D), we were able to identify possible answers of the 5 respondents that did not answer SQ^1_c by comparing the information on the readme.md files of the variant and mainlines. 30 of the 105 respondents provided links to documents (pull requests, issues, and blogs) relating to their choice of answers on motivation detail.

Fig 3 presents a Sankey diagram summarising the details of the respondents' choice of motivation based on the coded themes. The figure presents the distribution of the responses to all questions relating to $RQ1$ and how these responses relate to each other. The thickness of the edge represents the frequency of respondents between two entities.

Focusing on the axes of decision and motivation, we can confirm the observations from Fig. 2(b) that the majority of respondents had an individual and technical motivation. The majority of respondents that answered the question original developers? selected none implying that the majority of the variants were started by different developers. Since the answers to SQ^1_b were presented on a Likert scale, participants were asked to rank the appropriate motivation(s) to why they created the variant. While coding the motivations details, we identified respondents who ranked highly more than one motivation category and also provided a response in the open-ended question to support each highly ranked motivation category. In this scenario, each highly ranked motivation category would have a motivation detail for the same respondent. At the end we found that 105 of the survey participants chose 145 motivation categories, of which 84 technical, 34 governance, 3 legal and 24 others. Below we present the common motivation themes and some specific responses we found very interesting.

Technical. Maintenance is the most frequently mentioned reason for the technical motivation. 19 of the 84 survey participants who selected technical, mentioned phrases related to *performing bug/security fixes*.

- [R59] ranked highly both technical and governance and mentioned "*The PR to merge the fork's new capabilities into the mainline code was too large, [...] and my attempts to incorporate feedback into the PR [...] ended upsetting the primary maintainer who has been studiously ignoring the pull request for three years* ☹". The respondent also provided a GitHub link to his pull request to the mainline. Indeed, we found that the PR was made in February 2018 and was accompanied by a discussion of 218 comments between the mainline maintainer and the respondent. On October 2021, the PR was still open.

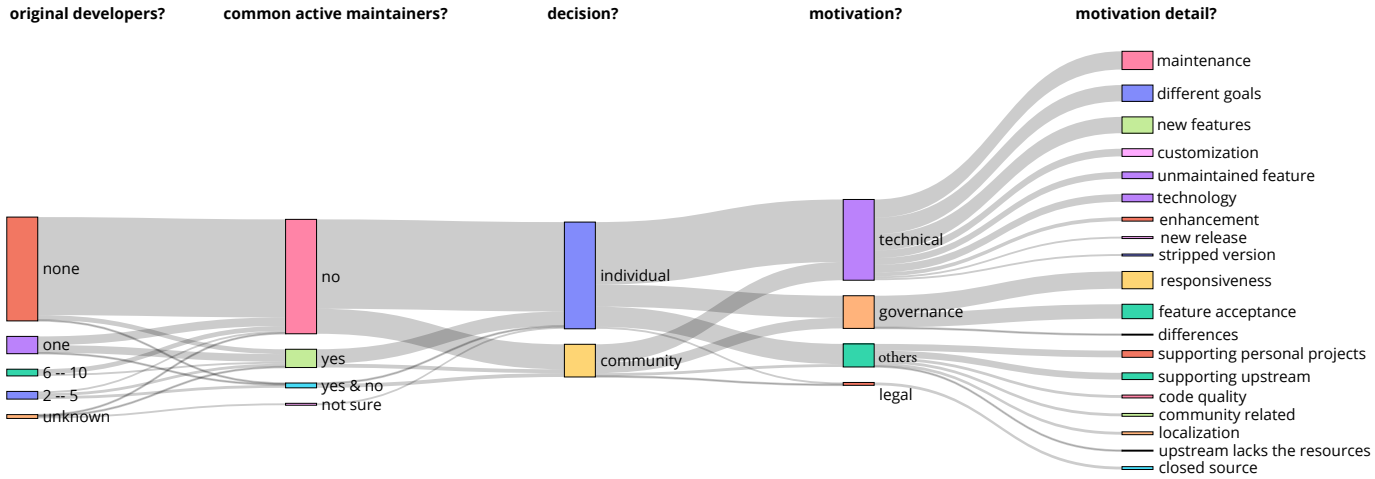


Fig. 3: Sankey diagram summarising the detailed motivations behind creating variant forks.

- "*I forked the original project in order to fix a bug. However, the way the original was architected made this very challenging, so I ended up rewriting it instead of submitting a patch to the original.*" [R79]

The next prominent technical motivation detail was different goals. 17 respondents who selected technical, mentioned phrases related to *variants present different goals / content / communities / directions*:

- "*[We] list websites that accept Bitcoin Cash cryptocurrency, as opposed to the mainline that lists websites with 2 factor authentication.*" [R1]
- "*The original goal of the mainline is completely different from the fork variant.* " [R4]
- "*We wanted to take the project in a different direction*" [R100].

An equally prominent technical motivation detail was new features. 17 respondents who selected technical, mentioned phrases related to *introduction of new features not in the mainline*:

- "*[...] to add support for a feature I knew would not get merged into the main project.*" [R53]
- "*Mainline developer only does bugfixes and eventual underlying runtime/SDK upgrades to stay current. He did not add new features due to lack of interest . . .* " [R67]
- "*Our variant introduces new experimental functionality that is not yet ready for use in the mainline.*" [R80]

Another technical motivation was customization. 8 respondents who selected technical, mentioned phrases related to *variant customizes the mainline features*:

- "*The "bones" were good, but I wanted to add some aesthetics [...] so, I forked it to make it pretty and my own.*" [R10]
- "*The new version is a vectorized, accelerated version of the original.*" [R37]
- "*[We] added some syntactic sugar and some improvements by itself . . .* " [R42]

The next technical motivation was unmaintained feature. 8 respondents who selected technical, mentioned phrases related to *one of the mainline feature used by the variant is no longer maintained*.

- "*The 'shiny' component of mainline was declared to be no longer maintained around the time I created our fork. [. . .] I did not like many of the architectural decisions of the original project, I opted to create a fork instead of volunteer to maintain the original.*" [R65]. The respondent provided an extra link. An issue about 'shiny' component was opened up in July 2015 and closed in July 2017. The issue contained 93 comments from 35 participants. When closing the issue the maintainer stated that "*[...] If somebody or bodies from the community wants to fork the source code and run with it, they have my blessing [...]*)". The variant was created on August 2017.
- "*The mainline project had made a radical shift from providing one set of features to a different, disjoint set of features. The maintainer had thought about it very well, but some users (including myself) had built their workflows around one of the old features. For this reason, I lifted that particular feature into a separate project that was also published under a different name to the package index.*" [R23]. The respondent also provided us a GitHub issue link, discussing the details. The issue was opened by the variant maintainer on July 2015 and was eventually closed on April 2018. The issue had 33 comments involving 17 participants.
- "*Mainline dropped support for a small subset of the code and asked for community support to create a fork to support that subset*" [R66].

A final technical motivation was technology. 7 respondents who selected technical, mentioned phrases related to *variant created to depend on a different technology*.

- "*Added support for Open Street Maps as an available map provider [. . .] mainline was not willing to accept this kind of contribution.*" [R8]. This was also ranked as a governance.

- "*The mainline wasn't updated to use .NET Core which I was using in my project, so I updated it*" [R29]
- "*[...] to keep the source code compatible with the language/-compiler version that we use (Swift / Xcode). [...] if the maintainer of the mainline is supporting a different one, then we could not compile our dependency anymore.*" [R54]

Governance. After technical, governance is the secondmost popular motivation, with responsiveness being the most prominent governance category. 18 of the 34 respondents who selected governance mentioned phrases related to *mainline was unresponsive to pull requests or issues for a long time*. Most of the respondents that ranked governance highly as their motivation, also ranked other options of motivations highly. Only 4 of the 34 ranked only governance.

- "*[They] had a series of commits that fixed functionality for newer PHP versions, but never made into a release. After waiting for more than a year for a release, a fork was done just to push a newer release into Composer/Packagist.*" [R21]
- "*We submitted some bug fixes [...], but didn't hear back from the maintainer for a while and needed to progress to meet our own goals so we forked. I followed up over email with the maintainer and he merged the patches about a month later, at which point we closed down and archived our fork and returned to using the mainline.*" [R15]. Merging back to the original corresponds to one the outcomes of variant forking reported in [2].
- "*[...] due to lack of response from mainline maintainer (more than months) and need of release. This lead to release of a new variant. [...] there is no intention to submit changes to mainline anymore (even when the first PR was merged into mainline after more than year).*" [R56]

The next governance motivation was feature acceptance. 15 respondents who selected governance, mentioned phrases related to *mainline hesitant to or not willing to accept feature*.

- *"TECHNICAL: Added support for Open Street Maps as an available map provider. GOVERNANCE: not exactly governance, but mainline was not willing to accept this kind of contribution"* [R8]. This was coded as technology in technical. The respondent also provided a GitHub PR link containing extra information. The PR included 45 conversations and 15 participants between June 2018 until March 2021 when it was closed.
- "*Mainline was not ready to accept those changes in part because the maintainers were not responsive. Since that time all of the issues have been dealt with and my variant is no longer needed, though the infrastructure for creating a new release of the variant remains in place in the event that it might be needed in the future.*" [R44]
- "*[...] even main repo maintainer was saying he is busy and please use your fork for thing X and Y. We don't know the exact reason why he stopped maintaining it and also did not allow us to maintain his repo*" [R89]. In one of the multiple choice answers, the respondent indicated that the variant was created through a community decision. The respondent also provided an extra link, revealing that three contributors from the community were interested in a couple of new features that were missing in mainline, but the mainline maintainer seemed busy. At the end, two members of the community took over the fork maintenance and introduced the missing features and advertised the additions in the readme.md file of the fork as well as in the issue.

Others. The most prominent motivation for others is supporting personal projects. 8 of the 24 respondents who selected others mentioned phrases related to *variant was created to support personal projects*.

- "*[The] maintainer was not interested in a PR that added functionality needed by a project I'm developing. [It] was considerably easier to add the logic into the [new] library than bolt it on.*" [R18]. This was ranked as technical, governance, and others. As we can see in the participant response we have phrases like "*adding logic*" (new features, technical), "*was not interested in a PR*" (feature acceptance, governance), and "*functionality needed by a project I'm developing*" (supporting personal projects, others).
- "*In Oct 2017 [. . .] has changed its API and these changes broke the mainline project. I used this project daily and needed to fix it ASAP. After quick fix I started to add my own features. [...] the mainline project has been fixed and refactored, but my other projects were already depending on my own fork.*" [R56]
- "*[...] to make sure that no matter what happen to the mainline repository, we can maintain source access to this library, which is an essential dependency of our project. . . .*" [R54]. This response is in line with Nyman et al. [1] who reported that forking provides a mechanism for safeguarding against despotic decisions by the project lead, who is thus guided in their actions to consider the best interest of the community.

The next motivation for others was supporting mainline, which was mentioned by 7 respondents who selected others:

- "*We have a fork that is the "main fork", which is [. . .], and the "development fork" is [FORKNAME]. In this case, our modeling tool [. . .] is only maintained as the fork [. . .] we synchronize everything between both forks while the [FORKNAME] one is mainly used to develop new features, which are then pushed as PRs to the main fork.*" [R61]
- "*Preparation of mainline pull requests. mainline repo should not be spammed by WIP PRs by students. Supervisors do coaching and try to improve the quality by the initial mainline pull request. [. . .] Keeping the PR open on the fork, reduces the number of PRs.*" [R73]
- "*We needed a repository for tracking our ideas to keep the number of issues of the main repository low.*" [R83]. The extra link that was provided revealed that the mainline and variant are owned by the same developer: "*this repository is used by [X] to make his ideas transparent. He collects the issues here to avoid flooding the "official" issue tracker. - Refined issues will be migrated to the official issue tracker*".

The next motivation detail for others was code quality. 3 respondents who selected others, mentioned phrases related to *mainline low code quality*.

- "*The mainline [...] was clearly written by someone who isn't a professional software engineer.*" [R63]
- "*The way the original was architected made this very challenging, so I ended up rewriting it instead of submitting a patch to the original.*" [R79]

Legal. The motivation of legal was least popular, corresponding to only 3 of the 105 respondents that indicated phrases related to closed source. Below we present their corresponding responses.

- "*[The] main reason is creating [an] open source and commercial product which has much more features*" [R7]. This motivation detail was also categorised as: (new features, technical) and (supporting personal projects, others).
- "*5 years ago the permissions model for GitHub and Travis is not what it is today. I wanted to use Travis but if I granted Travis access to my primary github account, it would have read access to all the github repos [...], which would expose private customer code. I forked the repo [but] the permissions model has evolved [and I] deleted the fork*" [R24].
- "*The founders of the mainline had been absent from the project for several years, but came back and booted the maintainers off and [...] shifted the project to a closed source.* " [R36]. The respondent provided a link with extra information showing that three of the maintainers that were booted from the original project and a fourth one from the community joined forces and are now maintaining the variant. The variant currently has over 739 stars, is used by 35 developers, has 101 pull requests and 195 issues.

B. Discussion and Implications

$RQ1$ mainly focused on determining the motivations for creating and maintaining variants, especially those that are actively being maintained in parallel with their mainline counterparts. We identified that the decision to create the variants is mostly initiated by individuals and less by the community. Our observations thereby confirm the findings in the literature. Our study also extends the state-of-the-art by providing fine-grained reasons for creating and maintaining variants relating to the reported motivations. Furthermore, our study revealed new reasons that have not been reported in literature (categorised as others in our survey) which include: 1) supporting the mainline, 2) variant supporting other personal projects, 3) localization purposes and 4) variant developers not trusting the code quality of the mainline. The reported findings are very useful to guide follow-up studies in investigating the co-evolution of mainline and variant projects.

Fig. 3 presented an overview of how the detailed motivations relate to who is involved in creating and maintaining the variants. The motivations majorly related to developers outside the core contributors of the mainlines (82%). We also observed quite a significant number of respondents (24%) reporting that the decision to create the variant was initiated by the community. We observed from the open-ended responses that, before the transition from social to variant fork, some variant maintainers engage with the mainline maintainers through discussions in issues and pull requests. This is inline with the Zhou et al. who reported that many variant forks start as social forks [8].

Besides the motivations for creating and maintaining variants, the respondents reported some interesting software reuse practices by the variants, like those categorized in the themes of: different goals, new features, customization, technology, supporting personal projects, supporting upstream, localization. A specific example of [R70] categorized in the different goals theme, stated that in the cryptocurrency world, all applications inherit code from the mother project bitcoin/bitcoin. Downstream applications also monitor their immediate upstream and other in the hierarchy for important updates like bug and security fixes as well as other specific updates. These cryptocurrency applications can be considered as a *software family* [21] or *software ecosystem* [28]. Variants are also likely to occur in other dedicated software ecosystems like Eclipse, Atom, Emacs, software library distributions for Java, C, C++, Python, Go, Ruby, and OS distributions for macOS, Linux, Windows, and iOS. To this end, our study opens up different research directions that can aim at deeply investigating different reuse practices in software families and software variants. A deeper understanding of these reuse practices can aid in developing tools that can support more effective software reuse.

***Summary – RQ1**: Many variant forks start as social forks. The decision to create/maintain the forks is either community-driven (contributing up to 24%) or individual (76%). The majority of the developers (82%) creating the forks are not maintainers of the mainlines. We identified 18 variant creation/maintenance motivation details categorized in the motivations of technical (accounting 58% of the responses), governance (24%), others (16%) and legal (2%). The detailed motivations in the others category are newly introduced since the social coding era.*

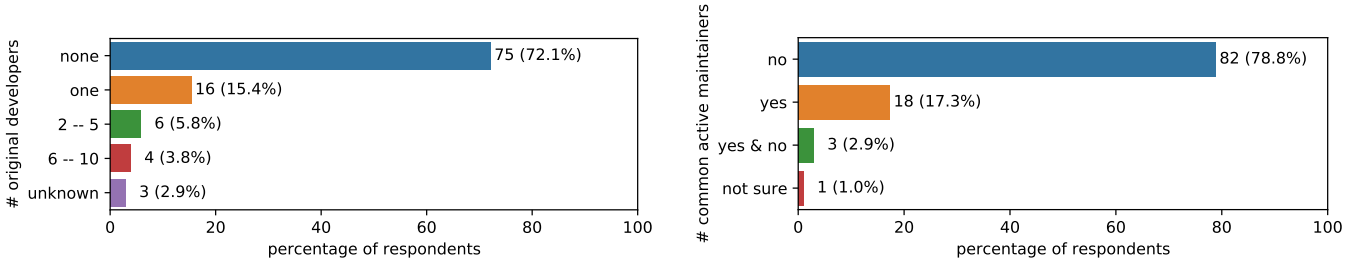
V. *RQ2:* How do variant projects evolve with respect to the mainline?

$RQ2$ aims to identify the impediments for co-evolution between the mainline and variant projects. This question lead to two specific focuses reflecting the *who* and the *how*, respectively. The *who* focus aimed at identifying who are the developers involved in maintaining variants. The *how* aimed to understand how variant forks evolve w.r.t. the mainline. As for $RQ1$ we refer to the responses using underlined, *italics* and [RN].

A. Results for the "who?" focus

To understand *who* is creating and maintaining variant forks, we asked two multiple-choice questions:
SQ^2_a: *How many of the original developers of the mainline maintained the variant in its first 6 months?*
SQ^2_b: *Do the variant and mainline have common active maintainers?*

Fig. 4(a) and Fig. 4(b) summarise the answers to SQ^2_a and SQ^2_b, respectively. The majority of the respondents chose the options of none for SQ^2_a (none of the creators of the variant were part of the mainline) and no for SQ^2_b (they do not have common active maintainers). This implies that



(a) SQ_a^2. How many mainline developers involved in creation of variant?

(b) SQ_b^2. Do the variant and mainline have common active maintainers?

Fig. 4: Who are the developers involved in creating and maintaining variants of a mainline project?

most developers involved in the creation and maintenance of variants are not core maintainers of the mainline from where the variant was forked. Fig. 3 reveals the difference in the numbers of participants who selected none for SQ_a^2 and no for SQ_b^2. Focusing at how responses of SQ_a^2–original developers? and SQ_b^2–common active maintainers? are associated, one can observe that most respondents that selected option none in SQ_b^2 went ahead to select option no in SQ_a^2. Other associations between responses of SQ_a^2 and SQ_b^2 can be observed as well.

Anecdotally, [R36] responded to SQ_a^2 that 6 – 10 developers from the mainline were involved in the creation of the variant, and responded to SQ_b^2 with the option yes & no –*"They used to have common maintainers in the early stages of the variant, but now the projects have technically diverged away from each other, there are no more common maintainers"*. Respondents [R51] and [R57] selected for SQ_a^2 the options 6 – 10 and 2 – 5 respectively, while selecting the option no for SQ_b^2. This implies that at least two maintainers involved in fork creation are not (or no longer) contributing to the mainline.

Summarising our observations for SQ_a^2 and SQ_b^2, we conclude that **variant forks are created and maintained by developers different from those in the mainline counterparts.** This observation concurs with the earlier findings of Businge et al. [10].

*B. Results for the "*how?*" focus*

To understand *how* variant forks evolve w.r.t. the mainline, we asked two additional questions:
SQ_c^2: *Do the variant forks and the upstream still discuss the main directions of the project?*
SQ_d^2: *Do the variant developers integrate changes to and from the upstream repository?*

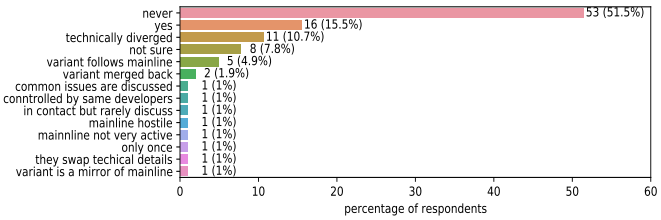
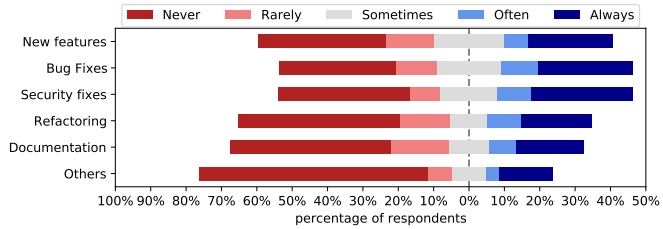


Fig. 5: SQ_c^2 Do the variant forks and the mainline still discuss the main directions of the project?

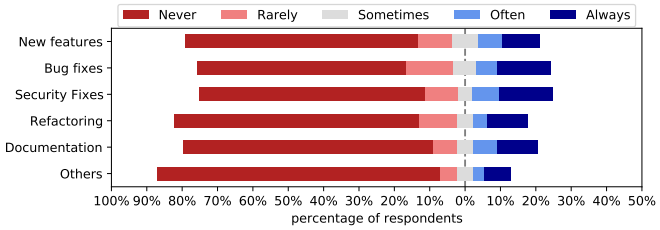
For SQ_c^2 we presented four multiple choice answer options, corresponding to the first four answers reported in Fig. 5, gathering the highest number of responses. We allowed respondents to provide an open-ended answer if they felt that their choice was not among the four proposed options. The open-ended answers were coded into themes (listed in Fig. 5 from variant follows mainline→to variant is a mirror of the mainline). Fig. 5 shows that more than half of the respondents chose the option of never (corresponding to: *no, there has never been any discussion since the creation of the variant*). Even if there was some discussion, 10.7% of the respondents signal that they technically diverged (corresponding to: "*They used to discuss but not anymore since the projects have technically diverged from each other*"). The open-ended answers also revealed variant responses that do not discuss the directions of the project, like mainline hostile to variant, not very active, in contact but rarely discuss and only once.

An explanation for the high number of variant developers that do not discuss with the mainline developers about the project direction can be derived from the findings of SQ_a^2 and SQ_b^2. The majority of the variants are created and maintained by developers that are not core developers of the mainline. Also, most of the *motivation details* in $RQ1$ could explain the high numbers of never. For example we observed that the majority of the variants in the *motivation details* category of different goals, unmaintained features in the mainline, those having issues with the mainline responsiveness, those whose features will not be accepted by the mainline (feature acceptance), selected never in SQ_c^2. We conclude that **the reasons for the majority of variant forks not to discuss the project directions with the mainline could be attributed to a diverging range of motivations for creating the variant as well as to the variant creators not being part of the mainline's core development team.**

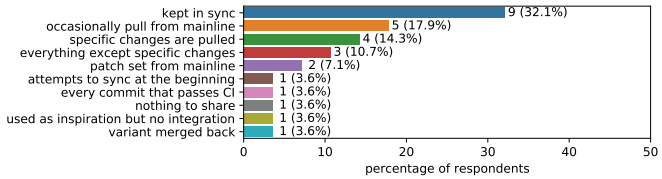
Anecdotally, 5 respondents indicated phrases related to variant follows mainline. Respondent [R77] indicated that "*in the crypto world, the mainline inherits changes from BITCOIN, for example, security commits, and the variant merges those changes in. So the variant is very interested in every change in the Mainline. However, the variant must maintain the specific new features that we added separately, and the Mainline is not interested in helping the Variant do this.*" We also observed two interesting cases where the variants merged back to the



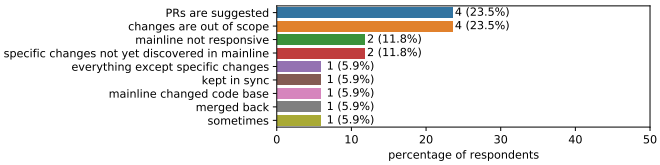
(a) Code integration back from the mainline



(b) Code integration from variant to mainline



(c) integration back from the mainline (coded themes)



(d) integration from variant to mainline (coded themes)

Fig. 6: *RQ2:* How do variant projects evolve with respect to the mainline?

mainline. This is in line with Robles and González-Barahona [2] who reported that one of the outcomes of forking is the fork merging back.

For SQ^2_d we asked respondents two closed-ended questions: (1) *How often do the maintainers of the variant integrate the following types of changes from the mainline?*; and (2) *How often do the maintainers of the variant integrate the following types of changes into the mainline?*. We provided Likert-scale options for the two questions. We presented optional follow-up questions with open-ended answers, for each of the two questions, allowing respondents to provide extra information.

Fig. 6(a) presents the answers from the respondents on what they value most when integrating changes back from the mainline. The highly scored changes are bug fixes and security fixes. One can observe that most respondents were leaning towards the negative side of the Likert scale, implying hat most variants are not interested in integrating changes from the mainline. Fig. 6(b) focuses on integrations from variants towards the mainline. We observe a similar trend to Fig. 6(a), with an even more pronounced negative inclination.

Fig. 6(c) and Fig. 6(d) present the coded themes of the extra information gathered from the open-ended answers corresponding to the results in Fig. 6(a) and Fig. 6(b), respectively. Fig. 6(c) summarises the results of 28 respondents who provided the extra information, while Fig. 6(d) summarises the results of only 17 respondents, most likely because most variants do not submit changes to the mainline. The most prominent response in Fig. 6(c) was related to being kept in sync, signaling the desire of variants to keep in sync with the changes made in the mainline. The next prominent response was related to occasionally pull from mainline implying that variants from time to time pull changes made in the mainline. Some respondents mentioned phrases related to specific changes are pulled; for example, [R63] indicated that "*It's mostly changes that make the library for specific iRobot Roomba models (new ones for example)*". Other respondents mentioned phrases related to everything except specific changes; for example, [R48] mentioned that "*All non-compiler specific changes are pulled*". In Fig. 6(d) there were two prominent answers: PRs are suggested, for example, "*Made PRs with changes but those have just been ignored. They're still "open" with 0 comments from the mainline dev*" [R67]. The other prominent answer is changes are out of scope, for example, "*We use this as a dependency in another project [. . .] which is often diverging from the language version of the mainline, so there is little reason for us to push this to mainline*" [R54].

C. Discussion and Implications

The results of $RQ2$ revealed that variants are created and maintained by developers that are not core developers of the mainline. We also observed limited interaction between the mainline and its variant(s). Although we found there is little code integration, the integration from mainline to variant is more frequent than from variant to mainline. Our study confirms and extends the findings of Businge et al. [10]: we provide concrete reasons relating to little integration between the mainline and variants that include:
1) *technical divergency*: variants and mainlines are offering different goals, implementing different technologies, variant is maintaining a part of the mainline that is frozen;
2) *governance disputes*: mainlines are unresponsive to pull requests and issues from the variants and mainlines not willing or hesitant to accept some features from the variants. One respondent also reported that mainline is actively very hostile to variants as a result of mainline's license changing to proprietary;
3) *distinct developers*: Another reason for the lack of code integration is because most of the variants are maintained by developers that are not part of the core team of the mainline. Furthermore, we observed that a few mainline–variant pairs that do interchange code are mostly interested in patch sets (security fixes and bug fixes).

Although maintenance and collaboration have improved through dedicated tooling, especially through distributed ver-

sion control systems like Git [29] and transparency mechanisms on social coding platforms like GitHub [30], these tools are only ideal for social forks which aim to sync all the changes between repositories. For example, code integration using pull requests and git tools like merge / rebase may not be the best when integrating changes in between mainline and variant forks since they involve syncing upstream / downstream with all changes missing in the current branch.

This study reveals that some variant maintainers are only interested in integrating commits with specific changes. A suitable integration mechanism would be commit cherry picking since the developers can choose the exact commits they want to integrate. However, GitHub's current setup does not make it easy to identify commits to cherry-pick without digging through the branch's history to identify relevant changes since the last code integration. Additionally, even though the variants have diverged from their mainlines, we do believe that since they share common code, some of the common code may go through maintenance to perform some bug and security fixing. Since these mainline–variant repository pairs are being maintained by uncommon developers, chances are that these fixes could be missed or they could be fixed at different times by different developers, resulting in duplicated effort.

Our findings are very relevant to code integration tool builders between mainline and variants to prioritise certain categories of mainline–variant pairs by targeting specific changes. Ideally, a tooling would help identify possibly important fixes in commits and recommend these commits to mainline or variant developers to support a more efficient reuse. Some promising studies in this direction have focused on providing the mainline with facilities to explore non-integrated changes in forks to find opportunities for reuse [31] and cross-fork change migration [32]. More experimental ideas have focused on virtual product-line platforms for unified development of multiple variants of a project [33]–[37].

Summary–RQ2*: Variant forks do not usually interact with the mainline during their co-evolution. The lack of interaction could be attributed to a variety of reasons including: (i) technical divergence, where variants and mainlines are offering different features or implementing different technologies having nothing to share; (ii) governance disputes, where mainlines are unresponsive to the requests from community and also uninterested in some features suggested by the community; (iii) distinct development teams that no longer interact; (iv) diverging licenses, where the mainline variant has changed the license and integration is no longer possible. As a result of these divergences, it is likely that important security or patch updates could be missed or are duplicated.*

VI. Threats to Validity

Construct validity. The response categories for the closed questions in the survey originated from a thorough literature review. The questions were carefully phrased to avoid biasing the respondent towards a specific answer. We validated the questions by consulting seven colleagues from three different universities and through trial runs of the survey with seven participants. Social desirability bias may also have influenced the answers [38]. To mitigate this issue, we informed participants that the responses would be anonymous and evaluated in a statistical form.

Internal validity. We used an open coding process to classify the participants responses received from open-ended questions. The coding process is known to lead to increased processing and categorization capacity at the loss of accuracy of the original response. To alleviate this issue lack of accuracy, we allowed more than one code to be assigned to the same answer.

Generalizability. Our study is limited to variants of mainline repositories that are hosted on GitHub. We do not claim that our findings generalize to other social coding platforms. In addition, the set of participants we interviewed corresponds to those who decided to make their e-mail public and who accepted to take part in our study. As such, they are not *de facto* representative of all maintainers of variant forks.

VII. Conclusions

Thanks to social coding platforms like GitHub, software reuse through forking to create variant projects is on the rise. We carried out an exploratory study with 105 maintainers of variants, focusing on answering two key research questions:

1) *Why do developers create and maintain variants on GitHub?* We observed that the motivations reported by studies carried out in the the pre-GitHub era, still hold. We identified 18 motivation details for variant creation and maintenance, categorized in the motivations of *technical* (58% of the responses), *governance* (24%), *others* (16%) and *legal* (2%). Some of these motivations are newly introduced in the social coding era.

2) *How do variants projects evolve with respect to the mainlines?* We have found that there is little interaction between the variants and their mainlines during the co-evolution and reported possible impediments to the lack of interaction. These include: (i) technical (i.e., diverging features), where variants and mainlines are offering different goals or implementing different technologies having nothing to share; (ii) governance (i.e., diverging interests), where mainlines are unresponsive to the requests from community and also uninterested in some features suggested by the community; (iii) legal (e.g., diverging licenses), where the mainline variant has changed the license and integration is no longer possible.

Our findings are very useful to guide follow-up studies in investigating the co-evolution and reuse practices between mainline and variants. A deeper understanding of these practices can aid code integration tool builders in developing tools to support more effective software reuse between mainline projects and their variant forks.

Acknowledgment

This work is supported by the joint FWO-Vlaanderen and F.R.S.-FNRS Excellence of Science project SECO-ASSIST under Grant number O.0157.18F- RG43.

REFERENCES

[1] L. Nyman, T. Mikkonen, J. Lindman, and M. Fougère, "Perspectives on code Forking and Sustainability in open source software," in *Open Source Systems: Long-Term Sustainability*, 2012, pp. 274–279.

[2] G. Robles and J. M. González-Barahona, "A comprehensive study of software forks: Dates, reasons and outcomes," in *Open Source Systems: Long-Term Sustainability*, 2012, pp. 1–14.

[3] R. Viseur, "Forks impacts and motivations in free and open source projects," *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 2, February 2012.

[4] L. Nyman and J. Lindman, "Code forking, governance, and sustainability in open source software," *Technology Innovation Management Review*, vol. 3, pp. 7–12, January 2013.

[5] L. Nyman and T. Mikkonen, "To fork or not to fork: Fork motivations in SourceForge projects," in *Open Source Systems: Grounding Research*, 2011, pp. 259–268.

[6] J. Gamalielsson and B. Lundell, "Sustainability of open source software communities beyond a fork: How and why has the libreoffice project evolved?" *Journal of Systems and Software*, vol. 89, pp. 128 – 145, 2014.

[7] G. Gousios, M. Pinzger, and A. van Deursen, "An exploratory study of the pull-based software development model," in *International Conference on Software Engineering*, 2014, pp. 345–355.

[8] S. Zhou, B. Vasilescu, and C. Kästner, "How has forking changed in the last 20 years? a study of hard forks on GitHub," in *International Conference on Software Engineering*. ACM, 2020, pp. 268–269.

[9] C. Sung, S. K. Lahiri, M. Kaufman, P. Choudhury, and C. Wang, "Towards understanding and fixing upstream merge induced conflicts in divergent forks: An industrial case study," in *International Conference on Software Engineering*. ACM, 2020, pp. 172–181.

[10] J. Businge, M. Openja, S. Nadi, and T. Berger, "Reuse and maintenance practices among divergent forks in three software ecosystems," *Journal of Empirical Software Engineering*, 2021.

[11] A. S. Laurent, *Understanding Open Source and Free Software Licensing*. O'Reilly Media, 2008.

[12] B. B. Chua, "A survey paper on open source forking motivation reasons and challenges," in *Pacific Asia Conference on Information Systems*, 2017.

[13] J. Dixion, "Different kinds of open source forks: Salad, dinner, and fish," https://jamesdixon.wordpress.com/2009/05/13/different-kinds-of-open-source-forks-salad-dinner-and-fish/, 2009.

[14] N. A. Ernst, S. M. Easterbrook, and J. Mylopoulos, "Code forking in open-source software: a requirements perspective," *ArXiv*, vol. abs/1004.2889, 2010.

[15] L. Nyman, "Hackers on Forking," in *The International Symposium on Open Collaboration*, 2014, pp. 1–10.

[16] E. S. Raymond, *The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary*. O'Reilly Media, Inc., 2001.

[17] P. Bratach, "Why Do Open Source Projects Fork?" https://thenewstack.io/open-source-projects-fork/, 2017.

[18] J. Jiang, D. Lo, J. He, X. Xia, P. S. Kochhar, and L. Zhang, "Why and how developers fork what from whom in GitHub," *Empirical Softw. Engg.*, vol. 22, no. 1, pp. 547–578, Feb. 2017.

[19] D. A. Wheeler, ""forking"," https://dwheeler.com/oss_fs_why.html#forking, 2009, revised as of July 18, 2015.

[20] T. de Raadt, "Theo de Raadt's dispute w/ NetBSD," https://zeus.theos.com/deraadt/coremail.html, 2006, retrieved October 2021.

[21] J. Businge, M. Openja, S. Nadi, E. Bainomugisha, and T. Berger, "Clone-based variability management in the Android ecosystem," in *International Conference on Software Maintenance and Evolution*. IEEE, 2018, pp. 625–634.

[22] F. Uwe, *An Introduction to Qualitative Research*. London: Sage Publications, 2014.

[23] J. Businge, A. Decan, A. Zerouali, T. Mens, and S. Demeyer, "An empirical investigation of forks as variants in the npm package distribution," in *The Belgium-Netherlands Software Evolution Workshop*, ser. CEUR Workshop Proceedings, vol. 2912. CEUR-WS.org, 2020.

[24] J. Businge, M. Openja, D. Kavaler, E. Bainomugisha, F. Khomh, and V. Filkov, "Studying Android app popularity by cross-linking GitHub and Google Play store," in *International Conference on Software Analysis, Evolution and Reengineering*, 2019, pp. 287–297.

[25] J. Businge, S. Kawuma, E. Bainomugisha, F. Khomh, and E. Nabaasa, "Code authorship and fault-proneness of open-source Android applications: An empirical study," in *PROMISE*, 2017.

[26] T. Zimmermann, "Card-sorting: From text to themes," in *Perspectives on Data Science for Software Engineering*. Elsevier, 2016, pp. 137–141.

[27] D. Garrison, M. Cleveland-Innes, M. Koole, and J. Kappelman, "Revisiting methodological issues in transcript analysis: Negotiated coding and reliability," *The Internet and Higher Education*, vol. 9, pp. 1–8, 03 2006.

[28] A. Decan, T. Mens, and P. Grosjean, "An Empirical Comparison of Dependency Network Evolution in Seven Software Packaging Ecosystems," *Empirical Softw. Engg.*, vol. 24, no. 1, pp. 381–416, Feb. 2019.

[29] C. Rodríguez-Bustos and J. Aponte, "How distributed version control systems impact open source software projects," in *Working Conference on Mining Software Repositories*. IEEE, 2012, pp. 36–39.

[30] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository," in *Conference on Computer Supported Cooperative Work*, 2012, pp. 1277–1286.

[31] L. Ren, S. Zhou, and C. Kästner, "Poster: Forks insight: Providing an overview of GitHub forks," in *The International Conference on Software Engineering: Companion (ICSE-Companion)*, 2018, pp. 179–180.

[32] L. Ren, "Automated patch porting across forked projects," in *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 1199–1201.

[33] M. Antkiewicz, W. Ji, T. Berger, K. Czarnecki, T. Schmorleiz, R. Lämmel, u. Stănciulescu, A. Wąsowski, and I. Schaefer, "Flexible Product Line Engineering with a Virtual Platform," in *Companion of the International Conference on Software Engineering*, 2014, pp. 532–535.

[34] S. Fischer, L. Linsbauer, R. E. Lopez-Herrejon, and A. Egyed, "Enhancing clone-and-own with systematic reuse for developing software variants," in *International Conference on Software Maintenance and Evolution*, 2014, pp. 391–400.

[35] L. Montalvillo and O. Díaz, "Tuning GitHub for SPL development: Branching models & repository operations for product engineers," in *International Conference on Software Product Lines*, 2015, pp. 111–120.

[36] J. Rubin and M. Chechik, "A framework for managing cloned product variants," in *International Conference on Software Engineering*. IEEE, 2013, pp. 1233–1236.

[37] S. Stanciulescu, T. Berger, E. Walkingshaw, and A. Wasowski, "Concepts, operations, and feasibility of a projection-based variation control system," in *International Conference on Software Maintenance and Evolution (ICSME)*, 2016, pp. 323–333.

[38] A. Furnham, "Response bias, social desirability and dissimulation," *Personality and Individual Differences*, vol. 7, no. 3, pp. 385–400, 1986.