An Exploratory Study to Find Motives Behind Cross-platform Forks from Software Heritage Dataset

Avijit Bhattacharjee  
University of Saskatchewan

Sristy Sumana Nath  
University of Saskatchewan

Shurui Zhou  
Carnegie Mellon University

Debasish Chakroborti  
University of Saskatchewan

Banani Roy  
University of Saskatchewan

Chanchal K. Roy  
University of Saskatchewan

Kevin Schneider  
University of Saskatchewan

ABSTRACT
The fork-based development mechanism provides the flexibility and the unified processes for software teams to collaborate easily in a distributed setting without too much coordination overhead. Currently, multiple social coding platforms support fork-based development, such as GitHub, GitLab, and Bitbucket. Although these different platforms virtually share the same features, they have different emphasis. As GitHub is the most popular platform and the corresponding data is publicly available, most of the current studies are focusing on GitHub hosted projects. However, we observed anecdotal evidences that people are confused about choosing among these platforms, and some projects are migrating from one platform to another, and the reasons behind these activities remain unknown. With the advances of Software Heritage Graph Dataset (SWHGD), we have the opportunity to investigate the forking activities across platforms. In this paper, we conduct an exploratory study on 10 popular open-source projects to identify cross-platform forks and investigate the motivation behind. Preliminary result shows that cross-platform forks do exist. For the 10 subject systems used in this study, we found 81,357 forks in total among which 179 forks are on GitLab. Based on our qualitative analysis, we found that most of the cross-platform forks that we identified are mirrors of the repositories on another platform, but we still find cases that were created due to preference of using certain functionalities (e.g., Continuous Integration (CI)) supported by different platforms. This study lays the foundation of future research directions, such as understanding the differences between platforms and supporting cross-platform collaboration.

CCS CONCEPTS
• Software and its engineering → Open source model.

KEYWORDS
OSS, Collaboration, Social coding, Cross-platform forks

1 INTRODUCTION
Fork-based development allows developers to start development from an existing codebase while having the flexibility and independence to make changes [13, 18]. Prior work studied the fork-based development mechanism from different perspectives, such as the collaboration efficiencies of software teams using forks [19], pull request management processes [13], sustainability of open-source communities [17], and different types of forks on GitHub [16]. However, most of these studies focus on GitHub. Although GitHub is the most popular platform that supports fork-based development, other reasonably popular platforms support the fork-based development mechanism as well, such as BitBucket and GitLab. BitBucket announced that till April 2019 they reached 10 million registered users and over 28 million repositories1. Similarly, GitLab is used by more than 100,000 organizations and its open-source codebase is contributed by 2,988 developers2. These platforms have different functionality emphasis such as DevOps solution, self-managed hosting, automatic code change monitoring, and Continuous Integration/Continuous Deployment (CI/CD). Anecdotal evidences show that people are struggling about which platform to choose for software development, and we observed projects migrating from one platform to another, such as the project Vim [11] was hosted on GitHub, and then was moved to GitLab because of the CI feature of GitLab and was renamed as mg-vim [4].

Prior work on understanding different types of forks defined two main types of active forks on GitHub [19]: social fork, which is created to contribute back to the main repository, and hard fork, which is aiming for superseding or replacing the original project. With the advances of the SWHGD [14, 15], which brings different social coding platforms data under single roof including GitHub, GitLab, Debian, and PyPI, we have the opportunity to study the forking activity among different social coding platforms. Specifically, we would like to understand why developers prefer one platform

1https://en.wikipedia.org/wiki/Bitbucket
2https://about.gitlab.com/company/
over another, what are the features and limitations of platforms that drive the migration of code, and see if there is space of improvement to current fork-based development mechanism.

In a nutshell, this paper conducts an exploratory empirical study using mixed methods (qualitative and quantitative) on ten popular projects to address the following two research questions:

- RQ1. How often do cross-platform forks happen?
- RQ2. What are the motives behind cross-platform forks?

The contributions of this paper are as follows. (1) We conducted the first-ever empirical study on cross-platform forks, (2) we propose an algorithm that automatically detects cross-platform forks, and (3) we provide a foundation of future research directions towards finding active cross-platform forks automatically and tracking their activities.

2  **RQ1: HOW OFTEN DO CROSS-PLATFORM FORKS HAPPEN?**

We would like to understand the frequency of cross-platform forks. To achieve our goal, we designed a two-step quantitative study to automatically detect the cross-platform fork candidates. First, we tried to detect forks of a target origin from the SWHGD. Second, we filtered out the forks of different platforms to find cross-platform forks. From our preliminary analysis with 10 representative projects from GitHub, we found 179 cross-platform forks for our subject systems.

2.1  **Subject Systems and Data Collection**

*SWHGD.* The *origin* table of the SWHGD [15] contains URLs of the repositories crawled. After some time interval, URLs of the *origin* table are visited again to capture new updates. These visits to an origin URL is stored in the *origin_visit* table. The *snapshot* table contains information about snapshots of an origin URL. Branches associated with each snapshot are stored in the *snapshot_branches* table. The *snapshot_branch* table stores the commits each branch of a snapshot points to. Finally, all commits of an origin URL are stored in the *revision* table which contains information about individual commits (i.e. author, committer, date, and message).

*Sampled dataset.* We randomly selected 10 projects on GitHub with a different number of forks using GitHub API [12]. To not bias our analysis by practices applied by the largest or by many small projects, we sampled 5 very frequently forked projects and 5 moderately forked projects, as shown in Table 1.

*Experiment setup.* We downloaded the full version of Software Heritage Dataset [1] in a VM with 24-core processor and 64GB RAM. The physical machine has dual Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz CPUs, with a total of 28 cores (56 threads). We used a local PostgreSQL server to load the instance. During the indexing process, we omitted the unnecessary tables to save time. We used Python code to run SQL queries on the PostgreSQL instance using SqlAlchemy [8] module. We uploaded the extracted data [3] and used Python program [6] to Zendoo.

<table>
<thead>
<tr>
<th>No</th>
<th>URL (<a href="https://github.com">https://github.com</a>)</th>
<th>Name</th>
<th>No. of Forks (From GitHub)</th>
<th>No. of forks (From SWHGD)</th>
<th>Time(Hr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>alicex/TextBlob</td>
<td>TextBlob</td>
<td>901</td>
<td>515</td>
<td>27min</td>
</tr>
<tr>
<td>2</td>
<td>/explosion/spacy</td>
<td>spacy</td>
<td>2700</td>
<td>756</td>
<td>45min</td>
</tr>
<tr>
<td>3</td>
<td>/flutter/flutter</td>
<td>Flutter</td>
<td>1080</td>
<td>483</td>
<td>8hr 7min</td>
</tr>
<tr>
<td>4</td>
<td>vim/vim</td>
<td>Vim</td>
<td>2600</td>
<td>1339</td>
<td>12hr</td>
</tr>
<tr>
<td>5</td>
<td>neovim/neovim</td>
<td>Neovim</td>
<td>2600</td>
<td>2284</td>
<td>5hr 15min</td>
</tr>
<tr>
<td>6</td>
<td>/bitcoin/bitcoin</td>
<td>Bitcoin</td>
<td>25000</td>
<td>10228</td>
<td>20hr 17min</td>
</tr>
<tr>
<td>7</td>
<td>/scikit-learn/scikit-learn</td>
<td>scikit-learn</td>
<td>19000</td>
<td>10551</td>
<td>10hr 31min</td>
</tr>
<tr>
<td>8</td>
<td>/facebook/react-native</td>
<td>ReactNative</td>
<td>18800</td>
<td>11785</td>
<td>16hr 5min</td>
</tr>
<tr>
<td>9</td>
<td>/nodejs/node</td>
<td>Node.js</td>
<td>16000</td>
<td>15510</td>
<td>30hr 12min</td>
</tr>
<tr>
<td>10</td>
<td>/tensorflow/tensorflow</td>
<td>TensorFlow</td>
<td>79400</td>
<td>27908</td>
<td>21hr 8min</td>
</tr>
</tbody>
</table>

**Algorithm 1 SQL queries to retrieve forks of a target URL**

1. `select id as origin_id from origin where url = :target_url`
2. `select snapshot_branch.target as interval_commits from origin_visit, snapshot_branches, snapshot_branch where snapshot_branch.object_id = snapshot_branches.branch_id and snapshot_branches.snapshot_id = origin_visit.snapshot_id and origin = :origin and status = 'full'`
3. `select distinct id as child_commits from revision_history where parent_id = ANY(interval_commits)`
4. `select url, snapshot_branch.target as rev from origin, origin_visit, snapshot_branches, snapshot_branch where origin.id = origin_visit.origin and origin_visit.snapshot_id = snapshot_branches.snapshot_id and snapshot_branches.branch_id = snapshot_branch.object_id and snapshot_branch.target = ANY(interval_commits + :child_commits) and url != :target_url`

2.2  **Research Method**

As SWHGD uses a deduplication technique to avoid storage of the same snapshot for multiple forks, all the forks including the original repository points to the same snapshot which allows us to detect forks over multiple platforms. To achieve generalization, SWHGD only stores project-specific data (commit, directory, release, snapshot, branch), and skipped platform dependent data such as issues, pull requests, forks, and others. Therefore, in order to detect forks, we need to rely on the comparison of commit history information and corresponding platform URL to find the cross-platform fork candidates. Specifically, we detect cross-platform forks by analyzing the following tables from the SWHGD: *origin, origin_visit, snapshot_branches*, and *snapshot_branch*.

**Step 1: Finding forks from commit information.** As shown in Algorithm 1, we detected forks of a repository from the SWHGD step by step. In (1), id of the target origin URL is retrieved. In (2), we extracted interval commits (last commit of each branch during a snapshot stored in *snapshot_branch* table) of an origin URL by joining *origin_visit, snapshot_branches, snapshot_branch* tables. (3) is to find child commits of the origin interval commits by searching all commits in *revision_history* table. Now, we need to map all the

---

commits to their corresponding URL so that we can get the forks for our target URL. Before going to (4), we merged the interval commits and their child commits from (2), (3) into a target commit list. Finally, the query of (4) is used to map target commits with their corresponding URL by joining origin, origin_visit, snapshot_branches, and snapshot_branch tables where commits are in target commit list and URL is different from our target URL. For more details of the approach, we would recommend the interested readers to look into our code made available via Zendoo [6]. We only selected commits pointed by snapshot_branch table to find forks. The reason behind this choice is when we select all commits of an origin URL, the query to map commits to their URL takes a long time which hinders our approach to investigate motives behind cross-platform forks. As the commit to URL map query of (4) of Algorithm 1 joins four large tables, the increase in number of commits increases the time required for the query. For example, when we tried to get forks for spaCy project using all the commits, we have found 802 forks which is a little bit more compared to the numbers reported in Table 1. However, the time taken for the algorithm is 12 hours where considering interval commits from the snapshot_branch table takes 45 minutes to find 756 forks.

Step 2: Finding cross-platform forks. In the second step, we used the forks extracted from the previous step to detect forks of a project on different platforms. While extracting forks, we used the commit information to detect whether a repository is a fork of a target origin. Hence, we expect to get forks from platforms different than GitHub, as our subject systems are from GitHub. Therefore, we filtered out the forks where URL has https://github.com as a prefix of their URL. We reported the number of cross-platform forks in Figure 2.

2.3 Findings

In Table 1, we presented URL of the repositories, with the number of forks collected from GitHub and our approach. We also reported time taken by our program to search forks from 88,290,221 repositories stored in the origin table of the SWHGD. In Figure 1, we plotted the percentage of forks we have been able to detect compared to the data from GitHub. We can see that for Node.js and Neovim our approach detected above 87% forks reported in GitHub. For TextBlob, Vim, scikit-learn, and ReactNative our approach detected approximately 56%-63% forks. For spaCy, Vim, Bitcoin and TensorFlow the percentage of forks is below 50%. As we only used interval commits stored in the snapshot_branch table and their child commits from the revision_history table, the number of detected forks depends on how many forks points to the interval commits and their child commits.

Figure 2 plots the number of extracted forks from GitLab by our approach. For Bitcoin and Node.js library, we found 57 forks from GitLab which is the highest among all our subject systems. TensorFlow repository got the 2nd position among 10 subject systems by getting 23 forks from GitLab. ReactNative, scikit-learn and Vim got 12-13 forks from GitLab. Whereas spaCy, TextBlob have no forks from GitLab. In the following section, we conducted a manual analysis to identify the motives behind the detected GitLab forks.

3 RQ2: WHAT ARE THE MOTIVES BEHIND CROSS-PLATFORM FORKS?

In 2018, when Microsoft acquisition of GitHub took place, GitLab reported a spike of importing GitHub projects to their platform [1]. As many open-source developers believe that Microsoft is not open-source friendly, the acquisition of GitHub might have motivated some of the developers to move their projects to GitLab. This is one of the motivations which caused cross-platform forks. Still, there might be other reasons for cross-platform forks. To better understand the motivation behind cross-platform forks, we conducted a qualitative study to find motives behind switching between platforms.
3.1 Research Method
To answer RQ2, two of the authors of this paper manually compared cross-platform forks with their origin repositories. For comparison between two repositories, title, description, commit history, and username of the fork owner are considered. To remove the subjectivity issue, two authors individually noted above-mentioned properties for studied 100 GitLab projects and their corresponding origin repositories. Later, a third author analyzed the notes and categorized them into five different categories which we described briefly below with examples.

3.2 Findings
We randomly sampled 100 cross-platform fork candidates and manually analyzing the corresponding information. We classified the forks in five categories based on their activity history.

3.2.1 Forks created using mirroring feature. During our manual analysis, we found that most of the forks are created using the mirror feature of GitLab. The mirror feature in GitLab crawls the original GitHub repository every five minutes to keep the forked repository in sync. For example, when we investigate cross-platform forks for Bitcoin [2], we find a project SeppPepper/bitcoin [7] in GitLab where the description section mentions the project is a mirror of the original Bitcoin project. Also, it mentions that the project was updated 17 minutes ago which means GitLab was crawling it with some time interval. Later, we found from GitLab documentation that their mirror can be used to push or pull from or to a remote repository which is automatically updated in 5-minute intervals. This feature is a key for a significant number of cross-platform forks.

3.2.2 Forks owner are also a contributor of the original project. Some cross-platform forks are owned by one of the contributors of the original repository and the last commit in the cross-platform fork belongs to the origin repository, whose committer is the fork owner. We verified that this commit belongs to the same person by comparing the usernames in GitHub and GitLab. For example, we found a project vectorci/tensorflow [10] which is a cross-platform fork of tensorflow/tensorflow [9] on GitHub, where the owner is vectorci, who is also the committer of the last commit in GitLab which belongs to the original project. We found the account under username vectorci does not exist anymore in GitHub. We suspect that the developer might want to leave GitHub and make GitLab as their new version control hosting platform. Therefore, to preserve their contribution to the main TensorFlow repository, the developer forked it to GitLab.

3.2.3 Forks of which title renamed after forking. Another kind of cross-platform forks has a different name compared to the original project. During our manual analysis, we came across these kind of forks. For example, we found a repository on GitLab named onyx-gameboost [5] in description section of which says "Onyx GameBoost Dev Repo". From the SWHGD we detected this repository as a cross-platform fork of the TensorFlow [9] repository, as all the commits in the GitLab copy are originated from the original TensorFlow repository. From observing this type of cross-platform forks, we can possibly conclude that some forks are created and renamed for some purpose. Later, developers did not go through their intentions.

3.2.4 Forks intended for an individual copy. We found almost 70% of the cross-platform forks are just merely for having a copy in different social coding platform accounts of the developers. As different social coding platforms provide different facilities, developers prefer to have copies of their own in different platforms for the projects they are interested in. All the cross-platform forks we found are one to four years older and they are inactive. Therefore, we can safely conclude that they are for keeping an individual copy.

3.2.5 Forks intended to continue development with another social coding platform. From our analysis, we managed to retrieve one worth mentioning example. We found a repository named mg-vim [4] on GitLab which is a cross-platform fork of original Vim [11] repository. We found a commit which is absent in the original repository. From investigating the commit, we detected that it was due to adding CI/CD support provided by GitLab for the repository. Additionally, we studied about CI/CD support provided by different social coding platforms. GitLab offers free CI/CD support developed by them. On the contrary, GitHub allows third-party CI/CD tools to be integrated on their platform but does not provide any support themselves when the fork created, although recently Github Actions started supporting free CI/CD for public repositories. Further investigation and interviews with the developers can reveal more concrete reasons over cross-platform forks.

4 THREATS TO VALIDITY
One might question the generality of our findings since we only used 10 projects from GitHub. In order to at least partially mitigate this issue, we carefully chose projects of diverse varieties and thus our findings could be generalizable to some extent. Of course analyzing more data might provide more insights for forking. However, by manually investigating the sampled projects, we reach the saturation and the findings already seem interesting and important.

5 CONCLUSION AND FUTURE WORK
To sum up, we conducted a mixed-method study (quantitative and qualitative) to identify cross-platform forks using the SWHGD. For our experiment, we started with 10 popular GitHub repositories and then detected their forks among all the platforms. Based on our qualitative analysis, we reported five types of cross-platform fork scenarios with concrete examples. In future, we plan to conduct a large scale analysis along with interviews with the developers to understand the pros and cons of different platforms, help stakeholders to make deliberate decisions on choosing code hosting platforms, and help platform providers to better facilitate social coding activities. Another research direction could be designing methods to track activities among cross-platform forks and generate a larger overview for cross-platform forks as a whole along with the idea proposed by Zhou et al. [18].

Acknowledgments: This work is supported in part by the Canada First Research Excellence Fund (CFREF) under the Global Institute for Water Security (GIWS).

---

1[https://docs.gitlab.com/ee/user/project/repository/repository_mirroring.html](https://docs.gitlab.com/ee/user/project/repository/repository_mirroring.html)

2[https://usersnap.com/blog/gitlab-github/](https://usersnap.com/blog/gitlab-github/)
An Exploratory Study to Find Motives Behind Cross-platform Forks from Software Heritage Dataset

REFERENCES


