Mining Fork-Including Software Development Traces

Iris Reinhartz-Berger[1](✉) and Amir Tomer[2]

[1] Information Systems Department, University of Haifa, Haifa, Israel
iris@is.haifa.ac.il
[2] Software Engineering Department, Kinneret College on the Sea of Galilee, Tzemach, Israel
tomera@mx.kinneret.ac.il

Abstract. Open-source software development is a common practice that encourages collaborative development and reuse across projects. Forking is a way to make a copy of an existing project and explore it for different purposes. Two types of forks are commonly mentioned in the literature: *contributing forks* which continue the development lines of the forked projects and aim at merging the contribution back to the forked projects; and *independently developed forks* which open new lines of development deviating from the forked projects. In this study, we aim to explore characteristics of fork-involving traces for better understanding collaboration and reuse considerations in software development. Analyzing 880 Java projects and their related action and observation events, with process mining and statistical techniques, we found that the occurrence of certain event types may predict the fork type, while the creation of certain fork types increase the involvement of users in the forked projects.

Keywords: Forks · Software development · Process mining · Development traces

1 Introduction

Nowadays information systems engineering and software development rely much on collaborative development which enables developers to learn from previously developed artifacts and reuse them. Forking is a well-known mechanism in open source software repositories for collaborative development; it supports the creation of new projects, named *forkees*, from an existing project, named *forked project*. The authors in [1] found high value in forking, especially in contribution to exposing and fixing software bugs and adding new features. In cases that the forked project is further developed and maintained, the literature distinguishes between two types of active forks [2, 6]: *contributing forks* which continue by developing new artifacts that are eventually merged back to the forked project via pull requests; and *independently developed forks* which lead towards a course of independent projects through commits that are unique to the forkees.

Forking is extensively studied. The work in [10], for example, explores the impact of fork type on project sustainability; the work in [8] studies the efficiency of forking

J. De Weerdt and A. Polyvyanyy (Eds.): CAiSE Forum 2022, LNBIP 452, pp. 100–109, 2022.
https://doi.org/10.1007/978-3-031-07481-3_12

practices. Several works (e.g., [4, 5, 7]) acknowledge the strong correlation between forking and project popularity, shedding light on the practice of forking. However, all these works neglect the behavioral, event-related aspects of forking and collaborative software development. Perceiving open source software repositories as event-driven management systems, we suggest focusing on the operations recorded by their API in order to mine the underlying processes and the features that characterize forking of different types.

In this study, we explored fork-inlcuding traces for better understanding collaboration and reuse considerations in software development. To this end, we analyzed the types of events in such traces, the partial traces *before* the forking operations and the continuation of the traces in the forked projects *after* the forking operations. By doing so, we try to reveal patterns of behavior leading towards (and maybe encouraging) forking, as well as to reveal post-operation patterns of behavior. Accordingly, we address the following research questions:

RQ1. What are the main types of events in open software development environments that support collaborative work?

RQ2. Which types of events lead to the creation of different types of forks in given projects?

RQ3.How does forking of different types influence the involvement of the users in the forked projects after the forking occurred?

The rest of the paper is structured as follows: Sect. 2 introduces a categorization framework for addressing RQ1; Sect. 3 presents the empirical study conducted for addressing RQ2 and RQ3; Finally, Sect. 4 discusses the implications and the threats to validity, while Sect. 5 summarizes the above and highlights some future research directions.

2 Categorizing Software Development Events

To address RQ1, we suggest the conceptual framework depicted in Fig. 1. This framework includes the main types of development events in open software development environments, in particular in GitHub. It further categorizes the events according to two dimensions: the involved elements and the event nature. The involved element may refer to the project itself – REPO[1]; its commits – COMMIT; its issues for suggesting improvements, tasks or questions – ISSUE; and its base pull requests for handling proposed changes – PR. The event nature distinguishes between action and observation: action events modify the information on the involved elements, while observation events add complimentary information and may indicate some user interest in the projects in general and in the specific elements in particular. An additional type of events that gets a special attention in our study is forking, i.e., fork creation – FORK. We concentrate here on contributing and independently developed forks, ignoring inactive forks, namely forks that are not further developed and maintained. Next we present the core definitions of event, trace (a sequence of events made by a *certain user* to a *certain project*) and contributing/independently developed forks.

[1] We used REPO for referring to project-related events, to avoid confusion with the general term 'project' which refers to the entire metadata of the software project.

Definition 1: An ***event*** is a tuple (p, u, ev, t, el, n), where p is the project to which the event belongs, u is the user who performed it, ev is the event (e.g., creating, watching, merging, etc.), t is its timestamp, el ∈ {REPO,COMMIT,PR,ISSUE, FORK} is the element to which the event relates and n ∈ {ACT,OBS} is the nature of the event (action or observation, respectively).

Definition 2: A ***trace*** is a sequence 'of events $<e_1 \ldots e_n>$ satisfying for each i, j, $e_i.p = e_j.p$ (same project), $e_i.u = e_j.u$ (same user) and $i < j \rightarrow e_i.t < e_j.t$ (sequence in time).

Definition 3: Let p, p' be projects satisfying fork(p, p')[2] and having sets of development events E, E', respectively.

- p' is a ***contributing fork*** if contribute(p', p)[3] holds.
- p' is an ***independently developed fork*** if it is not contributing and there is e '∈ E' such that e'.el = COMMIT and e'.n = ACT.

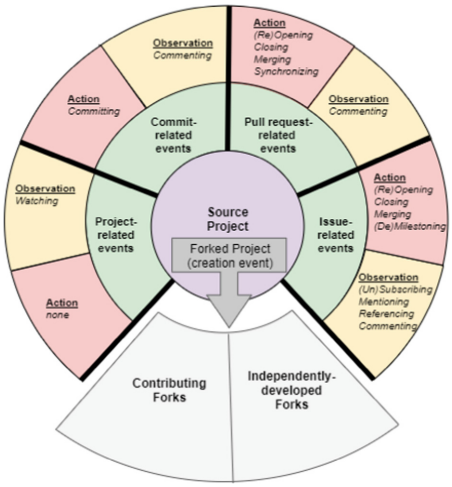


Fig. 1. The suggested framework

In summary, to address RQ1, we categorize the development events along two dimensions: the involved elements (REPO, COMMIT, ISSUE, PR) and the event nature (ACT, OBS), leading to 8 categories of events.

[2] fork(p, p') denotes the dependency between the forkee p' and its forked project p.

[3] contribute(p', p) is a predicate indicating whether (or not) the forkee p' contributes to the forked project p via a merging pull request event.

3 Characterizing Fork-Including Software Development Traces

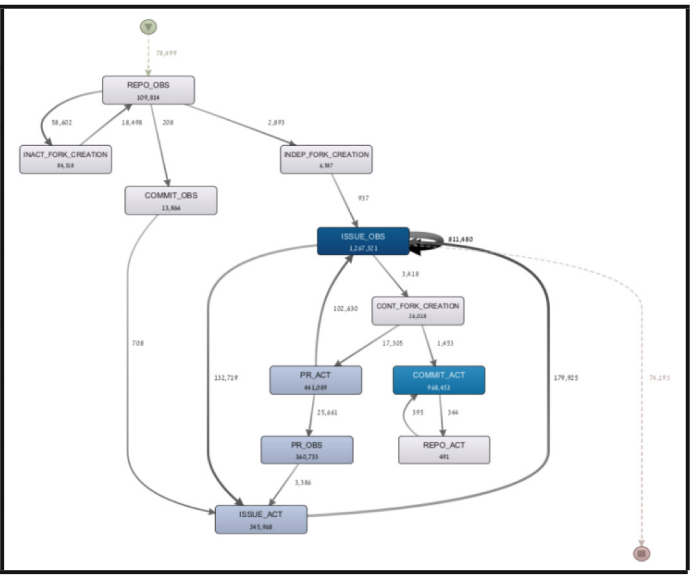
To address the two other research question (RQ2, RQ3), we conducted an empirical study, whose dataset, procedures and results are described below.

3.1 The Dataset

Our dataset is based on GHTorrent, which monitors the Github public event time line [3][4]. For each event, GHTorrent retrieves its contents and their dependencies, exhaustively.

Following observations from related works, and in order to concentrate on projects with significant characteristics relevant to our study, we used all projects satisfying the following conditions: (1) created during the year 2014 and were not deleted (i.e., have 4.5–5.5 years of existence by June 2019) (2) classified as written in Java, and (3) are highly forked (i.e., each yielded at least 100 forkees). Overall, we retrieved 880 projects and 366,631 forkees. After filtering out deleted forkees, we were left with 355,403 forkees to the 880 projects. The forked projects were related to 5,112,603 events of different types, 3,624,658 of which were involved in traces of length longer than 1. We refer to this set as our dataset (see Fig. 2(a) for details).

Characteristic	Value
Overall events	3,624,658
Events related to forked projects	3,507,735
Actions in forked projects	1,756,001 (50.06%)
Observations in forked projects	1,751,734 (49.94%)
Fork creation	116,923
Inactive forks	84,318 (72.11%)
Contributing forks	26,018 (22.25%)
Independently developed forks	6,587 (5.63%)



(a) (b)

Fig. 2. (a) The dataset characteristics; (b) The generated process map

Most events in the dataset (more than 3.5 million) related to the forked projects and almost 117,000 – to fork creation. About half of the events were actions and half had

[4] We particularly used the latest MySQL version of GHTorrent, dumped on June 1st, 2019; see https://ghtorrent.org/downloads.html.

an observation nature. Almost half of the events relate to issues, more than a quarter to commits and less than a quarter to pull requests. Finally, most forks were inactive, less than a quarter were contributing, and only 5.63% were used for opening an independent line of development[5].

3.2 Execution and Analysis Procedures

In order to analyze ("directly follows") edges in the relevant traces, we used process mining and particularly Disco software[6]. Figure 2(b) depicts the process map generated for our dataset. As can be seen, *independent fork creation* follows in many (about third) of the cases repository observation and in some cases involvement of the users in the forked projects can be observed (e.g., though issue observation events). *Contributing fork creation* may follow different events, but in many cases (two thirds of the cases), they were associated with active involvement of the users in the pull requests of the forked projects[7]. The process maps also shows *inactive fork creation*, which is quite common. Such creations are frequently associated with repository observation of the forked projects (watching events), either immediately before or immediately after the fork creation.

Grouping the traces according to the involved events and their order, 26,382 variants have been found. We filtered out those having only one occurrence, remaining with 3,392 variants. The maximal length of a trace in those variants was 675, but most of them were of length 2 to 10. Only 1,582 variants included at least one forking operation (37 variants included even 2 to 4 such operations).

For each of the 1,582 relevant variants, we recorded the following information: (1) Identity & occurrence information (variant number, number of cases); (2) Variant characteristics (variant length, number of forking operations, the step when the first forking operation appeared and its type)[8]; (3) Prefix traces (immediate event, numbers of events of each type before the first forking operation); (4) Suffix traces (immediate event, numbers of events of each type after the first forking operation).

We used a multinomial model for analyzing the data, where the fork type was the dependent variable and all other aforementioned features were independent variables in the same model. The factors were tested for $\alpha = 0.05$.

3.3 Results

In the context of the traces that lead to forks (RQ2), we found three significantly influencing factors: the immediate event ($\chi^2(14) = 488.68$, $p < 0.0001$), pull request actions ($\chi^2(2) = 560.70$, $p < 0.0001$) and commit actions ($\chi^2(2) = 12.69$, $p = 0.0018$). These

[5] The dataset, as well as its analyses, can be found at https://doi.org/10.5281/zenodo.6351644.

[6] https://fluxicon.com/disco/

[7] Note that these pull request actions refer to the forked projects and are not the merge operations of the forkees into the forked projects, as expected in contributing forks.

[8] Note that although some traces included more than one fork, this was very rare (happened in 37 out of 3,392 variants, and overall in 716 out of 175,414 cases). Hence, we considered in our analyses only the first forking operation in each trace/variant.

results led to the following outcomes. Due to page limitation constraints, Fig. 3(a) visually presents the results only for the immediate event.

Outcome 1. ***Contributing forks*** are characterized by either:

- Being created as the first communication of the user with the project, without any prior action or observation of the user in the forked project;
- Being created (eventually) after a pull request action in the forked project. In other words, the results show that the probability to have a pull request action in the forked project prior to the creation of a contributing fork is *high.*

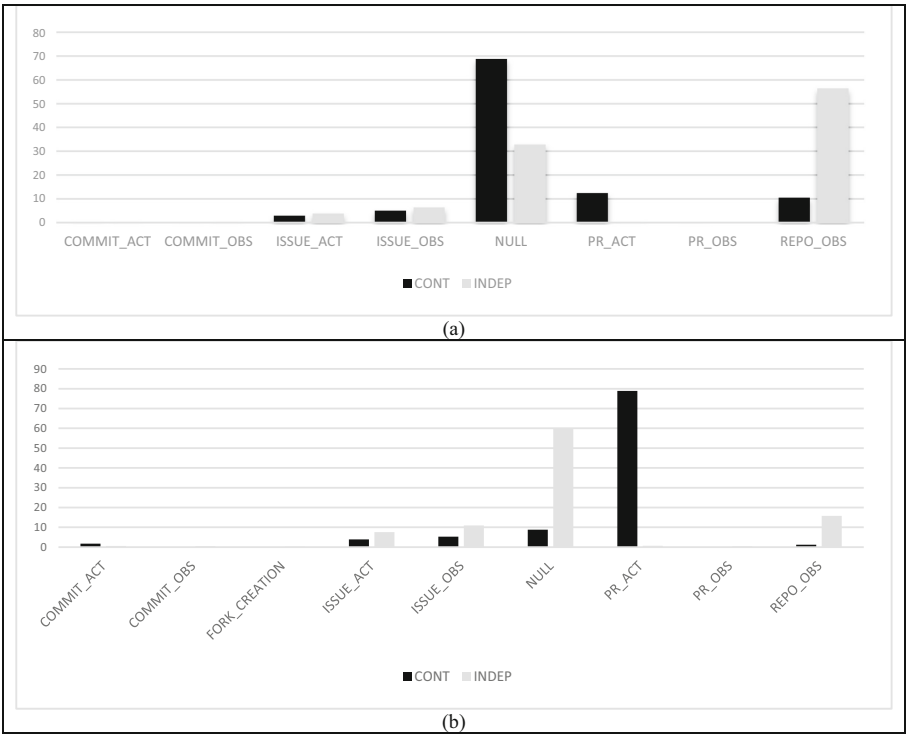


Fig. 3. (a) The dependencies between prefix traces and fork types – immediate (preceding) event; (b) The dependencies between suffix traces and fork types – immediate (following) event

Outcome 2. ***Independently developed forks*** are characterized by:

- Being created just after repository observation operations (namely, watching events)[9];
- Being created (eventually) after (several) commit actions in the forked project. Actually, the results show that the probability to create an independently developed fork

[9] However, this also characterizes inactive forks.

increases as the user performs more commits in the forked project prior to the fork creation.

These outcomes (RQ2) suggest that contributing fork creation commonly starts user traces or occurs after pull request actions, whereas independently developed forks are commonly created immediately after repository observation operations or subsequently after commit actions.

With respect to the continuation of traces after the first forking operation (RQ3), we found four significantly influencing factors: the immediate event ($\chi^2(14) = 658.87$, $p < 0.0001$), pull request actions ($\chi^2(2) = 1887.05$, $p < 0.0001$), issue observations ($\chi^2(2) = 15.38$, $p = 0.0005$) and repository observations ($\chi^2(2) = 7.82$, $p < 0.02$). These resulted in the following outcomes. Due to page limitation constraints, Fig. 3(b) visually presents the results only for the immediate event.

Outcome 3. ***Contributing forks*** are characterized by:

- Being followed (immediately or eventually) by pull request actions in the forked project. In other words, the results suggest that the probability to have a pull request action in the forked project after the creation of a contributing fork is *high*;
- Not being followed by an issue observation or a repository observation in the forked project (in other words, the probability to have either an issue observation or a repository observation in the forked project after the creation of a contributing fork is *low*).

Outcome 4. ***Independently developed forks*** are characterized by:

- Being the last event in the traces, and in some cases, being directly followed by repository observations;
- Not being followed by a pull request action, an issue observation or a repository observation in the forked project (in other words, the probability to have a pull request action, an issue observation or a repository observation in the forked project prior to the creation of an independently developed fork is *low*).

Our findings (RQ3) suggest that contributing fork creation commonly occurs (immediately or eventually) after pull request actions, whereas independently developed forks are commonly created at the end of traces (i.e., the users stop observing or acting on the forked projects).

4 Discussion and Threats to Validity

Analysis and interpretation of the results in the previous section show that creation of forks may involve the creators (the owners of the forkees) not only in the forkees, but also in the forked projects; this involvement can take place at early stages, before the fork is created, or afterwards. However, there appear to be specific types of events that are more significantly performed than others. Perceiving open source coding as a social activity, users may aim to increase their popularity and encourage the reuse of their code through active forking. Our findings may lead to "best practices" relevant to project owners to disseminate their projects and changes.

Project Dissemination: The reported results suggest that for disseminating projects their owners cannot rely only on the project community (e.g., committers). Watchers indeed tend to create independently developed forks, but the fork creators tend to get involved only after creating (contributing) forks, if at all. In some cases, involvement exists through pull request actions prior to contributing fork creation and through commit actions prior to independently developed fork creation. In these cases, project dissemination can be done also to pull request actors and committers.

Change Dissemination: After forkees have been created, it is important to be aware of the forked project evolution (changes). The owners of the forked projects may be interested in some involvement of the owners of their forkees. Our results suggest that while it is difficult to define the relevant community for independently developed forks, pull request actors may be targeted for this purpose.

Several threats of validity were identified during the study and deserve further consideration. First, we referred only to formal forks created in a single environment GitHub using its interface. The work in [9], for example, extends the definition of fork beyond the one obtained directly from GitHub metadata to forked projects generated on other platforms. Second, we currently investigated the traces individually, as information about actions and consequences are not implicitly available. Future research should explore dependencies among traces belonging to the same users, the same groups, or the same projects. Third, the categorization we suggested consolidates data relevant to the individual events. Further investigation and evaluation of the strengths or weaknesses of our conceptual framework is needed. Finally, the results are limited by the dataset we used and the statistical methods we applied. Although we have not used the specific characteristics of the dataset in the analysis procedure, replication of the study to different datasets is needed to verify generalizability.

5 Summary and Future Research

In this work, we discovered event-related dependencies in fork-including software development traces. To this end, a trace was defined as a sequence of events made by a *certain user* to a *certain project*. We introduced a conceptual framework, in which those events are categorized along two dimensions: the involved elements, namely projects/repositories, commits, base pull requests and issues, and the event nature which

distinguishes between action and observation operations. Using this framework, we analyzed potential dependencies between development events and forking, concentrating on contributing and independently developed forks, which support collaborative development and reusing. The analysis used process mining to identify interesting variants and statistical techniques to reveal patterns of behavior. The results were interpreted as "best practices", aiming to assist project owners who aim to utilize the social coding platform to disseminate their projects and changes through forkees.

This work may continue in various further directions. Particularly, we plan to explore interdependencies across traces, especially traces made by users related to each other, either by working in the same organization or collaborating on a number of projects. We also intend to apply additional techniques to the analysis. Machine learning, as one example, may be applied to the process mining results, in addition or alternatively to the statistical analysis, in order to reveal further mutual relations between development and forking events. Another example is text mining which may be applied to certain data items (e.g., issues, comments) in order to find both justifications and considerations which may support user decisions to perform certain operations, on top of the quantitative results.

Acknowledgement. This research is partially supported by the Israel Science Foundation under grant agreements 1065/19.

References

1. Biazzini, M.: "May the fork be with you": novel metrics to analyze collaboration on GitHub. In: Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics, pp. 37–43, June 2014
2. Cosentino, V., Izquierdo, J.L.C., Cabot, J.: A systematic mapping study of software development with GitHub. IEEE Access **5**, 7173–7192 (2017)
3. Gousios, G., Vasilescu, B., Serebrenik, A., Zaidman, A.: Lean GHTorrent: GitHub data on demand. In: Proceedings of the 11th Working Conference on Mining Software Repositories, pp. 384–387, May 2014
4. Jiang, J., Lo, D., He, J., Xia, X., Kochhar, P.S., Zhang, L.: Why and how developers fork what from whom in GitHub. Empir. Softw. Eng. **22**(1), 547–578 (2016). https://doi.org/10.1007/s10664-016-9436-6
5. Nyman, L., Mikkonen, T.: To fork or not to fork: fork motivations in SourceForge projects. Int. J. Open Source Softw. Process. (IJOSSP) **3**(3), 1–9 (2011)
6. Rastogi, A., Nagappan, N.: Forking and the sustainability of the developer community participation – an empirical investigation on outcomes and reasons. In: 2016 IEEE 23rd International Conference On Software Analysis, Evolution, and Reengineering (SANER), vol. 1, pp. 102–111. IEEE, March 2016
7. Robels, G., González-Barahona, J.M.: A comprehensive study of software forks: dates, reasons and outcomes. In: Hammouda, I., Lundell, B., Mikkonen, T., Scacchi, W. (eds.) IFIP International Conference on Open Source Systems, pp. 1–14. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33442-9_1
8. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: review and benchmark. ACM Trans. Knowl. Discovery from Data (TKDD) **13**(2), 1–57 (2019)

9. Zhou, S., Vasilescu, B., Kästner, C.: What the fork: a study of inefficient and efficient forking practices in social coding. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 350–361, August 2019
10. Zhou, S., Vasilescu, B., Kästner, C.: How has forking changed in the last 20 years? A study of hard forks on GitHub. In: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE), pp. 445–456. IEEE, October 2020