Quantifying Exogenous Software Forks

Antoine Pietri
Inria
Paris, France
antoine.pietri@inria.fr

Guillaume Rousseau
University Paris Diderot and Inria
Paris, France
guillaume.rousseau@univ-paris-diderot.fr

Stefano Zacchiroli
University Paris Diderot and Inria
Paris, France
zack@irif.fr

***Abstract*—A key research direction of software health and evolution is the study of *software forks*: projects which started off with an identical code base and development history, but whose development *forked* to end up being developed at different places, sometimes by different developer communities. Studying software forks not only allows researchers to get a more precise vision of where software projects are actively maintained, but also has important applications in software health: forks exhibit patterns of activity that can be used to determine common criteria for healthy software projects, by comparing successful forks to their inactive counterparts, as well as robust metrics independent of the distributed version control system (DVCS) used by the development team.**

Historically, studies of forks have mostly been using the metadata given by code hosting providers, most notably GitHub, as the source of truth for what a fork is. GitHub provides a public graph of forks that links forked repositories to their "parent" repositories. Relationships in this directed ancestry graph are created when, and only when, people press the "fork" button on GitHub. Alternatively, studies focusing on the tracking of source code artifact provenance use definitions based on intrinsic properties such as the "most fit fork", seeking to capture the main development lines. In this presentation, we expose flaws in trusting this information and argue that it can lead to selection biases when studying forks.

To fix that issue, we introduce the notion of *exogenous forks*, software projects that have some shared development history, as captured by the DVCS, but for which no forge-level metadata exist to identify them. To cover all possible cases of software forks, we have to generalize the notion of "fork" to a symmetric definition, where repositories are forks of each other without directionality. We detail different hypothetical cases where this approach should capture the reality of software evolution in a more precise way.

Through preliminary experiments on the Software Heritage graph dataset, we try to quantify the prevalence of these forks, in order to estimate the potential selection bias introduced when ignoring them. We observe that the main step for finding all the clusters of forks that share common development history is to search for all the connected components on the undirected subgraph of origins, snapshots, releases and revisions.

After running this experiment on the full graph dataset, we showcase the frequency distribution of the sizes of the connected components, and compare it with the GitHub fork graph extracted from the GHTorrent project. We attempt to explain the extremely disparate results between the two potential ground truths, as well as provide tools to investigate these differences.

We also discuss the caveat of this approach regarding the loss of directionality in the graph. We present potential workarounds to restore ancestry information in the general case: heuristics based on forge metadata, repository creation date, as well as the potential for partial ordering, and discuss research use cases where it may be applicable.

A potential for future exploitation of this work is to try to replicate previous experiments that used GitHub forks as their ground truth on the generalized definition, and see if the results change meaningfully.

***Index Terms*—software evolution, software health, software forks, source code, open source software, free software**

References

[1] Jean-François Abramatic, Roberto Di Cosmo, and Stefano Zacchiroli. Building the universal archive of source code. *Communications of the ACM*, 61(10):29–31, October 2018.

[2] Roberto Di Cosmo and Stefano Zacchiroli. Software Heritage: Why and how to preserve software source code. In *iPRES 2017: 14th International Conference on Digital Preservation*, 2017.

[3] MM Mahbubul Syeed, Imed Hammouda, and Tarja Systa. Evolution of open source software projects: A systematic literature review. *Journal of Software*, 8(11):2815–2830, 2013.

[4] Hongyu Pei Breivold, Muhammad Aufeef Chauhan, and Muhammad Ali Babar. A systematic review of studies of open source software evolution. In *Software Engineering Conference (APSEC), 2010 17th Asia Pacific*, pages 356–365. IEEE, 2010.

[5] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, pages 369–378, 1987.

[6] Jesus M Gonzalez-Barahona, Gregorio Robles, Martin Michlmayr, Juan José Amor, and Daniel M German. Macro-level software evolution: a case study of a large software compilation. *Empirical Software Engineering*, 14(3):262–285, 2009.

[7] Jeffrey Svajlenko and Chanchal Kumar Roy. Fast and flexible large-scale clone detection with cloneworks. In *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017 - Companion Volume*, pages 27–30, 2017.

[8] Yuichi Semura, Norihiro Yoshida, Eunjong Choi, and Katsuro Inoue. Ccfindersw: Clone detection tool with flexible multilingual tokenization. In *24th Asia-Pacific Software Engineering Conference, APSEC 2017, Nanjing, China, December 4-8, 2017*, pages 654–659, 2017.

[9] Suresh Thummalapenta, Luigi Cerulo, Lerina Aversano, and Massimiliano Di Penta. An empirical study on the maintenance of source code clones. *Empirical Software Engineering*, 15(1):1–34, 2010.

[10] Georgios Gousios and Diomidis Spinellis. GHTorrent: Github's data from a firehose. In *9th IEEE Working Conference of Mining Software Repositories, MSR 2012, June 2-3, 2012, Zurich, Switzerland*, pages 12–21, 2012.

[11] Antoine Pietri, Diomidis Spinellis, and Stefano Zacchiroli. The software heritage graph dataset, March 2019.

[12] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. The promises and perils of mining github. In *Proceedings of the 11th working conference on mining software repositories*, pages 92–101. ACM, 2014.

[13] Marco Biazzini and Benoit Baudry. 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*, pages 37–43. ACM, 2014.