Phylogenetic analysis of software evolution, a case study: VE&MINT

Alvaro Ortiz-Troncoso Corresp. 1

1 Science Programme Public Engagement with Science, Museum für Naturkunde, Leibniz Institute for Research on Evolution and Biodiversity at the Humboldt University Berlin, Berlin, Germany

Corresponding Author: Alvaro Ortiz-Troncoso
Email address: alvaro.ortiztroncoso@mfn.berlin

Open source projects may face a forking situation at some point during their life-cycle. The traditional view is that forks are a waste of project resources and should be avoided. However, in a wider technological and organisational context, forks can be a way to foster the creation of a software ecosystem. Either way, forking is explicitly allowed by open source licenses. Notwithstanding, methods for quantifying the evolution of forks are currently scarce. The present work attempts to answer the question whether a real-life project has forked. It does so by considering code and organisational characteristics of the project, and analysing these characteristics by applying methods ported from biological phylogenetics. After finding that the project is forked, implications for project governance are discussed.

Phylogenetic analysis of software evolution, a case study: VE&MINT

Alvaro Ortiz-Troncoso[1]

[1]**Museum für Naturkunde Berlin**

Corresponding author:
Alvaro Ortiz-Troncoso[1]

Email address: alvaro.OrtizTroncoso@mfn.berlin

ABSTRACT

Open source projects may face a forking situation at some point during their life-cycle. The traditional view is that forks are a waste of project resources and should be avoided. However, in a wider technological and organisational context, forks can be a way to foster the creation of a software ecosystem. Either way, forking is explicitly allowed by open source licenses. Notwithstanding, methods for quantifying the evolution of forks are currently scarce. The present work attempts to answer the question whether a real-life project has forked. It does so by considering code and organisational characteristics of the project, and analysing these characteristics by applying methods ported from biological phylogenetics. After finding that the project is forked, implications for project governance are discussed.

1 INTRODUCTION

In open source software development, a fork is a bifurcation from an existing project, resulting in an autonomous development strand, with its own name, infrastructure, code base, and community of developers (Robles and González-Barahona, 2012). Organisations adopting open source software development might face a fork situation during the software's life-cycle, or might voluntarily fork existing software as a means to solve technical-, license-, and team-related issues (Nyman and Lindman, 2013; Robles and González-Barahona, 2012).

Knowing whether a project will likely fork (or has forked already) can therefore facilitate project management. Phylogenetic methods, ported from evolutionary biology, can be used to estimate a phylogenetic tree of a project (Ortiz-Troncoso, 2018a), and hence provide a method for estimating whether two diverging development strands are more likely to fork than to merge. The present work applies phylogenetic methods to the evolution of a real-world project, VE&MINT.

The VE&MINT project is a cooperation between several German institutes: MINT-Kolleg Baden-Württemberg with the VEMINT-Konsortium, the Leibniz Universität Hannover, and the Technische Universität Berlin (VE&MINT, 2016). The project's purpose is to offer a preparatory mathematics course online[1]. Technically, course modules are written using the document preparation system LaTeX. The modules are transformed into formats suitable for online presentation using a software originally authored at the Karlsruhe Institute of Technology (KIT) and further developed at the Technische Universität Berlin (TUB). There are two main software development strands: the KIT and the TUB strands. These strands are developed by different teams, yet the name of the project is the same, and the teams have access to each other's code repositories. Therefore, the definition of fork proposed by Robles and González-Barahona (2012), introduced above, applies only in part. The purpose of the present analysis is to study the evolution of the VE&MINT project by answering the following question:

Research Question

Can the KIT and TUB software development strands be considered parts of one project or is the original project forked in two different projects?

[1]The version of the course developed at the Technische Universität Berlin can be accessed at: `https://www.math4refugees.tu-berlin.de/`

The answer to this question could be instrumental in directing the development effort in future iterations of the project.

2 MATERIALS AND METHODS

The "master" branch, maintained by the Karlsruhe Institute of Technology (KIT) and the "multilang" branch, maintained by the Technische Universität Berlin (TUB) were used for the analysis, as these are the two branches under active development.

The two development strands do not have a common release scheme. In order to quantify development, it was necessary to create releases. This was accomplished by creating a release for each month in each branch. As TUB development started in June 2016, the chosen time period spans from the 1st July 2016 to the 1st January 2018.

Software metrics were computed for each month in each branch. The choice of software metrics is based, with alterations, on the overview of software metrics provided by Nagappan et al. (2008). Metrics about team members were anonymised. A summary of the metrics retained and of their data types is given in table 1.

Table 1. Summary of metrics and their data types

Metric	Domain	Data type
Presence (or absence) of a file in a release	One measurement for each version of a file in a release	Boolean
File changes (checksums)	One measurement for each version of a file in a release	categorical
Code churn: line count for each code file	One measurement for each version of a code file in a release	integer
Team composition: presence (or absence) of a team member (anonymised) for each release	One measurement for each team member contributing to a release	Boolean
Edit frequency: count of edits by each team member (anonymised) for each release	One measurement for each team member contributing to a release	integer
Code ownership: count of commits by each team member (anonymised) for each release	One measurement for each team member contributing to a release	integer

A matrix of pairwise dissimilarity between releases was computed. As the measurements used are of different data types (table 1), the Gower algorithm for computing distance matrices was used, since this algorithm can combine measurements of mixed data types (D'Orazio, 2016).

The Neighbour-Joining (NJ) algorithm (Saitou and Nei, 1987) was used to estimate a phylogenetic tree from the distance matrix, as previous work on well-documented cases of forks, e.g., the MySQL / MariaDB fork (Ortiz-Troncoso, 2018a), suggests that the NJ algorithm produces the most accurate results when applied to the evolution of software. The tree was rooted at the last release prior to the start of development at the TUB (June 2016).

A cophenetic distance matrix was computed to obtain a numerical representation of the phylogenetic tree: The elements of the cophenetic distance matrix are the pairwise distances between the tips of the tree (R Development Core Team, 2008, p. 1275). The correlation between the distance matrix (obtained from code and team characteristics) and the cophenetic distance matrix (obtained from the tree) was used as a measurement of how well the tree represents the data.

Whether the VE&MINT project is forked was answered by performing a comparison of the mean cophenetic distance between releases in the same development strand ("branches") vs. across strands ("forks"), (building upon Ortiz-Troncoso (2018a)). The research question can be expressed as a statistical hypothesis:

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27159v1 | CC BY 4.0 Open Access | rec: 29 Aug 2018, publ: 29 Aug 2018

Statistical hypothesis

A project is forked when a branch can be told from a fork by examining the pairwise distance between releases: if the mean cophenetic distance between releases is significantly different ($p<0.05$) and larger depending on whether the releases are on the same or on different development strands, then the project is forked.

The research question can thus be answered by comparing the means of the pairwise cophenetic distances for two treatments ("branches" and "forks"). Two variables are examined: one nominal variable, whether a pair of releases are on the same development strand ("branches") or on different development strands ("forks"); and one measurement variable: the pairwise cophenetic distance between releases.

As the variance of the measurement variable is not assumed to be equal for the two treatments, the Welch two-sample t-test was chosen (McDonald, 2014, p. 130). A t-test assumes that the data is normally distributed. Performing a t-test on data that is not normally distributed increases the risk of false positives (McDonald, 2014, p. 128). Previous work has shown that the distribution of cophenetic distances between software releases is not normally distributed, but that the square root of distances better fits the normal distribution (Ortiz-Troncoso, 2018a). Therefore, the t-test was performed on square root transformed data.

The source code of the KIT development strand can be downloaded from `https://bitbucket.org/dhaase/ve-und-mint/src/master/`, and the TUB source from `https://gitlab.tubit.tu-berlin.de/stefan.born/VEUNDMINT_TUB_Brueckenkurs`. The data set composed of measurements of code and anonymized organisational characteristics can be downloaded from: `http://doi.org/10.5281/zenodo.1327424`. The analysis was done using Evorepo, a software for constructing phylogenetic trees of software evolution (Ortiz-Troncoso, 2018b). The source code to produce the phylogenetic analysis for this work can be downloaded from: `https://zenodo.org/record/1401151`.

3 RESULTS

The measurements obtained from the releases were stored in a database and exported to a spreadsheet consisting of one column for each release and one row for each measurement. Two data sets were created: one for the measurements obtained from code characteristics and one for the measurements obtained from team characteristics (table 1). A matrix of pairwise distances between releases was computed based on each data set, and a phylogenetic tree was estimated from the distance matrix obtained from each data set, applying the methods described above (figures 1 and 2). The trees represents the distance matrix accurately (code data: 0.9999605 correlation, $p < 2.2\times10^{-16}$; team data: 0.9843978 correlation, $p < 2.2\times10^{-16}$).

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27159v1 | CC BY 4.0 Open Access | rec: 29 Aug 2018, publ: 29 Aug 2018

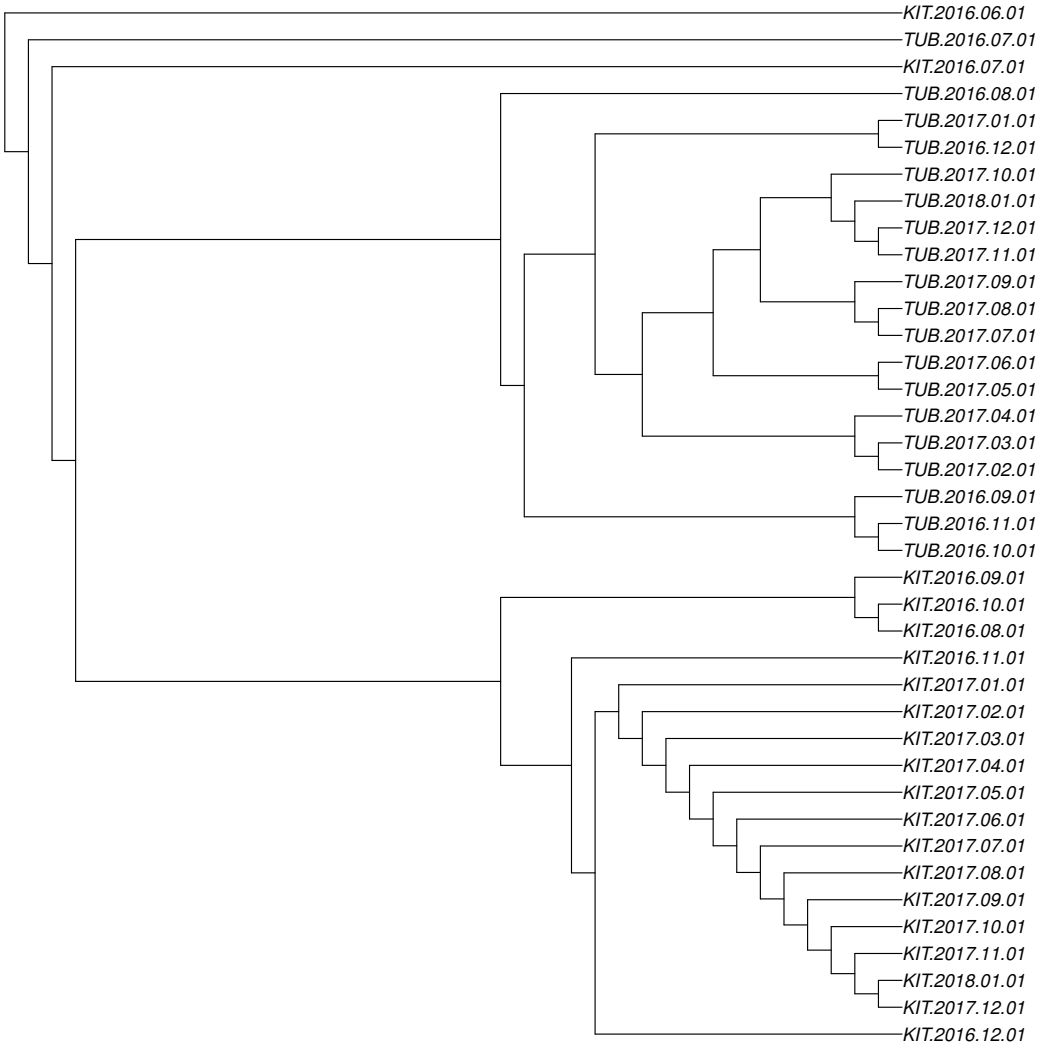

Figure 1. A phylogenetic tree of the VE&MINT project, obtained from **code** characteristics.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27159v1 | CC BY 4.0 Open Access | rec: 29 Aug 2018, publ: 29 Aug 2018

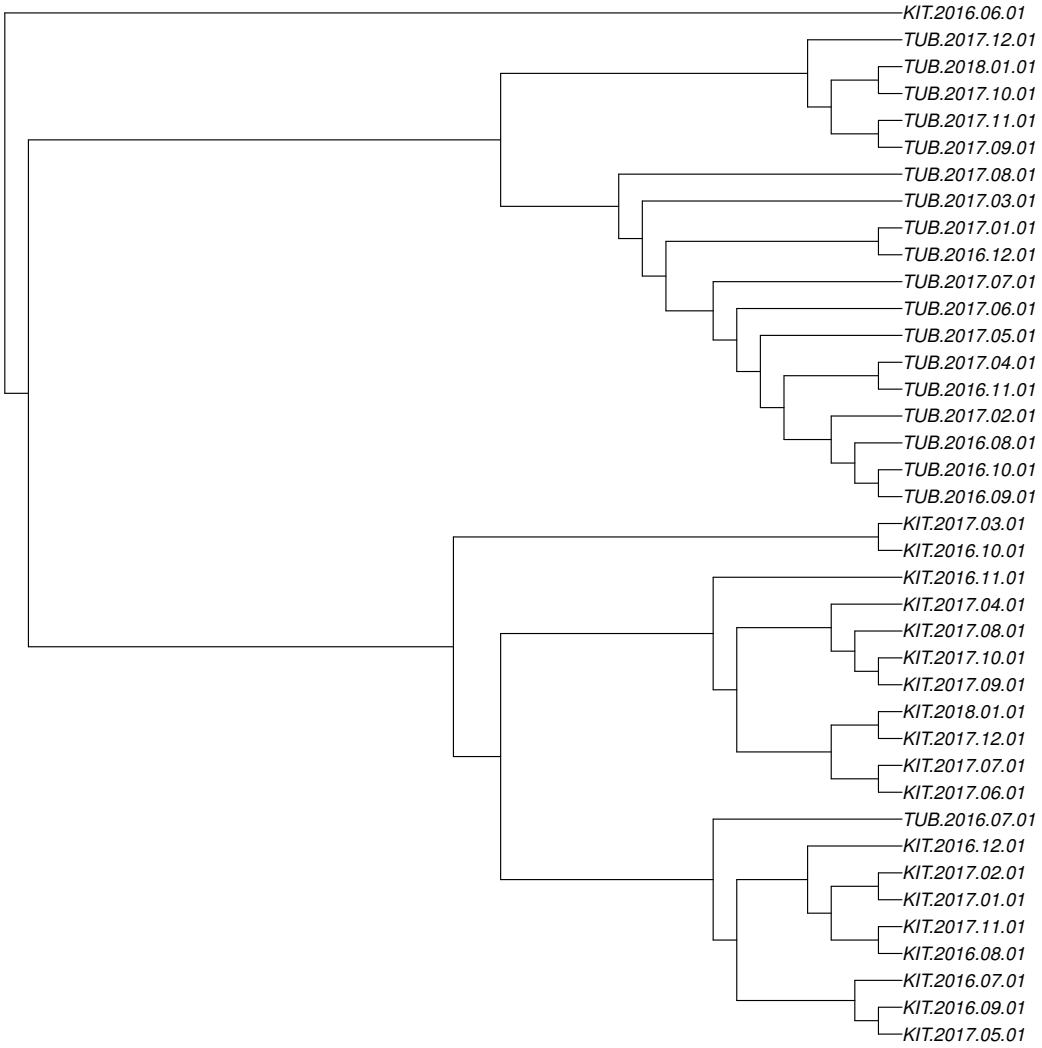

Figure 2. A phylogenetic tree of the VE&MINT project, obtained from **team** characteristics.

In order to answer the research question, a relation was sought between the main pairwise distance between releases and whether these releases are part of the same development strand ("branches") or are part of different strands ("forks"). The analysis was repeated for the data obtained from code characteristics and for the data obtained from team characteristics (tables 2 and 3). The analysis shows that the the null hypothesis can be rejected ($p < 2 \times 10^{-16}$) for each data set. Therefore, the distance between releases on the same strand ("branches") is significantly different from the distance between releases across strands ("forks").

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27159v1 | CC BY 4.0 Open Access | rec: 29 Aug 2018, publ: 29 Aug 2018

Table 2. Pairwise distances of branches vs. forks for the code dataset.

```
Welch Two Sample t-test

data:  values by vars
t = -56.836, df = 704.91, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.7676998 -0.7164323
sample estimates:
mean in group Branches    mean in group Forks
             0.1690282              0.9110943
```

Table 3. Pairwise distances of branches vs. forks for the team dataset.

```
Welch Two Sample t-test

data:  values by vars
t = -8.4289, df = 709.32, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.08226048 -0.05117884
sample estimates:
mean in group Branches    mean in group Forks
             0.3914378              0.4581575
```

4 DISCUSSION

The estimated phylogenetic trees (figures 1 and 2) show all releases as a monophyletic clade, i.e., they all descend from a common ancestor. The most recent common ancestor of all releases is the KIT release dated June 2016. After that, the algorithm estimated that the most parsimonious tree, i.e., the simplest and therefore the most likely tree, is a tree where most KIT and TUB releases are grouped separately.

The analysis of variance found that the distance between releases of different development strands is significantly larger than the distance inside development strands. This provides evidence for the hypothesis that the project could be said to be forked.

Nyman and Lindman (2013) discuss the implications of forking from the point of view of project governance. They argue that forking affects open source project at three levels:

- At the software level, a fork creates a redundancy, reduces the probability that a project might become obsolete, and hence increases the lifespan of a project.
- At the community level, a fork allows to tackle multiple issues, using different technologies, and hence increases the sustainability of a project's community of developers.
- At the business level, forks provide a mechanism for quickly responding to changing needs in the user base.

On the other hand, forking can result in competing products, i.e. competing for the participation of developers. Therefore, in order to take advantage of the possibilities offered by a fork, the governance of

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27159v1 | CC BY 4.0 Open Access | rec: 29 Aug 2018, publ: 29 Aug 2018

a forked project should focus on nurturing a community of practice (encompassing forked strands) and encouraging external participation Kogut and Metiu (2001).

5 CONCLUSIONS

A phylogenetic analysis of the development of the VE&MINT project found evidence that the development strands at the KIT and at the TUB are forked.

The phylogenetic analysis is neither capable of discerning the reasons for the fork nor to make predictions about the fork's outcome (Ortiz-Troncoso, 2018a). However, having obtained quantitative evidence that the project is forked could facilitate a discussion on the way the project should be steered, as well as on the validity of the method presented here.

6 ACKNOWLEDGEMENTS

I wish to thank Erhard Zorn of TUB for providing feedback on the first draft of this paper.

REFERENCES

D'Orazio, M. (2016). StatMatch: Statistical Matching.

Kogut, B. and Metiu, A. (2001). Open-source software development and distributed innovation. *Oxford Review of Economic Policy*, 17(2):248–264.

McDonald, J. H. (2014). *Handbook of Biological Statistics*. Sparky House Publishing.

Nagappan, N., Murphy, B., and Basili, V. (2008). The influence of organizational structure on software quality. In *Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on*, pages 521–530. IEEE.

Nyman, L. and Lindman, J. (2013). Code Forking, Governance, and Sustainability in Open Source Software. *Technology Information Management*, (January):7–12.

Ortiz-Troncoso, A. (2018a). *Evaluating phylogenetic methods for quantifying risks and opportunities presented by forks in open source software (master dissertation)*. Zenodo.

Ortiz-Troncoso, A. (2018b). evorepo version 0.9.1.

R Development Core Team (2008). R - A language and environment for statistical computing. *Social Science*, 3.

Robles, G. and González-Barahona, J. M. (2012). A Comprehensive Study of Software Forks: Dates, Reasons and Outcomes.

Saitou, N. and Nei, M. (1987). The neighbor-joining method : a new method for reconstructing phylogenetic trees. *Molecular and Biological Evolution*, 4(4):406–425.

VE&MINT (2016). Ve&mint. `http://www.ve-und-mint.de/`. Accessed: 2018-02-20.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27159v1 | CC BY 4.0 Open Access | rec: 29 Aug 2018, publ: 29 Aug 2018