Track 1 Lightning Talk: Forking as a Tool for Software Sustainability—An Empirical Study

Sarah Alhozaimy
School of Computer Science
University of Manchester, UK

Robert Haines
Research IT
University of Manchester, UK

Caroline Jay
School of Computer Science
University of Manchester, UK
caroline.jay@manchester.ac.uk

***Abstract*—*Forking*—the process of cloning a repository, allowing development to progress separately to the original project—has become an important part of source control functionality. As forking enables software reuse, it has the potential to play a role in *software sustainability*, which aims to improve the longevity of software. We studied the relationship between forking and the *sustainment*, or active life, of a software project. An examination of 9,118 projects hosted on GitHub shows a significant relationship between forking and software sustainment, with projects that have forks being, on average, sustained for longer than those that do not, a phenomenon that is true when considering both the length of the original project, and the length of the original project extended by any forks. The results provide evidence that maintaining software via forking is a sustainable software practice, and making software open source improves its sustainability through enabling reuse via forking.**

I. Introduction

The Software Sustainability Institute (SSI) defines 'sustainable software' as that which 'you use today will be available—and continue to be improved and supported—in the future'[1]. Ensuring sustainability has been identified as one of the key challenges in the development of software, particularly in research projects, where the software may continue to have value, but there are limited financial resources to maintain it. Whilst sustainability is generally linked to best practice [1, 2], precisely how to achieve it remains an open question [3].

This paper examines the link between the use of *forking*—the process of cloning a repository, such that development can progress separately to the original project—and *software sustainment* (the active life of the project). We find that forking increases the sustainment of the original project, and can increase the lifespan of software by allowing its extension.

II. Background

Fogel (2005) defines a software *fork* as occurring when a development team divides itself into two rival groups, to work on different, incompatible versions of the code [4]. The term has been used in a wider context over the last few years, and now generally describes a case where a new project has been developed through the replication of an existing codebase.

Forking in GitHub can be represented as a tree structure with the original repository, F_0, as the root [5]. Forks that are created directly from the original repository are termed primary forks, F_{1_n}. Secondary forks, F_{2_n}, are created from a primary fork. We examined three depths of fork, as shown in Figure 1.

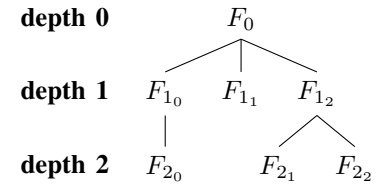

Fig. 1. A representative fork tree showing the depths of forks.

An application loses value over time if it fails to adapt [6]. The use of forks therefore aligns with the SSI's definition of sustainability, as it means a software project is open to modifications, increasing the potential utility of the code. Whilst forks can result from technical incompatibility or interpersonal disagreements [7], forking tends to be used by new developers to safely test a new idea before adding it to the original project [8]. Forking can represent a threat to a project, as software may end up 'watered down', or in competition [9, 5], but it can also extend its lifespan, as it allows it to be taken in a different direction [10].

Whilst there is a solid foundation for considering forking to be a useful tool in sustaining software, there is no definitive, quantitative evidence demonstrating a relationship to project lifespan. Here, we examine the extent to which it increases or decreases the longevity of both the original project, and the original project and its longest surviving fork.

III. Research Methodology

A subset of repositories was selected from GitHub based on the following criteria: the project was created between 1st January and 31st December 2009; the project had at least one commit; the first commit occurred on or after 1 st of January 2009.We sampled five continuous days from each month. For example, we choose the first five days of January 2009; and 6th to 10th of February 2009. The data and analysis code are available on Figshare [11].

In this paper, we use two definitions for software sustainment. Sustainment of the original project, S, is defined as the time period from the first commit of a repository through to its last commit, measured in days, as shown in equation 1 [12].

[1] www.software.ac.uk/about

S was calculated for the *default* branch of each repository, which in 99% (9,017) of projects was the master branch.

$$S = t_{last-commit} - t_{initial-commit} \quad (1)$$

Software sustainment with forking, SF, is defined as the number of days from the first commit of the original repository through to the last commit when considering all forks (in this study to a depth of two), as shown in equation 2.

$$SF = t_{last-commit-on-all-forks} - t_{initial-commit} \quad (2)$$

IV. Results

Of the 9,118 projects retrieved from GitHub, 3,314 (36%) had at least one fork. Of the projects with forks, 1,349 (40%) had one fork, 504 (15%) had two forks, and 45% had three or more.Whilst 90% of projects had fewer than 5 forks, a number of projects used forking extensively. The largest number of forks recorded for one project was 3,009. It should be noted that these numbers consider two depths of forking, F_1, *i.e.* forks from the original project, and F_2, *i.e.*, forks of forks.

A. Is forking related to sustainment of the original project?

Figure 2 shows the distribution of projects as a function of S. Projects that have forks are shown in blue, towards the back of the graph. The mean sustainment S, of projects where the number of forks ≥ 1 is 788 days, compared to 303 days for projects with 0 forks, a difference shown by a Mann-Whitney test to be statistically significant ($U = 5,885,866.5$, $p = .000$).

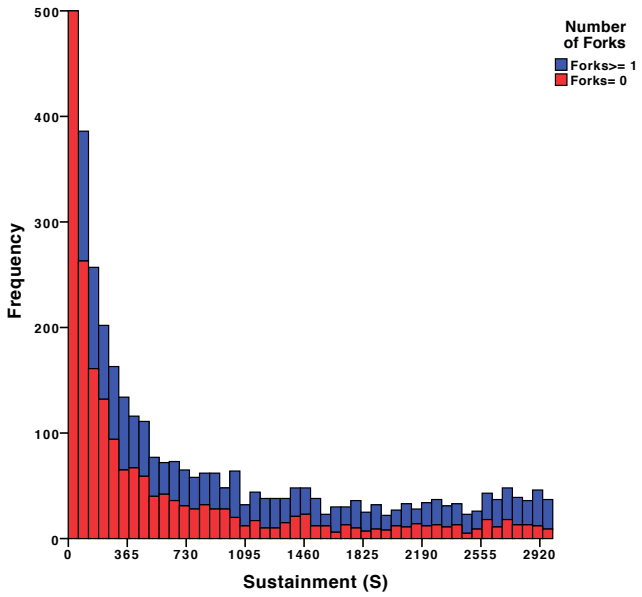

Fig. 2. Software sustainment as a function of S in days for projects in GitHub (The arbitrary (or frequency) value of the first column is 4,432 for projects where Forks $= 0$ and 1,076 for projects where Forks ≥ 1.)

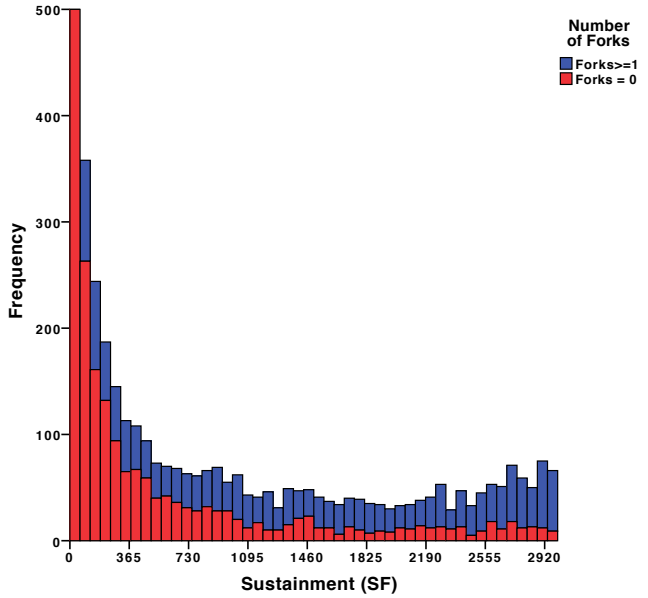

Fig. 3. Software sustainment as a function of SF in days for projects in GitHub (The arbitrary (or frequency) value of the first column is 4,141 for projects where Forks $= 0$ and 768 for projects where Forks ≥ 1.)

B. Does forking sustain software beyond the original project?

A second question is whether forking extends the life of software beyond the original project: if the original project ceases to be sustained, does activity continue in a forked project? Here we considered only those 3,314 projects with at least one fork. Figure 3 shows the distribution as a function of SF. The mean value of SF (1,111 days) is significantly greater than the mean value of S (788 days), a difference that a Wilcoxon test shows to be statistically significant ($Z = -32.631, P = .000$). Extension to the life of software offered by a fork occurs in just under a third of the projects examined. In 70% of projects the last recorded commit was in the original project; in the remaining 30%, the last commit was recorded in one of the forks.

V. Conclusion

We demonstrate a two-fold effect of forking on the sustainability of software. Projects that use forking are sustained, on average, for longer than those that do not, indicating that using forks to update or maintain software is a positive practice in terms of software sustainability. Forking also offers the opportunity to extend the life of the software beyond the original project. Open source repositories such as GitHub that support forking may play a key role in supporting the sustainability of software.

References

[1] M. de Souza et al, "Defining Sustainability through Developers' Eyes: Recommendations from an Interview Study," in *WSSSPE 2*, 2014.

[2] S. Betz and T. Caporale, "Sustainable Software System Engineering," in *2014 IEEE Fourth International Conference on Big Data and Cloud Computing*, 2014.

[3] C. Venters et al., "The Blind Men and the Elephant: Towards an Empirical Evaluation Framework for Software Sustainability," *JORS*, vol. 2, no. 1, 2014.

[4] K. Fogel, *Producing Open Source Software How to Run a Successful Free Software Project*, 2nd ed. LLC, 2005. [Online]. Available: http://producingoss.com/en/producingoss.pdf
[5] A. Rastogi, "Forking and the Sustainability of the Developer Community Participation - An Empirical Investigation on Outcomes and Reasons," *SANR*, 2016.
[6] L. Meir, "Programs, Life Cycles, and Laws of Software Evolution," *IEEE*, vol. 68, no. 9, Sep. 1980.
[7] H. Kuusirati, "Forks in Open Source Software Projects," *University of Oulu*, 2012. [Online]. Available: https://wiki.oulu.fi/download/attachments/28092087/ossd_2012_kuusirati_seppanen.pdf?version=1&modificationDate=1353314930000
[8] L. Nyman and T. Mikkonen, "To Fork or Not to Fork : Fork Motivations in SourceForge Projects," *Springer*, 2011.
[9] R. Viseur, "Forks impacts and motivations in free and open source projects," *IJACSA*, vol. 3, no. 2, pp. 117–122, 2012.
[10] L. Nyman, J. Lindman, and G. Moody, "Code Forking , Governance , and Sustainability in Open Source Software," *TIM*, pp. 7–12, Jan. 2013.
[11] S. Alhozaimy, "sustainability-and-SVC-usage-wssspe5.1.zip," 8 2017. [Online]. Available: https://figshare.com/articles/sustainability-and-SVC-usage-wssspe5_1_zip/5328730
[12] A. Aldabjan et al, "How should we measure the relationship between code quality and software sustainability?" *WSSSPE 4, CEUR Workshop Proceedings*, 2016.