Developer Turnover in Global, Industrial Open Source Projects: Insights from Applying Survival Analysis

Bin Lin
Università della Svizzera Italiana, Switzerland
Email: bin.lin@usi.ch

Gregorio Robles
Universidad Rey Juan Carlos, Spain
Email: grex@gsyc.urjc.es

Alexander Serebrenik
Eindhoven University of Technology, The Netherlands
Email: a.serebrenik@tue.nl

***Abstract*—Large open source software projects often have a globally distributed development team. Studies have shown developer turnover has a significant impact on the project success. Frequent developer turnover may lead to loss of productivity due to lacking relevant knowledge and spending extra time learning how projects work. Thus, lots of attention has been paid to which factors are related to developer retention; however, few of them focus on the impact of activities of individual developers.**

In this paper, we study five open source projects from different organizations and examine whether developer turnover is affected by when they start contributing and what types of contributions they are making. Our study reveals that developers have higher chances to survive in software projects when they 1) start contributing to the project earlier; 2) mainly modify instead of creating files; 3) mainly code instead of dealing with documentations. Our results also shed lights on the potential approaches to improving developer retention.

I. Introduction

Large open source software (OSS) projects have been known to be global software development projects. The research literature offers some examples of detailed descriptions of how such projects work in a distributed environment. Good examples are the GNOME desktop environment [9] and the FreeBSD operating system project [42]. Although since the late 1990s, the *traditional* software industry has to some extent collaborated with open source communities, currently we witness a new type of OSS projects, where many companies come together and collaborate. So, if before we had OSS projects with a majority of volunteer developers cooperating with paid employees, nowadays there are projects where volunteer developers are a minority and most of the tasks are done by professionals hired to work on the project [34, 33]. Projects such as WebKit (with several dozen companies) or especially OpenStack (with over 500 companies involved) are examples of these *industrial* open source software projects [44].

This type of projects are gaining lots of attention in recent times. Interestingly enough, to the challenge of being globally developed, these projects have to face a major complexity as, even if they often set up umbrella organizations such as foundations or associations, each industry partner has its own policy regarding human resources and task allocation.

Since developers play a key role in software development, it is first important to know the factors that can affect the retention of developers, and second to have tools and methods to properly analyze and manage retention. Hence, our study aims to shed some light on basic factors which can impact the will of developers to stay, and our purpose is that it serves as a fundamental step for further research on how to improve the retention rate of developers.

Therefore, we study five industrial OSS projects of different sizes (in terms of software size, contributors and number of companies involved) and examine whether the duration of developers staying in a project is related to following four factors: (i) the time of first contribution, (ii) the rate of maintaining own files, (iii) the main action type, and (iv) the main job type. We study these factors by means of applying survival analysis, a well-known technique from the medical sciences. Survival analysis is well-suited for the so-called *right-censored data*, i.e., data about patients/activities that are still alive/ongoing at the time of study. Not surprisingly, survival analysis has been applied to software evolution problems in the past [3, 10, 37]. Applying survival analysis, we find that 1) developers who start contributing to the project earlier have higher survival rates; 2) higher survival rates are found when developers balance maintaining files created by themselves with files created by others; 3) developers who mainly modify files survive longer than those who mainly create files; and 4) developers who mainly code have a higher survival rate than those who mainly work on documentation.

The remainder of the paper is structured as follows: our hypotheses are presented in Section II, while Section III introduces related research. The methodology used to conduct the research and a description of the OSS projects used as case study can be found in Section IV. Research findings are elaborated in Section V. Section VI discusses the results and its implications, including the threats to validity. Finally, Section VII draws conclusions and describes future work.

II. Hypotheses

Previous research has studied the reasons for developers leaving a job. By adapting earlier results to open source projects

we formulate the following hypotheses.

Yang et al. [52] described different learning trajectories between early joiners and late joiners in a Coursera MOOC: while early joiners were actively involved in forum discussions and course materials, the late joiners participated less actively. Ultimately, the drop-out rate of the early joiners was found to be statistically significantly lower.

Moreover, earlier engagement in the project activities can be expected to provide contributors with more flexibility to determine their own jobs and as such to achieve a better person-job fit, that has been shown to be beneficial for the retention of developers [38]. Thus, we posit:

H1. Developers who start contributing to the project earlier have higher survival rates.

Autonomy has been reported to have a positive impact on job satisfaction [48], as it is often reported as one of the general motivators for software developers [11]. We conjecture that maintenance of your own code can be seen as preserving more the autonomy of developers rather than maintaining code created by others. Furthermore, Lambert et al. [22] have found that job satisfaction has a significant impact on turnover intent, i.e., when workers have high job satisfaction they are less willing to leave. Hence, we link the focus on maintaining the developers' own files with longer engagement duration. However, developers that solely focus on their own contribution might be too isolated compromising the person-team fit, known to be beneficial for retention [38]. To balance these two observation we conjecture:

H2. Developers who balance maintaining files created by themselves with files created by others will stay longer than developers who predominantly focus on maintaining their own files and developers who predominantly focus on maintaining files created by others.

Martensen et al. [24] showed that creativity and innovation are important factors in job satisfaction. Software development has been reported as being perceived as "a form of puzzle solving, and it is reassuring to their [developers'] ego when they manage to successfully complete a difficult selection of code" [15]. As opposed to software development, software *maintenance* has been reported as being perceived as entailing "very little new creation and is therefore categorized as dull, unexciting detective work" [15]. Hence, we formulate the following hypothesis:

H3. Developers who mainly create files survive longer than those who mainly modify files.

Herbsleb et al. [12] stated that "the resistance to documentation among developers is well known and needs no emphasis." Also, in some modern software development methods such as agile software development [1], more emphasis is given to working on programming tasks than in a comprehensive and detailed documentation. Thus, we posit:

H4. Developers who mainly code have a higher survival rate than those who mainly work on documentation.

III. Related research

Developer turnover has been cited many times as one of the risks in global software development. For instance, Ebert et al. [7] suggest that staff turnover rates are directly related to the "drive for global talent allocation" and Dibbern et al. [5] note that "personnel turnover [was] found to increase client extra costs.". It is noteworthy that turnover in a global environment may be different from location to location. So, some authors argue that for surviving in a global software development, managers have to take into account that some global projects were canceled because of the high staff turnover rates, "which in other countries might be higher than in Europe" [6, 16].

In OSS projects, developer turnover is a phenomenon that has been paid already some attention [28, 38]. A major part of it is, however, centered on the joining process, as attracting new developers has been a major effort of many open source communities [43]. Research on the integration of new members that ranges from the first interactions [41] to the time a developer requires to become part of the leading *core team*, the most active contributors, can be found [13]. As opposed to these results, Zhou et al. [53, 54] have studied turnover. As opposed to our work, their results have been based on the data from the issue tracker augmented with interviews. Our work provides a complementary perspective on turnover as hypotheses H2–H4 cannot be answered unless the version control system can be accessed, while for H1 such an access provides a different operationalization.

The negative impact of developer turnover on quality in OSS projects has been studied as well [8]. It is known that turnover has a negative impact on the productivity due to knowledge loss and the extra time spent on learning how the project works [18]. Along similar lines, a recent work of Constantinou and Mens [4] relates the specialization of the leavers to the risk they cause to the ecosystem. As opposed to this line of work, in this paper we do not consider impact of turnover but rather try to identify developer subgroups that are more likely to leave the project.

As opposed to earlier research on properties of projects [51, 40] or project teams [39, 45] affecting turnover, we focus on the activities of developers and investigate how those activities can impact turnover.

Iqbal [17] investigated turnover patterns and concluded that members who own the role of a developer in Apache projects have more contributions than others. However, this study does not indicate what factors impact the turnover. Sharma et al. [40] analyze turnover considering both developer and project factors, including role, number of projects involved, past activity level, tenure, and age and size of the project. However, they only show that these factors were related to the turnover and further explanation would be required to reveal the relationship between turnover and these factors.

Finally, there is a study performed on the Wikipedia community that is similar to ours, not only because it has a similar focus and goals, but also because it uses as well survival analysis techniques [31].

IV. Methodology

A. Survival analysis

In the medical domain, survival curves describe the probability that a subject can survive beyond a specified period: the x-axis stands for the survival duration (i.e., how long an individual survives) and the y-axis stands for the probability that an individual can survive [32]. Ideally, to estimate the survival curves, one should have the complete data about death of all the individuals: however, as individuals might still be alive at the time the study finishes, their death time cannot be known. To address this special kind of missing data, known as *right-censored* data, Kaplan and Meier [19] proposed the "Kaplan-Meier curves". Furthermore, one is usually interested in comparing survival of groups of individuals rather than individuals themselves, e.g., those subject to treatment as opposed to the control group. When applied to groups, each of the groups is represented by one curve.

Kaplan-Meier curves were first used in the medical discipline and then applied to other domains including software engineering. For example, Bird et al. [3] use survival analysis to estimate the immigration events in Postgres. Samoladas et al. [37] applied survival analysis techniques to estimation of future development of an OSS project. Goeminne et al. [10] analyzed whether different database frameworks co-occur in OSS projects and whether some database frameworks are replaced by others over time.

In this paper, we use survival analysis to understand the impact of several factors on developers leaving a project. Since each project can contain several components (i.e., repositories), we can distinguish between local leavers, that leave the component (repository) but continue to be active within the project, and global leavers, that are no longer active in any of the components (repositories) [27]. Here, we focus on global leavers and define them as those contributors whose last commit was made 180 days ago: the choice of the threshold is motivated by a similar choice made in an earlier study by Foucault et al. [8]. However, since other threshold choices can be found in the literature [40, 18] we also study to which extent the value of this threshold (i.e., 180 days) influences our results, and therefore have repeated the study with different thresholds: 30 days and 90 days.

It is known that when a group consists of very few items (i.e., contributors in our case), the survival analysis might become inaccurate and does not have too much meaning. Thus, in the remainder of our analysis, we have removed those groups with less than or equal to five subjects.

B. Choice of the case studies

The goal of our research is to explore the validity and limitations of the hypotheses formulated in Section II. Hence, we have opted for a case study as recommended by Runeson and Höst [36].

The datasets we use in our study are obtained from Bitergia[1]. Bitergia has developed and maintained dashboards for many active OSS projects. Those dashboards visualize the information about gits, tickets, mailing lists and metrics about software development. The MySQL data dumps[2] used to develop the dashboards are available for many projects. The data dumps are generated by CVSAnalY[3]. We choose Bitergia projects for our study because these projects are large and active such that they can be both seen as representative of successful industrial OSS projects and provide enough data for analysis.

In addition to representativeness, diversity of the dataset is known to be important in selection of the case studies [30]. Hence, we select five projects from five different organizations, which are *WikiMedia*[4], *OpenStack*[5], *GlusterFS from Red Hat*[6], *Xen Project from the Linux Foundation*[7] and *Apache CloudStack*[8], to ensure that the results are not specific for one organization.

All case studies can be considered globally developed software projects. In the companion website of the paper[9], we have included the geographical distribution of developers by using data from their timezones. This information can be obtained from their e-mail activity in mailing lists or their commits in the versioning repositories. For all projects, but WikiMedia, it can be observed that a substantial part ($> 20\%$) of the contributions come respectively from the Americas, Europe and Asia. In the case of WikiMedia, Europe counts with over 60% and the Americas with over 30%.

All the projects count with a significant amount of full-time developers paid to work on the project, and the number of companies involved in them ranges from over 600 companies in OpenStack, 18 in CloudStack, 11 in GlusterFS, to 9 in Xen. WikiMedia projects are developed mainly with staff from the WikiMedia Foundation. Our focus on the industrialized open source projects means that the findings from these paper could potentially be applied to other, more traditional software development environments.

In the datasets, different aliases that are likely to belong to the same person are combined based on the user name and email addresses [20, 49]. Hence, we replace aliases with a unique ID. The numbers of commits, files and contributors are presented in Table I. We then clean our data by removing invalid items. Users with abnormal first commit year (e.g., 1970) are removed. In addition, the identity of contributors (i.e., user name) is checked and non-human contributors such as "Jenkins CI bot" are filtered out. The number of contributors after data cleaning can also be found in Table I.

C. Operationalization

To verify our hypotheses, we operationalize the hypotheses as follows:

[1]http://bitergia.com/projects.html

[2]Database schema details can be seen at http://gsyc.es/~carlosgc/files/cvsanaly.pdf.

[3]https://github.com/MetricsGrimoire/CVSAnalY

[4]https://wikimedia.biterg.io/

[5]http://activity.openstack.org/dash/browser/

[6]http://glusterfs.biterg.io

[7]http://xen.biterg.io

[8]http://projects.bitergia.com/apache-cloudstack/browser/

[9]https://dev-turnover.github.io/timezone.html

TABLE I: Statistics of the datasets

	Commits	Files	Contributors (original)	Contributors (after cleaning)
WikiMedia	707,844	932,662	2,491	2,488
OpenStack	400,805	316,895	4,065	4,050
GlusterFS	13,396	9,860	165	165
Xen Project	89,827	38,541	912	909
CloudStack	44,595	146,997	314	314

H1. Developers who start contributing to the project earlier have higher survival rates.

We use first commit year to represent the time of the first contribution, and then compare the Kaplan-Meier curve for each cohort of developers. To define the cohorts we use the year of the first contribution made by the developer. More fine-grained cohorts are at risk of producing smaller populations, and in combination with our decision to remove small groups, will lead to "gaps" between the cohorts hindering the interpretation of the curves. Similarly, more coarse-grained cohorts might blur up the distinctions between the groups.

If our hypothesis is supported by the data, the survival curve of early contributors will be above the one of developers who joined the project later.

H2. Developers who balance maintaining files created by themselves with files created by others will stay longer than developers who predominantly focus on maintaining their own files and developers who predominantly focus on maintaining files created by others.

To formalize the notion of "mainly maintaining files created by themselves" we define the rate of maintaining own files R of a contributor C as

$$R = \frac{\text{occurrences of files created by } C \text{ in all commits by } C}{\text{occurrences files in all commits by } C}.$$

For example, assume C has two actions in the log: 1) creating files F_1 and F_2 in the first commit; 2) modifying F_2 and F_3 in the second commit. In total, there are four occurrences of files in this log: F_1, F_2 (twice) and F_3; three of these occurrences belong to files created by C herself. The rate of maintaining own files R is hence $3/4 = 0.75$.

The definition of R refers to occurrences of files (also known as touches [2]) rather than the files themselves to represent the workload incurred by maintaining the files. We are aware of the shortcomings of the number of commits and related number of occurrences as measures of the maintenance effort or workload; however, these proxies are common in repository mining research (cf. discussion of Mens [26]).

If our hypothesis is supported by the data, the survival curve of very low R and very high R will be lower than the survival curve of the mid-range R.

H3. Developers who mainly create files survive longer than those who mainly modify files.

The type of each action performed by a developer can be identified from the versioning system. The different types of actions are: add file (A), modify file (M), delete file (D), replace file (R), rename file (V) and copy file (C). We determine the main action type of a contributor by finding the action type with the highest percentage in her commits. We calculate the distribution of main action types in projects. As can be seen in Table II, the most common actions in the project are "add file" (A) and "modify file" (M); very few developers conduct other types of actions as their main action type. Thus, in this paper we have decided to only consider and compare these two action types. Furthermore, to address the imbalance between the "add file" (A) and "modify file" (M) groups, we performed upsampling. We use upsampling instead of downsampling, because downsampling can cause information loss.

TABLE II: Distribution of main action types in projects

Type	OpenStack	CloudStack	GlusterFS	Xen Project	WikiMedia
A	6.32%	8.12%	5.39%	2.94%	13.29%
M	92.49%	88.44%	92.81%	95.76%	85.52%
D	0.47%	2.81%	1.20%	0.44%	0.68%
V	0.68%	0.31%	0.60%	0.87%	0.50%
R	0.00%	0.00%	0.00%	0.00%	0.00%
C	0.05%	0.31%	0.00%	0.00%	0.00%

If our hypothesis is supported by the data, the survival curve of creators (A) will be higher than the curve of maintainers (M).

H4. Developers who mainly code have a higher survival rate than those who mainly work on documentation.

The type of each file in the repository can be categorized depending on its content. We therefore use heuristics from other mining repository studies based on file extension and file name [35, 46]. Specifically, we label files as code, build, user interface (ui), internationalization (i18n), documentation, developer documentation (devel-doc), packaging, images, multimedia, or unknown. We identify the main job of a developer based on the type of files she touches. We do this by calculating the percentage of each file type in the commits of the contributor, and then consider the type with the highest percentage as the contributor's main job.

If our hypothesis is supported by the data, the survival curve of coders will be higher than the one of documentation writers.

Once we perform the specific queries that give an answer to the hypotheses, we use the Lifelines[10] software to obtain the Kaplan-Meier curves. The intermediate data and the code for survival analysis can be found on our companion website[11].

V. Results

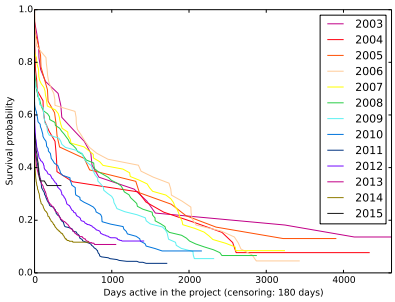
A. H1. Developers who start contributing to the project earlier have higher survival rates

Table III records the number of new contributors during the history of the projects. The table clearly shows that development teams are in general growing gradually in all projects, in spite of some fluctuation.

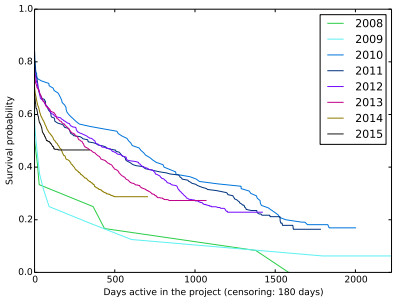
The survival curve based on the first commit year can be found in Figure 1.

[10]https://github.com/CamDavidsonPilon/lifelines

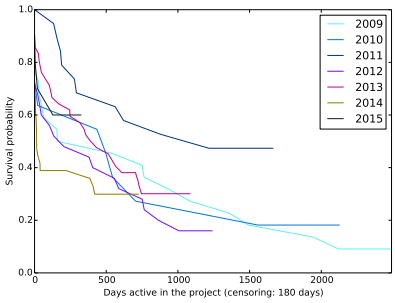
[11]https://dev-turnover.github.io/index.html



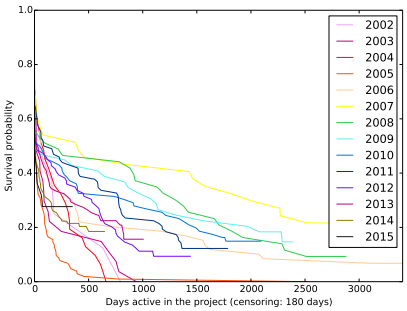
(a) WikiMedia



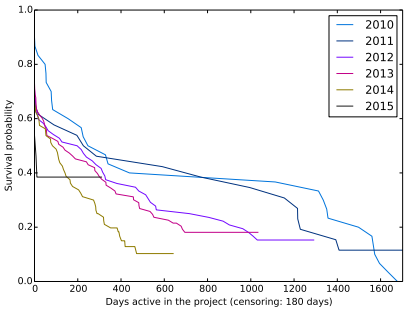
(b) OpenStack



(c) GlusterFS



(d) Xen Project



(e) CloudStack

Fig. 1: Survival analysis by first commit year

TABLE III: Number of new contributors per year: WM—WikiMedia, OS—OpenStack, CS—CloudStack

	WM	OS	GlusterFS	Xen	CS		WM	OS	GlusterFS	Xen	CS
2001	2	-	-	-	-	*2008*	91	12	-	43	-
2002	1	-	-	8	-	*2009*	86	16	22	49	-
2003	22	-	-	27	-	*2010*	108	110	11	86	30
2004	26	-	-	31	-	*2011*	296	254	19	98	26
2005	23	-	-	103	-	*2012*	532	548	25	89	72
2006	44	1	-	59	-	*2013*	602	1,050	42	102	93
2007	71	1	-	37	-	*2014*	522	1,285	36	130	80

By inspecting Figure 1, we observe that in all projects in the most recent years, newer developers are less willing to stay than older developers. This observation supports our hypothesis.

Since code complexity strongly influences contributions from new developers in a negative way [29], a potential reason for the trend in the graph could be that the growing code complexity forms an obstacle to contribution of more recently joining developers. Being unable to contribute harms the retention of developers. The survival rates in the first two years of OpenStack and the first five years of the Xen project are lower than for other years. The reason might be that the systems were still at the initial phase and code complexity was still acceptable for newcomers.

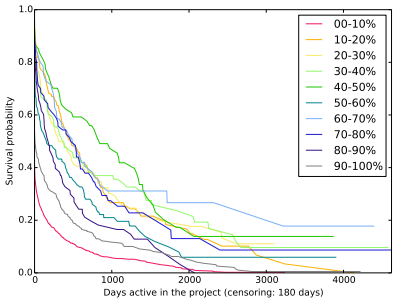
B. H2. Developers who balance maintaining files created by themselves with files created by others will stay longer than developers who predominantly focus on maintaining their own files and developers who predominantly focus on maintaining files created by others.

The survival curves based on the rate of maintaining own files can be found in Figure 2. We observe that, for instance, an interesting phenomenon is that developers with the lowest and the highest rates (0-10% and 90-100%) of maintaining their own files always have very low survival rates. The result supports our initial hypothesis; moreover, it provides some insights for when the best "balance" is reached. The balance does not mean developers should spend half of their effort on both activities. Instead, the results show that survival rates are higher when developers spend less than half of their effort on maintaining their own files.

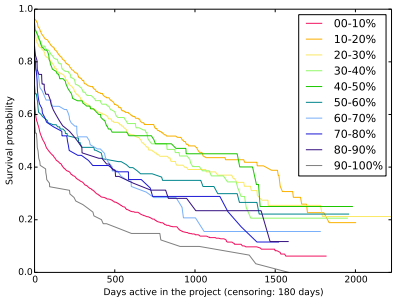
C. H3. developers who mainly create files survive longer than those who mainly modify files

The survival curves based on the main action type can be found in Figure 3. The survival curves in the figure contradict our assumption, as in all projects developers who mainly create files stay shorter than those who mainly maintain files.

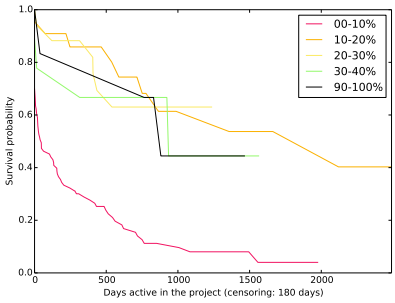
Since the data is highly unbalanced, we use upsampling to increase the number of "A—add file" samples by replicating them several times, such that (after the replication) the number is as close as possible to the number of "M—modify file" samples. The numbers of developers with main action type "A" (including after upsampling) and "M" can be seen in Table IV. The survival curves after upsampling, which can be seen in



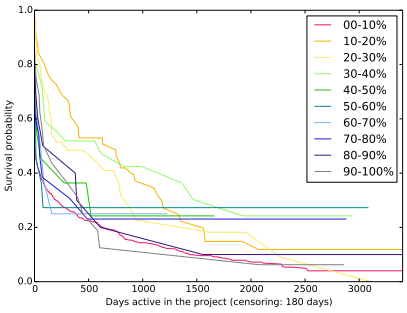
(a) WikiMedia



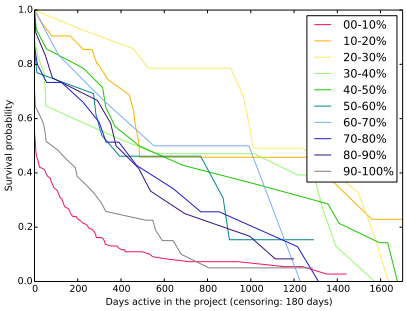
(b) OpenStack



(c) GlusterFS

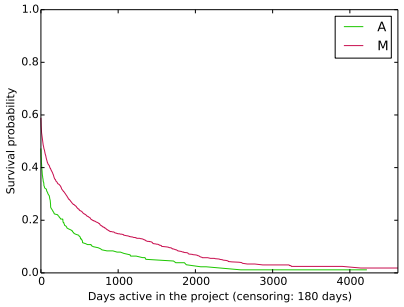


(d) Xen

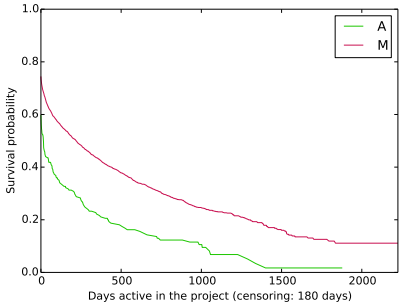


(e) CloudStack

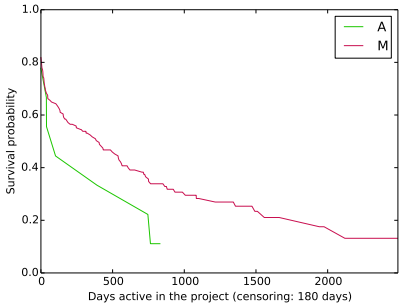
Fig. 2: Survival analysis by the rate of maintaining own files



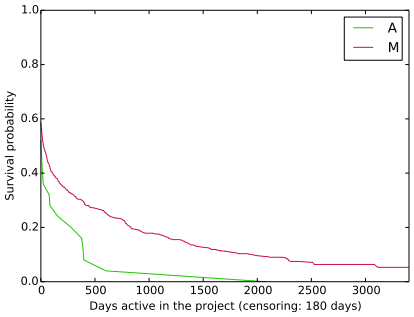
(a) WikiMedia



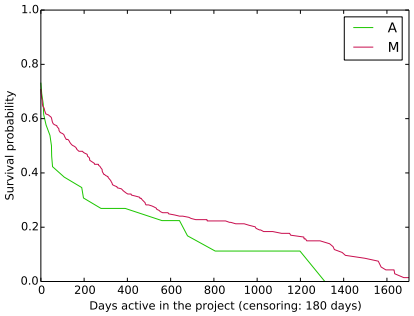
(b) OpenStack



(c) GlusterFS



(d) Xen



(e) CloudStack

Fig. 3: Survival analysis by the main action type

TABLE IV: Developers that predominantly add (A) and modify (M) files

	A	M	A (after upsampling)
WikiMedia	308	2,152	2,156
OpenStack	245	3,758	3,675
GlusterFS	9	154	153
Xen	25	875	862
CloudStack	286	277	271

Appendix C[12], show the same trends as those in Figure 3. To statistically support our perception, we applied log-rank test [47] to test "H3': developers who mainly modify files survive longer than those who mainly create files". We test both without and with upsampling.

If no upsampling is used, the statistical test confirms H3' for WikiMedia, OpenStack (for both projects the p-values are too small to be computed precisely) and Xen ($p \simeq 0.03$). When upsampling is applied to counteract the possible influence of the group imbalance on the results of log-rank test, H3' is confirmed for all projects[13].

The log-rank test results can be a strong evidence that developers whose main action is "A—add file" are more likely to leave the projects compared to those whose main actions is "M—modify file".

D. H4. developers who mainly code have a higher survival rate than those who mainly work on documentation

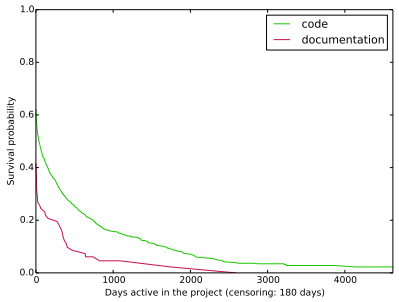
The survival curves based on the main job can be found in Figure 4. The figures show that developers who mainly write code stay longer than developers who mainly work on documentation. The possible reason is that coding usually takes more iterations than documentation, which forces users to stay longer to participate in the revisions. Another potential reason is that coding gives developers more sense of achievement, as it is known from the research literature that there is a lack of interest in documentation [25]. Since the data is unbalanced, we use the upsampling method to resize the "documentation" samples by completely duplicating them several times, such that the resized number is as close to the number of "coding" samples. The numbers of developers with main action type "documentation" (including after upsampling) and "code" can be see in Table V. The survival curves after upsampling, which can be seen in Appendix D[14], also show the same trends as in Figure 4.

Similarly to the study of H3, we also applied log-rank test. Dataset "GlusterFS" is excluded in this case, since it contains too few items and upsampling might cause a huge bias. The results for WikiMedia, OpenStack and Xen are statistically

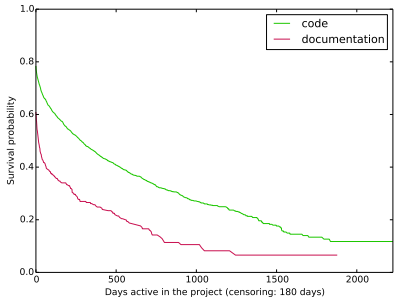
[12]Due to page limits, all appendices are uploaded online separately, Please refer to Appendix C at https://dev-turnover.github.io/download/appendix.pdf

[13]The p-values are too small to be computed precisely for WikiMedia, OpenStack and Xen. For GlusterFS $p \simeq 0.002$, for CloudStack $p \simeq 0.009$.

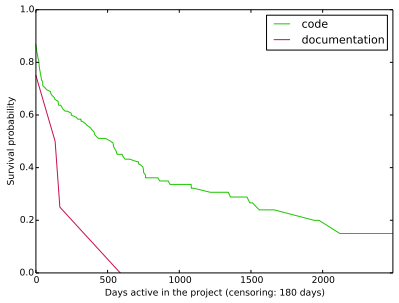
[14]Please refer to Appendix D at https://dev-turnover.github.io/download/appendix.pdf



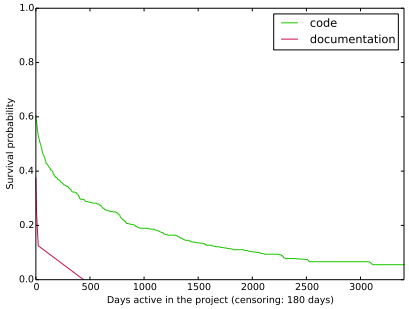
(a) WikiMedia



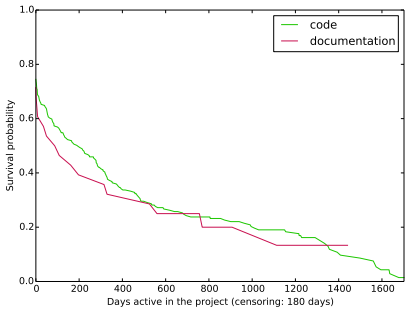
(b) OpenStack



(c) GlusterFS



(d) Xen



(e) CloudStack

Fig. 4: Survival analysis by the main job

significant, whether upsampling is used or not[15]. For these projects H4 can be confirmed, i.e., developers who mainly code have a higher survival rate than those who mainly work on documentation. For the remaining project, CloudStack, the evidence is not strong enough to support H4 (both when upsampling is used and when it is not used).

Hence, overall we can say that H4 is proved true in most cases. To obtain further insights in relation between the job type and turnover more projects can be investigated or other methods, e.g., qualitative studies, can be applied.

TABLE V: Numbers of developers with main job type "Code" and "Documentation"

dataset	Documentation	Code	Documentation (upsampling)
WikiMedia	82	1,818	1,804
OpenStack	349	2,860	2,792
GlusterFS	4	135	-
Xen Project	8	794	792
CloudStack	28	239	252

VI. Discussion and Threats to validity

A. Discussion

1) Developer retention improvement: We have revealed some factors related to developer turnover. Since high developer turnover threatens the success of software projects, we are interested in how we can improve developer retention, especially for globally distributed development teams.

We have found that developers who join a project earlier have higher chances to stay longer. Although the root cause of this phenomenon is not disclosed in our work, we can assume that developers might face difficulties to contribute to the project or are less loyal to the projects. No matter which is the real reason, it might be helpful to get newcomers more engaged in the project and have better onboarding assistance.

Another aspect to consider to improve developer retention is how to distribute tasks. Through the results obtained when examining H2, H3 and H4, we propose several strategies:

- Balance collaboration and individualism, i.e., when assigning maintenance tasks ensure that developers maintain both code developed by others and their own code.
- For those developers who mainly write new code, they could do more code maintenance tasks.
- Developers in charge of documentation should not only deal with documentations, instead, some coding tasks may increase their chances of staying in the projects.

2) Impact of the threshold for leavers: In this paper, we use 180 days as the threshold to determine whether a developer is a leaver or not. To understand the impact of the choice of the threshold, we have repeated the study with different values: 30 days and 90 days. The survival curves can be found in Appendix A and Appendix B, respectively[16]. The results show the same trends as we reported in previous sections.

B. Threats to validity

Several factors may affect the validity of our study results and some limitations exist in this study.

1) External validity: Although five projects from different organizations are used to conduct the study, the results obtained might not apply to other types of OSS projects, such as small scale OSS projects. Besides, all the projects in the study are maintained with git. It is unclear whether projects maintained with other version control tools such as Subversion will show a different pattern of the relationship between leavers and the factors presented in this report.

2) Internal validity: In our study, we consider the time to join the project, the rate of maintaining own files, the main action type and the main job type. However, during software project development, many other factors might impact the developers' will to stick to the project, such as the importance and maturity of the project. Due to the limitation of the datasets, we cannot take all factors into account. Furthermore, some human aspects such as the personal manners of the leaders might impact the will of developers to stay, which cannot be seen from our datasets. Additionally, some factors might interfere or cooperate with each other, nevertheless, we do not inspect this possibility in our study.

3) Construct validity: During the data processing, the contribution by developers might not have been computed correctly because the identities of developers are not correctly merged. That is, different developers may have been identified as the same person, and developers who have several accounts might not have been detected.

VII. Conclusions and Future Work

In this report, we examine whether four factors (the time to join the project, the rate of maintaining own files, the main action type and the main job type) affect the behavior of developers to stay in a project. We conduct survival analysis on five global, industrial OSS projects and find out 1) in general, earlier developers stay longer than later developers; 2) the survival rate of developers is not positively correlated to the percentage of maintaining own files; 3) developers who mainly modify files stay longer than those who mainly create files; 4) developers whose main job is coding do have higher survival rate than those who mainly maintain documentations.

While our study brings initial insights on the factors which impact how long developers survive, more research should be done to gain deeper understandings of why these phenomena happen. While the reasons behind the phenomena cannot be easily discovered with our dataset, a survey can be conducted among leavers to verify the potential reasons proposed in the report and the communication channels including mailing lists can be mined. In addition, many other factors can be involved in the study. Research has shown that the motivations

[15]When upsampling is applied the p-values for the three projects were too small to be calculated precisely. Without upsampling, for Xen $p \simeq 0.018$, WikiMedia 1×10^{-5} and for OpenStack the p-value was too small to be calculated precisely.

[16]Please refer to Appendix A and Appendix B at https://dev-turnover.github.io/download/appendix.pdf

of developers to participate in an OSS project are influenced by the identification of participants [14], the transformational leadership of leaders and an active management style [23], and the emotions of developers [50, 21]. We could take these factors into account and explore the role of these human aspects in developers' turnover intent. Furthermore, a more challenging task is to develop a model which predicts how long developers will stay in a project based on several factors.

ACKNOWLEDGMENTS

The authors would like to acknowledge financial support of SENECA (MSCA-ITN-2014-EID) and eMadrid (S2013/ICE-2715). We are very grateful to Bitergia for sharing their data with us, and to Mathieu Goeminne for his valuable feedback on the preliminary versions of this manuscript.

REFERENCES

[1] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al. Manifesto for Agile Software Development, 2001.

[2] K. Beecher, C. Boldyreff, A. Capiluppi, and S. Rank. Evolutionary success of open source software: an investigation into exogenous drivers. *Electronic communications of the EASST*, 2008.

[3] C. Bird, A. Gourley, P. Devanbu, A. Swaminathan, and G. Hsu. Open borders? Immigration in Open Source projects. In *MSR*, pages 6–11. IEEE, 2007.

[4] E. Constantinou and T. Mens. Socio-technical evolution of the Ruby ecosystem in GitHub. In *SANER*. IEEE, 2017.

[5] J. Dibbern, J. Winkler, and A. Heinzl. Explaining variations in client extra costs between software projects offshored to India. *MIS quarterly*, 32(2):333–366, 2008.

[6] C. Ebert and P. De Neve. Surviving Global Software Development. *IEEE Software*, 18(2):62–69, 2001.

[7] C. Ebert, B. K. Murthy, and N. N. Jha. Managing risks in Global Software Engineering: Principles and practices. In *ICGSE*, pages 131–140. IEEE, 2008.

[8] M. Foucault, M. Palyart, X. Blanc, G. C. Murphy, and J.-R. Falleri. Impact of developer turnover on quality in Open-Source Software. In *FSE*, pages 829–841. ACM, 2015.

[9] D. M. German. The GNOME project: A case study of Open Source, Global Software Development. *Software Process: Improvement and Practice*, 8(4):201–215, 2003.

[10] M. Goeminne and T. Mens. Towards a survival analysis of database framework usage in Java projects. In *ICSME*, pages 551–555. IEEE, 2015.

[11] T. Hall, H. Sharp, S. Beecham, N. Baddoo, and H. Robinson. What do we know about developer motivation? *IEEE Software*, 25(4):92–94, 2008.

[12] J. D. Herbsleb and D. Moitra. Global Software Development. *IEEE Software*, 18(2):16–20, 2001.

[13] I. Herraiz, G. Robles, J. J. Amor, T. Romera, and J. M. González Barahona. The processes of joining in Global Distributed Software projects. In *International workshop on Global software development for the practitioner*, pages 27–33. ACM, 2006.

[14] G. Hertel, S. Niedner, and S. Herrmann. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy*, 32(7):1159–1177, 2003.

[15] D. A. Higgins. *Data Structured Software Maintenance: The Warnier/Orr Approach*. Dorset House Publishing Company, 1986.

[16] H. Holmstrom, E. Ó. Conchúir, J. Agerfalk, and B. Fitzgerald. Global Software Development challenges: A case study on temporal, geographical and socio-cultural distance. In *ICGSE*, pages 3–11. IEEE, 2006.

[17] A. Iqbal. Understanding Contributor to Developer Turnover Patterns in OSS Projects: A Case Study of Apache Projects. *ISRN Software Engineering*, 2014, 2014.

[18] D. Izquierdo-Cortazar, G. Robles, F. Ortega, and J. M. González-Barahona. Using software archaeology to measure knowledge loss in software projects due to developer turnover. In *HICSS*, pages 1–10. IEEE, 2009.

[19] E. L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282):457–481, 1958.

[20] E. Kouters, B. Vasilescu, A. Serebrenik, and M. G. J. van den Brand. Who's who in gnome: Using LSA to merge software repository identities. In *ICSM*, pages 592–595, 2012.

[21] K. Lakhani and R. G. Wolf. Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. *Social Science Research Network*, 49(September):1–27, 2003.

[22] E. G. Lambert, N. L. Hogan, and S. M. Barton. The impact of job satisfaction on turnover intent: A test of a structural measurement model using a national sample of workers. *The Social Science Journal*, 38(2):233–250, 2001.

[23] Y. Li, C.-H. Tan, and H.-H. Teo. Leadership characteristics and developers' motivation in Open Source Software development. *Information & Management*, 49(5):257–267, 2012.

[24] A. Martensen and L. Grønholdt. Internal marketing: a study of employee loyalty, its determinants and consequences. *Innovative Marketing*, 2(4):92–116, 2006.

[25] B. Mehlenbacher. Documentation: not yet implemented, but coming soon. *The HCI Handbook: Fundamentals, Evolving Technologies, and Emerging Applications*, pages 527–543, 2003.

[26] T. Mens. Evolving software ecosystems A historical and ecological perspective. In M. Irlbeck, D. A. Peled, and A. Pretschner, editors, *Dependable Software Systems Engineering*, volume 40 of *NATO Science for Peace and Security Series, D: Information and Communication Security*, pages 170–192. IOS Press, 2015.

[27] T. Mens, M. Claes, P. Grosjean, and A. Serebrenik. Studying evolving software ecosystems based on ecological

models. In T. Mens, A. Serebrenik, and A. Cleve, editors, *Evolving Software Systems*, chapter 10, pages 297–326. Springer, 2014.

[28] V. Midha and P. Palvia. Retention and quality in Open Source Software projects. *AMCIS 2007 Proceedings*, page 25, 2007.

[29] V. Midha, R. Singh, P. Palvia, and N. Kshetri. Improving open source software maintenance. *Journal of Computer Information Systems*, 50(3):81–90, 2010.

[30] M. Nagappan, T. Zimmermann, and C. Bird. Diversity in software engineering research. In *ESEC/FSE*, pages 466–476, New York, NY, USA, 2013. ACM.

[31] F. Ortega and D. Izquierdo-Cortazar. Survival analysis in open development projects. In *FLOSS*, pages 7–12. IEEE, 2009.

[32] J. T. Rich, J. G. Neely, R. C. Paniello, C. C. Voelker, B. Nussenbaum, and E. W. Wang. A practical guide to understanding Kaplan-Meier curves. *Otolaryngology-Head and Neck Surgery*, 143(3):331–336, 2010.

[33] D. Riehle. How Open Source is changing the software developer's career. *IEEE Computer*, 48(5):51–57, 2015.

[34] D. Riehle, P. Riemer, C. Kolassa, and M. Schmidt. Paid vs. volunteer work in Open Source. In *HICSS*, pages 3286–3295. IEEE, 2014.

[35] G. Robles, J. M. Gonzalez-Barahona, and J. J. Merelo. Beyond source code: The importance of other artifacts in software development (a case study). *Journal of Systems and Software*, 79(9):1233–1248, 2006.

[36] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.

[37] I. Samoladas, L. Angelis, and I. Stamelos. Survival analysis on the duration of Open Source projects. *Information and Software Technology*, 52(9):902–922, 2010.

[38] A. Schilling, S. Laumer, and T. Weitzel. Who Will Remain? An Evaluation of Actual Person-Job and Person-Team Fit to Predict Developer Retention in FLOSS Projects. In *HICSS*, pages 3446–3455, 2012.

[39] A. Schilling, S. Laumer, and T. Weitzel. Together but apart: How spatial, temporal and cultural distances affect FLOSS developers' project retention. In *Proceedings of the 2013 annual conference on Computers and people research*, pages 167–172. ACM, 2013.

[40] P. N. Sharma, J. Hulland, and S. Daniel. Examining Turnover in Open Source Software Projects Using a Logistic Hierarchical Linear Modeling Approach. In *OSS*, volume 378 of *IFIP Advances in Information and Communication Technology*, pages 331–337, 2012.

[41] B. Shibuya and T. Tamai. Understanding the process of participating in Open Source communities. In *FLOSS*, pages 1–6. IEEE Computer Society, 2009.

[42] D. Spinellis. Global Software Development in the FreeBSD project. In *International workshop on Global software development for the practitioner*, pages 73–79. ACM, 2006.

[43] I. Steinmacher, M. A. Gerosa, and D. Redmiles. Attracting, onboarding, and retaining newcomer developers in Open Source Software projects. In *Workshop on Global Software Development in a CSCW Perspective*, 2014.

[44] J. Teixeira. Understanding coopetition in the open-source arena: The cases of webkit and openstack. In *Proceedings of The International Symposium on Open Collaboration*, page 39. ACM, 2014.

[45] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov. Gender and Tenure Diversity in GitHub Teams. In *CHI*, pages 3789–3798, 2015.

[46] B. Vasilescu, A. Serebrenik, M. Goeminne, and T. Mens. On the variation and specialisation of workload – A case study of the GNOME ecosystem community. *Empirical Software Engineering*, 19(4):955–1008, 2013.

[47] S. Wellek. A log-rank test for equivalence of two survivor functions. *Biometrics*, pages 877–881, 1993.

[48] S. G. Westlund and J. C. Hannon. Retaining talent: Assessing job satisfaction facets most significantly related to software developer turnover intentions. *Journal of Information Technology Management*, 19(4):1–15, 2008.

[49] I. S. Wiese, J. Teodoro, I. S. da Silva, C. Treude, and M. A. Gerosa. Who is who in the mailing list? Comparing six disambiguation heuristics to identify multiple addresses of a participant. In *ICSME*, page 13. IEEE, 2016.

[50] C.-G. Wu, J. H. Gerlach, and C. E. Young. An Empirical Analysis of Open Source Software Developers' Motivations and Continuance Intentions. *Information and Management*, 44(3):253–262, 2007.

[51] K. Yamashita, Y. Kamei, S. McIntosh, A. E. Hassan, and N. Ubayashi. Magnet or sticky? Measuring project characteristics from the perspective of developer attraction and retention. *Journal of Information Processing*, 24(2):339–348, 2016.

[52] D. Yang, T. Sinha, D. Adamson, and C. P. Rose. Turn on, tune in, drop out: Anticipating student dropouts in massive open online courses. In *NIPS Data-Driven Education Workshop*, pages 13–20, Lake Tahoe, NV, USA, 2013.

[53] M. Zhou and A. Mockus. Who will stay in the floss community? modeling participant's initial behavior. *TSE*, 41(1):82–99, Jan 2015.

[54] M. Zhou, A. Mockus, X. Ma, L. Zhang, and H. Mei. Inflow and retention in oss communities with commercial involvement: A case study of three hybrid projects. *TOSEM*, 25(2):13:1–13:29, Apr. 2016.