Issue Dynamics in Github Projects

Riivo Kikas, Marlon Dumas, and Dietmar Pfahl

Institute of Computer Science, University of Tartu, Estonia
{riivokik,marlon.dumas,dietmar.pfahl}@ut.ee

Abstract. Issue repositories are used to keep of track of bugs, development tasks and feature requests in software development projects. In the case of open source projects, everyone can submit a new issue in the tracker. This practice can lead to situations where more issues are created than what can be effectively handled by the project members, raising the question of how issues are treated as the capacity of the project members is exceeded. In this paper, we study the temporal dynamics of issues in a popular open source development platform, namely Github, based on a sample of 4000 projects. We specifically analyze how the rate of issue creation, the amount of pending issues, and their average lifetime evolve over the course of time. The results show that more issues are opened shortly after the creation of a project repository and that the amount of pending issues increases inexorably due to forgotten (unclosed) issues. Yet, the average issue lifetime (for issues that do get closed) is relatively stable over time. These observations suggest that Github projects have implicit mechanisms for handling issues perceived to be important to the project, while neglecting those that exceed the project's capacity.

1 Introduction

Issue trackers have become essential collaboration instruments in modern software development projects [1]. They are used for registering and tracking new feature requests, development tasks and bugs for example. In closed-source projects, usage of issue trackers is generally restricted and sometimes codified, so that new issues can only be opened by development team members, managers and a reduced set of stakeholders and may need to comply with established norms and minimum requirements [1].

On the other hand, in open source projects, for example in Github projects, it is common practice that everyone can open a new issue in the issue tracker of a project with basically no requirements placed on the content and quality of new issues [4,2]. This practice can lead to a potentially large and continuous inflow of issues exceeding the project's development team capacity, including low-quality issues or issues that are only marginally relevant to the project. As the inflow of issues exceeds the capacity of the project members, it is natural to conjecture that not all issues are effectively handled, and are either closed without resolution or implicitly ignored.

This paper aims to shed light into the extent to which open source projects, particularly those hosted in Github, cope with the inflow of issues they are subjected to throughout their lifetime. Based on a sample of more than 4000 Github projects, we analyze the temporal dynamics of issues in terms of how often they are created (arrival rate), the amount of pending issues, and their lifetime. Specifically, the paper addresses the following research questions:

- RQ1: What is the issue arrival rate and how does it change over time?
- RQ2: How do opened and pending issue numbers evolve over time?
- RQ3: What is the average issue lifetime and how does it change over time?

The rest of the paper is structured as follows. The next section presents related work. Then, Section 3 describes the dataset employed and the concepts used in the analysis. In Section 4 we present and discuss the results of our analysis with respect to the above research questions. Finally, Sections 5 and 6 respectively provide an analysis of threats to validity and present conclusions and opportunities for future work.

2 Related Work

There are several previous studies that focus explicitly on the analysis of issue in Github datasets. Closest to our research is a study by Bissyande et al. [2], giving a basic overview of issue tracker usage in 100,000 Github projects. Their work explores how many issues are tracked on average, labels and tags usage, who enters issues (developer or not), issue tracker usage and project success (number of watchers), and user community size and issue fix time. Compared to our work, they do not analyze the evolution of pending issues and their lifetimes. Cabot et al. [3] studies how tagging is used in Github issue trackers. Their results show that only small sets of projects use labels, and usage of labels correlates with a higher number of closed issues. Related to this, Izquierdo et al. [9] have presented a tool demo to explore issue label usage in projects.

The question of issue lifetime has been studied from different perspectives both in open source and closed source projects. Marks et al. [12] studied issue lifetimes in Mozilla and Eclipse. They found that 46% of bug reports in the Mozilla project and 76% of bug reports in the Eclipse project are closed within three months of their creation. Grammel et al. [8] study community involvement in closed source IBM Jazz projects. Their findings suggest that community created issues can be valuable, but they are handled differently than those created by project members. The average issue lifetime for community created issues is 39 days, whereas for the team issues it is 5.9 days.

Ko and Chilana [11] study why some issues in the Mozilla tracker are left open for long periods. Their findings suggest that issues resulting in fixes or code changes are proposed by a "group of experienced, frequent reporters". They conclude that open source projects benefit most from this group of experts.

Garousi [5] studies three open source projects with a focus on issue creation and resolution times. Their findings show that bugs and critical issues are handled faster than other issue types. For the jEdit and DrPython projects, the fractions of issues that are closed within the first day is 23% and 42% percent, respectively. Garousi also concludes that more bugs are submitted in the beginning of a project's lifetime. In addition, he shows that issues pile up over time and then are closed in batches.

Compared to these previous studies on issue lifetime, we employ a larger volume of projects and we consider not only issue lifetime, but also arrival rate and number of pending issues. The latter variable ("number of pending issues") has been studied separately by Kenmei et al. [10], who use a time-series modeling to analyze how the number of opened issues changes over time.

Other studies have considered how the arrival rate of new issues in a project and their resolution time can be used to plan future work [10] and to predict the lifetime of pending issues [13,6]. These latter studies are orthogonal to ours.

3 Dataset and Terminology

Github is an online platform for hosting git source code repositories. It offers a web interface for repositories, an issue tracker, and a mechanism for pull based software development and code review. Github has been gaining popularity in recent years, with usages from single-person projects to major companies such as Google and Microsoft using Github for hosting their open source projects.

We use a dataset from Github, collected by the GHTorrent project team [7]. Github provides a public stream of actions such as creation of new pull requests, commits, issues, but does not, for example, provide issue content text. GHTorrent project is augmenting this data by crawling the Github API to retrieve past data and consolidates the data from both sources into a single database.

3.1 Data Extraction and Filtering

Our GHTorrent data extraction is from April 2, 2015 and contains more than seven million projects (not counting forked projects). In order to exclude special cases and anomalies, we selected a subset of the extracted data, applying following filtering criteria.

- Projects must have been created between January 2012 and December 2014. We limited our observation period to this interval, because older data is often not fully available due to the crawling behavior of GHTorrent.
- Projects must have at least 100 opened issues and one closed issue. This criterion gives us projects that actually use the issue tracking capabilities.
- Projects must have at least five commits to the main repository. This criterion gives us projects where some issues generated development activity.
- Projects must not show any activity before repository creation date. In Github, it is possible to fork a repository and therefore inherit an already existing codebase which technically shows up as code committed before the project creation.

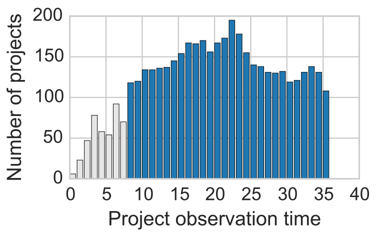
An examination of the selected data revealed projects with unexpectedly high issue tracker activity in short periods, such as several thousands of newly created issues during a single day. This can be caused by a data import from old tracking system or automatically created issues by using the Github API. To get rid of possible import behavior, we additionally filtered out projects that created or closed more than 2000 issues in a single month or created more than 500 issues during a single day.

3.2 Descriptive Statistics

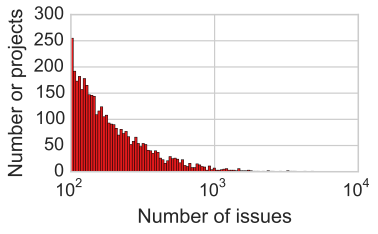
After filtering, 4452 projects met our criteria. Figure 1a shows the distribution of the projects' observation time lengths. Our sample contains projects with observation times ranging from 0 to 35 months. We observed that there were relatively few projects with a short observation time. We wanted to have at least 100 projects within each observation time interval. This resulted in the removal of all projects with observation less than eight months, i.e., projects created after April 2014. In Figure 1a, projects that were filtered out due to their observation time are marked in gray color.

The final dataset contains of 4024 projects, with 967,037 total issues, of which 675,970 (69.9%) have been closed at least once and 291,067 (30.1%) have not been closed during the observation period. The mean number of issues per project is 240.3. Issues can be re-opened and re-closed, and this affects 27376 (4.0%) issues.

The number of issues per project (Figure 1b) varies almost two orders of magnitude with the largest project having 4885 issues in total.



(a) Number of projects per project observation bucket (in months). Dark colored bars represent the sample used in the analysis.



(b) Number of projects per number of opened issues bucket (logarithmic scale).

Fig. 1. Basic properties of the dataset.

3.3 Terminology

The centerpiece of our analysis are issues, i.e., bug reports, new feature requests, and development related changes such as refactoring. In Github, issues are typically free text, can be submitted by anyone, support commenting, and can be referenced from other issues. The dataset does not give any information about issue text or content, but it records when an issue was created and by whom, and a set of events associated such as closing, reopening, being commented, being referenced from another issue. All these events are accompanied with the time of the action and which user is responsible for it.

In this paper we distinguish the following issue states:

- **Opened issue** - Newly created issue. Each issue is opened only once during its lifetime.
- **Pending issue** - Issue that has been opened but not yet closed. These issues denote unresolved cases that need attention or actual work.
- **Sticky issue** - Issue that does not get closed during our observation period. Sticky issues are a subset of pending issues.
- **Closed issue** - Issue that is marked closed in the issue tracking system. In practice an issue might be reopened and closed again, but here we use only the first closing event. We do not distinguish closed issues based on the resolution type, meaning that a closed issue might have been closed after the bug was fixed or closed without any activity.

One might consider our notion of pending issue as too simplistic since we do not take into account reopening and re-closing. The justification for this is the fact that reopening and re-closing affects only 4% of all issues.

Our dataset contains projects that are created at different points in time during our observation period and therefore have varying time periods during which we could observe project behavior. Below we list our time related terminology:

- **Issue lifetime** - Time from the first opening of the issue to the first closing of the issue.
- **Project observation time** - Number of months between project creation and the end of observation period (December 31, 2014). This number is obtained by calculating the number of days between the two dates, dividing by 30.4 (the average number of days in a month), and rounding down to the nearest integer.
- **Relative time** - Each project is transformed into a relative timescale. The relative timescale starts from repository creation, and after each 30.4 days, a new relative month starts. This results in the last month typically not being a full month, as projects can start on any day during a month, but our observation period ends with the 31st day of a month. Relative time 'zero' represents the first month of the project observation time, relative time month 'one' represents the second month of the project observation time.
- **Observation period** From January 2012 until end of December 2014. This is the period we have data about projects and we can use for the analysis.

3.4 Notations

In our analysis, we focus on the following metrics over time: opened issued, sticky issues, and pending issues. In the following, we give the definitions of these metrics.

Let N be the set of projects and $\mathcal{T}$ the set of all possible project observation times. Each project $i \in N$ has observation time of $T_i \in \mathcal{T}$.

We denote a single issue as a tuple (a_j, b_j) where j is a unique issue identifier, and a_j and b_j denote opening and closing times since project creation (measured in minute resolution). For each issue, it must hold $(a_j \leq b_j) \vee (a_j \leq T_i \wedge b_t = nil)$. Let PI_i denote the set of issues associated with project i. Even though T_i has discrete values, we assume that a_j and b_j are continuous and have minute level resolution in order to be able to derive exact ordering between closing and opening. Let $m(a_j)$ denote the corresponding relative month of a_j and $m^{-1}(t)$ the value in minutes for the corresponding month end date.

Let $o_{i,t}$ denote the number of total newly opened issues for a project i at a relative time $t \in \mathcal{T}$, then

$$o_{i,t} = |\{(a_j, b_j)|(a_j, b_j) \in PI_i, m(a_j) = t\}|.$$

We use $s_{i,t}$ to denote sticky issues, i.e.,

$$s_{i,t} = |\{(a_j, b_j)|(a_j, b_j) \in PI_i, m(a_j) \leq t \wedge b_j = nil\}|.$$

It represents sticky issues as total number of sticky issues by the end of month t.

Let $p_{i,t}$ denote the number of pending issues at time t. The number of pending issues is the number of opened - but not yet closed - issues at a certain point of time (measured in minute resolution). Thus, we devise the number of pending issues $p_{i,t}$ for a project i during a month t as follows:

$$p_{i,t} = \frac{1}{m^{-1}(t-1) + 1 - m^{-1}(t)} \sum_{d=m^{-1}(t-1)+1}^{d \leq m^{-1}(t)} \delta_{i,d}$$

where $\delta_{i,d}$ denotes the number of open issues for project i at minute resolution d, i.e.

$$\delta_{i,d} = |\{(a_j, b_j)|(a_j, b_j) \in PI_i, a_j \leq d \wedge (b_j = nil \vee b_j > d)\}|.$$

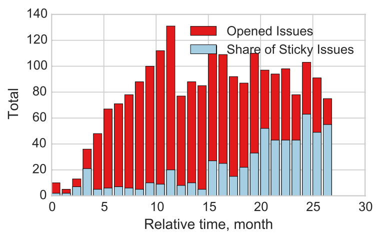
Finally, issue lifetime for issue j, denoted by LT_j, is the amount of days between issue creation and closing and can be calculated only for closed issues, i.e., $b_j \neq nil$:

$$LT_j = (b_j - a_j)/(60 * 24).$$

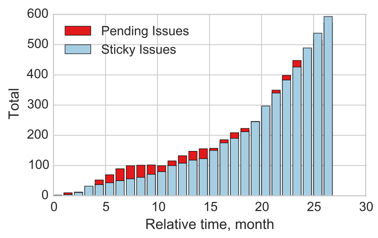
3.5 Examples

We illustrate our concepts with the help of an example project, Bootstrap[1], a front-end framework for creating user-interfaces in browsers.

Figure 2a shows the numbers of opened and sticky issues observed per month. Note that here we only show the share of sticky issues that correspond to issues opened in the month of observation. Each bar on the plot corresponds to $o_{i,t}$ and $s_{i,t} - s_{i,t-1}$ (except for the case $t = 0$, the sticky issues is equal to $s_{i,t}$). We see that Bootstrap had increasing numbers of opened issues during the first year of observation, then the number of opened issues leveled off. The monthly share of sticky issues started to rise around month 20. One reason for this could be that our observation period limits the available time for observing issues opened after month 20 being closed. Figure 2b plots the number of pending and sticky issues observed over time. We see that the number of pending issues increases and the majority of pending issues is made up by sticky issues. When comparing the relative share of sticky issues with opened issue per month, we observe that the majority of opened issues get closed, but the amount of work still to be done, represented by the amount of pending issues, is continuously increasing due to the amount of sticky issues.



(a) Opened issues for Bootstrap.



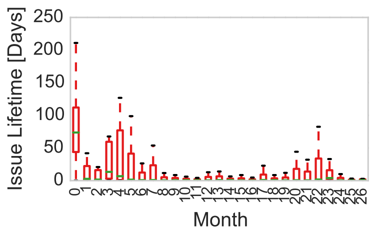
(b) Pending issues for Bootstrap.

Fig. 2. Opened, pending, and sticky issues for Bootstrap.

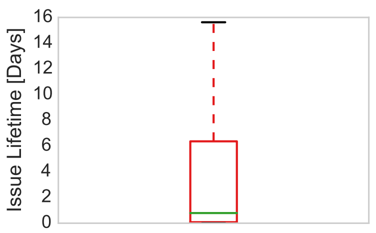
Figure 3a shows issue lifetime distributions as boxplots for groups of issues opened in a specific month of the project observation time (relative time in months). Note that in Figure 3a month = 0 is an abbreviation for the time up to the beginning of month = 1, i.e., representing the observation time interval [0, 1) months. Issue lifetimes remain stable on average over the project observation time (mean lifetime equals 12.9 days, median lifetime equals 0.78 days). We see, however, that issues created in month = 0 have a considerably higher average lifetime than those created in later months. One possible explanation is that during first month issues are entered that require additional development and this usually takes more time as simply correcting a bug. In this particular example, however, there were only 8 opened issues in month = 0. Therefore, this

[1] `https://github.com/twbs/bootstrap`

is not an important phenomenon. The overall issue lifetime distribution, shown in Figure 3b, indicates a small median (0.78 considering all issues) but large variation.



(a) Issue lifetime distributions over time for Bootstrap.



(b) Issue lifetime distribution for Bootstrap (all issues).

Fig. 3. Issue lifetime for Bootstrap. For both figures, outliers are removed. The maximums correspond to $1.5 * (75p - 25p) + 75p$, where 25p and 75p denote corresponding percentiles.

4 Results

In the following subsections we answer the research questions outlined in the introduction. First, we look at the opened issues rates, then we analyze the pending issues, and finally we analyze the issue lifetime distributions.

4.1 Issue Arrival Rate (RQ1)

To answer RQ1 we investigate the arrival rates of opened issues in our set of Github projects. We analyze projects with different project observation times separately, because we suppose that the length of the observation time has an effect on the opened issues. For example, projects with short observation times might (on average) have different numbers of opened issues during the first months of the observation time due to increased popularity of Github and the size and type of projects hosted in Github.

We classify projects based on observation times into buckets and calculate the average number of opened issues per month for all projects in a bucket separately. Figure 4 shows a line for each group of projects in the same bucket. There are $\mathcal{T}$ buckets in total and for each line l representing a bucket, a point t on the line represents the average number of opened issues at relative time since creation of the projects in the bucket (i.e., the start of the project observation time), given by the following formula:

$$\mathcal{O}_{t,l} = \frac{1}{\sum_{i \in N \wedge T_i = l \wedge t \leq l} 1} \sum_{i \in N \wedge T_i = l \wedge t \leq l} o_{i,t}$$

Looking at Figure 4, we observe a relatively higher average number of opened issues right after project creation as compared to a few months after project creation. This tendency is visible for all project buckets but most explicit for buckets of projects with shorter observation time. Overall, we see that the average number of opened issues is stable or shows slight negative trend over project observation time. For the first month, a project in Github receives on average 19.7 new issues, but one year later, during the 12th month, it receives 10.3 opened issues. The relative decline of opened issues after the first few months might be explained as a start-up effect, i.e., at the beginning many issues are submitted but never worked on because they represent wishful thinking regarding features to be included in the project. Furthermore, we observe that projects in buckets with shorter observation time seem to have significantly higher numbers of opened issues than those in buckets with longer observation times. For example, in the first month after project creation, projects in buckets with the shortest observation time have on average more than five times more opened issues than projects with the longest observation time. This might be explained by the relative growth of Github and the increasing size of projects in recent time. The drop-offs in the last months are caused due to a technical artifact. Namely, our relative time line starts with repository creation, and due to this, the last month is probably not a full month, as the observation period still ends during the end of a month. In addition, we observe that for some groups, there are outlier months, for example the outlier at month 12. We observe stable average numbers of opened issues for all buckets with relative times after month 28.

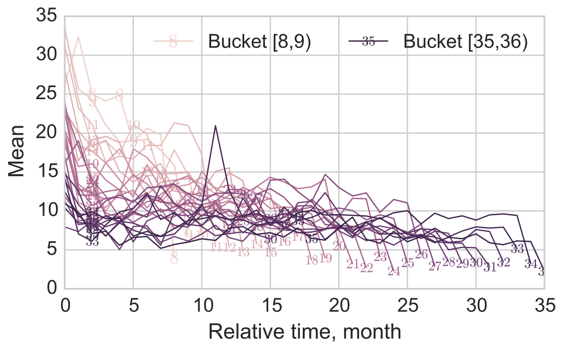


Fig. 4. Opened issues for projects with different lifetime. Color intensity varies with observation period length.

To understand what is the ratio of (the share of) sticky issues to opened issues for projects with different observation times, we calculated the ratio of sticky to opened issues for different buckets. In Figure 5, we display the ratios for each bucket and display the corresponding lines in four different graphs, each graph containing a subset of buckets. The purpose is to make patterns between groups of buckets more visible. The ratio of sticky to opened issues varies mostly between 0.2-0.6, meaning that still more issues get closed than stay open during observation time. We observe that the ratio of sticky issues is higher for early months, then levels off and finally starts to rise due to the technical effect that project observation time ends. The exception is the set of recent projects with short observation times. Compared to other buckets, projects in buckets with observation times 8-14 have less sticky issues in the early months followed by a steady growth of the ratio. One possible explanation for this phenomenon might be that issue submitters in recent projects are more realistic in their issue management and do not fill the issue tracker with issues that will never be worked on or are resolved only after a long time. Another explanation could be that due to the growth of the projects maintained in Github, more development capacity is available to work on issues and thus issues receive more attention and are resolved more quickly. This explanation, however, would not explain why the ratio for projects with short observation time strongly grows after a few months.

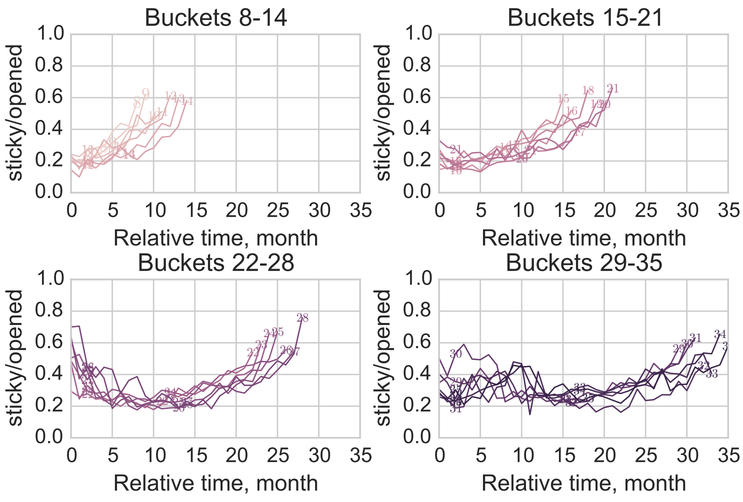


Fig. 5. Ratio of (shares of) sticky issues to opened issues.

To answer RQ1, we can conclude that the monthly rate of opened issues for projects in our data set decreases over time on average. During first 12 months, the average number of issues opened drops roughly by a factor of two. The ratio of sticky issues to opened issues changes differently over time for projects with shorter observation time as compared to projects with longer observation time.

4.2 Pending Issue Growth (RQ2)

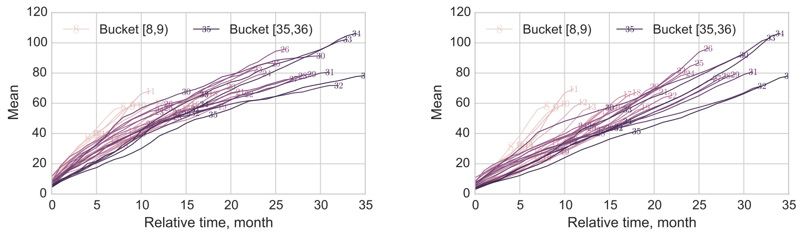
To answer RQ2 we analysed the dynamics of pending issues and sticky issues over time. Again, we consider projects with different observation times separately.

In Figure 6a, displaying the average numbers of pending issues over time, we classified projects with different observation times into buckets. For each bucket we show the average of pending issues over time as different lines T_i, the values of each line defined as:

$$P_{t,l} = \frac{1}{\sum_{i \in N \wedge T_i = l \wedge t \leq l} 1} \sum_{i \in N \wedge T_i = l \wedge t \leq l} p_{i,t}$$

where l denotes the project observation time and t is the relative time for projects with observation time l, thus also it must always hold that $t \leq l$. Similarly, in Figure 6b we plot the total number of sticky issues over time for each bucket.

One can see that pending issues are growing at approximately constant rates for all buckets. This phenomenon is underpinned by the growth pattern of sticky issues, issues that have not been resolved within our observation period.



(a) Pending issues. (b) Sticky issues.

Fig. 6. Pending and sticky issues. Even though, sticky issues are a subset of pending issues, we have plotted them on separate Figures for clarity.

The growth rates for both pending and sticky issues are different between buckets. Generally, there seems to be a tendency that more recent projects (shorter project observation time) have on average higher growth rates in the early months of observation time. On the other hand, there exist strong differences between growth rates of buckets with just one month difference of project observation time. For example projects in bucket 34 have in time interval [34, 35) 25 percent more pending (and sticky) issues than projects in bucket 35 in the same time interval.

As an answer to RQ2, we can say that pending issues are growing constantly and this comes mostly from the sticky issues, that do not get resolved during our observation period.

4.3 Issue Lifetime (RQ3)

So far, we observed that there is a steady arrival of new issues and an increasing number of pending (and sticky) issues. Although the amount of pending issues grows, most opened issues actually get resolved (closed) during project observation time. In this section, we answer RQ3 by analyzing issue lifetimes in Github projects.

Figure 7 shows issue lifetime distributions for issues opened during each month (relative time from the start of project observation time) of all projects in our data set, i.e., we do not classify data into buckets with the same project observation times. We consider all issues that are not sticky. Each boxplot represents issue lifetimes for corresponding ILT_t group, defined as $ILT_t = \{LT_j | (a_j, b_j) \in PI_i, i \in N, b_j \neq nil, m(a_j) = t\}$.

In Figure 7 one sees variation in maximum values but medians vary little over time. The median issue lifetime is 3.1 days for issues opened in month 0 (time interval [0, 1)), 4.1 days for issues opened in month 10 (time interval [9, 10)), 2.89 for issues opened in month 20 (time interval [19, 20)), and 1.78 for issues opened in month 30 (time interval [29, 30)). Thus, one can observe a slight increase from month zero until month 10 in the median value, and then a slight decrease. Around month 20, the median starts to drop more, but this might be a technical effect, caused by the fact that issued opened towards the end of our observation, in order to be included in the lifetime measurement must have been closed before the end of the observation period, thus, leaving out all issues that might be closed after a longer lifetime.

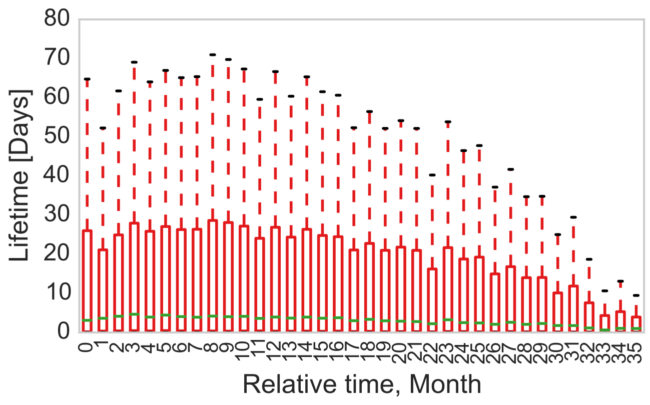


Fig. 7. Issue lifetime distribution depending on issue creation month, each month t lists distribution lifetime of ILT_t. Outliers have been removed.

Figure 8 shows distributions of issue lifetimes for groups of projects with the same project observation times l. Each boxplot represents the issue lifetime distribution for the set ILL_l, defined as $ILL_l = \{LT_j | (a_j, b_j) \in PI_i, i \in N, b_j \neq nil, T_i = l\}$. The observation that variation of issue lifetimes is growing for projects with longer observation time is not surprising, as in projects with longer observation time there is more time available to solve an issue. On the other hand, the median values seem to be stable. The median for projects with observation time 8 months is 2.64 days, for projects with observation time 20 months it is 3.4 days, and for projects with observation time 30 months it is 3.61 days. The variation over all projects is small, considering that the theoretical maximum difference can be more than four times for projects with observation time 8 months and projects with observation time 35 months.

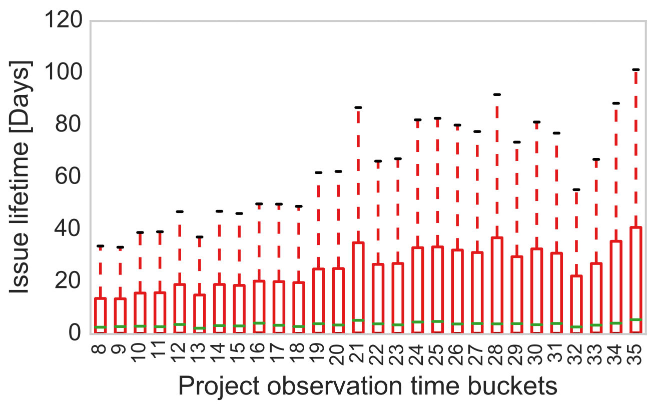


Fig. 8. Issue lifetime distribution depending on the project lifetime, each lifetime bucket l lists distribution of lifetime for issues in ILL_l.

As in the previous section, we also analyzed different project buckets separately. In Figure 9, we show median issue lifetimes for each group of projects with different observation times. Formally, there are again T lines in total and for each line l, we calculate a median for set of issues $MLT_{l,t}$, defined as $MLT_{l,t} = \{LT_j | (a_j, b_j) \in PI_i, i \in N, b_j \neq nil, T_i = l, m(a_j) = t\}$. We observe few outliers distort the big picture. Ignoring those outliers, we observe that medians are stable. The drop in the last months of the observation times results to some extent from the fact that issues requiring longer time for closing are excluded from the analysis. Interestingly, median values for projects in different buckets are very similar, especially true for projects with observation times between 15 and 35.

To answer RQ3, we can conclude that issue lifetimes are stable over project observation times.

4.4 Discussion

Our results partly confirm and partly extend published research. For example, our results related to RQ1 confirm results of Kenmei et al. [10] who studied trends of newly opened issues. They found for some systems (e.g., JBoss) that there is an increasing trend and for others (e.g., Mozilla) that there is not. Similarly, our results, which are averages over large numbers of projects do not show a trend of increasing numbers of opened issues over time for projects with comparable

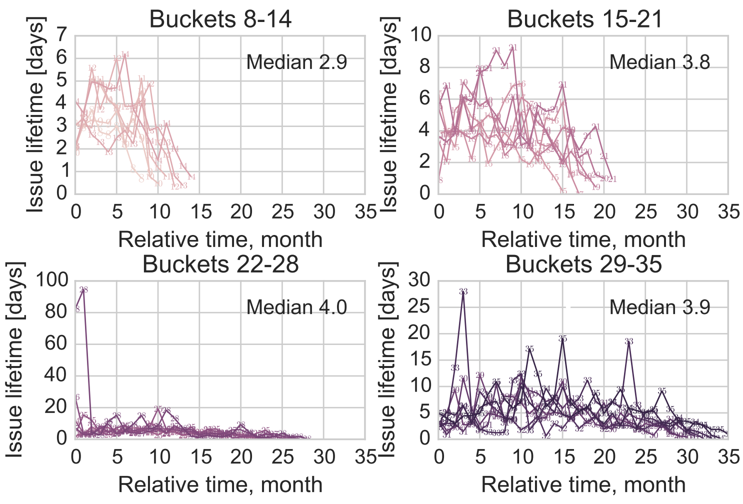


Fig. 9. Median issue lifetimes. For each group, we have calculated the median issue lifetimes over all issues over all months in that group.

observation time. However, we found that more recent projects, i.e., projects with a shorter observation time in our study period, tend to have generally higher volumes of opened issues than older projects, i.e., projects with longer observation time. Also, more recent projects seem to have a decreasing trend in numbers of opened issues, while older projects show a more stable behavior.

Garousi [5] analyzed three open source projects and found evidence of increasing work and short issue lifetimes. He showed that for jEdit and DrPython projects, the fraction of issues that are closed within the first day is 23% and 42% percent respectively. Related to RQ2, we found that for Github projects, the median of issue lifetimes varies between 2 to 4 days, thus more than 50% of issues get closed between at the latest after 4 days, and in many cases earlier. Thus, our results are closer to those of Garousi [5] as compared to those of Marks et al. [12] who found that for Mozilla 46% of the bugs and for Eclipse 76% of the bugs are closed after three months of bug creation. Grammel et al. [8] findings regarding closed source IBM Jazz projects suggest that community created issues can be valuable, but they are handled differently than those created by project members. The average issue lifetime for community created issues is 39 days, but for the team issues it is 5.9 days. These results are comparable to the 12.9 average issue lifetime we observed for the Bootstrap project. However, we also saw that median values are much smaller (e.g., 0.78 days for Bootstrap) and issue lifetime distributions extremely long-tailed. Thus, reporting mean values of issue lifetime might not be useful.

Our results regarding the trend of increasing pending and sticky issues over time (RQ2) seems to be related to the observation of highly positively skewed distributions of issue lifetimes.

5 Threats to Validity

In this section we briefly discuss threats to validity that may affect our results. *Construct validity* threats concern the relationship between theory and observation. In our study, these threats can be mainly due to the way how we measure the various types of opened, closed, pending, and sticky issues as well as to the quality of the data extracted from Github. Also, the fact that we neither distinguish between types and sizes of projects nor issue categories like bug fixes, enhancements, refactoring, and so on. We tried to address the issue of data quality by defining exclusion criteria that filter out projects with certain data anomalies, e.g., low activity, too small size, issue imports due to changes in issue tracker system used.

External validity concerns the generalization of the findings. Different to most of the studies presented in related work, our results rely on the analysis of more than 4000 Github projects from a time period of three years. We believe that our results are to some degree representative for open source projects in general. However, we noticed that there is some variation between projects depending on the length of the observation time. Also, we do not distinguish between types of projects and application domains. Finally, we would like to point out that closed source projects might have different issue behaviors due to the more controlled environment in which those projects are conducted.

Internal validity concerns external factors that may affect an independent variable. *Conlusion validity* concerns the correct application and interpretation of statistical methods. Both validity aspects are less relevant for our study since we neither conduct experiments nor statistical tests.

6 Conclusion and Future Work

Issue trackers are important for software projects to manage bugs and as a general task list indicating development actions needed to be done.

We analyzed issue dynamics of more than 4000 Github projects. Understanding issue volumes and issue lifetimes can be a source for understanding project performance and planning project resources. Once typical evolution patterns for issues are better understood, they might become an indicator for the state of a project and its future outlook.

We identified the following areas for future work. First, future work should look more into the semantics behind issues and distinguish between issue categories such as bugs or feature requests. The next step would be to study how issue growth actually impacts projects in terms of code changes, new releases or even project popularity.

7 Acknowledgement

This research was supported by the Estonian Research Council and by ERDF via the Software Technology and Applications Competence Centre - STACC.

References

1. Bertram, D., Voida, A., Greenberg, S., Walker, R.: Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams. In: Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW. pp. 291–300. ACM, Savannah, Georgia, USA (2010)
2. Bissyande, T.F., Lo, D., Jiang, L., Reveillere, L., Klein, J., Le Traon, Y.: Got issues? who cares about it? a large scale investigation of issue trackers from github. In: Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on. pp. 188–197. IEEE (2013)
3. Cabot, J., Canovas Izquierdo, J.L., Cosentino, V., Rolandi, B.: Exploring the use of labels to categorize issues in open-source software projects. In: Software Analysis, Evolution and Reengineering (SANER), 2015 IEEE 22nd International Conference on. pp. 550–554. IEEE (2015)
4. Crowston, K., Annabi, H., Howison, J.: Defining open source software project success. ICIS 2003 Proceedings p. 28 (2003)
5. Garousi, V.: Evidence-based insights about issue management processes: an exploratory study. In: Trustworthy Software Development Processes, pp. 112–123. Springer (2009)
6. Giger, E., Pinzger, M., Gall, H.: Predicting the fix time of bugs. In: Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering. pp. 52–56. ACM (2010)
7. Gousios, G., Spinellis, D.: Ghtorrent: Github's data from a firehose. In: Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on. pp. 12–21. IEEE (2012)
8. Grammel, L., Schackmann, H., Schröter, A., Treude, C., Storey, M.A.: Attracting the community's many eyes: an exploration of user involvement in issue tracking. In: Human Aspects of Software Engineering. p. 3. ACM (2010)
9. Izquierdo, J.L.C., Cosentino, V., Rolandi, B., Bergel, A., Cabot, J.: Gila: Github label analyzer. In: Software Analysis, Evolution and Reengineering (SANER), 2015 IEEE 22nd International Conference on. pp. 479–483. IEEE (2015)
10. Kenmei, B., Antoniol, G., Di Penta, M.: Trend analysis and issue prediction in large-scale open source systems. In: Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on. pp. 73–82. IEEE (2008)
11. Ko, A.J., Chilana, P.K.: How power users help and hinder open bug reporting. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 1665–1674. ACM (2010)
12. Marks, L., Zou, Y., Hassan, A.E.: Studying the fix-time for bugs in large open source projects. In: Proceedings of the 7th International Conference on Predictive Models in Software Engineering. p. 11. ACM (2011)
13. Weiss, C., Premraj, R., Zimmermann, T., Zeller, A.: How long will it take to fix this bug? In: Proceedings of the Fourth International Workshop on Mining Software Repositories. p. 1. IEEE Computer Society (2007)