The Journal of Systems & Software 142 (2018) 237–247

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Controversy Corner

Unusual events in GitHub repositories

Christoph Treude*,[a], Larissa Leite[b], Maurício Aniche[c]

[a] School of Computer Science, University of Adelaide, Adelaide, Australia
[b] Universitat Politècnica de Catalunya, BarcelonaTech, Barcelona, Spain
[c] Software Engineering Research Group, Delft University of Technology, Delft, The Netherlands

ARTICLE INFO

Keywords:
Awareness
Unusual events
GitHub

ABSTRACT

In large and active software projects, it becomes impractical for a developer to stay aware of all project activity. While it might not be necessary to know about each commit or issue, it is arguably important to know about the ones that are unusual. To investigate this hypothesis, we identified unusual events in 200 GitHub projects using a comprehensive list of ways in which an artifact can be unusual and asked 140 developers responsible for or affected by these events to comment on the usefulness of the corresponding information. Based on 2,096 answers, we identify the subset of unusual events that developers consider particularly useful, including large code modifications and unusual amounts of reviewing activity, along with qualitative evidence on the reasons behind these answers. Our findings provide a means for reducing the amount of information that developers need to parse in order to stay up to date with development activity in their projects.

1. Introduction

As part of their work, software developers create, modify, and delete many artifacts in any given day. While some of these artifacts follow regular patterns (e.g., an issue is closed by a new commit addressing the issue, or a pull request is merged quickly after a few code review comments), others are unusual: A difficult issue might take a particularly long time to address, a controversial pull request might attract an unusually large number of comments, and a disruptive commit might add or delete a lot of files at once.

For any developer participating in a large and active software project, it quickly becomes impossible to stay aware of all commits, issues, or pull requests that are being created or edited. Arguably, it is also not necessary to be aware of all details happening in a codebase or issue tracking system, and tools such as dashboards (Treude and Storey, 2010) or event feeds (Fritz and Murphy, 2011) have been designed to abstract away some of the details. In addition to the high-level awareness afforded by such tools, other tools have been proposed to bring developers' attention to activities in a project that have the potential of impacting them directly, such as Brun et al.'s Crystal (Brun et al., 2011) or WeCode (Guimarães and Silva, 2012) by Guimarães and Silva. However, these tools are very specific and provide little information about the project in general.

In a recent study (Treude et al., 2015) investigating the information that developers would like to be kept aware of, *unusual events* emerged from our qualitative data analysis as a major theme. In fact, we coded 121 out of 156 responses to be related to unusual events or one of its sub-codes. Our work identified a few anecdotal examples of such unusual events, namely an unusually long time between commits by a particular developer, an unusual commit message, or changes to a large number of files. Based on the answers (examples of unusual events that developers are interested in), we hypothesize that developers are interested in unusually large or small values for commit- and issue-related metrics (by generalizing the examples). In this work, we provide a systematic empirical investigation of the hypothesis that developers want to be kept aware of such events in their repositories.

Given the amount of data available in repositories on hosting sites such as GitHub, there is a large number of ways in which an artifact can be unusual. For example, a commit might delete an unusually large number of lines of code, an issue might have an unusually large number of labels, or a pull request might have an unusually large number of commits associated to it. In fact, in this work we found that more than half of all commits in a sample of 200 GitHub projects could be considered as unusual according to at least one metric, considering a comprehensive list of metrics that we defined based on previous work and the data available through the GitHub API.

However, we do not claim that all the different ways in which an artifact could be considered as unusual provide useful information to developers. In contrast, the goal of this work is to enumerate the subset of unusual events that developers consider useful to be kept aware of and to identify the reasons why some types of unusual events are useful to know about and others are not. We define an *unusual event* as an

* Corresponding author.
E-mail addresses: christoph.treude@adelaide.edu.au (C. Treude), larissaleite@gmail.com (L. Leite), m.f.aniche@tudelft.nl (M. Aniche).

https://doi.org/10.1016/j.jss.2018.04.063
Received 5 October 2017; Received in revised form 28 March 2018; Accepted 29 April 2018
Available online 04 May 2018

artifact that is unusual in at least one way (e.g., a commit with an unusually large number of files added), and an *unusual event type* as one way in which an artifact could be considered unusual (e.g., unusually large number of files added in a commit). One artifact could be unusual according to more than one unusual event type at any point in time. In this work, we consider commits, issues, and pull requests as artifacts, since they are the main artifacts on GitHub capturing developer activity.

To achieve our research goal of identifying the set of unusual event types that developers consider useful to be kept aware of, we presented 140 developers from 200 randomly sampled GitHub projects with a list of unusual events we had detected in their projects and asked them to rate the usefulness of the corresponding information. Based on a total of 2,096 ratings of different unusual events by the developers that were directly responsible for and/or affected by these unusual events and their reasoning, we compiled a list of types of unusual events that developers want to be kept aware of.

In particular, we investigated the following research questions:

RQ1 How are unusual events perceived by developers?
- **RQ1.1** Are unusual events perceived differently by developers?
- **RQ1.2** How do developers perceive artifacts affected by particular types of unusual events?

RQ2 Which types of unusual events do developers find most useful and why?
- **RQ2.1** Which types of unusual events do developers find most useful?
- **RQ2.2** Why do developers consider types of unusual events useful or not useful, respectively?

We found that information on unusual events in terms of number of lines of code deleted, added, and modified in a commit was considered particularly useful, along with the number of comments on issues and pull requests as well as the duration for which an issue had been open. These are also the types of unusual events that belong to artifacts perceived as difficult.

The contributions of this work are:

- A list of types of unusual events that developers want to be kept aware of, based on empirical evidence,
- the reasons for including and excluding specific unusual event types from this list,
- Data from 200 randomly sampled GitHub projects about the frequency of unusual events and their types, and
- An investigation to what extent different types of unusual events correlate with perceived difficulty and typicality of an artifact.

The remainder of this paper is structured as follows: Section 2 provides motivating examples for this work. In Section 3, we detail our sampling method for GitHub projects and we provide our definition of unusual events. Section 4 provides empirical data on how frequently the various unusual events occur in GitHub projects. Section 5 presents our research questions and methodology, before Section 6 presents the findings which are discussed in Section 7. Section 8 highlights the limitations, and Section 9 summarizes related work. We conclude the paper and outline future work in Section 10.

2. Motivating examples

RxSwift[1] is a GitHub project that ports ReactiveX, an API for asynchronous programming with observable streams, to Swift. When we downloaded its data, the repository contained 1605 commits, 352 issues, and 443 pull requests. A typical issue on RxSwift is closed after being open for less than 5 days (median: 4.65 days, first quartile: 21.74 hours, third quartile: 16.20 days). Considering these numbers, issue #206 is unusual: more than 10 weeks passed between the moment it was opened and the moment it was closed. When we pointed this out to one of RxSwift's contributors, they stated: *"I think the info is really useful actually, having a long standing issue could [...] be an indicator of a difficult issue"*.

Another project we analyzed for this work is LaTeXML,[2] a converter for LaTeX to XML, HTML, and other formats. The corresponding repository contained 4520 commits, 675 issues, and 119 pull requests when we downloaded its data. Out of the 675 issues, 21 were labeled with *wontfix*. These issues usually did not attract much discussion: the median number of comments for these 21 issues was 2, with the first quartile at 1 and the third quartile at 3.5. Issue #724 is unusual in this regard with 13 comments. When we asked one of LaTeXML's contributors about this unusual event, they responded: *"In this case it indicates an interesting discussion that spans beyond the concrete issue"*.

Finally, the Elixir repository[3] on GitHub hosts a dynamic, functional language for building scalable and maintainable applications, with 11,548 commits, 2402 issues, and 2696 pull requests at the time of our data download. Issue #3413 is unusual in terms of time between open and closed with a duration of almost 11 months, considering all issues in this project assigned to GitHub user josevalim. This user typically closes issues in less than 7 days (median: 6.94 days, first quartile: 21.26 hours, third quartile: 36.21 days). Given these numbers, one of his colleagues commented: *"This information is useful. Knowing José [...] closes issues quickly makes it appear that this was a difficult problem"*.

The goal of our work is to provide developers with useful insights such as the ones illustrated in these examples through a systematic investigation of different types of unusual events and their perceived usefulness.

3. Projects and definition of unusual

In this section, we explain our method for sampling GitHub projects and the definition of unusual used in this work.

3.1. Project selection

To systematically investigate awareness of unusual events, we randomly selected 200 original projects (excluding forks) from GitHub, limiting our sample to those projects that had at least 500 commits and at least 100 pull requests or 100 issues. The threshold of 500 commits had been used in previous work (e.g., Aniche et al., 2016) to filter out pet projects and small experiments developers host on GitHub. The additional filter on the number of issues and pull requests ensures that projects use at least one of these mechanisms to manage their development work.

To conduct the project selection, we randomly selected GitHub projects from the entire population of GitHub projects until we had 200 projects that fulfilled our criteria.[4] During this process, we disregarded 129,860 projects because they did not have enough commits and not enough issues or pull requests, 118,873 projects because they were forks, 1335 projects because they did not have enough issues or pulls (but enough commits), and 350 projects because they did not have enough commits (but enough issues or pull requests). In addition, we disregarded 94 projects that had been imported to GitHub using GoogleCodeExporter (i.e., their issue information was from Google Code rather than GitHub), we disregarded one project because it was a book writing project rather than a software project, and we disregarded one

[1] https://github.com/ReactiveX/RxSwift.

[2] https://github.com/brucemiller/LaTeXML.

[3] https://github.com/elixir-lang/elixir.

[4] We performed the random selection by randomly selecting GitHub project IDs between 1 and 70,000,000 and testing whether the corresponding projects fulfilled our sampling criteria.