Bug Seeding, On the importance of Previous Commit

Gema Rodriguez
University King Juan Carlos
Madrid, Spain
gerope@libresoft.es

Jesus M. Gonzalez-Barahon
University King Juan Carlos
Madrid, Spain
jgb@gsyc.es

Gregorio Robles
University King Juan Carlos
Madrid, Spain
grex@gsyc.es

ABSTRACT

Algorithms such as SZZ have been proposed to identify the cause of a *fixed* bug, some of these algorithms rely on the fact that bugs in source code have been originated in the previous commit, understanding previous commit as the fix-inducing commit. However, there has been no enough empirical evidences to prove this assumption, and taking into account that exist projects with different nature, ones more active that others, might happen that this assumption is not fulfilled in projects which present this characteristic.

In this paper, in order to ascertain the validity of such a claim, we have conducted an observational study that involved bug notifications from a very active project such as OpenStack, in which their source code is continuously evolving. Our results are promising in indicating that exists bug fixed which were not introduced in the previous commit.

Keywords

Bug-introduction, SZZ algoritm, Fix-inducing

1. INTRODUCTION

In order to investigate if this assumption is valid, we formulate following research questions:

- RQ1 : How can tickets which are bug reports be differentiated from those that are not?
- RQ2: How often is the bug caused in the previous commit?

2. THE CASE OF STUDY

2.1 Fist Stage: The Filtering

We use the tool to analyze 500 randomly tickets from the four principal repositories in OpenStack. This tickets could be tagged as either "Fix Commited" or "Fix Released", to be able to localize the patch implemented into de source code in the version repository. They are generally tracked in Launchpad `Nova`,`Cinder`,`Horizon` and `Neutron`[1]

The parameters analyzed for each ticket were the title and the description of the report and the description of the fix commit. Also, the code changes if neither the descriptions and the comments clarified the underlying ticket. Each ticket was then categorized into one of three following groups.

1. The ticket describes a bug report.
2. The ticket describes a feature, an optimization code, changes in test files or other not bug reports.
3. The ticket presents a vague description and cannot be classified without doubts.

Henceforth, we will refer to Group 1 as *Bug Report*, Group 2 as *not Bug Report* and Group 3 as *Undecided*.

2.1.1 The Tool

2.2 Second Stage: Responsability of Previous Commit

In the second stage, we only focused on the analysis of *Error* group, because this group contained the tickets that were bug reports. Discarding the *Undecided* group in favour of the *Error* group reduces the accuracy of the experiment due to the fact that some errors are being lost in *Undecided* group. However the *Undecided* group being only 16% of the bugs under consideration makes us think that this fact does not affect the validity of the final results.

We focused on analyzing characteristics of the commit such as the type of code, the number of previous commits involved, and the number of folders modified by the fixing commit to classify the bug report into one of three groups according to the responsibility of the previous commit:

1. It is responsible,
2. It is not responsible,
3. Undecided.

We use therefore the `diff` command to display linebyline difference between two files. `diff` is well-known algorithm, extensively explained in the research literature [**?**, **?**], which

[1]https://bugs.launchpad.net/NameOfRepository

MSR '16 Austin, Texas USA

DOI: 10.475/123_4

is included in any source code management system. `diff` examines both files and returns the differences found between them, showing the line number.

At this stage, we got a list with all the previous commits, which could be where the bug was introduced. We have to identify those previous commits responsible for inserting the line with the bug. When there are more than one commit implicated in the same file is because the bug was inserted in different lines of different commits, but not everyone has to be responsible for the commit, after the analysis the responsible can be only one of them or maybe two, in fact sometimes all of them were responsible. Therefore, comparing the status of the file, before and after the commit, we are be able to identify those previous commits that are responsible and those that are not, answering the RQ2.

3. RESULTS

3.1 Fist Stage

3.2 Second Stage

4. DISCUSSION

4.1 Threats to validity

The limited sample size of tickets used in this research is the major threat to its validity.

In addition our model has threats, external and internal, that make our model not 100% valid. The internal threats are following:

- We have not taken into account errors that have been classified into *Undecided*.
- There could be some lax criteria involving the subjective opinion of the reviewers.
- We are not experts in analyzing and classifying tickets, and our inexperience may have influenced the results of the analysis.
- We are only using part of the information that the ticket provides, like comments and text. There could be a recognized pattern, unknown at first sight, that involves other parts of the information, or the whole information.

The external threats, related to the researchers that have conducted the classification, are following:

- The word *bug* is continuously mentioned in the description and commit of a ticket even when we found it is not an error. This could lead to the incorrect classification during the reviewing process.
- Some tickets are not explicitly described, which could increase the percentage of *Undecided*. This is especially true if the reviewers are not from OpenStack.

Once we have all the tickets analyzed by diferents developers who have used a double blind, how to proceed if there are discordances between them:

1. Should they discuss after their analysis to reach a better classification?, Should the tool provide this?
2. Does the Bug report only the same ticket classified as Bug report for all the developers?

5. CONCLUSIONS AND FUTURE WORK

6. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

7. REFERENCES

[1] J. Śliwerski, T. Zimmermann, and A. Zeller. When do changes induce fixes? *ACM sigsoft software engineering notes*, 30(4):1–5, 2005.

Table 1: Classification statistics of each developer

	Bug Report	Not Bug Report	Undecided	Total
Developer 1	(184) 55%	(115) 34%	(35) 11%	334
Developer 2	(188) 76%	(54) 22 %	(7) 3%	249
Developer 3	(188) 56%	(116) 35%	(30) 9%	334

Table 2: Concordance between each developer in each repository

	Nova	Cinder	Horizon	Neutron	Total
D1 and D2	(44/63) 70%	(40/52) 77%	(37/62) 60%	-	68%
D1 and D3	-	(46/63) 73%	(48/63) 76%	(26/42) 62%	71 %
D2 and D3	(41/62) 66%	(10/10) 100%	-	-	71%