Contribution Title

Gema Rodriguez[1]

1 Introduction

Many efforts on how and why bugs are introduced in the software source code are underway in the software engineering research community. Software source code is affected by many changes, many of them due to failure of the software because of emergent bugs.

While a software system is being developed, software engineers use version repository to produce and manage their code. Developers and tester reported issues, which are stored in other repository known as issue-tracking system, when a wrong behavior or bugs are found in the systems.

Issue-tracking systems help solving these bugs, but their problem is the difficulty of distinguish the bug reports from other are not. These systems provide an interface to manage reports of maintenance activities. The developers report issues describing bug reports, features or optimizations. During the bug-seeding process is difficult distinguish bug reports from other issues due to the misclasification, a study describes that two of five issues was misclassified [2]. The misclassified causes bias predecting bugs whether non bug reports are taken into account.

To classify issues as bug reports or non bug reports we could use automatic classification system as is described in[1], but the vocabulary used in the issues could change as well as the policy depending on the project. Consequently, data validation is recomended in the studies[2].

Recording links between bug reports in a issue-tracking system and the corresponding fix-commit is a hard work, causing a lost time looking in issue-tracking systems and version repositories.

Traditionally, the methods used in link recovery [5, 6] are based on text patterns or the mining of key phrases. Unfortunately, this methods include several false negatives causing bias in data [7, 8]. Therefore other methods could to be developed such as Mlink approach that links bug report with fixes using feauters in the changed source files corresponding to commit logs in addition to the traditional textual features [4]. But all of them suppose that the issues are bug report. Hence, we propose a tool to display the data necessary to the developers/testers, who could decide whether the issue is a bug report or not. The developers know better than other the system,therefore the tool help their to choose only bug reports to be analyze in bug-seeding process removing any bias induced by non bug reports.

2 The tool

The tool works in the browser, displaying the main characteristics to distingush bug report from others in OpenStack project, the developers will be responsibles to classify the tickets as bug report or not, could explain his decisions in each ticket.

OpenStack was particularly of interest because of its highest scope and heterogeneous nature with hundreds of developers contributing, futhermore due to its short life, only 5 years, all history is saved and available in a version control system. The issues are called tickets in OpenStack and availables in the Launchpad, a web interface of ticket tracking system, classifying them as bug report or not.

2.1 Architecture

This project works with Launchpad as issue-tracking system and Gerrit as code review system. The image 1 presents the architecture used in the tool, which has been developed with JavaScript, Node, JQuery and HTML5 technologies. The server side works making queries to the API of Gerrit an Launchpad, and the client side is where the user can see the information displayed. The client side interacts with the server through events. Both sides share the information requiered using JSON files. Futhermore, to integrate some funtionatities from github, we use a third-party application between github and the browser.

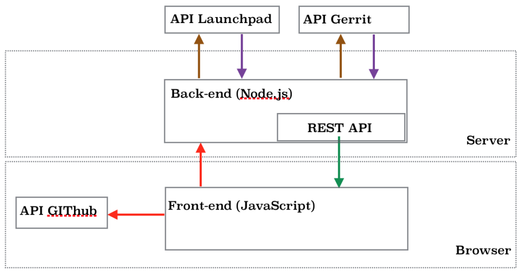


Fig. 1. Architecture

2.2 Main Features

The tool shows the id of the tickets, which are extracted randomly from each issue-tracking repository of OpenStack, and displays the information necessary to decide whether the issue is a bug report or not. We focused in display the main parameters that help in the classification, such as the title of the report and the description as well as the description of the fix commit.

The left side in the image 2, shows the information related with the ticket in Launchpad and its correspondent fix-commit in Gerrit. Some information displayed link with the original pages in Launchpad and Gerrit, thereby the developers have at their disposal extra information such as the comments that other developers have done. Could tracking the history since the ticket opens until the commit fixed the ticket.

The right side is guided to user's opinion, after reading all the information displayed, they have to classify the ticket as *Bug report* or *Not Bug report*. Due to unsophisticated description used in the ticket, the developers could doubt in the classification, for this reason we add an extra option in the classification, *Undecided*. Futhermore, the developers have a textarea to write their opinion about their classification in each ticket, as well as other textareas to write the keywords found in the title and the description which can help us building an automatic classification system in the future.

The tool allow carry out a double bind analisys, due to the data is saved in github user's account. In order that, the user can only see and modify their own data saved. Github is a control version system in which we have access to the whole information of each commit submitted by the developer. Thus, saving the data in Github, we could measure the time that each developer spend in the analisys, which tickets were more difficult to anaylize and other statistics that can help us undertanding the current problem of issues misclassification.

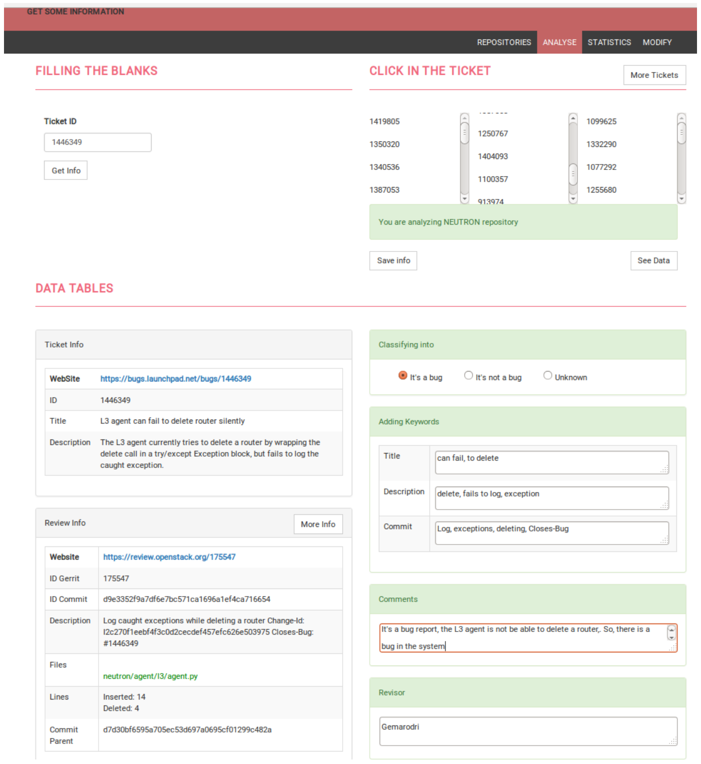
We continue developing the tool but an initial version is available [1], as well as a demonstration video[2]. It presents a licence type GPL 0 (General Public License) and you can find the code in my github's account[3]. Anyone can use it regardless of have github account or not. But, they only can save and modify their data, in addition, one of the requirements to save and modify data is create a new repository with the same name that the repository of OpenStack you want to analyze.

3 Validation Study

We use the tool to analyze randomly 500 tickets, reported in the last year, from the four principal repositories in OpenStack. This tickets could be tagged as either "Fix Commited" or "Fix Released", to be able to localizate the patch implemented into de source code in the version repository. They are generally tracked in Launchpad `Nova`,`Cinder`,`Horizon` and `Neutron`[4]

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 tiket. Each ticket was then categorized into one of three following groups.

[1] `bugtracking.libresoft.es`
[2] `https://www.youtube.com/watch?v=q0-TIvL4mqc&feature=youtu.be`
[3] `https://github.com/Gemarodri/BugTracking`
[4] `https://bugs.launchpad.net/NameOfRepository`



Ticket ID: 1446349

1419805, 1350320, 1340536, 1387053, 1250767, 1404093, 1100357, 913974, 1099625, 1332290, 1077292, 1255680

You are analyzing NEUTRON repository

Ticket Info

WebSite	https://bugs.launchpad.net/bugs/1446349
ID	1446349
Title	L3 agent can fail to delete router silently
Description	The L3 agent currently tries to delete a router by wrapping the delete call in a try/except Exception block, but fails to log the caught exception.

Review Info

Website	https://review.openstack.org/175547
ID Gerrit	175547
ID Commit	d9e3352f9a7df6e7bc571ca1696a1ef4ca716654
Description	Log caught exceptions while deleting a router Change-Id: I2c270f1eebf4f3c0d2cecdef457efc626e503975 Closes-Bug: #1446349
Files	neutron/agent/l3/agent.py
Lines	Inserted: 14 Deleted: 4
Commit Parent	d7d30bf6595a705ec53d697a0695cf01299c482a

Classifying into

[✓ It's a bug] It's not a bug Unknown

Adding Keywords

Title	can fail, to delete
Description	delete, fails to log, exception
Commit	Log, exceptions, deleting, Closes-Bug

Comments

It's a bug report, the L3 agent is not be able to delete a router,. So, there is a bug in the system

Revisor

Gemarodri

Fig. 2. Screenshot of Index

1. The ticket describes a bug report.
2. The ticket describes a feature, an optimitazion 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*, Gruop 2 as *not Bug Report* and Group 3 as *Undecided*.

In the analisys were involved three different developers who used a double bind review process, obtaining that each tickets was analyzed by two developers. Each developer analyzed 167 tickets plus the half tickets of his teammates.

4 Results

We have manually analyzed 500 tickets with support of the present tool. The table 1 show the statistics of each developer after analysed 334 tickets.

(should we corroborate that the statistics are similar with the previous study where only 100 tickets were analyse?

Table 1. Classification statistics of each developer

	Bug Report	Not Bug Report	Undecided
Developer 1	53.89%	34.13%	11.98%
Developer 2	–%	–%	–%
Developer 3	–%	–%	–%

results with 100 tickets: (63%) have been considered as Bug Report, (21%) as Not Bug Report, and (16%) as Undecided))

5 Discussion

How to procede if there are discordances beetween the developers:

- Should the developers disscuss after their analisys to reach a better classification? - Does the bug report only the same ticket classified as Bug report for all the developers?

5.1 Future Work

We would like know what grade of responsability, none or totally, practice the previous commit in the seeding of a bug in OpenStack, considering that currently exists an implicit assumption: the line that contains the error was caused by the inmediately previous commit[3]. The accuracy in our results depends on the quality of the data, thus we should focus only on bug reports discarding the other issues.

- Implement this part in the tool, displaying the code after and before the bug fixed and after and before the bug-introduction to determinate if the previous commit is responsible or not.

6 Knowleadges

References

1. Antoniol, Giuliano, et al. Is it a Bug or an Enhancement? A Text-based Approach to Classify Change Requests. 2008.
2. Herzig, Kim; Just, Sascha; Zeller, Andreas. It's not a Bug, it's a Feature: How Misclassification Impacts Bug Prediction. month, 2012.
3. J. Sliwerski, T. Zimmermann, and A. Zeller. When do changes induce fixes? ACM sigsoft software engineering notes, 30(4):15, 2005.
4. Nguyen, Anh Tuan, et al. 'Multi-layered approach for recovering links between bug reports and fixes." Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. ACM, 2012.

5. Zimmermann, Thomas; Premraj, Rahul; Zeller, Andreas. Predicting defects for eclipse. En Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007. International Workshop on. IEEE, 2007. p. 9-9.
6. Zimmermann, Thomas, and Peter Weigerber. "Preprocessing CVS data for fine-grained analysis." Proceedings of the International Workshop on Mining Software Repositories. 2004.
7. Bird, C., Bachmann, A., Aune, E., Duffy, J., Bernstein, A., Filkov, V., & Devanbu, P. (2009, August). Fair and balanced?: bias in bug-fix datasets. In Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (pp. 121-130). ACM.
8. Nguyen, T. H., Adams, B., & Hassan, A. E. (2010, October). A case study of bias in bug-fix datasets. In Reverse Engineering (WCRE), 2010 17th Working Conference on (pp. 259-268). IEEE.