International Journal of Open Source Software and Processes
Volume 13 • Issue 1

Bug Triage Automation Approaches: A Comparative Study

Madonna Fanoos, British University in Egypt, Egypt*
Abeer Hamdy, British University in Egypt, Egypt
Khaled A. Nagaty, British University in Egypt, Egypt

ABSTRACT

Bug triage is an essential task in the software maintenance phase. It is the process of assigning a developer (fixer) to a bug report. A personnel (triager) has to analyze the developers' profiles and bug reports for the purpose of making a suitable assignment. Manual bug triage consumes time and effort, so automating this process is a necessity. The previous research studies addressed the triage problem as an information retrieval problem, where the new bug report is the query. Other researchers tackled this problem as a classification problem and utilized traditional machine learning or deep learning techniques. A handful of research studies handled this problem as an optimization problem and utilized optimization algorithms such as Hungarian. This paper briefs and analyzes the previous bug triage approaches in addition to conducting an empirical comparison among five of the previous approaches.

KEYWORDS

Bug Triage Automation, Information Retrieval, Machine Leaning

1. INTRODUCTION

During the maintenance phase of large open source software projects, issues and defects usually show up. Bug reports are submitted to document these defects. The bug report includes the issue's information; such as bug id, summary, reporter name, severity, priority and submission date. Issue tracking systems (ITS) such as Bugzilla (Bugzilla, 2014) and Jira (Jira Software, n.d.) are used to manage and track the submitted reports (A. Hamdy & G. Ezzat 2020). Bug triage process is the assignment of each bug report to one of the developers, who is qualified enough to fix the assigned bug (Alenezi, Banitaan, & Zarour, 2018). A triager (personnel) analyzes each of the submitted bug reports and the developers' profiles to assign each developer to a bug report based on their experience and skills. Manual triage is a time-consuming process; the trigger, as a human, is not able commit to memory the qualifications of each developer and the skills required in each bug report; specially with the massive number of submitted bug reports to the ITS. For example, in November 2019, Eclipse

DOI: 10.4018/IJOSSP.313183 *Corresponding Author

ITS recorded 500,000 issue report (Anvik, Hiew, & Murphy, 2006) and 1,600,000 issue report are submitted in Mozilla repository. Also, 800,000 issue reports are received by Mozilla in October 2012, with around 300 new changes every day (Anjali, January 2015). These numbers indicate how this problem is sophisticated with respect to human resources and financial aspects. What makes it more complicated is the tossing actions. Tossing a bug report is a process of reassigning it to another developer in case the first one failed to solve it. Around 37% of the received bug reports go in reassignment (tossing) process (Gondaliya, Peters, & Rueckert, 2018). These tossing actions not only waste financial and human resources but also it delays the fixing time. In other words, an overloaded developer can take much time to address the problem, also a less qualified developer may fail to fix the assigned bug. In both cases the bug requires longer time to be fixed. Accordingly, more human and financial recourses will be required. Therefore, tossing actions should be minimized as much as possible.

The essential point behind the triage problem is assigning the bug report to the developer who is definitely able to make the required changes. Given the above-mentioned statistics, it is clear that a triager cannot equally distribute the newly submitted bugs over the developers. In addition, there may be sever issue that have critical consequences on the whole project. Thus, incorrect assignments decisions may lead to an increase in the fixing time and cost. Also, inaccurate assignment between the developers and bug reports waist the human resources, because of the tossing actions. According to the National Institute of Standards and Technology, handling software bugs requires over $59.5 billion per year (NIST, May 2002). Additionally, in Aka company, the payment of a senior bug fixer is $129,328 per year (Glassdoor, n.d.). Because of these big numbers, triaging systems must be optimized.

This paper summarizes and compares among the previous bug triage approaches, in a period from 2006 to 2022. The main contribution is summarized in a four-fold:

- Summarizing the state of art into a categorization schema; showing the model, algorithms, datasets used for each paper and results concluded by each of them.
- Providing a comparative analysis among previously published research papers.
- Introducing an experimental comparison for the performance of five classifiers in developer prediction.
- Providing some open discussion, which leads to find a gap to work on, in the future.

This paper is organized as follows: Section 2 provides a background and an explanation for some terminologies, that are necessary to understand the research area. Section 3 introduces a brief summary for the state of art. Section 4 discusses the previously proposed approaches in a criticizing manner. Section 5 shows experimental results of some classifiers used in bug triage problem. Section 6 introduces some ideas that could be explored in the future and conclusion.

2. BACKGROUND

This section introduces the main terminologies and concepts used in bug triage in addition to the main algorithms adapted by previous triaging approaches.

2.1 Bug Report

A term bug can be defined as an unexpected system response, as a result from a code logical or syntax mistake. When the tester finds a bug in a software under test, he uses a bug report document to raise. Bug report is a standard form consists of a group of fields to define the raised issue, such as:

- Bug ID: unique number to identify the bug.
- Bug Description: a short paragraph describing the issue and the its module.