Information and Software Technology 000 (2017) 1–18

Contents lists available at ScienceDirect

Information and Software Technology

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

Analyzing and predicting effort associated with finding and fixing software faults

Maggie Hamill[a], Katerina Goseva-Popstojanova[b,*]

[a] School of Informatics, Computing, and Cyber Systems, Northern Arizona University, Flagstaff, AZ USA
[b] Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506, USA

ARTICLE INFO

Article history:
Received 21 January 2016
Revised 4 September 2016
Accepted 10 January 2017
Available online xxx

Keywords:
Software faults and failures
Software fix implementation effort
Case study
Analysis
Prediction

ABSTRACT

Context: Software developers spend a significant amount of time fixing faults. However, not many papers have addressed the actual effort needed to fix software faults.

Objective: The objective of this paper is twofold: (1) analysis of the effort needed to fix software faults and how it was affected by several factors and (2) prediction of the level of fix implementation effort based on the information provided in software change requests.

Method: The work is based on data related to 1200 failures, extracted from the change tracking system of a large NASA mission. The analysis includes descriptive and inferential statistics. Predictions are made using three supervised machine learning algorithms and three sampling techniques aimed at addressing the imbalanced data problem.

Results: Our results show that (1) 83% of the total fix implementation effort was associated with only 20% of failures. (2) Both post-release failures and safety-critical failures required more effort to fix than pre-release and non-critical counterparts, respectively; median values were two or more times higher. (3) Failures with fixes spread across multiple components or across multiple types of software artifacts required more effort. The spread across artifacts was more costly than spread across components. (4) Surprisingly, some types of faults associated with later life-cycle activities did not require significant effort. (5) The level of fix implementation effort was predicted with 73% overall accuracy using the original, imbalanced data. Oversampling techniques improved the overall accuracy up to 77% and, more importantly, significantly improved the prediction of the high level effort, from 31% to 85%.

Conclusions: This paper shows the importance of tying software failures to changes made to fix all associated faults, in one or more software components and/or in one or more software artifacts, and the benefit of studying how the spread of faults and other factors affect the fix implementation effort.

1. Introduction

The cost of software faults is very high, not just because finding and fixing faults increases the development and testing cost, but also because of the consequences of field failures due to these faults. According to a report from Cambridge University, software developers spend on average 50% of their time finding and fixing bugs, which leads to an estimated cost to the global economy of $312 billion per year [1]. Similarly, Griss found that between 60 and 80% of the total development cost was spent on maintenance and rework [2], while Boehm and Basili claimed that software projects spend 40–50% of their effort on avoidable rework [3]. This paper is focused on analyzing factors that affect the effort needed to fix software faults and the ability to predict it using the information provided in software change requests (SCRs). Better understanding of software fixes associated with high effort and factors that affect them support more cost-efficient software debugging and maintenance. Furthermore, predicting the levels of software fix implementation effort is useful for planning and proactively adjusting resources in long-lived software systems that require sustained engineering.

Before continuing, we provide definitions of the terms used in this paper, which were adapted from the ISO/ IEC/ IEEE 24765 standard [4]. A *failure* is the inability of a system or component to perform its required functions within specified requirements. A *fault* is an accidental condition or event, which if encountered, may cause the system or component to fail to perform as required. Although

* Corresponding author.
E-mail address: Katerina.Goseva@mail.wvu.edu (K. Goseva-Popstojanova).

http://dx.doi.org/10.1016/j.infsof.2017.01.002
0950-5849/

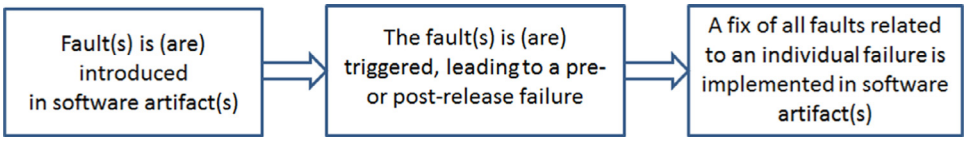

Fig. 1. The cause-effect relationships among faults, failures and fixes.

every failure is caused by one or more faults, every fault does not necessarily lead to a failure because the condition(s) under which the fault(s) would result in a failure may never be met. To prevent a specific failure from (re)occurring all faults associated with that failure must be corrected. Therefore, we define an additional term, *fix*, which refers to all changes made to correct the fault(s) that caused an individual failure [5]. With respect to definitions appearing elsewhere, the most similar to the term fix (as it is used in this paper) appears to be the term repair, which is defined in the ISO/IEC/IEEE 24765 standard [4] as the correction of defects that have resulted from errors in external design, internal design, or code. Fig. 1 illustrates the relationships among faults, failures, and fixes. Note that we considered both observed failures that occurred during operation and potential failures that were prevented from happening by detecting and fixing faults during development and testing.

Understanding the inner-workings of the complex relationships among faults, failures and fixes, as well as the effort associated with investigating failures and fixing faults, are important for cost-efficient improvement of software quality. However, while a significant amount of empirical studies have been focused on studying software faults and/or failures, very few works have tied the faults to the failures they caused and studied the relationships among faults, failures, and fixes. One of the main reasons for this is the difficulty to extract relevant information. Thus, even though most software development projects use some sort of bug, problem or issue tracking system, these systems often do not record how, where, and by whom the problem in question was fixed [6]. Information about fixes is typically hidden in the version control system that records changes made to the source code. In most cases, it is not identified if a change was made because of fixing fault(s), enhancement(s), or implementation of new feature(s). Furthermore, not many projects keep track of the effort needed to investigate failures and fix the corresponding faults.

This paper is part of a larger research effort aimed at characterizing and quantifying relationships among faults, failures and fixes, using the data extracted from a large safety-critical NASA mission. The fact that the mission kept detailed records on the changes made to fix fault(s) associated with each failure allowed us to close the loop from failures to faults that caused them and changes made to fix the faults. In our previous work we showed that software failures are often associated with faults spread across multiple files [7]. Our results further showed that a significant number of software failures required fixes in multiple software components and/or multiple software artifacts (i.e., 15% and 26%, respectively), and that the combinations of software components that were fixed together were affected by the software architecture [5]. More recently, we studied types of faults that caused software failures, activities taking place when faults were detected or failures were reported, and severity of failures [8]. Our results showed that components that experienced more failures pre-release were more likely to fail post-release and that the distribution of fault types differed for pre-release and post-release failures. Interestingly, both post-release failures and safety-critical failures were more heavily associated with coding faults than with any other type of faults.

In this paper we systematically explore the effort associated with investigating and reporting the failures (i.e., *investigation effort*), as well as the effort associated with implementing the fix to correct all faults associated with an individual failure (i.e., *fix implementation effort*). We also focus on predicting the fix implementation effort using the data provided in the SCRs, when the failure was reported. Some related works in this area used the term "fault correction effort". We use the term "fix implementation effort" to emphasize that our focus is on the effort needed to fix (i.e., correct) all faults associated with an individual failure, rather than individual software faults. Based on the processes followed by the NASA mission and our discussions with project analysts, it appears that the values of the effort data used in this paper are of high quality. However, it should be noted that, as in any observational study, it was infeasible to postmortem fully verify and validate the accuracy of self-reported data, such as the investigation effort and fix implementation effort.

It should be noted that investigation effort and fix implementation effort were not studied in our previous works. In general, the analysis and prediction of software fix implementation effort are much less explored topics than the analysis and prediction of software development effort (see [9] and references therein).

The first part of our study is focused on analysis of investigation effort and fix implementation effort and factors that affect them. We start with the research question:

RQ1: How are investigation effort and fix implementation effort distributed across failures?

Then, we study if investigation effort and fix implementation effort are affected by several factors, such as when failures were detected / observed, severity of failures, spread of faults across components and/or across software artifacts, and by types of faults and types of software artifacts fixed. Each of these factors individually is addressed in research questions RQ2–RQ5:

RQ2: Are investigation effort and fix implementation effort associated with post-release failures higher than corresponding efforts associated with pre-release failures?
RQ3: Are investigation effort and fix implementation effort associated with safety-critical failures higher than corresponding efforts associated with non-critical failures?
RQ4: Do failures caused by faults spread across components or across software artifacts require more effort?
RQ5: Is fix implementation effort affected by types of faults that caused the failure and software artifacts being fixed?

We conclude the analysis part with RQ6 focused on the effect of combinations of factors on the fix implementation effort with a goal to uncover important interactions among factors that cannot be observed by one-factor-at-a-time analysis.

RQ6: How do interactions among factors affect the fix implementation effort?

The second part of our study focuses on predicting the level of fix implementation effort, that is, explores the research question:

RQ7: Can the level of fix implementation effort (i.e., high, medium, or low) be predicted using the information entered in the SCRs, when the failure was reported?

Please cite this article as: M. Hamill, K. Goseva-Popstojanova, Analyzing and predicting effort associated with finding and fixing software faults, Information and Software Technology (2017), http://dx.doi.org/10.1016/j.infsof.2017.01.002