Empirical Study of Bug History with JUnit Tests

Siddarth Udayakumar
The University of Texas at Dallas
800 W Campbell Rd
Richardson
TX 75080
sxu140530@utdallas.edu

Keerthi Santhalingam
The University of Texas at Dallas
800 W Campbell Rd
Richardson
TX 75080
kxs142830@utdallas.edu

Ishan Dwivedi
The University of Texas at Dallas
800 W Campbell Rd
Richardson
TX 75080
ixd140630@utdallas.edu

ABSTRACT
This research project is to perform an empirical study on the bug history of ten real world Java projects using JUnit tests. This project also makes use of version control systems like GitHub to track errors reported. The end goal of this research is to examine the projects and find which type of bugs are more likely to be encountered over the course of the JUnit tests and which type of bugs. To achieve this study, we are going to make use of the issue tracking systems from each respective project and find which types of bugs have occurred more frequently. Using this, we would be able to find which bugs were detected during internal testing and which bugs were reported by users.

Categories and Subject Descriptors
[**Software and its engineering**]: software testing and debugging.

General Terms
Software, Projects, Development, Testing

Keywords
Bugs, JUnit, history, unit, parsing.

1. INTRODUCTION
Before we get started with the details of this paper, we should understand some of the commonly occurring terms related to software testing which appear in this paper.

Software testing is a major part of the software development lifecycle because it allows for quality of the software programs and ensures that the software functions as it is intended to. Software testing in itself can be classified into different categories but here, we will be concentrating on the very basic form of testing – unit testing. Unit testing involves taking the smallest functioning module of the software program under consideration and ensuring that the input provided to it produces the expected output. This can be performed using the JUnit framework. JUnit is a simple framework to write repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks [1].

The goal of JUnit is to eliminate classical testing and debugging. Some of the examples include a) Printing statements for test output, b) Debugger expressions for watchpoints and c) Manually written test scripts [2]. This allows us to write tests easy to run and reusable.

The second most commonly featured term in this document is "bug". A software bug can be defined as a fault in a software program that causes the program to produce an output that is very different from the actual output expected. This bug may arise due to errors in the code caused by developers, the framework of the base system on which the software is running on or it could be due to incorrect compilation by compilers. [3]

There are a lot of different type of bugs that can be encountered by a user, a software developer or a software tester. Some of the most common type of bugs are as follows [4]

1) Arithmetic bugs
2) Logic bugs
3) Syntax bugs

2. THE PROBLEM
There are a lot of software projects being undertaken every day and in the course of software development, there is a definite possibility of running into software bugs and faults. These bugs may either be simple bugs or sometimes major bugs depending on how the software project was and is currently being maintained. An old bug which was ignored earlier may be the cause for a major issue occurring in the system and may require lots of time and effort for solving it or it may be a smaller bug that can be solved or rectified with a small code change (This is mostly of the syntax bug variety).

To avoid bugs at the later stage of the software development process or during the release time, unit tests can be performed for each simple module and ensure that the module is functioning as expected. Some of the bigger projects on Github and other online repositories consist of a lot of code and the bugs associated with them. These bugs are reported in each of the specific software's issue tracking systems. These bugs are reported by the developer of the software or by testers for the software. In some cases, there are bugs raised by the users of the system. This is usually the case in open source software projects. For example, in the Android Open Source Project (AOSP), the users can report any bugs they encounter on the bug tracking system and once the raised bug is confirmed by the developers, it is officially included in the bug tracking system. The issues tracking system also includes the history of bugs that were raised and fixed.

These projects also involved lots of existing JUnit tests associated with them to check for proper functioning of the system modules. They may be new or previously existing JUnit tests. Our project aims to study the bug history of such software projects and find which type of bugs are more likely to be revealed by the JUnit tests in internal testing and which type of bugs more likely to be reported by users after release of the software.

3. DESIGN FOR PROJECT TECHNIQUES:

Since we would be performing an empirical study of software projects for comparing the number of bugs found through JUnit tests and the number of bugs raised by the users, we will be using open source Java projects from GitHub repositories. We will also be making use of the issue tracking systems of the projects to allow looking into software bugs. We will be considering the following projects:

1) AspectJ
2) Apache ANT
3) Apache Lucene
4) Mozilla Rhino
5) Eclipse JDT/Core
6) Apache Tomcat
7) JabRef
8) OpenMRS Core

We will be parsing each of the repositories and create a corpus for each of the Software projects we have chosen. Since we are using Java projects, we would be parsing all the java files and looking for specific terms such as “tests”, “bugs”, “JUnit”. This involves going through the entirety of the source code and finding all the terms required for our research. We could also find which java files have the terms we are looking for. This allows us to find the JUnit test cases that have been used for testing the respective systems.

In case of the bugs reported, we will be parsing all the bugs in the issues tracking system into a flat file and then crawling the flat file to find details about the bug and who raised the bug and the type of the bug. This allows us to make comparisons on which type of bugs were found by developers using internal testing and which types of bugs were reported by users.

We are yet to choose two more open source Java Projects on which our project can be applied to. The existing list of projects may or may not change based on the data available to us in each of the project repositories.

4. IMPLEMENTATION PLAN

We hope to implement the above design technique using the following software tools

a) Apache Lucene

b) A Python parser

c) Natural Language Toolkit (NLTK)

Apache Lucene is a high performance full featured text search engine library written entirely in Java. It is suitable for any process that requires full text search functions [5]. We will be using Apache Lucene to parse the java projects available and extract the specific terms related to this project from there.

NLTK is a platform for building python programs to work with human language data. We will be using this toolkit in tandem with the Python parser to parse bug reports for the respective projects and obtain information about the bugs raised.

The information obtained over the course of this process will be documented and observed in the future for inference.

5. ACKNOWLEDGEMENTS

Thanks to ACM SIGCHI for allowing us to modify templates they had developed.

6. REFERENCES

[1] http://junit.org/about

[2] Gamma, Erich, and Kent Beck. "JUnit." (2006)..

[3] A. Vahabzadeh, A. M. Fard and A. Mesbah, "An empirical study of bugs in test code," *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, Bremen, 2015, pp. 101-110. doi: 10.1109/ICSM.2015.7332456 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7332456&isnumber=7332440

[4] https://en.wikipedia.org/wiki/Software_bug#Common_types_of_computer_bugs.

[5] https://lucene.apache.org/core/