Study the relationship of time needed to fix bugs by heroes and non-heroes

Daniel Hanspeter 6129

Thomas Schievenin 5701

Abstract

The purpose of this document is to show how the project of the course “Empirical Software Measurement” has been conducted by us and what the adopted experimental procedure was.

The purpose of this project is to identify heroes among developers, based on their commit on the version control system and then the amount of time spent by them to fix a bug compared the amount required by non-heroes.

The focus was on the Project KUSER, subproject of KDE.

1. Problem

A hero is a developer that alone maintains most of a software system. Thus a hero is the only developer with a high knowledge the code he/she maintains exclusively. So when a hero is requested to fix a bug, it should take a short to fix very familiar code. However, the hero could be overloaded, as he/she is the only able to work on a large portion of code.

So what the main topic of the project will be is to compare the amount of time spent by heroes to fix a bug with the amount required by non-heroes.

RQ: do heroes take a shorter or a longer time to fix a bug than non-heroes?

2. Experimental definition and planning / Considerations taken

In order to get all the needed information to get heroes for the project we had to connect to the SVN repository using a library called SVNKit. We fetched all historical changes for each file currently (IMPORTANT for the secondary conclusions) in the project and added counters for modifications by other users and the original author.

Authors where flagged as heroes taking into consideration the original definition given in the slides provided during the course:

Hero= Developer that EXCLUSIVELY edits/commits a number of files >=α%

Once all authors & existing heroes where retrieved, we connected to Bugzilla and downloaded all bugs related to the project. The search had to be well defined since there exist various kinds of resolutions for bugs. We have taken into consideration only those kinds who consider modifications in the code.

2.1.Hypothesis formulation and variable selection

Based on the study definition the subsequent hypotheses can be formulated:

H0: there is not difference in the amount of time required by heroes and non-hero developers to fix bugs

H1: there is a difference in the amount of time required by heroes and non-hero developers to fix a bug

These two hypotheses are two-tailed because there is no a-priory knowledge on the expected trend that should favor heroes versus non heroes to fix bugs. On the one hand heroes know better the code so, probably, they require less time to fix. On the other hand, we cannot be sure that a hero knows exactly where to find the bug in prior.

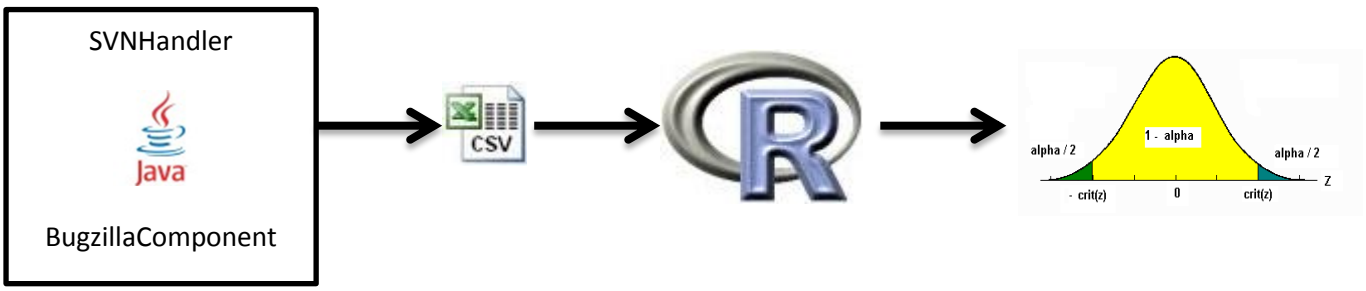
The dependent variable is the fix time. It is computed by summing up time needed to fix a bugs taking into consideration whether it has been fixed by a hero or not.

The independent variable (the main factor of the experiment) is the property of being hero or not.

2.2.Experimental material and procedure

The development of the software has been realized using the Java programming language, and in addition to the standard libraries, in order to perform data collection, connection libraries for the SVN and Bugzilla repositories where used.

The output of the software is a CSV (comma separated value) file which is then given to R as input such to gather all needed statistical results to make conclusions out of it.



2.3.Analysis method

We do not make any assumption on the normal distribution of experimental data, so we use a non-parametric statistical test to check the total fix time (H0). As we collect two measurements, one for bugs fixed by heroes and one for non-heroes, data are paired, so we use the Wilcoxon to check the hypothesis.

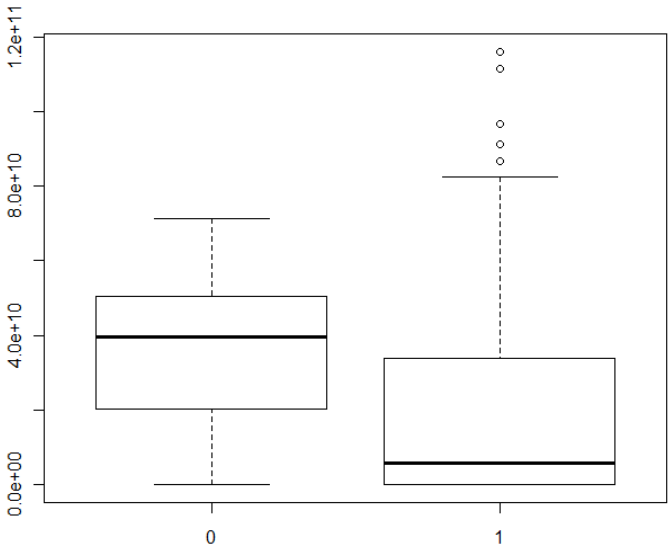
We fix the confidence level at 95%, which means that we reject the null hypothesis when p-value is lover that 0.05.

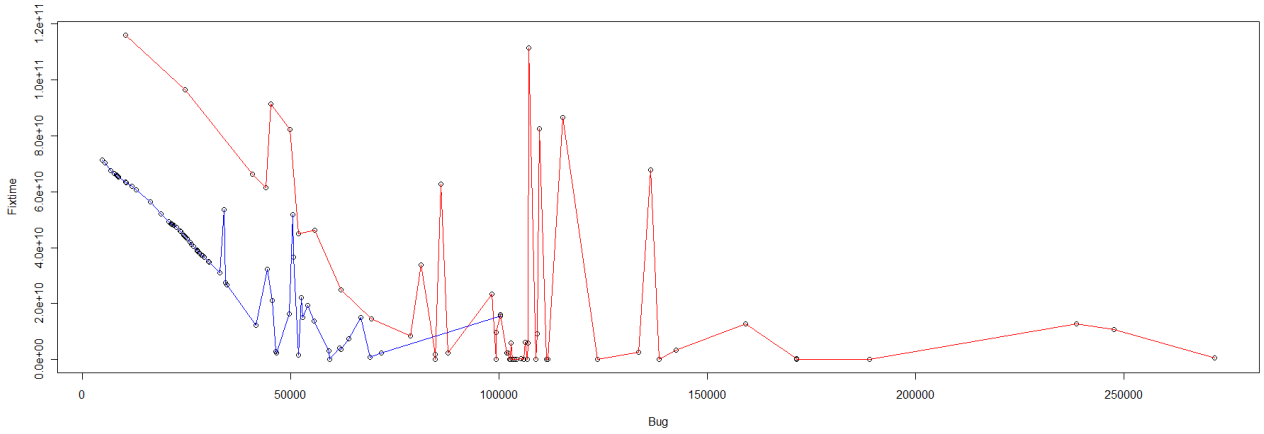
3. Preliminary Results

data: Fixtime by ByHero

W = 2635, p-value = 3.857e-05

alternative hypothesis: true location shift is not equal to 0





In red we can see bugs fixed by heroes, in blue fixed by non-heroes.

4. Conclusion

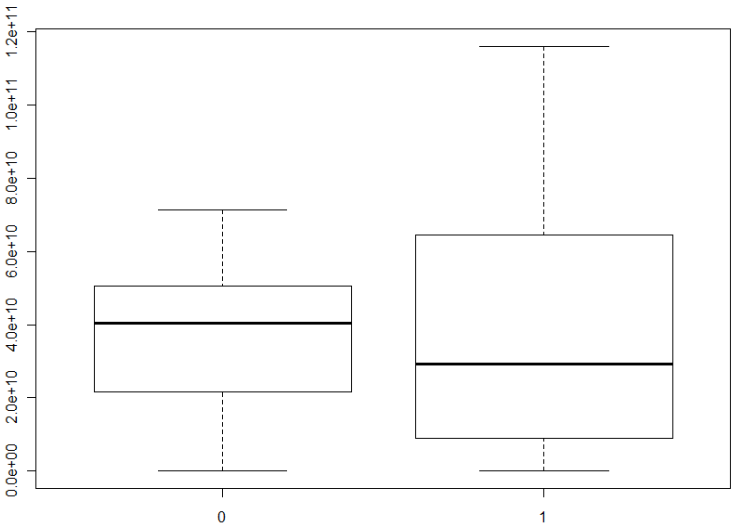
The key result of this experiment is that there is difference in time between heroes and non-heroes, meaning that the Null Hypothesis is rejected. But this result is affected by the fact that after bug 100443 approximately, there were no more bugs fixed by non-heroes. The R script in an automated way gets the correct last bug number and then plots the result to a new output device (windows()).

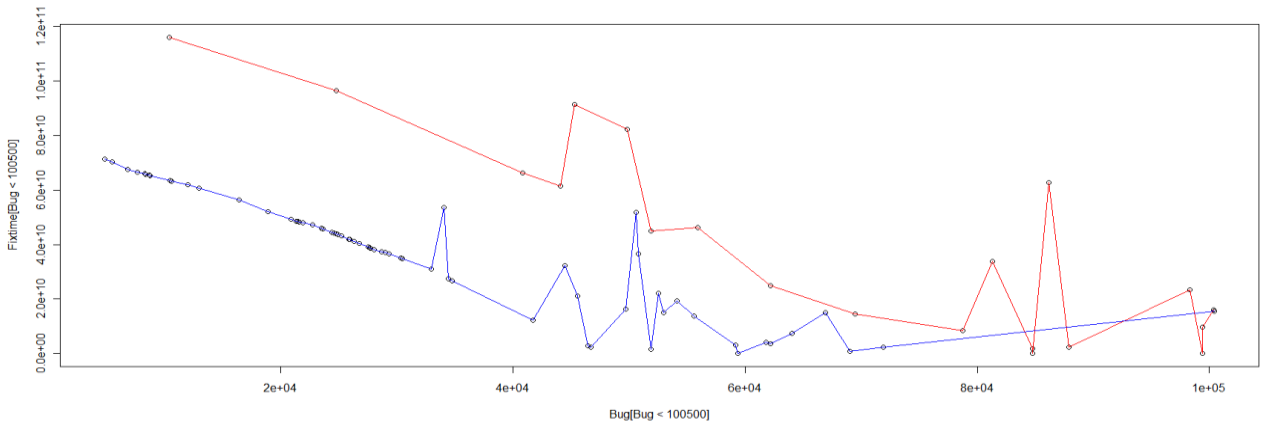
So to take better conclusions we can restrict the analysis on the first part of the project where fixes were done by both.

data: Fixtime[Bug < 100500] by ByHero[Bug < 100500]

W = 766, p-value = 0.6187

alternative hypothesis: true location shift is not equal to 0





In this case we notice that the Null Hypothesis is accepted, also by observing the graph, containing lot of bugs fixed by non-heroes that compensate those big ones fixed by heroes.