Examining the Relationship Between Developer Experience and Defects

Jake Read*

Paris District High School, Paris ON

E-mail: jakeread99@gmail.com

Abstract

All software will contain bugs at one stage or another in its development. Fixing bugs takes a large amount of both time and money. It is therefore extremely useful to be able to predict the circumstances under which such a defect could occur. One such way of predicting defects, developer experience, has yet to be widely looked into.

In this paper, we study the correlations between average developer experience and number of bugs in a file, and maximum developer experience and number of bugs in a file. In addition to this we also take a look at the correlation between the lines of code in a file and the number of bugs. To do this we studied the ActiveMQ repository and created programs in Python to gather all of the required data to form the correlations. We based each developer's experience on the number of past commits they'd made.

Our main findings are as follows: (1) The higher the average experience of the developers who have worked on a file, the lower the number of bugs. This means that companies may want to allocate their most important files to more experienced developers. (2) The higher the maximum experience of developers who have worked on a file is, the more bugs the file will have. This means that the most experienced developers are likely under greater stress and may therefore rush projects to meet

deadlines (3) More lines of code in a file means that the file will develop more bugs. Therefore larger files require further testing. This is consistent with past literature.

Introduction

The purpose of this paper is to show the correlations between pieces of data from a GitHub repository. Fixing bugs takes a lot of time and money, so we decided to complete this research to help predict where bugs are likely to appear in the future, and thus reduce expenditure of resources. This section will explain the concepts needed to understand what this information is and what it means.

Firstly, what is a GitHub repository? A GitHub repository is a place to store code so that anyone you allow can request to make changes. Whenever someone makes a change to a file in a repository, a log of their changes is added to the Git Log. This is called a commit. We can access these commits via Git Log commands in the command line and find all the information about any change that's ever been made in a repository. This is where we found the data for this report.

Secondly, what is Python? Python is the programming language that we used to mine the data in the Git Repository. It's simple to use and allows file manipulation. To gather data, you first need a file containing all the raw information, which can be created using git log commands in the command line. After this you can make a program to find all of the information you want from the file and then create a new file containing everything you need. This is how we gathered and formatted the required data for this report.

Lastly, what is a correlation? A correlation is a measure of the linear association of two variables. The values of a correlation vary between 1 and -1. A correlation of one means a strong positive correlation, a correlation of -1 means a strong negative correlation, and a correlation of 0 means no correlation. All decimal values between these are simply different strengths. Ex. A correlation of 0.2 is a weak positive correlation.[1] To find the correlations

for this report we used Microsoft Excel, which allows for easy data manipulation.

The correlations we found are between number of bugs fixed in a file and lines of code in the file, number of bugs fixed in a file and average developer experience, and number of bugs fixed in a file and maximum developer experience. Many similar studies have been done in the past, such as number of bugs and time of the commits,[2] and the correlations between bug inducing and bug fixing commits,[3] but no-one has used the correlations we are using.

The purpose of this research is to discover whether a file will get more bugs of an experienced or inexperienced developer worked on it. There are two main possibilities:

- It's possible that due to more experienced developers having a greater workload they'll make more mistakes in an attempt to rush to the next item on their list, while an inexperienced developer could spend more time fixing a bug due to a lower workload, and therefore reduce the chance of more bugs being created.
- It's also possible that an inexperienced developer will not yet have the required knowledge to fix an issue in the simplest way and may therefore create more bugs.

This research will provide answers to some of these questions.

Materials and Methods

First, we created a program to take all of the basic info from the Git Commit log. This information included the hash, developer, date, comment, and changed files for each commit. This information was then placed in a CSV file. After this we checked the ActiveMQ repository and downloaded a file containing the issue key of each commit that fixed a bug. We used this list to narrow down the larger CSV file to only contain the commits that fixed bugs. After this we shrank the file down further by removing all commits containing test files. Once these steps were complete the file was only 241 lines long.

At this point we found the experience of each developer working on ActiveMQ. We did this by creating a program to count the number of times each developer has made a commit

to the ActiveMQ repository, and then used that value as their experience. Next, we found the number of bugs that have been fixed in each file by going through the file we created previously and counting the times each file occurs. After this we created a list of developers who fixed bugs for each file and made a CSV file containing each file, how many bugs were fixed in it, and which developers worked on it. Now that we had a file where each line contained a separate file name, we shortened the information in this file by removing all lines in which the file was not java. This took the line count in this file to 292 lines from 364 lines.

From this new CSV file and the previously calculated experience for each developer, we found the average experience of the developers who fixed bugs in each file and added this to each line. Using a command in git bash, we found the number of lines in each file and placed this in the file also. We then used this information in Microsoft Excel to find a correlation between number of fixed bugs and average developer experience, and number of fixed bugs and lines of code.

We later decided to use the information we obtained to find the maximum developer experience per file. We used this information to find a correlation between number of bugs fixed and maximum developer experience.

For information regarding all correlations see the results section.

Results

Each of the following sections will show the results of one of the correlations we found.

Number of Bugs/Average Experience

For number of bugs and average experience we found a correlation of -0.020389527. This means that a higher average experience of the developers on a file will result in slightly fewer bugs. This is a rather weak correlation however, so the effects will likely not be too

noticeable.

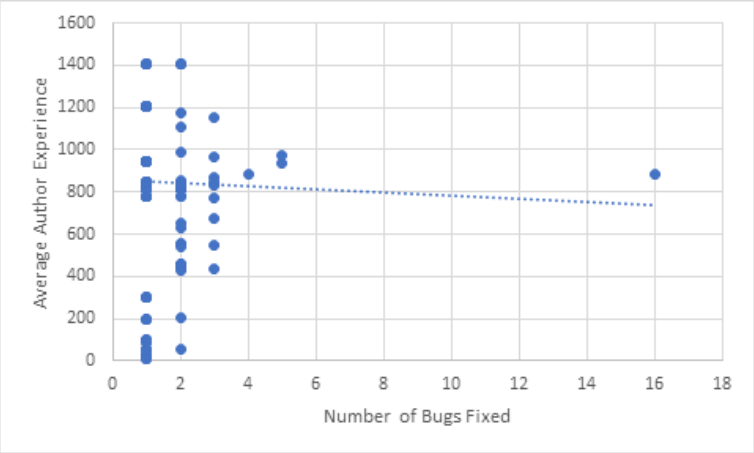

Figure 1: Bugs/Average Experience Correlation Graph

Number of Bugs/Maximum Experience

For number of bugs and maximum experience we found a correlation of 0.198651072. This means that the higher the experience of someone who has worked on the file, the more likely it is that the file will have more bugs in the future. Therefore, a file will contain less bugs if it hasn't been worked on by the most experienced developers.

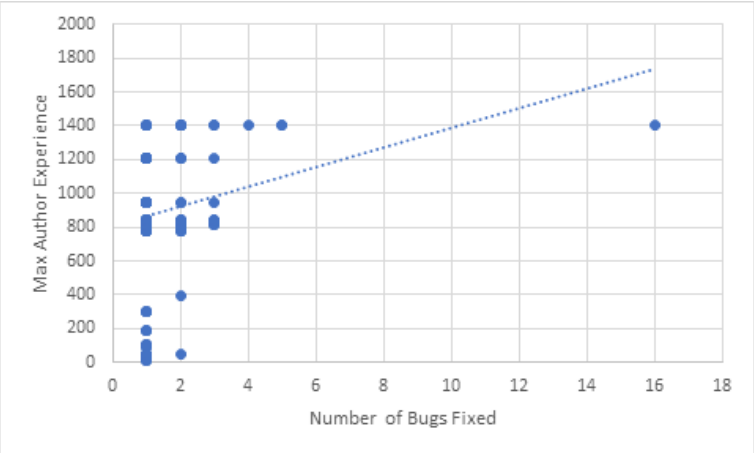

Figure 2: Bugs/Maximum Experience Correlation Graph

Number of Bugs/Lines of Code

For number of bugs and lines of code we found a correlation of 0.298881. This means that the more lines of code a file has the more likely it is that bugs will be created.

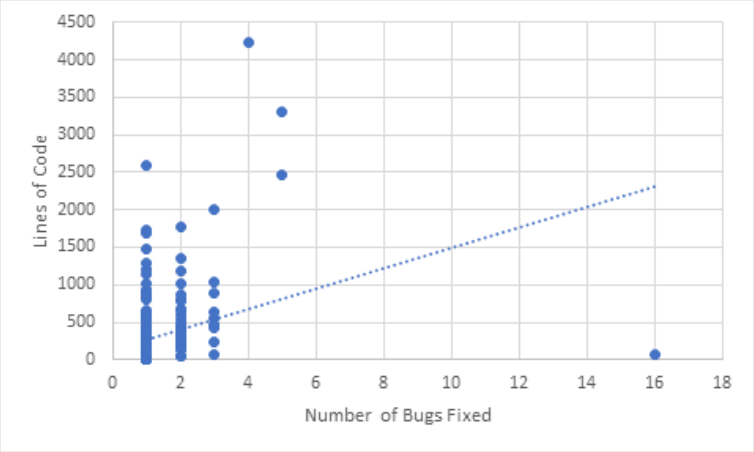

Figure 3: Bugs/Lines of Code Correlation Graph

Model

We created a model of our findings. With a statistical significance of ($P < 0.05$), we found that both maximum developer experience and number of lines of code in a file have a positive relation to the number of bugs that are created in files. The number of lines have a slightly larger impact. This means that a larger file will almost always have more bugs, however if two files contain the same or similar number of lines then the one on which someone with a higher experience has worked is more likely to contain a greater number of bugs.

As an example of this, one of the files we looked at had 16 bugs. This was the most of any file in the ActiveMQ repository. It had around an average number of lines for the repository, but it had the highest maximum experience (1408). This clearly shows that max experience has an effect on number of bugs. Similarly, the commit with the most lines of code (4230), was also one of the commits with the most bugs, showing the effect that lines of code can have.

Discussion

To sum up the results, higher average experience means slightly less bugs, higher maximum experience means more bugs, and more lines of code means more bugs

If these findings prove consistent then this is something that companies should take into account in order to improve work quality. There are a few possible reasons that a developer who is more experienced may end up causing more bugs. (1) They likely have a greater workload than an inexperienced developer. This means they're under more stress to complete tasks and could cause code to be rushed in order to meet deadlines. (2) More experienced developers are given more difficult tasks which could lead to more bugs being created. (3) Inexperienced developers will likely spend more time creating a piece of code due to their lack of experience, allowing for more time to spot bugs and patch them before making a commit.

To deal with this companies could try to reduce stress for developers by ensuring that workloads remain manageable and an appropriate amount of time can be spent writing code to reduce the chance of bugs. This can also be accomplished through a good work environment. Money invested in making a workplace stress free would likely pay for itself with money saved due to less bugs.

To test the validity of our findings we found some other papers that also relate to developer experience. Eyolfson et al. found that in general more experience makes less bugs, which agrees with the negative correlation involving average experience that we found, but also noticed that in one of the two repositories they looked at there is a sudden spike in number of bugs created by the most experienced users.[4] This could explain the higher correlation of maximum experience compared to average experience.

Mockus and Weiss found that developer experience and number of bugs had a negative correlation, meaning that greater experience results in less bugs. This matches our finding regarding average experience but contradicts our findings on maximum experience.[5] This means there are some inconsistencies between different repositories. Some possible reasons

for this are that developers on different projects or at different companies may have varying workloads and work condition, leading to less stress and more time to work on removing bugs from their code. However, this, along with the previously mentioned paper finding a general decrease in bugs with higher developer experience shows that the correlation between maximum developer experience and number of bugs likely needs further research.

For lines of code and bugs produced the correlation makes perfect sense. In general, the more lines there are in a file the more complex it is, and it is therefore more likely that making a small change may have a butterfly effect and create more bugs in other sections of the file. In short, more lines mean more opportunities for error.

This may be a slightly more difficult issue for companies to deal with, as some files simply have to be long in order to get the desired result. There are however some ways that files can be shortened to reduce risk of bugs. The primary solution would be to use more efficient code when possible. Since the most efficient method of completing a particular task may be obscure it would be a good idea for companies to teach new developers a simple and effective way of solving the most common issues with as few lines as possible.

Overall, the relations between average experience and number of bugs appears to be accurate based off of other papers results, but maximum experience and number of bugs may require further research to find a more consistent result. For lines of code and number of bugs a positive correlation makes sense and is likely consistent across most repositories.

Conclusion

In conclusion, we found three correlations during this research.

The first correlation is between average developer experience and number of bugs. We found a negative correlation of -0.020389527. This means that the higher the average experience of the developers who work on a project, the lower the number of bugs will be. Therefore, companies should try to ensure that more experienced developers work on the

most important files to slightly reduce bugs and reduce costs.

The second correlation is between maximum developer experience and number of bugs. We found a positive correlation of 0.198651072. This means that if someone with a high experience has worked on a file at any point then the file is likely to have more bugs in the future. This is likely due to stress placed on the most experienced developers in a company. Therefore, companies should work on relieving the stress of their more experienced developers to improve the quality of their work, reducing both bugs and the costs related to them. So far as we know, we're the first to cover this correlation.

The final correlation we found is between the number of lines of code in a file and the number of bugs. We found a positive correlation of 0.298881. This means that the more lines of code there are in a file the more likely it is to have bugs in the future. This is due to such files being more complex and therefore causing more opportunities for mistakes to be made. Therefore, companies should try to reduce the lines of code in files by teaching newer developers the most efficient means to accomplish common tasks.

The next research steps in regard to these correlations would be to check for consistency in a large number of repositories. This is particularly true for the correlation involving maximum experience, as other studies have contradicting results, implying that it may vary from repository to repository. More tests for each correlation would help secure these findings and discover a more consistent average for most repositories.

Keywords

Defect Prediction, Developer Experience, Mining Software Repositories

Acknowledgements

I would like to thank my mentor, Mei Nagappan, for his excellent guidance over the duration of this research.

I would also like to thank The Foundation for Student Science and Technology (FSST) for allowing me to undertake their program and giving me the opportunity to complete this research.

Finally, I would like to thank my co-op teacher, Maria Vink, for selecting me to be a part of this program.

References

(1) Kirch, W., Ed. *Encyclopedia of Public Health*; Springer Netherlands: Dordrecht, 2008; p 1090—1091.

(2) Eyolfson, J.; Tan, L.; Lam, P. Correlations between bugginess and time-based commit characteristics. *Empirical Software Engineering* **2014**, *19*.

(3) Wen, M.; Wu, R.; Liu, Y.; Tian, Y.; Xie, X.; Cheung, S.-C.; Su, Z. Exploring and Exploiting the Correlations between Bug-Inducing and Bug-Fixing Commits. Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York, NY, USA, 2019; p 326–337.

(4) Eyolfson, J.; Tan, L.; Lam, P. Do Time of Day and Developer Experience Affect Commit Bugginess? Proceedings of the 8th Working Conference on Mining Software Repositories. New York, NY, USA, 2011; p 153–162.

(5) Mockus, A.; Weiss, D. Predicting risk of software changes. *Bell Labs Technical Journal* **2002**, *5*, 169–180.