Do More Experienced Developers Introduce Fewer Bugs?

Daniel Izquierdo-Cortazar, Gregorio Robles, Jesús González-Barahona

Do more experienced developers introduce fewer bugs?

Daniel Izquierdo-Cortázar, Gregorio Robles, and Jesús M. González-Barahona

GSyC/LibreSoft, Universidad Rey Juan Carlos
{dizquierdo,grex,jgb}@libresoft.es

Summary. Developer experience is a common matter of study in the software maintenance and evolution research literature. However it is still not well understood if less experienced developers are more prone to introduce errors in the source code than their more experienced colleagues. This paper aims to study the relationships between experience and the bug introduction ratio using the Mozilla community as case of study.

As results, statistical differences among developers with different levels of experience has not been observed, when the expected result would have been the opposite[1].

1 Introduction

Software development is a task that demands high intellectual effort. Fixing errors is in among those tasks that are especially difficult. Even more when those errors are located in parts of the source code with which the developer is not *familiar*. Being *familiar* with a piece of code can be considered as having some *expertise* on/with it or previous *experience* with it (or similar one).

The goal of this paper is to study this assumption from a quantitative point of view, studying differences among developers of different levels of experience using several modules from Mozilla. To this aim, three ways of measuring experience will be used: a) number of commits, b) number of fixing commits, and c) *territoriality* (measured as the number of files only *touched* by one developer). Six null hypothesis have been defined: correlation between the three ways of calculating experience and the *bug seeding ratio* (being the percentage of *buggy* commits out of the total number of commits), and another three with the relationship among the several experience metrics used.

[1] This work been funded in part by the European Commission under project ALERT (FP7-IST-25809) and by the Spanish Gov. under project SobreSale (TIN2011-28110).

2 Empirical Approach

Some Mozilla Foundation projects have been the selected case of study for this paper. The analysis is based on the Mercurial repository[2] which offers a list of repositories that can be easily *cloned*. As a summary, 19 projects were analyzed, with more than 100,000 commits, more than 2,500 authors and around 4 years of history up to June 2011. In this scenario, we define following concepts:

- **Bug fixing commit**: commit that fixes an issue.
- **Bug seeding commit** (or *buggy* change): commit whose action has caused a later fixing action.
- **Bug seeding ratio**: the value represents the percentage of *buggy* changes done by a developer.
- **Territoriality**: number of files *touched* only by one committer.
- **Bug fixing activity:** this activity is detected analyzing the log message left by developers when performing a commit. In our case study, the developers of the Mozilla usually specify the fix of an error by means of the key word *Bug* followed by an integer. This integer matches with an *id* of an issue in the bug tracking system (BTS). We have validated our process and tested the heuristic we used, and have obtained 91,7% of precision and 89,65% of recall.
- **Bug seeding activity:** in general, the method is based on three steps: a) detection of fixing activity as previously detailed; b) identification of the lines involved; and, c) identification of previous commits where those lines were involved. The identification of commits inducing fixing actions is based on the *SZZ* algorithm [9], a method that has been also used in other research works [6]. The main assumption of this algorithm is that *modified* or *removed* lines in a fixing commit are the ones *suspicious* of having caused the fixing action (or being part of the error). Tracing back in the source code management system (SCM) to the time when they were *added* or *modified* results in the commit where the error was introduced. That commit is the one that is considered as the bug seeding commit.

3 Related Research

Expertise in developing teams is a recurrent piece of study. As claimed by some authors: "Finding relevant expertise is a critical need in collaborative software engineering, particularly in geographically distributed developments" [8].

From a quantitative point of view, experience has been found to be measured in several ways: a) *Number of commits* [8, 7, 2]; b) *Fixing activity* [1]; and, c) *Ownership* of the source code [3, 4].

[2] `http://hg.mozilla.org/`

Regarding our methodology, the approach used follows the assumption made by the SZZ algorithm where the lines that are part of a fixing activity are suspicious of being causing the fixing action (or being at least part of the problem) [9].

4 Results

Testing hypothesis $H_{0,a}$, $H_{0,b}$ and $H_{0,c}$: these hypotheses say that there is not correlation between bug seeding ratio and a) number of commits, b) number of fixing commits, and c) territoriality.

Table 1 shows the results we have obtained, offering the values of R^2 and the p-value for the projects under study. This table provides information about the comparison between bug seeding ratio and two types of experience: number of commits and number of fixing commits. Regarding to the territoriality study ($H_{0,b}$), all of the projects dataset is aggregated given that there are few values of study per project. In this case the R^2 is 0.037 and the p-value is 0.06. In all of the cases the inspection of the values provides low correlation.

textbfProject	R^2	**p-value**	R^2	**p-value**
ActionMonkey	0.006	0.432	0.019	0.166
ActionMonkey Tamarin	0.193	0.276	0.409	0.088
Camino	0.004	0.825	0.083	0.319
Chatzilla	0.001	0.953	0.005	0.876
Comm Central	0.007	0.557	0.037	0.157
Dom Inspector	0.000	0.941	0.238	0.077
Graphs	0.038	0.712	0.480	0.127
Mobile Browser	0.000	0.978	0.004	0.794
Mozilla Central	0.010	0.068	0.041	0.000
Tamarin Central	0.329	0.065	0.402	0.036
Tamarin Redux	0.009	0.651	0.001	0.897
Tamarin Tracing	0.419	0.043	0.005	0.849

Table 1. Testing hypothesis $H_{0,a}$: Number commits *vs.* bug seeding ratio and $H_{0,c}$: Correlation of bug fixing activity and bug seeding ratio

When visually analyzing the graphs for the projects, we have observed a behavior that is worth mentioning. We expected that developers with a high number of commits would have lower bug seeding ratios, graphically appearing on the left upper corner of the charts. However, some of the projects present a surprising pattern: the area the dots covers is similar to a triangle. This can be observed especially well in the case of *Mozilla Central* (see Figure 1), where those developers contributing with more than 800 commits have values of bug seeding ratio between 0.5 and 0.75, while we can find developers with less than 200 commits mostly in the range that goes from 0.25 to 0.85.

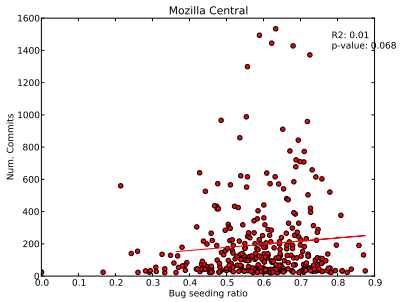

Fig. 1. Correlation study between number of commits and the bug seeding ratio. Each dot represents a developer of the Mozilla Central repository (Firefox project).

Although the pattern is not that clear as with the *Mozilla Central*, due probably to having a lower number of developers, the rest of the projects show a similar behaviour.

Testing hypothesis $H_{0,a-b}$, $H_{0,a-c}$ and $H_{0,b-c}$: these hypotheses say that there is not correlation among the three ways of measuring experience. Table 2 shows the results for the comparison between the number of commits and number of fixing commits. In addition, the comparison between number of commits and territoriality has provided a R^2 of 0.085 and a p-value of 0.004. Besides the study of the correlation between the number of bug fixing commits and the territoriality has raised a R^2 of 0.006 and a p-value of 0.444.

Project	R^2	**p-value**
ActionMonkey	0.885	0.000
ActionMonkey Tamarin	0.076	0.510
Camino	0.880	0.000
Chatzilla	0.987	0.000
Comm Central	0.963	0.000
Graphs	0.676	0.045
Mobile Browser	0.995	0.000
Mozilla Central	0.967	0.000
Tamarin Central	0.033	0.592
Tamarin Redux	0.980	0.000
Tamarin Tracing	0.630	0.006

Table 2. Testing hypothesis $H_{0,a-c}$: Study of the correlation between typical activity and fixing activity.

Table 2 has shown that a high linear correlation between these two variables exist, with high values of R^2. As the p-values are below the specified threshold, the results are statistically significant. On the contrary, no signifi-

cant values were found when correlating territoriality with number of commits and number of fixing commits.

5 Discussion

Results are detailed in Table 3. It is surprising to find that there is no correlation between any of the ways we measure experience and the bug seeding ratio. Our intuition and related research (see Eyolfson *et al.* [2]) had made us expect that developers would increase the quality of their code the more they contribute to a project. However, both results could be compatible. Eyolfson *et al.* study developers on an individual basis and consider their evolution, finding that the quality of the contributions improve over time. Our study is done in a specific snapshot and does not consider changes over time.

ID	Null hypothesis	Result
$H_{0,a}$	No correlation between BSR and # commits	FR except in Tam. Tracing
$H_{0,b}$	No correlation between BSR and territoriality	Rejected, although no linear correlation is found
$H_{0,c}$	No correlation, BSR - fixing activity	FR except in Mozilla and Tamarin Central
$H_{0,a-b}$	No correlation, # commits - territoriality	Rejected, although no linear correlation is found
$H_{0,a-c}$	No correlation, # commits - fixing activity	Rejected and strong linear correlation is found
$H_{0,b-c}$	No correlation, territoriality - fixing activity	FR

Table 3. Null Hypotheses and results. BSR: bug seeding ratio. FR: fail to reject.

Our results raise some very interesting research questions. One first question is to look for the reason for which experienced and less-experienced developers introduce bugs to the same extent. This may lie in the peer review policy that some of the Mozilla Foundation projects follow[3].

A second question is related with the tendency observed in several graphs where more experienced developers tend to a constant bug seeding ratio. This could mean that there exists an inherent difficulty for any software project that will make developers introduce a given number of bugs, and that this cannot be circumvented with more experienced developers.

Our results show that among the various ways of defining experience, territoriality may be the better choice. But further research should study if other aspects, such as an optimized peer review process, would not provide better output.

[3] `http://www.mozilla.org/hacking/committer/`

According to the reproducibility classification criteria proposed in [5], detailed information can be obtained at `http://gsyc.urjc.es/~grex/oss2012`.

6 Conclusions

In this paper we have analyzed the relation between various ways of measuring developer experience and the ratio at which bugs are introduced into the code. We have found no linear correlation between experience and bug seeding ratio; more experience does not imply that less buggy code is included in the project.

We have seen that for the various ways of measuring experience (number of commits, territoriality, bug fixing activity), territoriality seems to be the best choice to have low number of bugs introduced, although this fact is not strongly supported by our results. Number of commits and bug fixing activity are highly correlated, so for studies that handle developer experience one of them can be left out.

Our study opens the door to new research, among which we propose to study the importance of peer reviewing, the existence of a project-specific characteristics that make more experienced developers tend towards a value of bug seeding ratio, a better understanding of developer territoriality and implications of our findings in recommender systems.

References

1. S.N. Ahsan, M.T. Afzal, S. Zaman, C. Gütel, and F. Wotawa. Mining effort data from the oss repository of developer's bug fix activity. *Journal of Information Technology in Asia*, 3:67 – 80, 2010.
2. Jon Eyolfson, Lin Tan, and Patrick Lam. Do time of day and developer experience affect commit bugginess. In *Proceeding of the 8th working conference on Mining software repositories*, MSR '11, pages 153–162, New York, NY, USA, 2011. ACM.
3. Thomas Fritz, Gail C. Murphy, and Emily Hill. Does a programmer's activity indicate knowledge of code? In *Proceedings of the the 6th ESEC-FSE*, pages 341–350, New York, NY, USA, 2007. ACM.
4. Daniel M. German. Using software trails to reconstruct the evolution of software: Research articles. *J. Softw. Maint. Evol.*, 16:367–384, November 2004.
5. Jesús M. González-Barahona and Gregorio Robles. On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. *Empirical Software Engineering*, 17(1-2):75–89, 2012.
6. Sunghun Kim, E. James Whitehead Jr., and Yi Zhang. Classifying software changes: Clean or buggy? *IEEE TSE*, 34(2):181–196, March/April 2008.
7. Shawn Minto and Gail C. Murphy. Recommending emergent teams. In *Proceedings of the 4th International Workshop on MSR*, pages 5–, 2007.
8. A. Mockus and J.D. Herbsleb. Expertise browser: a quantitative approach to identifying expertise. In *Proc of the 24rd ICSE*, pages 503 –512, may 2002.
9. Jacek Śliwerski, Thomas Zimmermann, and Andreas Zeller. When do changes induce fixes? In *Intl Workshop Mining software Repositories*, pages 1–5, 2005.