Replication of Quantitative Analysis of Fault Distributions on Open Source Software Systems

ANA VRANKOVIĆ and TIHANA GALINAC GRBAC, University of Rijeka

An analysis of fault distribution in software systems was conducted in the classical work by Fenton and Ohlsson. Their study was replicated twice in order to support the findings. They investigated the Pareto principle, fault persistence, relation between code metrics and fault number, as well as fault density and fault density similarities across project releases. In this study, we aim to replicate the study on the data set from completely different environment in order to see if the evidence obtained from the previous studies can be obtained for software systems that were developed in the open source environment. Our study is conducted on two Eclipse projects in evolution involving 12 releases each. The results are in agreement with the previous studies, except that there is a difference in the behavior of fault densities between pre and post release faults.

1. INTRODUCTION

In software engineering, in order to show that results of a study are significant and can be generally used, replication studies are required. This kind of studies are valuable for statistically supporting the results of the original study [Mantyla et al. 1983]. In order to say that one result of a study is final, there needs to be a number of studies analyzing the same statement [J. 2005]. A replication study has to be conducted with the same methodology as the one that is being replicated, but in a different environment. There has been a great amount of work published on the subject of software fault distribution. One of those studies is the classical work by Fenton and Ohlsson [Fenton and Ohlsson 2000]. This study has already been replicated twice, in [Andersson and Runeson 2007] and in [Galinac Grbac et al. 2013], but the two replications did not manage to replicate the study in full. In this study, we aim to replicate the first study as a literal replication. We follow the methodology of the previous studies in order to obtain comparable results. The most considerable difference between previous studies and this one is in the data set. Data set for this study comes from completely different environment. Our data sets have more releases and more modules, but the number of faults is not too different. We kept the same hypotheses, as well as methods for proving the given hypotheses. Hypotheses that included analyzes between system test data and function test data were not conducted, because the data sets we worked with did not contain the information about the testing phases. In all previous studies data sets came from a large scale software system for telecommunication applications. In the original study [Fenton and Ohlsson 2000], data set was obtained from Ericsson Telecom AB, which develops large software-intensive systems for telecommunication applications. It had two releases. First replication study [Andersson and Runeson 2007] used empirical data from a large telecommunications company

This work is supported by Croatian Science Foundation's funding of the project EVOSOFT (UIP-2014-09-7945) and the University of Rijeka Research Grant 13.09.2.2.16.
Author's address: Ana Vranković, avrankovic@riteh.hr
Tihana Galinac Grbac, tgalinac@riteh.hr

In: Z. Budimac (ed.): Proceedings of the SQAMIA 2018: 7th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications, Novi Sad, Serbia, 27–30.8.2018. Also published online by CEUR Workshop Proceedings (http://ceur-ws.org, ISSN 1613-0073)

and it was divided into three projects. The last replication study [Galinac Grbac et al. 2013] used data from Mobile Switching Center (MSC), a functional node within the Third Generation (3G) Core network that is built on Ericssons proprietary AXE telephone exchange. This set had five releases of the software.

The paper is organized as follows. In Section 2, we present an overview of the previous studies as well as the similar studies conducted in an open source environment. In Section 3, we describe the study design, hypotheses and data collection. In Section 4, we describe the data sets and present the analysis results. The discussion is offered in Section 5 and we conclude the paper in Section 6. Due to space limitation, the supplementary material for this paper, containing the details of the projects and obtained results, are summarized in Appendix 1 available at the web-site `http://www.seiplab.riteh.uniri.hr/wp-content/uploads/2018/06/Appendix1-Replication-of-Quantitative-Analysis-of-Fault.pdf`

2. BACKGROUND

The original study analyzing the fault distributions in complex software systems was conducted in 2002 by N. E. Fenton and N. Ohlsson [Fenton and Ohlsson 2000]. They tested four groups of hypotheses:

—Hypotheses related to the Pareto principle of fault distribution,

—Hypotheses related to the persistence of faults,

—Hypotheses about the effects of module size and complexity on fault proneness, and

—Hypotheses about the quality in terms of fault densities.

The original study was replicated twice following the same hypotheses and the same methodology. The first replication was conducted by C. Andersson and P. Runeson in 2007 [Andersson and Runeson 2007]. and the second one by T. Galinac Grbac, P. Runeson, and D. Huljenic in 2013 [Galinac Grbac et al. 2013].

The first group of hypotheses was constructed so that it would prove that some software modules have a greater concentration of faults than others, following the Pareto principle [Juran and F.M. Gryna]. In all studies, it was confirmed that the small part of modules contain the greatest amount of faults. Hypotheses related to the persistence of faults were confirmed. The hypothesis stating that the great number of faults in the function test indicates the great number of faults in the system test, had limited support in the original study, but was strongly confirmed in the replications. The hypothesis stating that the number of faults in pre-release testing influences the number of faults in the post-release testing was rejected in the original study, but strongly supported in the two replications. None of the studies showed any support for the hypotheses related to module size. Statement about lines of code being good predictor for any type of faults was not confirmed. The idea of complexity metric acting as a possible predictor was investigated in the original study, but there was no support for it either. In the replication studies, this hypothesis was not tested due to limited data sets that did not contain the information about module complexity. The last group of hypotheses was directed towards the fault densities. There were two hypotheses. First one is stating that the fault densities between releases are constant, and the other one stating that the fault densities between similar environments are also similar. This hypotheses were confirmed or supported in all of the studies. In order to validate those results we conducted the same study on different data sets. Our data sets come from the open source community, while in other three studies, this data sets were from the industry. We study the data from 12 releases of each of the two Eclipse projects, JDT and PDE. Similar studies have been conducted for years. Some of those studies were conducted on open source projects. In the [English et al. 2009], as well as in [Zhang et al. 2010], it is shown that the Pareto principle can be applied

on open source code, also using the Eclipse data set, but they only used JDT project with only the first three releases. In the work [Denaro and Pezze' 2002] the Pareto principle is also tested with compelling results. For the first three releases of Eclipse projects there have been a few studies investigating the influence of lines of code and complexity on fault appearances. In [Zimmermann et al. 2007] and [Shihab et al. 2010] they obtained high correlation coefficients between pre-release and post-release faults. In those studies, they also found that there is no proof for using code size and complexity metrics as the predictors. Both studies were performed only on the first three releases of one project while in this study we are analyzing twelve releases of two Eclipse projects.

3. STUDY DESIGN

3.1 Hypotheses

Hypotheses in this study are taken from the replication studies and the original one. Hypotheses are divided into four groups. The group of hypotheses that are related to the different phases of testing process are not replicated, as our data set did not contain the required information. The hypotheses are as follows:

G1 *Hypotheses related to the Pareto principle of fault* distribution:
- H1a A small number of modules contain most of the faults detected during pre-release testing.
- H1b If a small number of modules contain most of the pre-release faults, then it is because these modules constitute most of the code size.
- H2a A small number of modules contain most of the faults detected during post-release testing.
- H2b If a small number of modules contain most of the post-release faults, then it is because these modules constitute most of the code size.

G2 *Hypotheses related to the persistence of faults: (Was not tested)*
- H3 A higher incidence of faults in function testing implies a higher incidence of faults in system testing
- H4 A higher incidence of faults in pre-release testing implies a higher incidence of faults in post-release testing and use.

G3 *Hypotheses about effects of module size and complexity on fault proneness:*
- H5a Smaller modules are less likely to be fault-prone than larger ones.
- H5b Size metrics are good predictors of pre-release faults in a module.
- H5c Size metrics are good predictors of post-release faults in a module.
- H5d Size metrics are good predictors of a modules pre-release fault density.
- H5e Size metrics are good predictors of a modules post-release fault density.
- H6 Complexity metrics are better predictors than simple size metrics of fault and fault-prone modules

G4 *Hypotheses about the quality in terms of fault densities:*
- H7 Fault densities at corresponding phases of testing and operation remain roughly constant between subsequent major releases of a software system.
- H8 Software systems produced in similar environments have broadly similar fault densities at similar testing and operational phases.

3.2 Data Collection

We used two data sets from Eclipse open source community, Java Development Tools and Plug-in development environment. Eclipse is an integrated development environment (IDE). The Java development tools (JDT) and Plug-in development environment (PDE) are often analyzed in research, as it has an

open source repository on Bugzilla with information about defects and testing method. Data was collected using Bug Code (BuCo) Analyzer tool [G. et al. 2014]. The data collection approach is described in details in [Mauša et al. 2014]. We conducted our research on 12 releases of the JDT project and 12 releases of the PDE project. The data set contains information for each class within the software system together with over 50 metrics of each class. The metrics we use are lines of code (LOC), cyclomatic complexity, post-release and pre-release fault number. Unlike in the last two replication, our dataset contained information about cyclomatic complexity and therefore some tests that were not presented in those two replications are presented here. Within the data sets, unfortunately, there is no information about testing phases that is needed in order to replicate all the hypotheses, and therefore, some of them are left out.

3.3 Data Analysis Techniques

In order to be consistent, the techniques we use for data analysis are the same as the techniques in the other studies. Alberg diagrams [Ohlsson and Alberg 1996], scatter plot diagrams, and correlation analysis is used. For correlation analysis, we use the Pearson and Spearman correlation coefficient. All analysis is done in Matlab software.

4. DATA ANALYSIS AND RESULTS

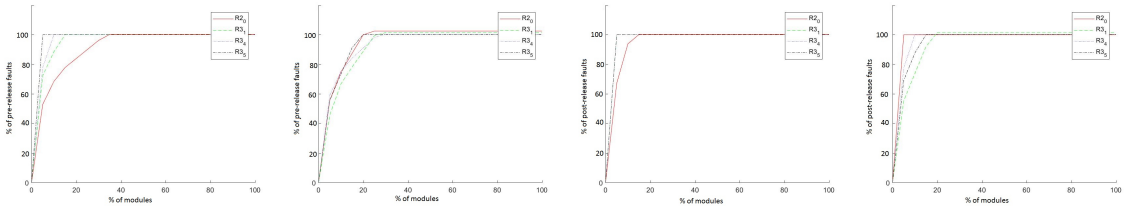
In this section we describe the data analysis and obtained results for each hypothesis in a separate subsection. In this study, our data set consisted of two Eclipse projects, Java development tool (JDT) and Plug-in development environment (PDE). In the further text and tables, they will be addressed as Project 1 (JDT) and Project 2 (PDE). Each project in this study had 12 versions. In comparison to the other three studies, this is a much larger number of releases. The tables with detailed analysis results are omitted in this paper, due to space limitation, but may be found in Appendix 1 available at the website `http://www.seiplab.riteh.uniri.hr/wp-content/uploads/2018/06/Appendix1-Replication-of-Quantitative-Analysis-of-Fault.pdf`

4.1 Hypotheses Related to the Pareto Principle of Fault Distribution

This group of hypotheses tests for the realization of the Pareto Principle that states that 20% of the software classes are responsible for 60% of the faults in the software. But we also test to see if those 20% are the larger part of the whole software in terms of code size.

Hypothesis 1a. A small number of modules contain most of the faults detected during prerelease testing. In this hypothesis, we analyze the relationship between the percentage of modules and the percentage of the pre-release faults. In Figures 1a and 1b the Alberg diagram showing the percentage of modules versus the percentage of pre-release faults for Project 1 and Project 2 are given. The modules are sorted in decreasing order with respect to the number of pre-release faults, while the scale is presented in percentages. In the graphs only five selected releases of each project are given, since graphs for all twelve versions would not be noticeable. The releases selected for both projects are releases *R2_0, R3_2,R3_4* and *R3_5*. Results of these graphs are not similar to the ones in the previous studies. It visually confirms the statement that 80% of pre-release faults are indeed within the 20% of modules, while in the other studies these numbers were lower, around 60% to 70% and in the last replication around 66% to 80%. We can say that the hypothesis is supported just as in the previous studies.

Hypothesis 1b. If a small number of modules contain most of the prerelease faults, then it is because these modules constitute most of the code size. The 10% of the modules with the highest number of pre-release faults contains around 40% of all code in Project 1. It can still be said that these modules do not constitute most of the code size as this number is below 50%. In Project 2



(a) Modules versus the percentage of pre-release faults in Project 1

(b) Modules versus the percentage of pre-release faults in Project 2

(c) Modules versus the percentage of post-release faults in Project 1

(d) Modules versus the percentage of post-release faults in Project 2

Fig. 1: Modules versus the percentage of faults

that percentage is even smaller as the modules contain only around 10%-30% of the system size. With that the hypothesis 1b is rejected, as in the previous studies.

Hypothesis 2a. A small number of modules contain most of the faults detected during postrelease testing. Looking at Figures 1c and 1d, we can see that the Pareto principle is also sustained for post-release faults. In the previous studies results were as follows: in the original study 100 and 80%, in the first replication 63, 74, and 59% for the three projects, and in the second replication 62, 39, 40, 55, and 53% for the five projects. In Project 1 almost 100% of the faults is contained within the 10% of system modules. In the Project 2 that percentage is between 70% and 100%. These results strongly support the hypothesis. In the original and the first replicated study this hypothesis is confirmed at the 10% of modules, just as in this study, while in the second replication it is confirmed first at the 20%. Just as in the previous studies, this observation has been stronger for post-release faults than for the pre-release faults. Hypothesis H2a is confirmed more strongly than hypothesis H1a.

Hypothesis 2b. If a small number of modules contain most of the post-release faults, then it is because these modules constitute most of the code size. Comparing the percentage of the modules with the most post-release faults against the system size, yields the same results as for pre-release faults. In the twelve versions of Project 1, the 10% of the modules contain 35%-47% of the system size. In Project 2, those numbers are between 9% and 32%. In none of the three previous studies is hypothesis H2b supported. This study also rejects hypothesis 2b.

4.2 Hypotheses related to the persistence of faults

As it was stated in previous sections, hypothesis 3 that higher incidence of faults in function test implies higher incidence of faults in system test was not tested because of the lack of data.

Hypothesis 4. A higher incidence of faults in prerelease testing implies higher incidence of faults in postrelease.

The original study found the reversed conclusion for this hypothesis. They stated that almost all of the faults detected in prerelease testing appear in modules that had no postrelease faults. The result that they gathered stated that 93% and 77% of the faults in prerelease testing occurred in modules that had no postrelease faults for the two releases. The first replication stated that 36%, 29%, and only 13% of the faults in prerelease testing occur in modules that have no postrelease faults. The first replication, unlike the original study, confirmed the hypothesis. In the second replication, this hypothesis could not be literally replicated as the data set used did not have necessary information. They used faults detected during system testing as post-release faults. The results that they obtained confirmed the finding of the first replicated study and confirmed the hypothesis. In this study we used correlation coefficients to analyze the relationship. The results indicated a moderate to strong

connection in some releases, while in others there is no correlation. When observing the percentage of pre-release faults that have no subsequent post-release faults, the results are quite diverse. For Project 1, those percentages are between 40% and 50%, while in Project 2 those percentages are between 40% and 75%. We could not come to any conclusions regarding this hypothesis as the results are not consistent.

4.3 Hypotheses about the Effects of Module size and Complexity on Fault Proneness

Most of the fault prediction techniques rely on the software metrics as the indicator of fault number. In the original and the first replicated study, the effects of modules metric on the fault proneness were analyzed by comparing LOC and Cyclomatic complexity with the number of faults and the fault density. In the second replication, only the LOC metric was compared as they lacked the necessary data for the analysis of complexity. In this study, we followed the original and the first replicated study and conduct the analysis on both LOC and complexity metrics.

Hypothesis 5a. Smaller modules are less likely to be failure prone than larger ones. In order to study the hypothesis we use Pearson and Spearman correlation coefficients. The correlation coefficients are quite diverse. There are releases for which there is no correlation, and there are some releases in which the correlation is moderate to strong positive relationship but the results are not consistent. We obtained similar results for Project 2 as for the Project 1, but the correlation coefficients were higher. In the last two version there was no correlation, but in the rest of the releases the correlation was moderate to strong. In the previous replication studies, results were similar to ours. For some releases, there was no correlation, but there were some cases where the correlation was strong. Hence, we cannot accept the hypothesis that states that smaller modules are less likely to be failure prone based on the results of this analysis.

Hypothesis 5b. Size metrics are good predictors of prerelease faults in a module. The results are considerably similar to the results for hypothesis 5a. The correlation coefficients are not similar between releases. We again have releases with no correlation, but also some with moderate and strong relationship between lines of code and the number of pre-release faults. Previous studies show that there is no correlation in this case. In the original study, the conclusion was that size metrics are in fact good predictors in this case, but it was based strictly on scatter plots without any statistic analysis.

Hypothesis 5c. Size metrics are good predictors of postrelease faults in a module. In this case, we have diverse results for both projects. There are some releases that show a strong correlation, and there are some where there is no correlation. Just as in this study, in the previous studies, this hypothesis was rejected as there were no indicators of correlation. We can say that the module size is not a good predictor of post-release faults.

Hypothesis 5d. Size metrics are good predictors of a modules prerelease fault density. Based on our results, there is no or a very weak relationship between size metrics and pre-release fault density. We can reject the hypothesis with certainty. In the previous studies results were the same. There was no basis to claim that size metrics are in fact good predictors of module fault density.

Hypothesis 5e. Size metrics are good predictors of a modules postrelease fault density. All previous studies say that there is no support for this hypothesis. Our analysis concludes that there is no relationship between size metric or post-release fault density, and therefore, the hypothesis is also not supported. We also tested to see if there is a regularity between fault densities of the different releases, but there was none.

Hypothesis 6. Complexity metrics are better predictors than simple size metric of fault and fault-prone modules In the original study, they concluded that there is some small correlation between the complexity and number of faults. They say that the earlier graphs for size versus faults

reveal a similar pattern. We run a correlation analysis, in the same manner as we did for LOC and number of faults. Our correlation coefficient for Pearson and Spearman correlation were even lower than in the case of LOC and the number of faults. Based on that analysis, we conclude that there is no eveidence that complexity metric is a better predictor than size metric for the number of faults.

4.4 Hypotheses about the Quality in Terms of Fault Density

We could not analyze the hypotheses stating that fault densities at corresponding phases of testing and operation remain roughly constant between subsequent major releases of software system and that software systems produced in similar environments have broadly similar fault densities at similar testing and operational phases, since we did not have appropriate data set to test it.

Hypothesis 8: Software systems produced in similar environments have broadly similar fault densities at similar testing and operational phases. For this hypothesis, as we do not have information about system and functional tests, we did the analysis using all the faults in the software system. Our results were that the pre-release densities are between 0 and 6.4 while post-release are between 0.1 and 3.1. There is even a case in which post-release fault density is higher than the pre-release in PDE release R3_4. The results are too diverse to make a conclusion based on them.

5. DISCUSSION

In this section, we discuss the results of the previous studies and compare to this study. We have to take into consideration the differences between the data sets in each study.

For the group G1 of hypotheses (1a,1b,2a and 2b), results of all three previous studies were the same. The first hypothesis (1a and 2a), that states that a small number of modules contains most of the faults, was confirmed. They prove that the 20% of modules contain 60% of faults, both pre-release and post-release. In this study, we confirmed their findings. In our test cases within 10% of modules, there was already 60% of faults. The second hypothesis (1b and 2b) from the group G1 states that, if there really is 60% of faults within 20% of modules, it is only because those 20% are a great part of the whole software system. In the other studies it is shown that this statement is false. Their results show that those 20% of modules occupy less than 40% of the system size. In this study, that number was higher. The 10% of modules that contained most of the faults have around 40% of the system size, which is still small enough to say that the hypothesis can not be confirmed. In the similar studies that were conducted on open source projects, the results were similar. In paper [English et al. 2009], they show that there is 82% of faults inside the 20% of the system size.

Hypothesis 3 related to the persistence of faults could not be tested in this study in terms of the system test and function test. Hypothesis 4 that a higher incidence of faults in pre-release testing implies a higher incidence of faults in post-release testing could not be supported. The results for this hypothesis in this study were very diverse. Great differences between releases are possible because of the big number of releases. In most of the later releases of both projects the overall number of faults is decreasing and releases tend to have a very small to none post-release faults. The original study strongly rejected the hypothesis, while the other two confirmed it. There seems to be underlaying differences in the data sets that are not straight forward or not measured in this studies and are affecting analysis of this hypothesis.

The group G3 of hypotheses tests the effects of module size on fault proneness. In the previous studies, there was no support for this group of hypotheses. In our study, we also concluded that we cannot support the hypothesis. Therefore, we agree with the conclusions of the previous studies. The hypothesis that the complexity metrics are better predictors than the size metric, in this study, is rejected. The correlation test shows that there is less correlation between complexity metric and number of faults, than the size metric and the number of faults. In the replication studies this hypothesis was not tested

due to lack of data, but in the original study, based strictly on the scatter graph, they came to the same conclusion as we did. In the studies that are conducted on the open source projects [Shihab et al. 2010; Zimmermann et al. 2007], they conclude that the number of lines of code is the most stable predictor .

The last tested hypothesis 8 that there is a similar behavior of fault densities between pre and post release number of faults within a similar environment. Both of our projects were from the same environment, but neither between the projects, nor between releases, we could not find similarities. All previous studies confirmed this hypothesis, but based on our results we could not accept the hypothesis. It is possible that this difference is somehow related to the fact that our projects came from the open source community, in which there are great differences within the environment between open source and industry.

6. CONCLUSION

In this paper, we present the third replication of the classical work by Fenton and Ohlsson [Fenton and Ohlsson 2000]. It has been replicated twice [Andersson and Runeson 2007; Galinac Grbac et al. 2013] and as in the previous replications we followed the original study to conduct a replication as similar as possible to the previous studies. Part of the team working on the replication study on the industrial data [Galinac Grbac et al. 2013], is the same as in this study. For that reason data collection was the same as it was for the industrial data which minimizes the possibility of gathering different results due to data collection differences. Our results are similar to the replication studies. For the hypotheses testing the Pareto principle, we came to the conclusion that there is indeed more than 60% of faults within the 20% of modules. There is still no support for using the number of lines of code for predicting the number of faults, neither in pre-release testing, nor in post-release, as well as using complexity metric. Unlike in other studies, we could not find evidence for stating that fault densities behave similar within similar environments. This study was conducted solely on Java software from Eclipse environment and the results are not representative for all the open source software. We can conclude that the study results from previous replication studies are almost the same as in this study, despite great differences in the environment and data sets.

REFERENCES

C. Andersson and P. Runeson. 2007. A Replicated Quantitative Analysis of Fault Distributions in Complex Software Systems. *IEEE Transactions on Software Engineering* 33, 5 (2007), 273–286.

G. Denaro and M. Pezze‘. 2002. An Empirical Evaluation of Fault- Proneness Models. *Proc. 24th Intl Conf. Software Eng.* (May 2002.), 241– 251.

M. English, C. Exton, I. Rigon, and B. Cleary. 2009. Fault Detection and Prediction in an Open-Source Software Project. *Proc. Fifth Intl Conf. Predictor Models in Software Eng.* (May 2009.), 17:1–17:11.

N.E. Fenton and N. Ohlsson. 2000. Quantitative Analysis of Faults and Failures in a Complex Software System. *IEEE Trans. Software Eng.* 8 (Aug 2000.), 797–814.

Mauša G., Perković P., Galinac Grbac T., and Štajduhar I. 2014. Techniques for BugCode Linking. *Proceedings of SQAMIA 2014* (2014), 47–55.

T. Galinac Grbac, P. Runeson, and D. Huljeni. 2013. A Second Replicated Quantitative Analysis of Fault Distributions in Complex Software Systems. *IEEE Transactions on Software Engineering* 39, 4 (2013), 462–476.

Miller J. 2005. Replicating software engineering experiments: a poisoned chalice or the Holy Grail. *Inf Software Technology* 4 (2005), 233244.

J.M. Juran and Jr. F.M. Gryna. *Quality Control Handbook, fourth ed.*

Mika V. Mantyla, Casper Lassenius, and Jari Vanhanen. 1983. Software Eng. (1983), 230–237.

G. Mauša, Galinac Grbac T., and Dalbelo Baši´ c B. 2014. Software defect prediction with bug-code analyzer a data collection tool demo. *Proceedings of SoftCOM 14* (2014).

N. Ohlsson and H. Alberg. 1996. Predicting Fault-Prone Software Modules in Telephone Switches. *IEEE Trans. Software Eng.* 12 (1996.), 886–894.

E. Shihab, Z.M. Jiang, W.M. Ibrahim, B. Adams, and A.E. Hassan. 2010. Understanding the Impact of Code and Process Metrics on Post-Release Defects: A Case Study on the Eclipse Project. *Proc. ACM/IEEE Intl Symp. Empirical Software Eng. and Measurement* (Sept 2010.), 4:1–4:10.

H. Zhang, A. Nelson, and T. Menzies. 2010. On the Value of Learning from Defect Dense Components for Software Defect Prediction. *Proc. Sixth Intl Conf. Predictive Models in Software Eng.* (Sept 2010.), 14:1– 14:9.

T. Zimmermann, R. Premraj, and A. Zeller. 2007. Predicting Defects for Eclipse. *Proc. Third Intl Workshop Predictor Models in Software Eng.* (May 2007.), 9:1–9:7.