Bug-fix Time Prediction Models: Can We Do Better?

Pamela Bhattacharya Iulian Neamtiu
Department of Computer Science and Engineering
University of California, Riverside, CA, USA
{pamelab,neamtiu}@cs.ucr.edu

ABSTRACT

Predicting bug-fix time is useful in several areas of software evolution, such as predicting software quality or coordinating development effort during bug triaging. Prior work has proposed bug-fix time prediction models that use various bug report attributes (e.g., number of developers who participated in fixing the bug, bug severity, number of patches, bug-opener's reputation) for estimating the time it will take to fix a newly-reported bug. In this paper we take a step towards constructing more accurate and more general bug-fix time prediction models by showing how existing models fail to validate on large projects widely-used in bug studies. In particular, we used multivariate and univariate regression testing to test the prediction significance of existing models on 512,474 bug reports from five open source projects: Eclipse, Chrome and three products from the Mozilla project (Firefox, Seamonkey and Thunderbird). The results of our regression testing indicate that the predictive power of existing models is between 30% and 49% and that there is a need for more independent variables (attributes) when constructing a prediction model. Additionally, we found that, unlike in prior recent studies on commercial software, in the projects we examined there is no correlation between bug-fix likelihood, bug-opener's reputation and the time it takes to fix a bug. These findings indicate three open research problems: (1) assessing whether prioritizing bugs using bug-opener's reputation is beneficial, (2) identifying attributes which are effective in predicting bug-fix time, and (3) constructing bug-fix time prediction models which can be validated on multiple projects.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging—*Debugging aids*; D.2.9 [**Software Engineering**]: Management—*Time estimation*

General Terms

Economics, Experimentation, Human Factors, Management, Measurement

Keywords

Bug report triage, issue tracking, statistical model, bug-fix time

1. INTRODUCTION

Predicting bug-fix time is useful in several areas of software evolution, such as predicting software quality [9] or coordinating effort during bug triaging [8]. To this end, prior efforts have constructed bug-fix time prediction models, based on machine learning algorithms, [1] on both open source and commercial projects.

Prior studies on open source projects [8, 5, 1] have used various bug report attributes (e.g., number of developers involved in fixing the bug, bug severity, number of comments) as features for building predictors. On commercial software, an empirical study by Guo et al. [6] has found that the likelihood that a bug reported would be fixed, and the time taken to fix it, are highly correlated with the bug-opener's reputation in the project community—in other words, a bug opened by a developer who has been successful in fixing bugs she/he has reported in the past has a higher likelihood of being fixed, and is fixed faster. Hooimeijer et al. [8] also used bug-opener's reputation to build a linear model for predicting bug-fix time for Mozilla Firefox, in addition to other attributes, like bug severity. However, they did not perform an analysis to show if there is a correlation between bug-fix time and bug-opener's reputation. In summary, all these previous studies have focused on feature-based classification models for predicting bug-fix time. To find out the predictive power of existing models, in this paper we measure the significance of the attributes these models use; we perform both univariate and multivariate regression analyses to capture the significance of the features selected in building prior models.

We used a total of 512,474 bug reports from Chrome, Eclipse and three products from the Mozilla project (Firefox, Seamonkey and Thunderbird) for our analysis. We chose these projects because (1) they are active, real-world projects with a sizable code base, and large numbers of users and developers, and (2) bug reports from these projects have been used previously in several studies involving bug report analyses [3, 13, 2, 4]. We used multivariate regression testing to point out the low predictive power of existing models; we found that goodness of fit varied from 0.30 to 0.49, which indicates that additional attributes are needed to increase prediction accuracy (Section 4.2.1). We found, via univariate regression testing, that most bug report attributes are *not* correlated with the likelihood of a bug getting fixed, or with bug-fix time (Section 4.2.2). Similarly, we tested whether there exists any relationship between the bug-opener's reputation, the likelihood of the bug getting fixed, and the bug-fix time (i.e., whether the bug-fix time is shorter when the bug-opener has a higher reputation) in open source projects. We found that in open source projects each bug receives equal at-

[1] A machine learning algorithm (or a classifier in this case) can be trained using input attributes and desired output classes; after training, when presented with a set of input attributes, the classifier predicts the most likely output class.

tention, regardless of bug opener's reputation (Section 4.2.3).

The results of our empirical investigation put forward three open research questions: (1) is prioritizing bugs based on bug-opener's reputation, as observed in commercial projects, beneficial for maintenance?, (2) can we find attributes which correlate highly with bug-fix time?, and (3) can we construct bug-fix time prediction models that can be validated on multiple projects?

2. RELATED WORK

Guo et al. [6] performed an empirical study to characterize factors that determine which bugs get fixed in Windows Vista and Windows 7. They found that bugs reported by people with higher reputation were more likely to get fixed, as were bugs opened by people on the same development team and working in geographical proximity; we tested whether reputation has any influence on bug-fix likelihood in open-source software. We found that there is no linear relationship between a bug-opener's reputation and the likelihood of a bug being fixed.

Hooimeijer et al. [8] designed a model that uses bug report attributes such as bug severity and bug-opener's reputation to predict whether a bug report will be triaged within a given amount of time. They used bug reports from the Mozilla Firefox project to test their model, and report around 75% prediction accuracy. Panjer [12] used several different data mining models to predict Eclipse bug lifetimes by training the models using various bug report attributes. We use a wider range of open source projects and show that bug-opener reputation and other attributes used in these prediction models do not correlate highly with bug-fix time, which suggests there is room for improving the prediction accuracy of prior models [7].

Anbalagan et al. [1] presented results from an empirical study on 72,482 bug reports from nine releases of Ubuntu, a popular Linux distribution. They found that there is a strong linear relationship between the number of people participating in a bug report and bug-fix time, and proposed a linear regression model to predict bug-fix time; they report an R^2 of up to 0.98. Kim et al. [9] computed the bug-fix time of files in ArgoUML and PostgreSQL by identifying when bugs are introduced and when they are fixed. They reported two bug-fix time statistics: average bug-fix time, and files whose bug-fix time were above average and suggested that the files which took above average time to fix should be refactored. Giger et al. [5] studied six projects: Eclipse JDT, Eclipse Platform, Mozilla Core, Mozilla Firefox, Gnome GStreamer and Gnome Evolution. They found that using post-submission data of bug reports (i.e., number of comments made to a bug and number of developers involved) improves bug-fix time prediction accuracy. Additionally their model could predict how promptly a new bug report will receive attention. We measured how attributes used by these prediction models correlate with bug-fix time, and found correlation values to be low. Mockus et al. [11] performed an empirical study to report the bug-fix time trends in Apache and Mozilla. For Mozilla, their bug reports cover the period July 1998–July 2000, whereas our Mozilla analysis covers the period May 1998–March 2010. They found that on average, bugs of priority 1 or 3 are resolved in 30 days or less, bugs of priority 2 are resolved in 80 days or less, while the median resolution time of bugs of priority 4 and 5 exceeds 100 days. In our study we look at a wider range of bug history, and more projects, to test how various factors affect bug-fix time.

3. DEFINITIONS

Developer reputation. Similar to Hooimeijer et al. [8] and Guo et al. [6], we used the following metric to quantify reputation:

$$Reputation(D) = \frac{Opened(D) \cap Fixed(D)}{Opened(D) + 1} \qquad (1)$$

As shown in Equation 1, the reputation of a developer is measured as the ratio of the number of bugs a developer D has opened and fixed to the number of bugs that D has opened, plus one. The significance of the "+1" in the denominator is to help prevent developers who have fixed fewer bugs achieve artificially high reputations.

Bug severity. To indicate the urgency of a bug, bugs are assigned a *severity*. For example in Mozilla and Eclipse, bug severity is a number from 1 to 7 which corresponds to "blocker," "critical," "major," "normal," "minor," "trivial," and "enhancement."

Number of developers. If more developers are involved in the bug-fix process, i.e., the bug is tossed from one developer to another before it is finally resolved, the bug-fix time increases, as shown in previous work [3].

Attachments. An increase in number of attachments (patches submitted by developers to fix a bug) should increase the bug-fix time, as each patch has to go through the review process before the bug status can be confirmed to be resolved.

Dependencies. When a bug depends on other bugs, e.g., bug B_1 manifests itself only if bugs B_2 and B_3 are present in the source code, it is important to resolve B_2 and B_3 in order to completely resolve B_1. This will therefore increase the time taken to fix B_1.

4. EXPERIMENTS

We now present our data set, hypotheses, and results.

4.1 Data Set

For conducting our experiments we used bug reports from May 1998 to March 2010 for three Mozilla projects: 129,053 bug reports from Firefox, 19,885 bug reports from Seamonkey and 7,276 bug reports from Thunderbird. For Eclipse, we considered bugs numbers from 1 to 306,296 (October 2001 to March 2010). For the Chrome project, we studied 49,964 bug reports (August 2008 to November 2010). We used bug reports from all projects for quantifying the effect of bug-opener's reputation on the bug-fixing process (Section 4.2.3). We used bug reports from Mozilla and Eclipse only [2] for the remaining experiments (Section 4.2.1 and 4.2.2).

4.2 Hypotheses and Results

In this section we test the relationship between bug report attributes (that were used in prior work for building bug-fix time prediction models) and bug-fix time, via univariate and multivariate regression testing. We then test the influence of the bug-opener's reputation on the bug-fix process.

4.2.1 Multivariate Regression Testing

Multivariate regression is a technique that estimates how well a collection of independent variables can predict a dependent variable. Multivariate regression testing yields the coefficient of determination (R^2) that provides a measure of how well future outcomes are likely to be predicted by the model, an F-value which indicates the accuracy of the model, and a corresponding p-value (derived from the F-value). In addition, each independent variable has an associated t-value that indicates its statistical significance (for uniformity, instead of t-values we report corresponding p-values).

Results. We performed a multivariate regression test where the dependent variable was bug-fix time, in days, and the indepen-

[2]Mozilla and Eclipse use Bugzilla as bug tracker, while Chrome uses the Google Code bug tracker. Bugzilla archives information about patches and the list of developers that the bug was assigned to, sorted by date, which contains the data required for our regression testing; Google Code does not provide this information.

Project	**Adjusted R^2**	**F-Value**	**p-value**	**p-value for independent variables**			
				Number of developers	Severity	Attachments	Dependencies
Firefox	0.401	1857.51	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
Thunderbird	0.498	985.11	<0.0001	<0.0001	<0.0001	<0.0001	0.0292
Seamonkey	0.366	2473.95	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
Eclipse	0.301	7079.16	<0.0001	<0.0001	<0.0001	<0.001	<0.0001

Table 1: Multivariate regression results; the dependent variable is bug-fix time, in days.

Attribute	**Correlation with attribute** (FF=Firefox, SM=Seamonkey, TB=Thunderbird, EC=Eclipse)															
	Number of developers				**Severity**				**Attachments**				**Dependencies**			
	FF	SM	TB	EC	FF	SM	TB	EC	FF	SM	TB	EC	FF	SM	TB	EC
Days	0.625	0.601	0.692	0.526	-0.186	-0.178	-0.207	-0.253	0.125	0.164	0.153	-0.001	0.121	0.115	0.157	0.035
Developers					-0.187	-0.210	-0.197	-0.235	0.304	0.324	0.371	0.144	0.271	0.243	0.308	0.194
Severity									-0.066	-0.134	-0.132	-0.009	-0.061	-0.039	-0.08	-0.056
Attachments													0.394	0.211	0.398	0.128

Table 2: Univariate regression results.

dent variables were: bug severity, number of attachments (patches), bug dependencies and number of developers involved in the bug-fix process. In Table 1 we present the results of our multivariate regression testing; the first column contains the project name, the second column shows goodness of fit, the third column shows the F-value, and the remaining columns contain the p-values, for the model and for each independent variable. We find that the R^2 ranges between 30% and 49% for the projects we consider, which denotes the low predictive power of the models. For example, for Firefox (row 2) the R^2 value is 0.4016, which means that the independent variables used in building the regression model only contribute to 40.16% of accurate prediction. However, the low p-values associated with the independent variables indicate that all features contribute to the prediction model, but more features (independent variables) would need to be incorporated to increase the goodness of fit.

Conclusions. In summary, the variables used in prior work for building prediction model are useful, however the low R^2 values indicate that more independent variables need to be used for improving the prediction accuracy of the model.

4.2.2 Univariate Regression Testing

Machine learning research [7] has shown that effective feature sets should contain features that are highly correlated with (predictive) of the output class, but are uncorrelated with (not predictive of) each other. Moreover, prior efforts [1] have used linear univariate prediction models. Therefore, in addition to multivariate regression testing, we also perform univariate regression testing to find out how features selected in prior classification-based bug-fix time prediction models correlate with each other and with bug-fix time. Univariate regression analyses (e.g., Pearson and Spearman's) return correlation values and an associated $|t|$-value with each correlation, to show the statistical significance of the correlation. For brevity, we present Pearson correlation coefficients only.[3]

Results. In Table 2 we present the results of our univariate regression testing. For brevity, we omit presenting the p-values for correlations, but we found all of them to be less than 0.0001, indicating that they are valid at the 1% significance level. The results indicate that features such as bug severity, number of attachments or bug dependencies, do not correlate well with bug-fix time, which calls into question the use of these attributes for building univariate linear prediction models. However, similar to Anbalagan et al. [1]'s observation for the Ubuntu project, we found that bug-fix time increases when more developers are involved in a bug fix (see the higher values in the "Days-Developers" cells).

Conclusions. The results of the univariate regression testing show that most of post-submission bug report features, such as number of attachments or bug severity, do not exhibit high correlation with bug-fix time. Therefore, we suspect that the successful predictions made in prior work might be due to a combination of data set choice and feature selection, a problem known as "optimistic bias" in machine learning [10]. To confirm that there is no such bias in the prediction, two steps can be taken for designing prediction models in the future: (1) training the model with larger data sets, preferably with the entire bug history of an application, and (2) choosing multiple applications to verify the generality of the prediction model.

4.2.3 Influence of Bug-opener's Reputation on Bug-fix Likelihood

Guo et al. [6] have found that for two Microsoft products, Windows Vista and Windows 7, bugs reported by developers with higher reputation were (1) more likely to get fixed, and (2) fixed faster. They found that there is a linear relationship between a bug-opener's reputation and the number of bugs he/she has fixed. Hooimeijer et al. [8] also used bug-opener's reputation to predict whether a new bug report will receive immediate attention or not. Therefore, we verified whether, in the five projects we considered, a bug-opener's reputation correlates with bug-fix time or with the probability of the bug being fixed. In Figure 1 we plot the bug-opener's reputation (x-axis) versus the percentage of the bugs he/she opened that eventually got fixed (y-axis); as we can see, there is no clear relation between these two variables. For completeness, in Table 3 we show the results of two correlation analyses: (1) the correlation between the bug-opener's reputation and the percentage of the bugs he/she opened that eventually got fixed, and (2) the correlation between the bug-opener's reputation and the average time taken to fix a bug reported by him/her. As can be seen in the table, for the projects we considered, we found low correlation values between a bug-opener's reputation and bug-fix likelihood or bug-fix time. We also found that the bug-fix time is independent of whether the bug-opener is involved in the bug-fixing process or not.

[3]The Spearman's correlation results have the same trend for our data sets as Pearson correlation results.

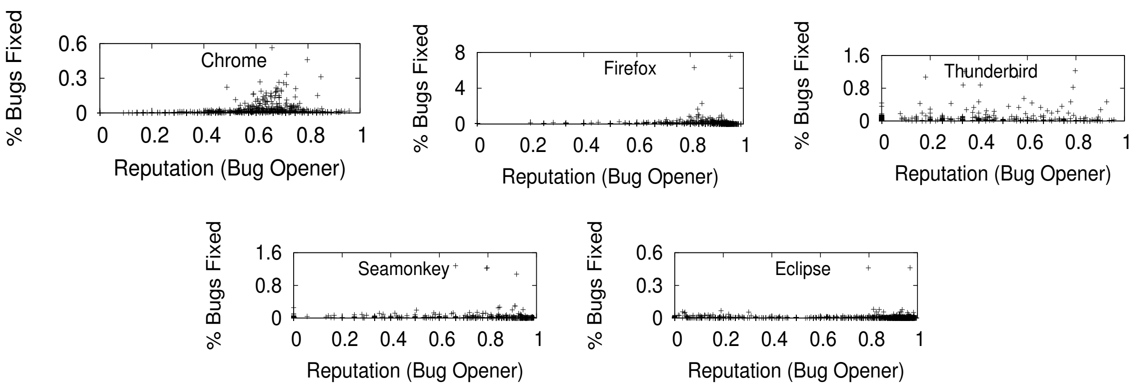

Figure 1: Bug-opener's reputation v. percentage of bugs reported by him/her that eventually got fixed.

Attribute	Correlation with bug-opener's reputation				
	Firefox	*Seamonkey*	*Thunderbird*	*Eclipse*	*Chrome*
Percentage of bugs eventually fixed	0.0559	0.0568	0.0471	0.0185	0.1507
Average bug-fix time	0.0023	0.1646	0.3278	0.4837	0.0878

Table 3: Correlation between bug-opener's reputation, percentage of bugs fixed and average time taken to fix a bug.

5. FUTURE WORK

The results of our investigation point to three significant future research problems and directions:

Quantifying the importance of bug-opener's reputation in the bug-fixing process. Empirical studies on commercial software have demonstrated how the reputation of the bug-opener's influences whether a bug gets fixed [6]. In our study we found that no such influence drives the bug-fixing process in widely-used open source software. The first research question we pose is: is there an advantage in prioritizing bugs based on bug-opener's reputation, as observed in the commercial software development process?

Selecting attributes that correlate with bug-fix time to design prediction models. We found that most attributes used by prior work to predict bug-fix time do not correlate with bug-fix time when analyzed in isolation. Feature-selection research in machine learning has shown the importance of correlation of attributes with the predicted variable for improving accuracy [7]. Therefore, the next research challenge that follows from our results is: which attributes should we choose so that they correlate with bug-fix time when tested in isolation, to improve existing prediction models?

Building generalized prediction models. We found that existing prediction models are based on assumptions that do not generalize well to other projects; this is particularly problematic as Mozilla and Eclipse have been used as bug study and prediction benchmarks in prior work. The challenge is to build bug-fix time prediction models that can be validated on a wider range of software projects. Additionally, validation on multiple projects would discard the chances of bias in feature selection.

6. CONCLUSIONS

Bug-fix time prediction is useful in software evolution, especially for coordinating the development effort during bug triaging. In this paper we demonstrate that, unlike in commercial projects, the bug-fix time in open source projects is not influenced by the bug-opener's reputation. We also show, using regression analysis, that various bug report attributes which have been previously used to build bug-fix time prediction models do not always correlate with bug-fix time. These findings put forward the research challenges of finding additional attributes for designing more general bug-fix time prediction models, and assessing the advantage of using bug-opener's reputation in the bug-fixing process.

7. REFERENCES

[1] P. Anbalagan and M. Vouk. On predicting the time taken to correct bug reports in open source projects. In *ICSM*, 2009.

[2] N. Bettenburg et al. What makes a good bug report? In *FSE*, 2008.

[3] P. Bhattacharya and I. Neamtiu. Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging. In *ICSM*, 2010.

[4] G. Canfora and L. Cerulo. Supporting change request assignment in open source development. In *SAC*, 2006.

[5] E. Giger, M. Pinzger, and H. Gall. Predicting the fix time of bugs. In *RSSE*, 2010.

[6] P. J. Guo et al. Characterizing and predicting which bugs get fixed: An empirical study of Microsoft Windows. In *ICSE*, 2010.

[7] M. A. Hall. *Correlation-based Feature Subset Selection for Machine Learning*. PhD thesis, Department of Computer Science, University of Waikato, 1999.

[8] P. Hooimeijer et al. Modeling bug report quality. In *ASE*, 2007.

[9] S. Kim et al. How long did it take to fix bugs? In *MSR*, 2006.

[10] L. Li, J. Zhang, and R. M. Neal. A Method for Avoiding Bias from Feature Selection with Application to Naive Bayes Classification models. *Bayesian Analysis*, 2008.

[11] A. Mockus et al. Two case studies of open source software development: Apache and Mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3), 2002.

[12] L. D. Panjer. Predicting eclipse bug lifetimes. In *MSR*, 2007.

[13] T. Zimmermann et al. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In *FSE*, 2009.