A Model for Predicting Bug Fixes in Open Source Operating Systems: an Empirical Study

Paolo Ciancarini
Università di Bologna and CINI
Italy
paolo.ciancarini@unibo.it

Alberto Sillitti
Innopolis University and CINI
Russian Federation
a.sillitti@innopolis.ru

***Abstract*—This paper proposes an adaptation to the open source environment (Linux Kernel and OpenSolaris) of a model for predicting which bugs get fixed in the Microsoft Windows operating system. We have analyzed the entire bug repositories containing 16,136 bug reports reported in about 8 years of activity of the project (from 2002 to 2010) for the Linux Kernel and 16,301 bug reports reported in about 3 years of activity of the project (from 2007 to 2010) for OpenSolaris. According to the data analyzed and the descriptive models produced, we have found that (a) bugs reported by people with better reputation and bugs in which more people are involved are more likely to get fixed, (b) reassigning or re-opening bugs are not affecting the fix likelihood, and (c) managing bugs in the same location increases the fix likelihood. The predictive model defined has a precision of 61% and a recall of 39% for the Linux Kernel and a precision of 76% and a recall of 73% for OpenSolaris. These results are comparable with the ones for Microsoft Windows. document.**

Keywords-bugs; predictive models; open source

I. INTRODUCTION

Bugs are one of the major problems in software development since the very early era of computers (Brooks, 1995). In any kind of software products, not all bugs are fixed. This is because different bugs have a different impact on the user and bug fixes requires a significant amount of time and effort (Ciancarini et al., 2015; Ciancarini et al., 2016; Di Bella *et al.*, 2013b; Pedrycz et al., 2015; Remencius *et al.*, 2016). Moreover, in popular products, the number of bug reported is usually very high and there are not enough resources to fix them all. As discussed in Guo *et al.* (2010), a significant amount of effort is devoted to bugs that remain unfixed: all this effort is wasted. For these reasons, it is essential to identify as fast as possible the bugs that are fixed and the one that remain not fixed. To support this identification many different approaches are possible based on the analysis of the development process (Abrahamsson *et al.*, 2007; Di Bella *et al.*, 2013; Petrinja *et al.*, 2010; Sillitti *et al.*, 2012) and the analysis of the code (Jermakovics et al., 2008; Jermakovics et al., 2011; Pedrycz *et al.*, 2002).

This paper adapts the work of Guo *et al.* (2010) developed on the Microsoft Windows operating system to open source alternatives, namely the Linux Kernel and OpenSolaris. To adapt better the model, we have considered the development of the two systems till 2010 since the original study was of that year. In this study, we have investigated the Linux Kernel that was about 12 MLOC and OpenSolaris, about 20 MLOC.

To this end, the major outcomes of the work of Guo *et al.* (2010) and ours are summarized in Table I.

TABLE I. MAJOR FINDINGS OF THE PAPER OF GUO *ET AL.* (2010) COMPARED TO OUR STUDY

Guo *et al.* (2010)	**Our study**
"People more successful in getting their bug fixed in the past are more likely to get their bugs fixed in the future."	Confirms
"The more people are involved in the bug life cycle, the more likely it is fixed."	Confirms
"Reassignments of bugs are not always detrimental to the likelihood of bug fix."	Confirms
"Re-openings of bugs are not always detrimental to the likelihood of bug fix."	Confirms
"Bugs assigned across teams or locations are less likely to get fixed."	Confirms
Definition of a model for predicting the probability of fix of a given bug at the time of opening.	Similar model

This paper is structured as follows: Section II presents the state of the art of the area; Section III describes the research methodology and the data; Section IV and V provide a descriptive and a predictive statistical models; finally, Section VI draws the conclusions and provides directions of future work.

II. RELATED WORK

There are several studies related to bug reports. In particular, the are several papers related to bug analysis and prediction. However, in our knowledge, the only study related to the likelihood of fix of a bug is the one by Guo *et al.* (2010) related to the bug reports of the Microsoft Windows operating system. Moreover, no studies are available in the open source area.

Readers interested in an overview of the state of the art and of the related literature can refer to the paper of Guo *et al.* (2010).

III. RESEARCH METHODOLOGY

A. Environment

We have performed a quantitative analysis extending the approach followed by Guo *et al.* (2010) to the analysis of open source projects. In particular, we have considered the bug

DOI reference number: 10.18293/SEKE2016-199

repositories of the Linux Kernel and OpenSolaris till 2010. We have selected such software because they are both open source operating systems and the availability of a large amount of information. In particular, all the bug reports are publicly available. Moreover, the timeframe is limited to 2010 since we compare with the study of that year.

The main differences with the study of Guo *et al.* (2010) are related to the open nature of the considered projects, in particular: a) we had no possibility to propose a questionnaire to the developers; b) we had no information about the actual geographical location of the developers; c) we had no information about the organizational structure of the contributors, except from the information we can get from their email addresses that is basically the organization they belong to. In particular, we were not aware if people in the same organization share the same boss and/or belong to the same working team.

Even with the listed differences, it was possible to compare the core part of the studies and propose an approach to make a similar analysis possible in any public bug repository.

B. Data

We have considered all the bugs stored in the bug tracking systems of the Linux Kernel and the OpenSolaris operating systems till 2010.

For each bug report, we extracted the history of the bug and the following set of information:

- **Editor:** who has modified the bug report
- **State:** if the bug is still open or not
- **Component:** the name of the affected component
- **Severity:** potential impact of the bug on the system: blocking, major, minor, trivial, etc.
- **Opener:** who has opened the bug
- **Assignee:** who is assigned to manage the bug
- **Resolution:** if the bug has been resolved. The study considers only bugs that are marked as resolved as fixed or not. Such other status includes bugs that are identified as duplicates, bugs that will not be fixed, etc.

Compared to the study of Guo *et al.* (2010), we were unable to collect the following information:

- **Bug source:** the source of the bug report such as internal testing, a customer, code review, etc.
- **Bug type:** bug in code, specification, test, etc.

In the projects considered, the typical life cycle of a bug is the following: when a bug is opened, all the described fields are filled in by the opener, than the bug is edited one or more times (including reassignments to other developers) until the bug is resolved. However, it might happen that a resolved bug is reopened because it was not solved properly.

Our goal is to characterize bugs that have been resolved successfully and in the case of re-openings, we consider only the final status in 2010.

Beside the bug reports, Guo *et al.* (2010) presented data related to a questionnaire that was filled in by a large number of developers and data related to the organization of the development teams in Microsoft. In our study, these data were not available since our data come from open source projects. The data we miss are the following:

- **Geographical data:** information about the co-location of people in the same office, building, or campus.
- **Organizational data:** information developers belonging to the same team.
- **Qualitative data:** information collected through a questionnaire asking the developers about the factors that influence the likelihood for a bug of being fixed.

According to their results, geographical and organizational data have some impact on the likelihood for a bug of being fixed. However, these factors have a limited impact in open source projects since most of them are extremely distributed and do not have a hierarchical organization.

IV. Descriptive Statistical Models

A. Building the models

We have built two models to predict the probability that a bug will be fixed in the two operating systems considered. To build the models, we have used the same approach described in Guo *et al.* (2010). The identified factors and the related coefficients are listed in Table II. We also found that all the significant factors are statistically significant at $p < 0.001$, according to an Analysis of Deviance chi-square test (Hosmer and Lemeshow, 2000). We have checked for interactions between factors and we have verified that there are no statistically significant interactions.

TABLE II. Descriptive logistic regression model for bug fix probability. "N.A." values refer to factors that are not available in the specific model; "N.S." values refer to factors that are not statistically significant in the specific model.

Factor	Coefficient		
	Windows Vista	***Linux Kernel***	***OpenSolaris***
Reputation of bug opener	2.19	2.03	1.83
Reputation of 1st assignee	2.46	0.26	2.39
Opened by a temporary employee	-0.12	n.a.	n.a.
Initial severity level	0.03	n.s.	0.06
Opener/any assignee same manager	0.68	n.a.	n.a.
Opener/any assignee same building	0.27	n.a.	n.a.
Severity upgrade	0.26	n.s.	0.28

DOI reference number: 10.18293/SEKE2016-199

Factor	Coefficient		
	Windows Vista	*Linux Kernel*	*OpenSolaris*
Editors	0.24	n.s.	0.11
Assignee buildings	-0.26	n.a.	n.a.
Re-openings	-0.13	n.s.	n.s.
Component changes	-0.23	-0.20	-0.20
Opener and 1st assignee in the same organization	n.a.	0.47	n.s.
Opener/any assignee same organization	n.a.	n.s.	0.47
Opener and 1st assignee are the same person	n.a.	n.s.	1.04
Assignees	n.a.	0.18	n.s.
Comments	n.a.	0.01	n.s.
Severity changes	n.a.	n.s.	-0.31

The purpose of this model is to describe the factors affecting the bug fixes but it cannot be used to make predictions since it includes factors that are not available at the time of creation of the bug report. A predictive model is described in Section V.

B. Meaning of the regression coefficients

As it happens in Guo *et al.* (2010), the meaning of the parameters of the logistic regression model is intuitive. The list of factors presented includes all the factors coming from a quantitative experimentation (in the case of Windows Vista, the study of Guo *et al.* (2010) included some factors collected through questionnaires, therefore these have been omitted).

Moreover, some of the factors considered in Guo *et al.* (2010) are not considered in our models:

- **Opened by a temporary employee:** this factor does not have any meaning in open source projects since a significant amount of contributions is provided by volunteers (the percentage of paid developers and volunteers varies a lot in different open source projects)
- **Opener/any assignee same manager:** this factor is not considered in our models for the same reason as the previous factor.
- **Opener/any assignee same building:** this factor is not considered in our models since we are not aware of the physical location of the developers. In many open source projects most of the developers are geographically distributed.

The last 6 factors are not included in Guo *et al.* (2010) for different reasons:

- **Opener and 1st/any assignee in the same organization:** these were obviously not considered since all the development was inside Microsoft.
- **Opener and 1st assignee are the same person:** this factor is analyzed in the paper (even if no values are reported for confidentiality) but it is not present in the proposed model.
- **Assignees:** the authors in Guo et al., 2010 preferred to use the number of assignee buildings. However, in our models we are unable to do it since we do not know the location of each developer.
- **Comments:** this factor is not considered in the original paper. In our models it has a very small effect and it is statistically significant only for the Linux Kernel.
- **Severity changes:** this factor is not considered in the original paper that considers only the fact that the severity has been upgraded, downgraded, or not modified. This factor has a relevant effect in the OpenSolaris model while it is not significant in the Linux Kernel one.

C. Interpretation of the models

According to the two developed models, the following factors are correlated with the bug fix probability:

- **Reputation of bug opener and of the 1st assignee:** as discussed in Section IV.C, these factors are strongly positively correlated with the likelihood of fix in both the analyzed projects (actually the correlation is weaker for the 1st assignee in the Linux Kernel but it is still relevant). This could mean that bug openers with a relevant track record in having fixes are going to receive more fixes in the future. This could be because the of effectiveness of their reports. The same is true for the 1st assignee since he could be effective in fixing he bug (effective management of the assignation of bugs to the correct person) or he can redirect the bug to the best person able to fix it. In any case, it is liked with an effective management of the assignation of the bugs.
- **Initial severity level:** this factor has a positive correlation with the likelihood of fix of a bug. However, this is statistically significant only in the OpenSolaris model. This could be related to the fact that the number of severity changes and severity upgrades are not significant in the Linux Kernel model. Therefore, the significance of this factor could also be related to the same motivation: a more structured management of the severity levels that might occur in OpenSolaris compared to the Linux Kernel. This is also supported by the fact that Guo *et al.* (2010) in Microsoft reports a similar value.
- **Severity upgrade and Severity changes:** changes in severity affects the likelihood of bug fix. However, in our two data sets, they affect only OpenSolaris, while they are not significant in the Linux Kernel. This behavior could be linked with a more structured management of the severity levels that might occur in OpenSolaris compared to the Linux Kernel since a

DOI reference number: 10.18293/SEKE2016-199

consistent part of its development was done inside Sun Microsystems (now Oracle) (therefore, the corporate organization could be more similar to the Microsoft one reported in Guo *et al.* (2010)).

- **Editors:** the number of editors are positively correlated with the likelihood of fix, however this factor is significant only for the OpenSolaris model. This could also be related to the more structured contributions from Sun Microsystems, as editors and edits could be linked more strictly with the knowledge of the system and a deeper investigation of the bug (and maybe more information in the bug reports).
- **Component changes:** this factor is negatively correlated with the likelihood of bug fix. This happens in both models and could be related to a vague bug report that is not able to identify precisely the source of the error.
- **Opener and 1st assignee in the same organization and Opener/any assignee same organization:** these factors are positively correlated with the likelihood of bug fix, however only one of them is statistically significant in each of the two models. In any case, they identify a link between the opener and the management of the bug inside the same organization. This could relate the interest/usage of a specific set of functionalities of a product and the involvement in its development.
- **Opener and 1st assignee are the same person:** this factor is positively correlated with the likelihood of bug fix. However, it is statistically significant only in the OpenSolaris model. This could be related to the size of the community since Linux is more adopted than OpenSolaris, therefore the factor is not affecting significantly the likelihood of bug fix. On the contrary, in OpenSolaris the community of reporters maybe overlap more with the community of developers.
- **Assignees:** this factor is positively correlated with the likelihood of fix, however this factor is significant only for the Linux Kernel model. In any case, this factor is linked to the number of people involved in the fix of a bug. This could be related to the community approach to software development in open source projects (even if OpenSolaris is open source, its history as an open product is quite recent and a large part of its community was linked to Sun Microsystems).
- **Comments:** this factor has a positive correlation with the likelihood of fix of a bug. However, this is statistically significant only in the Linux Kernel model. This could be related to the community approach to software development in open source projects as described in the previous factor.

D. Comparison with other studies

In our knowledge, the only comparable study is the one of Guo *et al.* (2010). The main similarities are the following:

- The reputation of bug opener and of the 1st assignee are very important to determine if a bug will be fixed. For the bug opener our two models present coefficients close to the value of the study of Guo *et al.* (2010), for the 1st assignee the strongest similarity is between the original study and the OpenSolaris model. This could happen because even if OpenSolaris is open source, its roots are in the Solaris operating system developed at Sun Microsystems that provides a large amount of contributions to the project.
- Re-opening or reassigning a bug does not affect the bug fix likelihood.
- Bugs assigned across teams (organizations in our data sets) are less likely to get fixed.
- Changing the component to which the bug is associated decreases the bug fix likelihood. It is an indication that it is not clear where the problem is and/or the bug report is not adequate to identify the problem.

And the main differences are:

- We are unable to evaluate the role of the hierarchy of the organization in the bug fix likelihood. This is because in our projects the interactions among developers are more complex than in the study of Guo *et al.* (2010). In open source projects there is a mixture of contributions including different companies, associations, and volunteers. However, if the opener and assignees belongs to the same organization (or even it is the same person) the bug fix likelihood increases.
- The number of comments in a bug report increases the bug fix likelihood in the Linux Kernel model. This could be because of the collaborative approach of the development in open source projects (OpenSolaris does not have a long tradition of community-based development).
- Severity upgrades and changes affect positively and negatively only the fix likelihood of OpenSolaris, this could be related to a more structured management of such information as it happens in Microsoft.

Moreover, the two studies differ also about some collected data due to the different environments considered.

Some of the factors considered in the study of Guo *et al.* (2010) are not included in our model:

- **Opened by a temporary employee:** this factor does not have any meaning in open source projects since a significant amount of contributions is provided by volunteers (the percentage of paid developers and

DOI reference number: 10.18293/SEKE2016-199

volunteers varies a lot in different open source projects)

- **Opener/any assignee same manager:** this factor is not considered in our model for the same reason as the previous factor.
- **Opener/any assignee same building:** this factor is not considered in our model since we are not aware of the physical location of the developers. In many open source projects most of the developers are geographically distributed.

However, we have extended the model proposed by Guo *et al.* (2010) with 6 factors that are relevant for open source projects:

- **Opener and 1st/any assignee in the same organization:** these are obviously important in open source development since the development is performed in a distributed way with contributions of several organizations.
- **Opener and 1st assignee are the same person:** this factor has the same motivation as the one before.
- **Assignees:** our models use such information while the paper of Guo *et al.* (2010) preferred to use the number of assignee buildings. Considering open source development, it is usually difficult to know the location of each developer.
- **Comments:** comments are very useful in open source development since the communication among developers is usually mediated by tools. However, in our models, it has a very small effect and it is statistically significant only for the Linux Kernel.
- **Severity changes:** this factor could be an indicator of the policies for managing bugs in the project. It has a relevant effect in the OpenSolaris model, while it is not significant in the Linux Kernel one.

The model we have proposed is easily replicable in any open source project since it is based on information that nearly all open source projects provide.

V. Predictive Statistical Models

A. Building the models

Using the experience in the development of the descriptive model described in the Section IV, we have built two models to predict the probability that a bug will be fixed in the two operating systems considered. To do that, we have replicated the approach of Guo *et al.* (2010). The identified factors and the related coefficients are listed in Table III. Also in this case, we found that all the factors are statistically significant (mostly at $p < 0.001$) according to an Analysis of Deviance chi-square test (Hosmer and Lemeshow, 2000). We have checked for interactions between factors and we have verified that there are no statistically significant interactions.

In particular, the predictive models include only factors that are known at the time of the bug submission, therefore only a subset of the factors of the descriptive models are included in the new one. Obviously, the related coefficients are slightly different.

TABLE III. Predictive logistic regression model for bug fix probability. "n.a." values refer to factors that are not available in the specific model; "n.s." values refer to factors that are not statistically significant in the specific model; "*" is significant at $p < 0.05$; "**" is significant at $p < 0.01$.

Factor	Coefficient		
	Windows Vista	***Linux Kernel***	***OpenSolaris***
Reputation of bug opener	2.19	1.75	1.74
Reputation of 1st assignee	2.39	0.18*	2.63
Opened by a temporary employee	-0.04	n.a.	n.a.
Initial severity level	0.06	n.s.	0.04**
Opener/any assignee same manager	0.27	n.a.	n.a.
Opener/any assignee same building	0.03	n.a.	n.a.
Opener and 1st assignee in the same organization	n.a.	0.48	0.53
Opener and 1st assignee are the same person	n.a.	n.s.	0.80

B. Performances of the models

We have investigated the precision and the recall of the developed models and even if our models are based on less information than the ones of Guo *et al.* (2010), we have obtained comparable performances. In particular, we have obtained a precision of 0.61 and a recall of 0.41 for the Linux Kernel model and a precision of 0.76 and a recall of 0.73 for the OpenSolaris one.

Such results have been obtained applying the same approach described in Section IV.

C. Comparison with other studies

Since in our knowledge, the only similar study is the one of Guo et al. (2010), we have compared our model with theirs (Table IV). In particular, the Linux Kernel model performs worse than the original model, while the OpenSolaris one performs better. This could be related to the fact that even if OpenSolaris is an open source product, it comes from a closed source system and still mostly developed by Sun Microsystems.

TABLE IV. Comparison of the precision and recall of the models

DOI reference number: 10.18293/SEKE2016-199

	Guo *et al.*, 2010		Our study	
	Windows Vista	***Windows 7***	***Linux Kernel***	***OpenSolaris***
Precision	0.67	0.68	0.61	0.76
Recall	0.68	0.64	0.41	0.73

VI. Conclusions and Future Work

This study aimed at extending in an open source environment an empirical investigation performed inside Microsoft. Our results has confirmed the claims of the study of Guo *et al.* (2010) and was able to build two predictive models with comparable performances using less information.

We think that our study provides a contribution to the generalizability of the findings of empirical studies in software engineering and we aim at replicating the study in several more open source projects.

Acknowledgment

This paper has been partially supported by Consorzio Interuniversitario Nazionale per l'Informatica (CINI) through the ECSEL project MANTIS (662189).

References

[1] Abrahamsson P., Moser R., Pedrycz W., Sillitti A., Succi G., "Effort Prediction in Iterative Software Development Processes – Incremental Versus Global Prediction Models", 1st International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), Madrid, Spain, 20 - 21 September 2007.

[2] Brooks F. P., *The Mythical Man-Month: Essays on Software Engineering*, Addison-Wesley, 2nd edition, 1995.

[3] Ciancarini P., Poggi F., Rossi D., Sillitti A., "Improving bug predictions in multi-core cyber-physical systems", 4th International Conference on Software Engineering for Defense Applications (SEDA 2015), Rome, Italy, 26 - 27 May 2015.

[4] Ciancarini P., Poggi F., Rossi D., Sillitti A., "Mining Concurrency Bugs", Embedded Multi-Core Systems for Mixed Criticality Summit 2016 at CPS Week 2016, Vienna, Austria, 11 April 2016.

[5] Di Bella E., Sillitti A., Succi G., "A multivariate classification of open source developers", *Information Sciences*, Elsevier, Vol. 221, pp. 72 - 83, February 2013.

[6] Di Bella E., Fronza I., Phaphoom N., Sillitti A., Succi G., Vlasenko J., "Pair Programming and Software Defects – a large, industrial case study", Transaction on Software Engineering, IEEE, Vol. 39, No. 7, pp. 930 - 953, July 2013.

[7] Guo P. J., Zimmermann T., Nagappan N., Murphy B., "Characterizing and Predicting Which Bugs Get Fixed: An Empirical Study of Microsoft Windows", *32nd International Conference on Software Engineering (ICSE 2010)*, Cape Town, South Africa, 2 - 8 May, 2010.

[8] Hooimeijer P., Weimer W., "Modeling bug report quality", *22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007)*, Atlanta, GA, USA, 5 - 9 November, 2007.

[9] Hosmer D. W., Lemeshow S., *Applied Logistic Regression*, John Wiley & Sons, 2nd edition, 2000.

[10] Jermakovics A., Moser R., Sillitti A., Succi G., "Visualizing Software Evolution with Lagrein", 22nd Object-Oriented Programming, Systems, Languages & Applications (OOPSLA 2008), Nashville, TN, USA, 19 - 23 October 2008.

[11] Jermakovics A., Sillitti A., Succi G., "Mining and Visualizing Developer Networks from Version Control Systems", 4th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2011) at ICSE 2011, Honolulu, HI, USA, 21 May 2011.

[12] Pedrycz W., Succi G., Chun M. G., "Association Analysis of Software Mesures", International Journal of Software Engineering and Knowledge Engineering, Vol. 12, No. 3, pp. 291 - 316, 2002.

[13] Pedrycz W., Succi G., Sillitti A., Iljazi J., "Data description: A general framework of information granules", Knowledge-Based Systems, Elsevier, Vol. 80, pp. 98 - 108, May 2015.

[14] Petrinja E., Sillitti A., Succi G., "Comparing OpenBRR, QSOS, and OMM Assessment Models", 6th International Conference on Open Source Systems (OSS 2010), Notre Dame, IN, USA, 30 May - 2 June 2010.

[15] Remencius T., Sillitti A., Succi G., "Assessment of software developed by a third-party: A case study and comparison", Information Sciences, Elsevier, Vol. 328, pp. 237 - 249, January 2016.

[16] Sillitti A., Succi G., Vlasenko J., "Understanding the Impact of Pair Programming on Developers Attention: A Case Study on a Large Industrial Experimentation", 34th International Conference on Software Engineering (ICSE 2012), Zurich, Switzerland, 2 - 9 June 2012.

DOI reference number: 10.18293/SEKE2016-199