Developer Involvement Considered Harmful?

An Empirical Examination of Android Bug Resolution Times

Subhajit Datta
Singapore University of Technology and Design
subhajit.datta@acm.org

Proshanta Sarkar
Heritage Institute of Technology, Kolkata, India
proshant.cse@gmail.com

Subhashis Majumder
Heritage Institute of Technology, Kolkata, India
subhashis.majumder@heritageit.edu

ABSTRACT

In large scale software development ecosystems, there is a common perception that higher developer involvement leads to faster resolution of bugs. This is based on conjectures around more "eyeballs" making bugs "shallow" – whose validity and applicability are not without dispute. In this paper, we posit that the level of developer attention as well as its extent of diversity influence how quickly bugs get resolved. We report results from a study of 1,000+ Android bugs. We find statistically significant evidence that attention and diversity have contrasting relationships with the resolution time of bugs, even after controlling for factors such as interest, importance, dependency etc. Our results can offer helpful insights on team dynamics and project governance.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Life cycle*

General Terms

Experimentation

Keywords

Android, attention, diversity, social network analysis, LDA

1. INTRODUCTION

Background: As discussed at length in Raymond's *Cathedral and the Bazaar*, often the success of open source software comes out of effectively harnessing the social processes that underlie the highly interactive enterprise of open source development [11]. Over the last decade and half, development and maintenance of large scale open source systems has become a predominantly collective enterprises involving many individuals. Concomitantly, the conventional wisdom has come to recognize the extent and variety of developer involvement to be beneficial to development activities, most notably bug resolution. The so called Linus' Law [11], opinions around it [5] and efforts at empirically validating it [10],[4], point to the subtleties around establishing whether, and how much developer involvement helps in faster resolution of bugs. In this paper, we argue that the *variety* as well as *extent* of developer involvement stands to influence the resolution time of bugs. (Due to lack of space, the study of reopened bugs is outside the scope of this work.) We report results from investigating the following research question on a corpus of 1,000+ Android[1] bugs: *How does the extent and variety of developer involvement relate to the resolution times of Android bugs?*

Hypotheses: We refine the research question into the following hypotheses:

- *H1: Bugs which attract higher developer attention get resolved faster.*
- *H2: Bugs which attract more diverse developer interest get resolved faster.*

Related Work: To understand the influences on bug resolution time we consider the following established factors influencing outcomes in software development: *Dependency* between units of work is critical in determining how quickly development activities are completed [3]. The *importance* of a task, in terms of its priority or severity is also known to be a key factor in ensuring its early closure [7], [3], [15]. The level of project *responsibility* of the owner of a task plays an important role in the timing of the task's completion [3], [15]. Additionally, in an open source project, community *interest* plays a very important part in guiding debugging activities. Large scale software development today is highly interactive. A common mechanism of collaboration between a large group of developers is co-commenting on units of work. Commenting and responding to comments play a vital role in knowledge circulation in the development ecosystem, and influence the outcome of development activities [15]. This is not an exclusive list of factors influencing outcomes in software development, but they are the most relevant in the context of our study as established in literature.

Organization of the paper: In the next section we describe the methodology of our study, followed by an outline of model development and threats to validity. The paper ends with a discussion of results and conclusions.

2. METHODOLOGY

Collecting data: We collected the Android bug report data from a publicly available online repository [12]. We parsed the XML file and stored the data in a specifically designed MySQL database to help easy querying. Each bug

[1] www.android.com

SSE'14 , November 16, 2014, Hong Kong, China

has a unique bug identifier, along with the following fields: title, status, owner, date opened, date closed, type, priority, component, stars, reported by, description. Each comment has the attributes - identifier of the bug commented upon, commenter, date of comment, textual content of the comments.

Cleaning and filtering data: Resolution time for each bug was computed as the integral number of days between date the bug was opened and the date it was closed. For our analysis, we filtered the original data-set of 2,546 bugs by only including bugs which have been commented by more than one developer. From this set, bugs with missing attributes or incorrectly recorded attributes (e.g. the opened date being later in time than the closed date) were excluded. Finally, the data-set was pruned to bugs which had a resolution time of one year or less. This was done under the assumption that bugs which have not been resolved for more than a year are unlikely to have attracted notable interest in the community. The final data-set analyzed consists of 1,016 bugs, and 73 unique developers who own at least one of these bugs. When we mention a "developer's bugs" in subsequent discussion, it means bugs which are owned by that developer. The onus of resolving a bug rests with its owner.

Generating the bug similarity network: In a large scale software development enterprise such as Android, when a bug is raised, its title and textual description are used to decide how similar it is to other bugs that were raised earlier [8]. On the basis of this judgment, the ownership of a bug is most likely to be given to a developer who has resolved similar bugs earlier. Thus the critical step of assignment ownership is based on evaluating the similarity between bugs.

To detect how similar bugs are to one another, we used a Latent Dirichlet Allocation (LDA) based approach. LDA considers a document to be a mixture of a limited number of topics, with each word in the document attributed to one of these topics [2]. Given a corpus of documents, LDA *discovers* a set of topics, the set of keywords associated with each of the topics and the mixture of these topics for each document in the corpus. To build the LDA based topic models, we have used the collapsed Gibbs sampling method [13].

In our case, the set $\mathcal{P}$ of title and description for all bugs in our data-set has been used as the text corpus (each document in this corpus is a stemmed set of keywords obtained from a bug's title and description) from which LDA discovers a set of topics $\Gamma = \{\tau_1 \cdots \tau_k\}$. From a text corpus LDA creates two sets of probability distributions. One of these sets models topic mixture over documents (denoted as $\Theta = \{\theta_p | p \in \mathcal{P}\}$) and the other set models keyword mixture over topics. For a bug p, we get a probability distribution θ_p over topics, and for a given topic, we get a probability distribution of keywords. In LDA, these two are taken to be Dirichlet distributions with parameters α and β respectively. Arriving at the optimal number of topics for a given corpus is an empirical process based on repeated trials. We need to vary α, β, the number of iterations (N) and the number of topics (K) to get the log likelihood value for the model which indicates its highest level of effectiveness [16]. Iterating over these parameters several thousand times, we selected 30 topics for our study. Beyond this number, the instances of repetitions in the keywords across the topics were going up substantially, thus indicating a low possibility of identifying further distinguishable topics. Having obtained the probability distribution over topics for each bug, we calculate the similarity between all pairs of bugs in our data-set using the symmetric Kullback Leibler Divergence (KLD) [9]. KLD is a distance measure between two probability distributions, given by the following equation, where $\bar{p}$ and $\bar{q}$ are probability distributions or normalized feature vectors:

$$KLD(\bar{p}, \bar{q}) = \sum_i p(i) log(\frac{p(i)}{q(i)}) + \sum_i q(i) log(\frac{q(i)}{p(i)})$$

We define a *Bug Similarity Network* (BSN), whose vertices (nodes) are bugs [4]. In BSN, two bugs are connected by an edge (undirected link) if they are *similar* to one another by the LDA and KLD based measure explained above. As we are interested in detecting the most significant level of similarity between bugs (to reduce false positives), we only connected two bugs by an edge in BSN if the corresponding KLD value was in the 96th percentile or above.

Examining hypotheses: To examine the hypotheses presented earlier, we develop multiple linear regression models as described next.

3. MODEL DEVELOPMENT

Independent variables: To validate the first hypothesis (H1), we need to identify a metric that captures developer attention on a bug. As mentioned earlier, commenting is a critical aspect of collaborative software development. We can reasonably assume that the number of comments on a bug is a reflection of the level of *attention* the bug receives from the development community. However, a bug may be commented many times by a small group of developers, including the bug's owner. To validate the second hypothesis (H2), we need to capture the level of *diversity* in the interest generated around a particular bug; this can be measured from the number of unique developers who do not own the bug, but comment on it. Thus, as independent variables in our model, we take ***Attention*** and ***Diversity*** as measured from the number of comments on a bug and the number of unique non-owner commenters on a bug, respectively.

Dependent variable: In both hypotheses H1 and H2, we are concerned with how quickly a bug gets resolved. Thus, as the dependent variable we consider ***TimeToResolve***, which is calculated from the resolution time of each bug.

Control variables: The objective of building statistical models is to investigate how the independent variables *Attention* and *Diversity* relate to the dependent variable *TimeToResolve* in the light of our hypotheses. But to establish such relationships unambiguously, we need to *control* for the peripheral influences on the dependent variable. These influences are reflected in the control variables. Based on earlier discussion, we consider ***Dependency***, ***Importance***, ***Responsibility***, and ***Interest*** as our control variables. These are calculated in the following ways.

The *degree* of a bug in BSN is the number of other bugs it is connected to via edges. As an established network metric, the degree of a vertex is a measure of the extent of its connection [1]. Bugs which are connected to many other bugs can be expected to have a higher level dependency on other parts of the system. We thus measure *Dependency* of a bug from its degree in BSN. The *Importance* of a bug is reflected in the priority field of Android bug reports. Each bug has a single owner – the developer who is responsible for

Table 1: Descriptive statistics of model variables

	Mean	Stdev	Skewness	Kurtosis
TimeToResolve	3.25	1.78	0.38	1.03
Dependency	5.79	2.74	0.41	0.13
Importance	0.7	0.08	1.12	9.7
Responsibility	130.15	109.99	0.81	-0.77
Interest	1.15	0.21	0.86	15.66
Attention	1.71	0.67	0.49	-0.34
Diversity	1.19	0.87	0.39	-0.51

resolving it. How much a developer can focus on a particular bug is dependent on the extent of his/her overall ownership responsibility of other bugs. Thus the *Responsibility* control variable for a particular bug is measured from the total number of bugs owned by the owner of that bug. Finally, to get a sense of how much interest a bug has generated in the Android community, we consider the "stars" field of bug report. It is "used in order to represent the number of people following a bug" [6]. For a bug, *Interest* is derived from the number of stars accumulated by the bug.

We recognize that *Dependency*, *Importance*, *Responsibility*, and *Interest* are not the only factors with notable influence on the resolution times of Android bugs. We considered several other factors in our preliminary analysis, but finally selected these to arrive at the most parsimonious model using multiple linear regression. Other modeling approaches such as Poisson regression, and negative binomial regression were considered but not pursued with, as our dependent variable is not a typically *count* measure, and its mean and variance are not close to another.

Model assumptions and variable transformations: Multiple linear regression rests on the assumptions of linearity, normality, and homoscedasticity of the residuals, and lack of multicollinearity between the independent variables. The residual properties were verified using histogram, Q-Q plot and scatter plot of the standardized residuals. After inspecting the descriptive statistics of the model variables, the following transformations were applied to the corresponding raw measurements, to ensure closer conformance to normality – natural logarithm for *TimeToResolve*, square root for *Dependency*, natural logarithm for *Importance*, eighth root for *Interest*, natural logarithm for *Attention*, and natural logarithm for *Diversity*. The descriptive statistics for the transformed variables used in the model are given in Table 1. Among the variables, *Importance* and *Attention* had a relatively high correlation (around 0.55), which is expected as high priority bugs tend to get commented more. Since the Variance Inflation Factors (VIF) of all variables were found to be below the upper limit of 10 in both the base and refined models (which are described below), no notable multicollinearity was detected [14]. On the basis of the above evidence, we conclude that assumptions of multiple linear regression hold within reasonable limits in our analysis [14].

Model description, validation, and results: Column I of Table 2 presents parameters of the *base model*, considering only the effects of control variables on the dependent variable, while column II gives details the *refined model* that also includes the independent variables. Superscripts of the coefficients give the range of their p values, as specified in the table's caption. The p value for each coefficient is calculated from the t-statistic and the Student's t-distribution. In the table's lower portion, model overviews are provided:

Table 2: Results of regression for the effects on bug resolution time.(Superscripts '***', '**', '*', '†' denote $p \leq 0.0001$, $p \leq 0.001$, $p \leq 0.01$, $p \leq 0.05$, respectively)

	I	**II**
	Base model	*Refined model*
Intercept	0.945	2.126^{**}
	(0.655)	(0.676)
Control variables		
Dependency	-0.04^{*}	-0.045^{*}
	(0.02)	(0.02)
Importance	-0.32	-0.382
	(0.652)	(0.642)
Responsibility	0.002^{**}	0.001^{**}
	(0.001)	(0.000)
Interest	2.397^{***}	1.327^{**}
	(0.463)	(0.509)
Independent variables		
Attention		-0.427^{*}
		(0.195)
Diversity		0.671^{***}
		(0.15)
N	1016	1016
R^2	0.0394	0.071
df	1011	1009
F	10.37	12.93
p	< 0.001	< 0.001

N is the number of data points used in building the model, in our case the number of bugs. R^2 is the coefficient of determination – the ratio of the regression sum of squares to the total sum of squares; indicator of the goodness-of-fit of the regression model in terms of the proportion of variability in the data-set that is accounted for by the model. *df* gives the degrees of freedom. F is the Fisher F-statistic – the ratio of the variance in the data explained by the linear model divided by the variance unexplained by the model. The p value for the overall model is calculated using the F-statistic and the F-distribution, and it indicates the statistical significance of the model. For the coefficients as well as the overall regression, if $p \leq$ *level of significance*, the corresponding result is taken to be statistically significant, based on null hypothesis significance testing.

From columns I and II of Table 2, we notice that both the base and refined models are statistically significant overall, and by adding the independent variables, the R^2 value increases by 81% from the base to refined models and the F-statistic also increases. Thus the independent variables have increased the explanatory power of the model. We highlight the influence of the model variables in the Discussion and Conclusions section.

4. THREATS TO VALIDITY

This is a study of existing data, rather than a controlled experiment. Therefore, correlation does not imply causation in the statistical models we developed. Threats to **construct validity** is concerned with whether model variables are measured correctly. In this paper, an approach based on LDA has been used to detect bug similarity. There are other approaches such as cosine similarity which may be simpler, but they do not take into account clusters of keywords that may occur together. Since BSN is an undirected network, the direction of dependence between bugs can not be inferred from it. The calculation of the dependent variable assumes

that the elapsed time between bug opening and closure denotes the actual time taken to resolve the bug. Calculation of the independent variables is based on the acts of commenting on bugs by developers; there may be other ways to measure levels of attention and diversity. We recognize the existence of these threats, but they are consistent with similar studies [15]. **Internal validity** establishes that a study is not affected by systemic errors and biases. As we only use data from Android, this threat is mitigated in our study. **External validity** ensures the generalizability of results. We have studied only one data-set in this paper. Also, the R^2 values of the models indicate that there may be several other factors relating to the resolution times of bugs which have not been considered. We plan to include them in our future work. Thus we recognize our results are not generalizable as yet. A study demonstrates **reliability** when the results are reproducible. Our results are reproducible, given access to the Android data-set.

5. DISCUSSION AND CONCLUSIONS

The results from the regression models given in Table 2, can be used to validate our hypotheses. We see that the independent variable *Attention* has a statistically significant relation with the dependent variable *TimeToResolve*. From the negative sign of the coefficient, it is evident that higher level of attention is associated with lower resolution times and vice-versa. Hence, hypothesis H1 is supported. However, the other independent variable *Diversity* has a statistically significant relation with *TimeToResolve*; but higher levels of diversity is seen to associate with higher resolution times. Hence hypothesis H2 is contradicted.

These results have a number of implications. We find evidence that a bug which attracts more developer attention – in the form of comments – is likely to be resolved faster; but a bug which has more diverse involvement – in the form of different commenters – is likely to take more time to be resolved. More attention brings in a range of perspectives on resolving a bug; it also introduces the potential for a lack of alignment in the bug resolution process. This tension is inherent in the variety of views that intermingle as many individuals work together. Understanding and addressing this tension is essential for harnessing the full power of social processes that underlie software development in the large today. At the **individual level**, these insights can lead to an awareness about the issues in involving many peers for an assigned task. For **project governance**, these can facilitate the setting up and maintenance of communication channels between and across teams. Since many large scale software projects today are distributed, these results can aid resource and work allocation across geographies at the **organizational level**.

In conclusion, this paper highlights a duality around the effects of developer involvement on bug resolution times. Results from studying 1,000+ Android bugs offer insights on how it is important to be cognizant about the benefits and pitfalls of interactive activities in large scale software development.

6. REFERENCES

[1] Albert, R., and Barabasi, A. Statistical mechanics of complex networks. *cond-mat/0106096* (June 2001). Reviews of Modern Physics 74, 47 (2002).

[2] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *J. M. L. R.* (Mar. 2003).

[3] Cataldo, M., Herbsleb, J. D., and Carley, K. M. Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement* (New York, NY, USA, 2008), ESEM '08, ACM, p. 2âĂŞ11.

[4] Datta, S., Sarkar, P., Das, S., Sreshtha, S., Lade, P., and Majumder, S. How many eyeballs does a bug need? an empirical validation of linusâĂŹ law. In *Agile Processes in Software Engineering and Extreme Programming*, G. Cantone and M. Marchesi, Eds., vol. 179 of *LNBIP*. Springer International Publishing, 2014, pp. 242–250.

[5] Glass, R. L. *Facts and Fallacies of Software Engineering*. Addison Wesley Professional Pearson Education [distributor], Boston; Old Tappan, 2002.

[6] Guana, V., Rocha, F., Hindle, A., and Stroulia, E. Do the stars align? multidimensional analysis of android's layered architecture. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on* (2012), pp. 124–127.

[7] Herbsleb, J., and Mockus, A. An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering 29*, 6 (2003), 481–494.

[8] Jeong, G., Kim, S., and Zimmermann, T. Improving bug triage with bug tossing graphs. In *ESEC/FSE '09* (New York, NY, USA, 2009), ACM, pp. 111–120.

[9] Kullback, S., and Leibler, R. A. On information and sufficiency. *Ann. Math. Statist. 22*, 1 (1951), 79–86.

[10] Meneely, A., and Williams, L. Strengthening the empirical analysis of the relationship between linus' law and software security. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (New York, NY, USA, 2010), ESEM '10, ACM, pp. 9:1–9:10.

[11] Raymond, E. S. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly, 2001.

[12] Shihab, E., Kamei, Y., and Bhattacharya, P. Mining challenge 2012: The android platform. In *The 9th Working Conference on Mining Software Repositories* (2012).

[13] Steyvers, M., and Griffiths, T. *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum, 2007, ch. Probabilistic topic models.

[14] Tabachnick, B., and Fidell, L. *Using Multivariate Statistics*. Boston: Pearson Education, 2007.

[15] Wagstrom, P., Herbsleb, J. D., and Carley, K. M. Communication, team performance, and the individual: Bridging technical dependencies. *Academy of Management Proceedings 2010*, 1 (Aug. 2010), 1–7.

[16] Wallach, H. M., Murray, I., Salakhutdinov, R., and Mimno, D. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), ICML '09, pp. 1105–1112.