11-2014

Perspectives on task ownership in mobile operating system development [invited talk]

Subhajit DATTA
Singapore Management University, subhajitd@smu.edu.sg

Perspectives on Task Ownership in Mobile Operating System Development (Invited Talk)

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

ABSTRACT

There can be little contention about Stroustrup's epigrammatic remark: our civilization runs on software. However a caveat is increasingly due, much of the software that runs our civilization, runs on mobile devices today. Mobile operating systems have come to play a preeminent role in the ubiquity and utility of such devices. The development ecosystem of Android - one of the most popular mobile operating systems - presents an interesting context for studying whether and how collaboration dynamics in mobile development differ from conventional software development. In this paper, we examine factors that influence task ownership in Android development. Our results can inform project governance decisions at the individual and organizational levels.

Categories and Subject Descriptors

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

General Terms

Experimentation

Keywords

Android, reviews, code changes, merging

1. INTRODUCTION

Background: Deciding who does what in the assignment of functionality to software components has been recognized as a "desert island skill" [6]. Similarly, in the delegation of tasks to members of a software development team there are many open questions as to what constitutes the most optimal distribution of responsibility. Large scale software systems are being increasingly built out of interactions between many members of distributed teams. In such ecosystems, whether developers should be encouraged to focus closely on tasks they own, or participate widely in shared responsibilities with peers, needs to be better understood.

DeMobile'14, November 17, 2014, Hong Kong, China
ACM 978-1-4503-3225-5/14/11
http://dx.doi.org/10.1145/2661694.2661702

Related work: There have been studies on whether more eyeballs indeed make a bug shallow [4], and the influences on resolution times of defects [5] or modification requests [3]. In addition to bug resolution or implementation of modification requests, another critical activity underpins much of iterative software development – collective review of code changes. In this paper, we study how developer involvement in such review activities relate to the time taken for corresponding code changes to be integrated into the main code base (i.e. "merged"), using data from Android development. The wide currency of Android as a popular operating system for mobile devices makes it imperative that the code base undergoes frequent refinements [1]. An essential element of this refinement process is the production and review of new units of code, and based on their review outcome by developers, arrival at the decision on whether or not to merge the code. In the context of the Android review data, we examine the following hypothesis in this paper:

Hypothesis: *Code changes owned by developers who are involved in more code reviews take higher amount of time to be merged.*

2. METHODOLOGY

To validate this hypotheses, we develop a multiple linear regression model using data on merged Android reviews from a publicly accessible repository[1]. Lack of space prevents a detailed discussion of the data-set; interested readers may refer to the curation and structure of the data in [7]. In the following discussion, a "review" will be taken to mean a unit of code change that is scrutinized by multiple developers ("reviewers"), and based on "approval" scores given by each of them, a decision is taken whether or not to "merge" the unit with the main body of code.

As the **dependent variable** of the model, *TimeToMerge* was calculated as the elapsed time between the time-stamp of review creation and the time-stamp of its last updating. As the **independent variable**, we took *OwnerReviewCount*; for each review (say x), this is the number of reviews the owner of x has reviewed and given an approval score. The *OwnerReviewCount* is taken to reflect the level of involvement of a review's owner in the overall review process. To isolate the relation between the dependent and independent variables, we identified the following **control variables** which capture the peripheral effects on the independent variable: *ExtentOfCodeChange* - calculated as the number of patches associated with a review; *QualityOfChange* - the median of the approval scores received by

[1]https://github.com/mmukadam/gerrit-miner.git

Table 1: Descriptive statistics of model variables

	Mean	Stdev	Skew	Kurt
TimeToMerge	17.61	21.48	2.31	6.25
ExtentOfCodeChange	2.50	2.01	2.98	4.44
QualityOfChange	0.43	0.58	1.13	0.35
DependencyLevel	300.52	192.55	0.65	0.13
NoOfReviewers	4.70	1.79	1.50	3.98
NoOfNnOwnrComnts	2.53	1.19	2.89	6.9
OwnerWorkload	60.63	67.02	1.14	0.13
OwnerReviewCount	190.07	228.18	2.01	5.51

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	$-2.86^{\dagger}$	-1.89
	(1.65)	(1.65)
Control variables		
ExtentOfCodeChange	1.01^{***}	1.12^{***}
	(0.29)	(0.28)
QualityOfChange	-4.02^{***}	-2.96^{***}
	(0.763)	(0.75)
DependencyLevel	0.005^{*}	0.004^{*}
	(0.002)	(0.002)
NoOfReviewers	1.235^{**}	1.086^{***}
	(0.260)	(0.25)
NoOfNnOwnrComnts	4.45^{***}	3.6^{***}
	(0.53)	(0.52)
OwnerWorkload	0.017^{***}	0.054^{***}
	(0.006)	(0.006)
Independent variable		
OwnerReviewCount		-0.024^{***}
		(0.002)
N	2243	2243
R^2	0.186	0.232
df	2236	2235
F	85.2	96.4
p	< 0.001	< 0.001

a review; *DependencyLevel* - calculated as the degree centrality of review in a network based on review similarity; *NoOfReviewers* - the number of different developers participating in the review process; *NoOfNnOwnrComnts* - the number of comments on a particular review by developers who *do not* own it;*OwnerWorkload* - for each review, the number of reviews owned by its owner. *TimeToMerge* and *NoOfNnOwnrComnts* are transformed by taking the square root, to be closer to normal distribution.

Table 1 gives the descriptive statistics of the above model variables. To understand the relation between the dependent and independent variable, after accounting for the effects of the control variables, we first build a base model with only the control variables and the dependent variable, and then augment it into a refined model by adding the independent variable; the corresponding columns of Table 2 gives the parameters of these models. The underlying assumptions of multiple linear regression were found to be within reasonable limits for our analysis.

3. RESULTS AND DISCUSSION

As we observe from Table 2, the refined model represents a better fit to the data compared to the base model and both the models are statistically significant. Adding the independent variable increases the R^2 value by 25%. In both the models, the variables are statistically significant at $\alpha = 0.05$.

The threat to **construct validity** comes from our definition of similarity between reviews using a Latent Dirichlet Allocation [2] based approach, as well as the calculation of some of the model variables based on available data. Since we use curated data from a single source, there is no notable threat to **internal validity**. As we have only studied data from a single source and the R^2 values indicate there are influences on the independent variable which have not been captured, we recognize that threats to **external validity** are present in the study. Our results establish **reliability** as they can be reproduced, given access to the data-set. And finally, as this is an observational study rather than a controlled experiment, correlation can not be taken to indicate causation.

From the model parameters, we observe that higher *OwnerReviewCount* relates to lower *TimeToMerge*; thus our *hypothesis is not supported by the empirical evidence*. This is a counter-intuitive result, especially as we find that higher *OwnerWorkload* does indeed relate to more time being required for merging code changes. In contrast, higher involvement in reviewing appears to make developers more aware of the development ecosystem, leading to more effective ownership of code changes, which is reflected in quicker merging of code.

4. REFERENCES

[1] Asaduzzaman, M., Bullock, M., Roy, C., and Schneider, K. Bug introducing changes: A case study with android. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on* (June 2012), pp. 116–119.

[2] Blei, D., Ng, A., and Jordan, M. Latent dirichlet allocation. *Journal of Machine Learning Research 3* (2003), 993–1022.

[3] Cataldo, M., Wagstrom, P. A., Herbsleb, J. D., and Carley, K. M. Identification of coordination requirements: implications for the design of collaboration and awareness tools. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work* (New York, NY, USA, 2006), CSCW '06, ACM, p. 353âĂŞ362.

[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] Koru, A. G., and Liu, H. Building defect prediction models in practice. *IEEE Softw. 22*, 6 (Nov. 2005), 23âĂŞ29.

[6] Larman, C. *Applying UML and Patterns*. Prentice Hall, 1997.

[7] Mukadam, M., Bird, C., and Rigby, P. C. Gerrit software code review data from android. In *Proceedings of the 10th Working Conference on Mining Software Repositories* (Piscataway, NJ, USA, 2013), MSR '13, IEEE Press, pp. 45–48.