Evaluating Bug Severity Using Crowd-based Knowledge: An Exploratory Study

Yang Zhang, Huaimin Wang, Gang Yin, Tao Wang, Yue Yu
Key Lab. of Parallel and Distributed Computing, College of Computer, National University of Defense Technology, Changsha, 410073, China
{yangzhang15, hmwang, jack.nudt, taowang2005, yuyue}@nudt.edu.cn

ABSTRACT

In bug tracking system, the high volume of incoming bug reports poses a serious challenge to project managers. Triaging these bug reports manually consumes time and resources which leads to delaying the resolution of important bugs. StackOverflow is the most popular crowdsourcing *Q&A* community with plenty of bug-related posts. In this paper, we explore the correlation between bug severity and the crowd attributes of linked posts. Two typical types of projects' bug repositories are studied here, *e.g.* Mozilla (user-centric project) and Eclipse (developer-centric project). Our results show that the bug severity is consistent with the crowd-based knowledge both in Mozilla and Eclipse, *i.e.* the linked posts of severe bugs have higher score *etc.* in StackOverflow than non-severe bugs. This interesting phenomenon inspires us that we can optimize the existing evaluation methods of bug severity by incorporating the crowd-based knowledge from a third-party in future.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*process metrics*

General Terms

Experimentation

Keywords

Bug severity, Bugzilla, crowd-based knowledge, StackOverflow

1. INTRODUCTION

Managing the incoming bug reports in a large software project within the limited time and resources is a challenging task [1]. In order to handle the highly important bugs first and allocate time and resources appropriately, the triager (the person who analyzes the bug reports) needs to prioritize the incoming bug reports by their order of importance. Bug importance indicates the order in which bugs should be fixed (priority) or the impact that the bugs have on the system (severity) [2].

Thus, researchers are interested in building classifier models to automatically classify bugs as important or non-important by using machine learning techniques [3][4]. Existing research mainly focuses on the textual description (summary, description and comments) of bug reports [3] and bug location (product, component *etc.*) [4]. These features are extracted from the bug tracking system themselves. As yet, bug importance, like severity in general, is largely dependent on triagers' point of view.

StackOverflow[1] (SO), a question and answer (Q&A) community, is widely used by developers to search and publish answers to technical issues and concerns in software development. Some researchers integrate bug tracking systems with SO to help developers find relevant information to solve programming problems [5]. In SO, in order to engage more participants, available quantitative information such as the score, view count, answer count and other characteristics of posts are recorded. These post evaluation characteristics capture much practical insights about the crowd-based knowledge, *i.e.* the opinions from a large number of different users. There are many bug reports that contain URL links to the posts of SO within user comments. Additionally, many answers or comments submitted on SO contain URL links to relevant bug reports. The study presented in this paper is motivated by our belief that we can enhance existing bug tracking system by exploiting these interesting characteristics of SO posts. We believe that evaluating bug severity by using crowd-based knowledge is a relatively unexplored area and our motivation is to throw light on this topic.

The specific research aims and contributions of the study include:

1) We investigate the correlation between the bug severity in bug tracking system and the linked posts' crowd-based knowledge in SO.

2) We focus on two types of projects' bug repositories, Mozilla (*user-centric*) and Eclipse (*developer-centric*) for our case study. Using statistical analysis, we find that there exists correlation between bug severity and the linked posts' crowd-based knowledge.

[1]http://stackoverflow.com

INTERNETWARE '15, November 6, 2015, WuHan, China

3) Using regression modeling on our dataset, we verify that we have the potential to evaluate bug severity by exploiting crowd-based knowledge. This is an interesting and promising research direction, which would guide future software engineering tool innovations as well as practices.

The remainder of this paper is structured as follows. Section 2 presents our empirical study methodology. In Section 3, we introduce the study results on Mozilla and Eclipse. We present conclusions in Section 4.

2. METHODOLOGY

In this section, we first give our research questions, and then we present our datasets, the process of mining bug-post link pairs and our statistical analysis methods.

2.1 Research Questions

According to the different product users, projects can be divided into two types, *user-centric* projects and *developer-centric* projects. The products of *user-centric* projects are mainly used by users with less development experience. While users of *developer-centric* projects have more knowledge about software development. Based on the different type of projects, we present our research questions:

RQ1. *What is the correlation between bug severity and the linked posts' crowd-based knowledge? Is there some difference between user-centric projects and developer-centric projects?*

RQ2. *Is there the potential to enhance existing evaluation methods of bug severity in a bug tracking system by incorporating a third dimension of crowd-based knowledge?*

2.2 Datasets

For the bug datasets of different types of projects, we use bug reports from two large open-source projects using Bugzilla[2] as their bug tracking system: Mozilla[3] and Eclipse[4].

Mozilla: Mozilla is a suite of programs for web browsing and collaboration. Mozilla is a *user-centric project*. Its products are mainly used by less savvy users [3]. The bug database we obtained contains all reports submitted from April 1998 to March 2015.

Eclipse: Eclipse is an integrated development platform to support various aspects of software development. Eclipse is a *developer-centric project*. It is mainly supported and used by developers [6]. The bug database contains all reports submitted from October 2001 to March 2015.

For the dataset of posts, we collect the most recent data dump (up to March 2015) from SO. The dumped data contains data of seven categories, *badges*, *posts*, *user*, *votes*, *post history*, *post links* and *comments*. All information extracted was saved in a database for the manipulation in the subsequent phases.

2.3 Mining bug-post Link Pairs

For automated extracting the bug-post link pairs, we first mined the URLs which link to the SO posts from the bug reports' summary, description and comments. Then we mined the URLs which link to the bug reports from the posts' title, body, answer and comments. We refer to these links as *direct-link*. There are many posts that link to other posts in SO, which we consider as related posts [7]. For example, as shown in Fig.1, if Bug X has a *direct-link* to Post A, Post A has a URL which links to Post B or Post B has a URL which links to Post A, we consider that Bug X has an *indirect-link* to Post B. If Post C is also related to Post B but Post C is not related to Post A, we consider Post C has a weak relationship with Bug X because it is difficult to determine whether it is a real related post after several URL links. So we do not consider these links in our study.

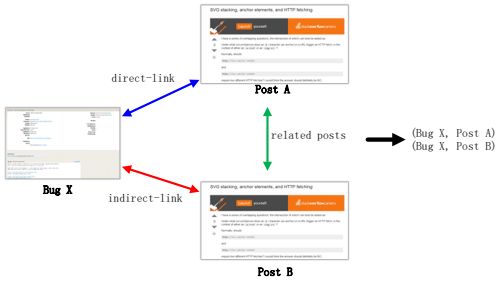


Figure 1: Example of mining bug-post link pairs

In a bug report's life-cycle, there are six states: *new*, *assigned*, *resolved*, *verified*, *closed* and *reopen* [2]. We only focus on the bug reports marked as *resolved*, *verified*, or *closed* because the severity of such reports is set by the triager or developer after resolution of the bug. The values of the severity field varies from *trivial*, *minor*, *normal*, *major*, *critical*, to *blocker*. We can also use this field to indicate whether a bug is an enhancement request by setting the severity as "enhancement". The *normal* severity is the default option for selecting the severity when reporting a bug and it represents the grey zone. Similar to the approach proposed by Ahmed Lamkanfi *et al.* [3], in our study, the *normal* and *enhancement* severities are deliberately not taken into account and we treat the severities *blocker*, *critical* and *major* as *severe*, while we treat severities *minor* and *trivial* as *non-severe*. Finally, we collect 957 *bug-post* pairs (176 bugs and 810 posts) for Mozilla and 3501 *bug-post* pairs (528 bugs and 2732 posts) for Eclipse. Table 1 presents basic statistics about our dataset. The number of *severe* and *non-severe* bugs in Mozilla and Eclipse are shown in Table 2. In Mozilla, the average value of each bug's related posts is 6.2 and the value is 7.6 in Eclipse.

Table 1: Basic statistics for our bug-post dataset

	Mozilla			Eclipse		
	bug	post	bug-post	bug	post	bug-post
direct-link	176	227	238	528	669	728
indirect-link	89	626	776	276	2208	2937
total (duplicates removed)	176	810	957	528	2732	3501

Table 2: Basic statistics for severe and non-severe bugs

	severe				non-severe		
	blocker	critical	major	total	minor	trivial	total
Mozilla	8	40	94	142	27	7	34
Eclipse	56	109	308	473	48	7	55

[2]http://www.bugzilla.org
[3]http://bugzilla.mozilla.org
[4]http://bugs.eclipse.org/bugs

2.4 Statistical Analysis

After mining the bug-post link pairs, we seek to answer our research questions by using statistical analysis. We divide our analysis metrics into two parts, *bug-level* and *post-level*. In the *bug-level*, the basic analysis metric is the severity. In addition, we propose two other feature metrics.

ccList: Number of the list of people who get mail when the bug changes.

Bug Description Complexity (BDC): Total number of words in bug report's summary and description.

In the *post-level*, we propose six feature metrics that capture much practical insights about the crowd-based knowledge of the posts.

Score: Total score of the post.

View Count (VC): Number of times the post is viewed. Because the order of magnitude of *VC* is larger than other metrics, in our statistical analysis, the *VC* is log transformed to stabilize variance and reduce heteroscedasticity.

Answer Count (AC): Number of answers a post gets.

Comment Count (CC): Number of comments posted on a post.

Favorite Count (FC): Number of favorites a post gets.

Up Votes (UV): Total number of up votes a post gets.

In our study, we use the entirety of these *post-level* metrics to represent the crowd-based knowledge of linked posts. We first investigate the distribution of *post-level* metrics in *severe* and *non-severe* bugs. Then we use the Mann-Whitney-Wilcoxon (*MWW*) test to compare these *post-level* metrics in the two sets, *severe* and *non-severe*. The *MWW* test is a non-parametric statistical test used for assessing whether two independent distributions have equally large values. In order to explore the potential of evaluating bug severity using crowd-based knowledge, we use the multiple linear regression to analyze the correlation between bug severity and the crowd-based knowledge. The outcome measure is the severity. We build two models, *Model 1* only considering the *bug-level* metrics, and *Model 2* adding the *post-level* metrics.

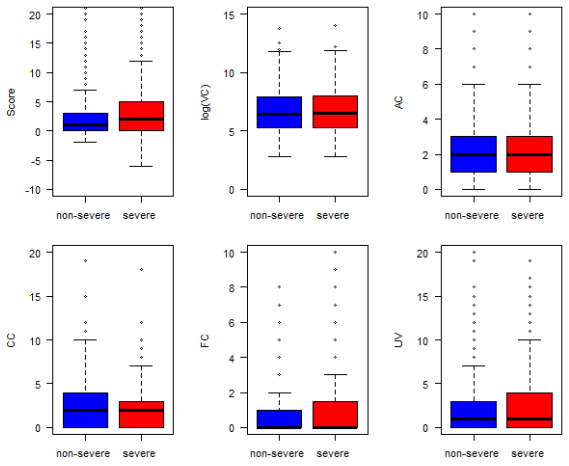
3. STUDY RESULTS ON MOZILLA AND ECLIPSE

In this section, we present the results of our empirical study on Mozilla and Eclipse.

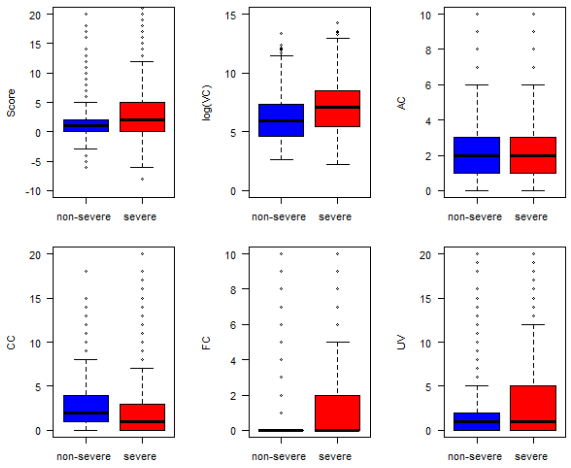
3.1 RQ1. The correlation between bug severity and the crowd-based knowledge

Table 3 and Table 4 separately display the min, 25%, median, mean, 75% and max values of *post-level* metrics in Mozilla and Eclipse. In Mozilla, the average *Score* of linked posts for *non-severe* bug is 10.2 (median: 1.0), while the value increases to 30.5 (median: 2.0) for *severe* bug. In Eclipse, the average *Score* of linked posts for *non-severe* bug is 9.8 (median: 1.0), while the value increases to 13.3 (median: 2.0) for *severe* bug. Except *CC*, the similar phenomenon is observed with respect to the *VC*, *AC*, *FC* and *UV*.

In order to give a clear description of the distribution of *post-level* metrics in *severe* and *non-severe* bugs, we present the boxplots as shown in Fig.2. In Mozilla, Fig.2-a shows that the linked posts of *severe* bugs are more likely to have higher *Score*, more *VC*, more *AC*, more *FC* and more *UV* than



(a) Mozilla



(b) Eclipse

Figure 2: The distribution of the *post-level* metrics in *severe* and *non-severe* bugs

Table 3: The basic statistical results on Mozilla

		Min	25%	Median	Mean	75%	Max
Score	severe	-6.0	0.0	2.0	30.5	5.0	4928.0
	non-severe	-2.0	0.0	1.0	10.2	3.0	1326.0
VC	severe	17.0	200.3	662.5	13331.5	2996.0	1254732.0
	non-severe	16.0	187.0	598.0	7852.1	2672.0	974177.0
AC	severe	0.0	1.0	2.0	2.6	3.0	36.0
	non-severe	0.0	1.0	2.0	2.6	3.0	25.0
CC	severe	0.0	0.0	2.0	2.3	3.0	24.0
	non-severe	0.0	0.0	2.0	2.8	4.0	27.0
FC	severe	0.0	0.0	0.0	10.7	1.3	3960.0
	non-severe	0.0	0.0	0.0	7.7	1.0	869.0
UV	severe	0.0	0.0	1.0	28.0	4.0	4544.0
	non-severe	0.0	0.0	1.0	9.6	3.0	1365.0

Table 4: The basic statistical results on Eclipse

		Min	25%	Median	Mean	75%	Max
Score	severe	-8.0	0.0	2.0	13.3	5.0	4375.0
	non-severe	-6.0	0.0	1.0	9.8	2.0	1990.0
VC	severe	9.0	234.0	1160.0	11972.1	4793.0	1596898.0
	non-severe	14.0	100.8	366.0	5994.4	1548.8	645282.0
AC	severe	0.0	1.0	2.0	2.9	3.0	39.0
	non-severe	0.0	1.0	2.0	2.8	3.0	35.0
CC	severe	0.0	0.0	1.0	2.3	3.0	22.0
	non-severe	0.0	1.0	2.0	3.1	4.0	24.0
FC	severe	0.0	0.0	0.0	7.8	2.0	1778.0
	non-severe	0.0	0.0	0.0	2.9	0.0	698.0
UV	severe	0.0	0.0	1.0	12.0	5.0	3594.0
	non-severe	0.0	0.0	1.0	8.8	2.0	1680.0

Table 5: The MWW test results on Mozilla and Eclipse

	Score	VC	AC	CC	FC	UV
MWW in Mozilla	5.8e-04	6.7e-03	1.9e-03	9.2e-04	2.9e-03	2.7e-03
MWW in Eclipse	<2.2e-16	<2.2e-16	5.3e-03	2.1e-11	<2.2e-16	1.1e-15

non-severe bugs. Similarly, in Eclipse, as shown in Fig.2-b, the linked posts of *severe* bugs are more likely to have more crowd-based knowledge than *non-severe* bugs which is cosistent to the results on Mozilla. The *MWW* test results in Table 5 show that in Mozilla and Eclipse, all the differences between *severe* bugs and *non-severe* bugs are statistically significant (p<0.05).

RQ1. There is some correlation between bug severity and the linked posts' crowd-based knowledge. No matter in developer-centric project Eclipse or user-centric project Mozilla, bug severity has a positive relationship with the crowd-based knowledge, i.e. the linked posts of severe bugs have higher score etc. than non-severe bugs.

3.2 RQ2. The potential of using crowd-based knowledge to evaluate bug severity

In order to explore the potential of using crowd-based knowledge to evaluate bug severity, we use the multiple linear regression to compare the two models as described in Section 2.4. All variance inflation remained well below 3, indicating absence of multicollinearity. For each model variable, we report its coefficients, standard error, and significance level. We consider coefficients important if they were statistically significant (p<0.05). We obtain effect sizes from *ANOVA* analyses. The resulting multivariate linear regression models are shown in Table 6 and Table 7.

As expected, both in Mozilla and Eclipse, *ccList* and *BDC* play a dominant role in explaining the variance in the data. Both in Mozilla and Eclipse, *Score*, *VC*, *AC*, *FC* and *UV* all have a positive effect, while *CC* has a negative effect, which is consistent with the previous results of statistical analysis. Compared to *Model 1*, *Model 2* offers a significantly better fit (Mozilla: *adjusted* R^2 raises 7.2%, Eclipse: *adjusted* R^2 raises 9.1%). It indicates that incorporating a third dimension of crowd-based knowledge into the evaluation of bug severity is helpful to some extent.

RQ2. There is the potential to enhance existing evaluation methods of bug severity by exploiting the crowd-based knowledge.

Table 6: The multiple linear regression results on Mozilla

	Model 1		Model 2	
	Coeffs(Errors)	Sum Sq.	Coeffs(Errors)	Sum Sq.
(Intercept)	2.866e+00(7.329e-02)***		3.112e+00(1.919e-01)***	
ccList	1.864e-02(1.865e-03)***	273.68***	1.884e-02(1.871e-03)***	273.68***
BDC	7.678e-04(6.489e-05)***	284.53***	7.556e-04(6.523e-05)***	284.53***
Score			7.637e-04 (1.534e-02)*	1.78.
log(VC)			2.744e-02 (3.013e-02)**	1.06
AC			3.002e-03(2.495e-02)**	1.39*
CC			-2.487e-02(1.497e-02).	5.48.
FC			7.787e-04(4.872e-03)*	6.05*
UV			1.215e-03 (1.676e-02)*	0.01
Adjusted R^2	0.1519		0.2234	
P-value	<2.2e-16		<2.2e-16	
'***' p<0.001, '**' p<0.01, '*' p<0.05, '.' p<0.1				

Table 7: The multiple linear regression results on Eclipse

	Model 1		Model 2	
	Coeffs(Errors)	Sum Sq.	Coeffs(Errors)	Sum Sq.
(Intercept)	4.698e+00(2.330e-02)***		4.279e+00(6.868e-02)***	
ccList	1.862e-02(9.038e-04)***	409.4***	1.673e-02(9.197e-04)***	409.4***
BDC	5.319e-05(3.262e-06)***	303.0***	4.818e-05(3.274e-06)***	303.0***
Score			4.201e-03(3.826e-03)*	0.5*
log(VC)			8.606e-02(1.084e-02)***	60.1***
AC			2.461e-02(7.515e-03)**	20.0***
CC			-2.602e-02(6.320e-03)***	18.7***
FC			1.902e-03(7.626e-04)*	5.4*
UV			6.005e-03(4.814e-03)*	1.7*
Adjusted R^2	0.0812		0.1724	
P-value	<2.2e-16		<2.2e-16	
'***' p<0.001, '**' p<0.01, '*' p<0.05, '.' p<0.1				

4. CONCLUSIONS

The study presented in this paper shows evidences of the bug severity in bug tracking system has some relationship with the crowd-based knowledge in SO. Our statistics results indicate that no matter in *developer-centric* project Eclipse or *user-centric* project Mozilla, bug severity has a positive relationship with the linked posts' crowd-based knowledge, *i.e.* the linked posts of *severe* bugs have more *Score etc.* than *non-severe* bugs. Our study shows that we certainly have the potential to enhance existing evaluation methods of bug severity by exploiting the crowd-based knowledge of SO posts to incorporate a third dimension of knowledge.

5. ACKNOWLEDGMENTS

The research is supported by the National Natural Science Foundation of China (Grant No.61432020, 61472430 and 61502512) and the Postgraduate Innovation Fund (Grant No.CX2015B028).

6. REFERENCES

[1] Herraiz I, German D M, Gonzalez-Barahona J M, et al. Towards a simplification of the bug report form in eclipse. In *Proceedings of the International Working Conference on Mining Software Repositories*, pages 145-148. ACM, 2008.

[2] D'Ambros M, Lanza M, Pinzger M. "A Bug's Life" Visualizing a Bug Database. In *Proceedings of the International Workshop on Visualizing Software for Understanding and Analysis*, pages 113-120. IEEE, 2007.

[3] Lamkanfi A, Demeyer S, Giger E, et al. Predicting the severity of a reported bug. In *Proceedings of the International Working Conference on Mining Software Repositories*, pages 1-10. IEEE, 2010.

[4] Tian Y, Lo D, Xia X, et al. Automated prediction of bug report priority using multi-factor analysis. *Empirical Software Engineering*, pages 1-30, 2014.

[5] Bacchelli A, Ponzanelli L, Lanza M. Harnessing stack overflow for the ide. In *Proceedings of the International Workshop on Recommendation Systems for Software Engineering*, pages 26-30. IEEE, 2012.

[6] Banerjee S, Helmick J, Syed Z, et al. Eclipse vs. Mozilla: A Comparison of Two Large-Scale Open Source Problem Report Repositories. In *Proceedings of the High Assurance Systems Engineering*, pages 263-270. IEEE, 2015.

[7] Wang T, Yin G, Wang H, et al. Linking stack overflow to issue tracker for issue resolution. In *Proceedings of the Asia-Pacific Symposium on Internetware on Internetware*. pages 11-14. ACM, 2014.