LETTER

A Quantitative Analysis on Relationship between an Early-Closed Bug and Its Amount of Clues: A Case Study of Apache Ant

Akito SUNOUCHI[†], *Nonmember*, **Hirohisa AMAN**[††a)], *and* **Minoru KAWAHARA**[††], *Members*

SUMMARY Once a bug is reported, it is a major concern whether or not the bug is resolved (closed) soon. This paper examines seven metrics quantifying the amount of clues to the early close of reported bugs through a case study. The results show that one of the metrics, the similarity to already-closed bug reports, is strongly related to early-closed bugs.

key words: *bug report, early close, amount of clues, metrics, POS tagger*

1. Introduction

Software failures may occur during an operation of a software product, and these failures are reported to the development project as bug reports. A bug report usually provides failure information including the symptom and the status, etc. The development project confirms the details of reported failure and starts to detect and fix the cause (fault). Although it is ideal to solve all bugs[*] soon, some bugs take a long time, e.g., one year or more, until they are resolved (closed). If a bug fix requires a long time, we have to take an alternative approach in order to continue the operation while avoiding the failure. Thus, it is a significant challenge to predict whether a reported bug is early closed or not.

In recent years, there have been studies on data of bug reports which are useful for predicting early-closed bugs [1]–[3]. While the posted bug report itself is an essential information source to resolve the corresponding bug, we believe it may also be worthwhile to focus on the relationships of the bug report with other available artifacts—other bug reports and source files. In this paper, for each reported bug, we focus on not only the posted bug report but also other already-closed bug reports and source files which were available at the time when the bug was reported. Then, we quantify the amount of clues from these available data, and analyze which kind of data are related to early-closed bugs.

2. Available Information When a Bug is Reported

2.1 Motivation

As mentioned above, when a bug is reported, the development project starts to analyze the bug and resolve it. If a bug report provides richer information about the bug, it is easier to detect the corresponding fault for developers, and the bug would be closed earlier. Karim et al. [3] empirically showed that one of notable points is whether or not an error message is given in the bug report. While a bug report is a useful information source about the bug, we have yet another question if other available information can also play an important role for an early close of the bug. That is to say, since the following items are also available when a bug is reported, we will focus on them as well: (1) already-closed bug reports and (2) source files. The main reasons why we focus on these two kinds of items are as follows. (1) If there is an already-closed bug which is related to or similar to the newly reported bug, the fix information of the already-closed bug may help an early close of the new bug. (2) Since [3] showed that an error message given in a bug report is a useful clue and such an error message must be printed by a source code, associations of the reported bug with source files would also be noteworthy to detect the corresponding fault.

2.2 Quantifying the Amount of Clues

In order to quantify the amount of clues, we extract tokens from bug reports and source files, then perform the morphological analysis, i.e., we transform them into their basic forms and decide their parts of speech (POS). For each bug report and source file, we single out nouns (or potential nouns) as the feature terms according to the previous work by Abebe and Tonella [4], and form the term set corresponding to the bug report/the source file; "potential nouns" are the terms which we could not automatically decide their POS by the POS tagger we used. Since some variable names or path names are not successfully processed by natural language processing tools, we decided to include these terms into our feature term set as well. In this paper, we will use the Tree Tagger [5] as the tool (POS tagger) according to the results of the comparative study conducted by Tian and Lo [6].

We now explain how to extract the feature terms from (1) a bug report and (2) a source file in detail. (1) For a bug report, we first make a text file which contains the "description" part and the "comment" parts of the corresponding bug report: the former is the original bug report by the corresponding reporter and the latter ones are comments or replies to the report. In the Bugzilla XML format,

Manuscript received April 27, 2018.
Manuscript revised June 11, 2018.
Manuscript publicized June 22, 2018.

[†]The author is with Department of Computer Science, Ehime University, Matsuyama-shi, 790–8577 Japan.
[††]The authors are with Center for Information Technology, Ehime University, Matsuyama-shi, 790–8577 Japan.
a) E-mail: aman@ehime-u.ac.jp

DOI: 10.1587/transinf.2018EDL8094

[*]We will use term "bug" to indicate both software failure and fault in this paper. A bug fix means that the corresponding fault is fixed and the corresponding failure does not occur.

they can be easily extracted by focusing on `<thetext>` ··· `</thetext>` tags. Then we parse the text file by using the Tree Tagger, and single out the feature terms (nouns) from the output. Since the Tree Tagger automatically extracts tokens from the document, and performs the stemming and the decision of POS, we do not perform any special preprocessing for the text file, but just use the text file as an input document. (2) For a source file, we have no special preprocessing; we just regard a source file as a document. That is to say, we directly parse a source file by using the Tree Tagger, and extract feature terms from the output. Although we considered removing stop words from bug reports and source files, we decided to use a naive way without any preprocessing—just using a POS tagger for both a bug report and a source file—in this study; Since we extract only (potential) nouns, many English stop words (e.g., "an," "at," "is") are excluded. Moreover, because we have no reasonable set of stop words which are common to both types of documents (bug reports and source files), we considered that making a common stop word list can be yet another threat to validity. We would like to perform a further experiment involving such preprocessing techniques as our future work.

We evaluate the amount of clues from two points of view: (i) the posted bug report itself, and (ii) similarities of the bug report to other already-closed bug reports or to source files. For the point of view (i), we focus on features of the term set corresponding to the bug report. If the bug report has a richer vocabulary, it may provide useful clues. For (ii), we evaluate a similarity between bug reports or a similarity between a bug report and a source file by comparing the corresponding term sets. When a new bug is reported, if there is an already-closed bug report which is similar to the new bug report, the precedent bug report may be a useful clue toward an early resolution of the new bug. Moreover, if there is a source file having a high similarity to the bug report, such a source file might be a helpful clue as well.

To quantify the amount of clues, we define seven metrics shown in Table 1. Metrics TC and UTC measure the appearance count of terms and the unique count of them in the bug report, respectively. Metric VOL is an integrated score which is made from TC and UTC; the integration is inspired by the program volume metric in Halstead's theory [7]. Each of the remaining four metrics quantifies a similarity between bug reports or between a bug report and a source file. We consider that two artifacts are similar as they have more common terms, and we quantify the similarity using two measures: the Jaccard index and the number of common terms. Then, we evaluate the amount of clues by using the metric value of the most similar pair.

Table 1 Metrics for quantifying amount of clues.

metric	description
TC(b)	# of terms appearing in b
UTC(b)	# of unique terms appearing in b
VOL(b)	VOL(b) = TC(b) $\cdot \log_2$ {UTC(b)}
MJB(b)	$\max_i \{Jac(b, b_i)\}$
MCB(b)	$\max_i \{Com(b, b_i)\}$
MJS(b)	$\max_j \left\{Jac(b, s_j)\right\}$
MCS(b)	$\max_j \left\{Com(b, s_j)\right\}$

$$\left\{\begin{array}{l} b : \text{the bug report of interest.} \\ b_i : \text{an already-closed bug when } b \text{ is reported.} \\ s_j : \text{a source file existing when } b \text{ is reported.} \\ Jac(\cdot,\cdot) : \text{Jaccard index between two documents.} \\ Com(\cdot,\cdot) : \text{\# of unique terms appearing in both documents.} \end{array}\right.$$

3. Case Study

In order to examine the above seven metrics, we conducted a case study where we analyzed 1,280 bug reports in the Apache Ant project: all of them were closed by January 1st, 2018. We report the results in this section.

Our aim of this study is to show whether or not the amount of clues mentioned above is related to an early close of bug. We examined different seven ways of quantifying the amount of clues by the metrics shown in Table 1. In this study, let a bug be an early-closed bug if its status in Bugzilla was changed to "RESOLVED," "VERIFIED" or "CLOSED" within 30 days from the bug was reported. Notice that all of these bug reports are new bugs and not duplicated ones. In our dataset, 369 out of 1,280 bug reports (28.8%) were early-closed ones. Table 2 presents the distribution of days that bug reports required to be closed: The median (see "50%" column of Table 2) is longer than 1 year, so there are many bug reports which are not closed for a long period in the real. This distribution also shows the importance of predicting whether a posted bug report will be early closed or not.

For each of seven metrics, we categorize the set of bug reports into four subsets B_1, B_2, B_3 and B_4, according to their metric values as follows. Let $v(b)$ be a metric value of bug report b. If $v(b) \leq Q_1$ then $b \in B_1$; if $Q_1 < v(b) \leq Q_2$ then $b \in B_2$; if $Q_2 < v(b) \leq Q_3$ then $b \in B_3$; if $v(b) > Q_3$ then $b \in B_4$, where Q_1, Q_2 and Q_3 are the 25 percentile, the median and the 75 percentile, respectively. B_i is a set of bug reports having richer information than B_j if $i > j$.

We computed the rate of early-closed bug reports (the early-close rate: ECR) in B_i (for $i = 1, 2, 3, 4$) and compared them. Table 3 and Fig. 1 present the results.

As a result, for only metric MJB, the ECR values monotonically increased from B_1 toward B_4 (emphasized in Fig. 1), and the difference of ECR values between B_1 and B_4 was statistically significant at $\alpha = 0.01$ ($\chi^2 = 76.586$, $df = 1$, $p < 2.2 \times 10^{-16}$). Moreover, these ECR values in B_1 and B_4 are the minimum value and the maximum one among all results shown in Table 3, respectively. Thus, metric MJB seems to be strongly related to whether a bug report is early closed or not. Metric MJB quantifies the amount of clues as the highest similarity of the new bug report to other already-closed bug reports, in terms of the Jaccard index. The presence of such a similar bug report would become a

Table 2 Distribution of days to bug close.

min	25%	50%	75%	max
0.001	15.621	392.214	913.855	4072.212

Table 3 Comparison of the ECRs among bug report categories.

	ECR			
metric	B_1	B_2	B_3	B_4
TC	0.311	0.329	0.282	0.243
UTC	0.326	0.280	0.291	0.255
VOL	0.316	0.300	0.290	0.246
MJB	0.158	0.234	0.249	0.463
MCB	0.276	0.291	0.268	0.311
MJS	0.332	0.285	0.248	0.288
MCS	0.306	0.289	0.292	0.269

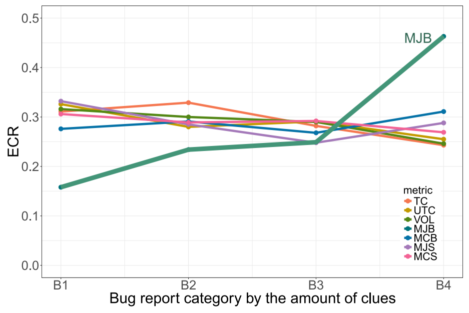

Fig. 1 Comparison of the ECRs among bug report categories.

help for resolving the newly reported bug.

Three metrics regarding the amount of descriptions in a new bug report itself—TC, UTC and VOL—did not show increase trends of ECR values[†]. That is to say, even if a bug report contains more terms (words), its ECR does not get higher; A more detailed data collection like [3] would be needed for predicting if it is early closed or not.

Two metrics focusing on relationships with source files—MJS and MCS—also did not seem to be related to ECR values. Since we analyzed source files in the same manner as we did for bug reports, the set of terms extracted from source files might be a mixture of wheat and chaff in this study. A more sophisticated analysis would be required to make a useful link between a source file and a bug report. For example, there are identifiers consist of two or more words, e.g., "`dataFile`" which can be split into "`data`" and "`file`." Moreover, there are also abbreviations of words in identifiers such as "`idx`" which is an abbreviation of "`index`." They might become useful clues if those compound names are properly split and abbreviations are appropriately expanded before the POS tagger analysis. We plan to perform such preprocessing for identifiers by leveraging the identifier-split/expansion studies [8], [9] in the future. Furthermore, a semantic analysis such as the topic modeling [10], [11] would also be useful to mine yet another link between a bug report and a source file. This is also our important future work.

4. Conclusion

For a newly reported bug, we proposed to measure the amount of clues to the resolution (close) by seven metrics. Then, in order to examine these metrics, we conducted a case study using a set of bug reports in the Apache Ant project. As a result, one of these metrics, the maximum similarity (Jaccard index) between bug reports, showed a positive relationship with the early close rate of bug report. That is to say, when a new bug is reported, if there is a more similar bug report which is already closed, the new bug is expected to be closed earlier. Since the proposed metric can be automatically and easily measured, the result of this empirical study would be a useful help toward a better prediction if a reported bug is closed or not.

Our future work includes: (1) to conduct an empirical study involving a larger set of bug reports in order to clarify the generality of our results; (2) to perform a more detailed analysis on the content of bug reports and source files, together with a proper stop word elimination and an appropriate identifier splitting/expanding, for enhancing the accuracy of prediction; (3) to do further studies focusing on the number of similar and already-closed bug reports, and/or the properties of already-closed bugs such as their priorities and their difficulties to resolve, etc.

[†]On the other hand, the ECR values showed a monotonic decrease trend for metric VOL. However, its difference between B_1 and B_4 was not significant ($\chi^2 = 3.4983$, $df = 1$, $p = 0.06143$).

Acknowledgments

This work was supported by JSPS KAKENHI #16K00099. The authors would like to thank the anonymous reviewer for the helpful comments to an earlier version of this paper.

References

[1] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?," IEEE Trans. Softw. Eng., vol.36, no.5, pp.618–643, Sept. 2010.
[2] S. Davies and M. Roper, "What's in a bug report?," Proc. 8th ACM/IEEE Int'l Symp. Empir. Softw. Eng. and Measurement, pp.26:1–26:10, Sept. 2014.
[3] M.R. Karim, A. Ihara, X. Yang, H. Iida, and K. Matsumoto, "Understanding key features of high-impact bug reports," Proc. 8th Int'l Workshop Empir. Softw. Eng. in Practice, pp.53–58, March 2017.
[4] S.L. Abebe and P. Tonella, "Natural language parsing of program element names for concept extraction," Proc. IEEE 18th Int'l Conf. Program Comprehension, pp.156–159, June 2010.
[5] H. Schmid, "Tree Tagger." http://www.cis.uni-muenchen.de/˜schmid/tools/TreeTagger/.
[6] Y. Tian and D. Lo, "A comparative study on the effectiveness of part-of-speech tagging techniques on bug reports," Proc. IEEE 22nd Int'l Conf. Softw. Analysis, Evolution, and Reeng., pp.570–574, March 2015.
[7] M.H. Halstead, Elements of Software Science, Elsevier, New York, 1977.
[8] E. Hill, D. Binkley, D. Lawrie, L. Pollock, and K. Vijay-Shanker, "An empirical study of identifier splitting techniques," Empir. Softw. Eng., vol.19, no.6, pp.1754–1780, Dec. 2014.
[9] D. Lawrie, H. Feild, and D. Binkley, "Extracting meaning from abbreviated identifiers," Proc. 7th IEEE Int'l Working Conf. Source Code Analysis and Manipulation, pp.213–222, Sept. 2007.
[10] T. Hofmann, "Probabilistic latent semantic indexing," Proc. 22nd Annual Int'l ACM SIGIR Conf. Research and Development in Inf. Retrieval, pp.50–57, Aug. 1999.
[11] D.M. Blei, "Probabilistic topic models," Commun. ACM, vol.55, no.4, pp.77–84, April 2012.