USING BUG REPORTS AS A SOFTWARE QUALITY MEASURE

Liguo Yu
Indiana University South Bend, USA
ligyu@iusb.edu

Srini Ramaswamy, Anil Nair
Industrial Software Systems
ABB Corporate Research
Bangalore India 560048
srini@ieee.org, anil.nair@in.abb.com

Abstract: Bugzilla is an online software bug reporting system. It is widely used by both open-source software projects and commercial software companies and has become a major source to study software evolution, software project management, and software quality control. In some research studies, the number of bug reports has been used as an indicator of software quality. This paper examines this representation. We investigate whether the number of bug reports of a specific version of a software product is correlated with its quality. Our study is performed on six branches of three open-source software systems. Our results do not support using the number of bug reports as a quality indicator of a specific version of an evolving software product. Instead, the study reveals that the number of bug reports is in some ways correlated with the time duration between product releases. Finally, the paper suggests using accumulated bug reports as a means to represent the quality of a software branch.

Key Words: Software Quality, Bug Reports.

INTRODUCTION

Bugzilla [1] is an online software bug reporting and bug tracking system. It is being used by a large number of companies, organizations, and projects [2], including the Linux kernel project, Eclipse, Facebook, and NASA ITOS. Because Bugzilla defines and implements a full bug lifecycle model, it can provide detailed information about the bug reported to a software system. Therefore, it has been widely used by Software Engineering researchers to study software evolution, software project management, and software quality control [3] [4] [5] [6] [7] [8] [9] [10] [11] [12].

In some research articles, the number of bugs or the number of bug reports has been used as the indicator of software quality. For example, in [13], Zhang and Kim used the number of bugs reported in each month to represent the change of product quality. In other studies, number of bugs reported in each month have been found correlated with the number of uploads and packages [14]. In our previous work [15], we analyzed the feasibility of using the distribution of bug reports against time to represent software quality (specifically, maintainability) and concluded that number of bug reports might not be a good candidate of maintainability measure. In this paper, we closely study the possibility of using the number of bug reports to represent software quality. Using statistical methods, we analyze the correlation between number of bug reports and software changes in six branches of three open-source software systems.

The remainder of this paper is organized as follows. Section 2 presents the background knowledge of this study. Section 3 describes the data source and data mining process. Section 4 presents the results and the analysis. Conclusions are in Section 5.

BACKGROUND

Continually evolving software systems, especially open-source software systems, are frequently changed

and released. On the one side, a change could fix a bug or add a new feature to the system. On the other side, a change might introduce new bugs to the system. Consider a software evolution branch with six releases. As shown in Figure 1, bugs are reported to each of the six releases. If we assume the number of bugs (bug reports) can be used to represent the product quality of each release, it means (1) all the bugs reported in current version (vi) are introduced during the modification to its previous version (vi-1); (2) all the bugs introduced in modifying current version (vi) will be detected, reported, and fixed in next release (vi+1). In other words, we need to assume that most bugs can only live for one version and they cannot be carried for two or more versions.

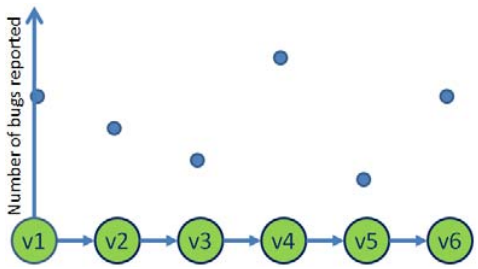


Figure 1. The bugs reported to six releases of a software system.

It has been proved in other studies that the number of bugs introduced in each release is correlated with the amount of modification made to its previous version [16] [17] [18]. Therefore, if the number of bugs (bug reports) can be used to represent the product quality, we will be able to see the correlation between the number of changes to version vi and the number of bugs reported for version vi+1. We accordingly formulate the following null hypothesis.

H_{01}: In a software branch with *n* releases, there is no linear correlation between the number of changes to version v_i and the number of bugs reported in version v_{i+1}($i<n$).

In this study, the amount of changes made to a version of a software product is measured with the size changes of the source code. The size of a software product can be measured as (1) the number of lines of code, which includes comment lines and blank line; (2) physical lines of code, which does not include blank lines; (3) logical lines of code, which only contains statements (no blank lines, no comment lines); or (4) McCabe Cyclomatic complexity. Since blank and comment lines are generally not considered factors of software complexity, we will use physical lines of code, logical lines of code, and McCabe Cyclomatic complexity to measure the amount of changes to a version of a software product. Accordingly we make the following definitions.

- **Definition 1**. In a software branch with *n* releases, *Diff_Physical* of version v_i ($1<i\leq n$) is the absolute value of the difference of the physical lines of code of version v_i and version v_{i-1}.
- **Definition 2**. In a software branch with *n* releases, *Diff_Logical* of version v_i ($1<i\leq n$) is the absolute value of the difference of the logical lines of code of version v_i and version v_{i-1}.
- **Definition 3**. In a software branch with *n* releases, *Diff_Complexity* of version v_i ($1<i\leq n$) is the absolute value of the difference of the McCabe Cyclomatic complexity of version v_i and version v_{i-1}.

In most cases, we will see the increase of size of source code in new releases, because of the addition of new features. Occasionally, we can see the decrease of size of source-code in new releases. Therefore, absolute difference is used in these definitions. In a continually evolving software system, if the time duration between two adjacent releases is relative long, most of the bugs could be detected and reported in current releases. However, if the time duration between two adjacent releases is relative short, most of the bugs could not be detected and reported in current releases. The number of bugs reported to each version

might depend on the time duration to next release. It is a natural speculation that the longer a product version is used, the more likely the scenario that more bugs will be detected and reported.

Software users normally like trying new releases. Suppose after a product version vi is released, it will be widely used and tested until next version vi+1 is released. During the period between vi is released and vi+1 is released, most bugs will be reported to version vi. After the releases of vi+1, vi will receive less attention and most bugs will be reported to version vi+1. To test whether our speculation is correct, we formulate the following hypothesis.

H_{02}: In a software branch with n releases, there is no linear correlation between the number of bugs reported to version v_i and the time difference of the release dates of version v_i and version v_{i+1}(i<n).

Statistical tests, such as Person's correlation test and Spearman's rank correlation test can be used to evaluate Hypotheses H01 and H02. If the correlation is positive and significant at the 0.05 level, we will reject the hypothesis. For Hypothesis H01, a significant positive correlation indicates the number of bugs (bug reports) can be used to represent the product quality; an insignificant correlation indicates the number of bugs (bug reports) cannot be used to represent the quality of a specific version of a product. For Hypothesis H02, a significant positive correlation indicates the number of bugs (bug reports) depends on the time duration between adjacent releases; an insignificant correlation indicates no such relations.

Hence the objective of this study is to find the affecting factors of the number of bugs reported to each version of a software product and determining whether it can be used to represent software quality.

DATA SOURCE AND DATA MINING PROCESS

In this study, three open-source products are analyzed. They are Apache httpd, Apache Tomcat, and Apache Ant. The source code of these products is downloaded from their source code repositories [19]. The bug reports are mined from their Bugzilla site [20].

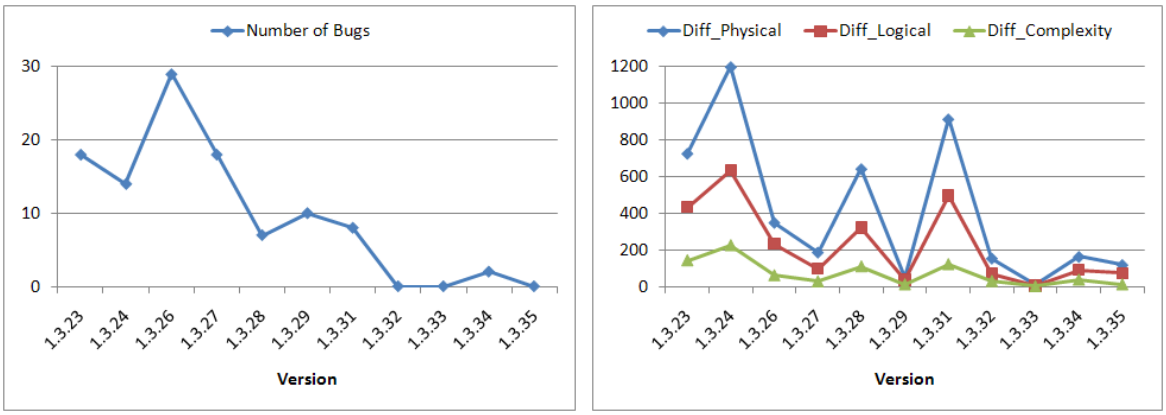
Software version system is a tree structure. There could be a trunk and zero or more branches. A software system with a single evolution line only contains one branch—its trunk. Six branches are selected from these three software systems. They are httpd branch 1.3, httpd branch 2.0, httpd branch 2.2, Tomcat branch 5.5, Tomcat branch 6.0, and Ant trunk. Table 1 describes the release information in these six branches. It should be noted that only the releases with bug reports data are included in this study. Early releases without bug reports, such as httpd 1.3.22 are not included in this study.

In measuring the size and the complexity of each product, a CASE tool—LocMetric [21] is used. Httpd is written in C. Therefore, "*.c" and "*.h" are considered as the source code. Tomcat and Ant are written in Java, and accordingly "*.java" files are considered as the source code. Three measurements of each version of six branches are recorded. They are physical lines of code, logical lines of code, and McCabe Cyclomatic complexity. The release date of each product version is also recorded.

In mining bug reports, only confirmed and fixed bug reports are mined. Unconfirmed or duplicated bug reports are not included. The bug reports are dated since the release date of the version of the product until September 29, 2010.

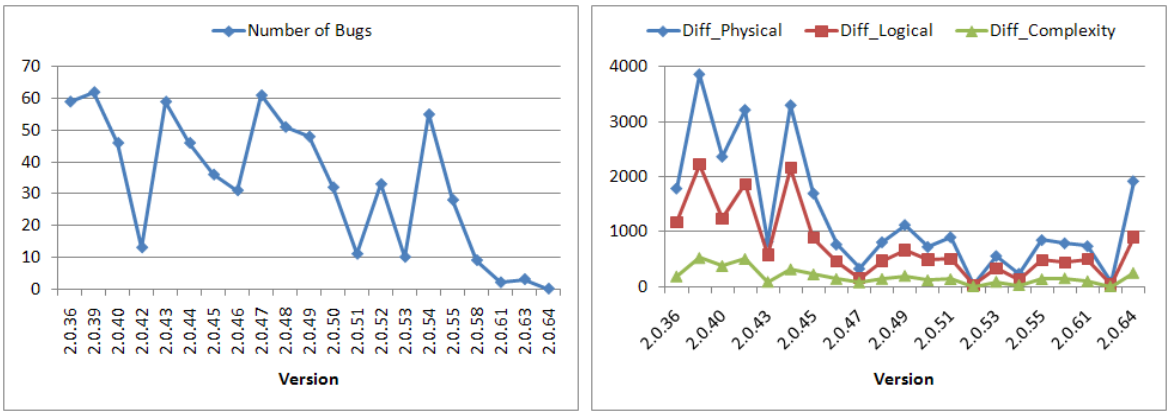
ANALYSIS AND RESULT

Figure 2 through Figure 7 illustrate the number of bugs reported to each version and the size changes of each version compared with its previous version for the six software branches.



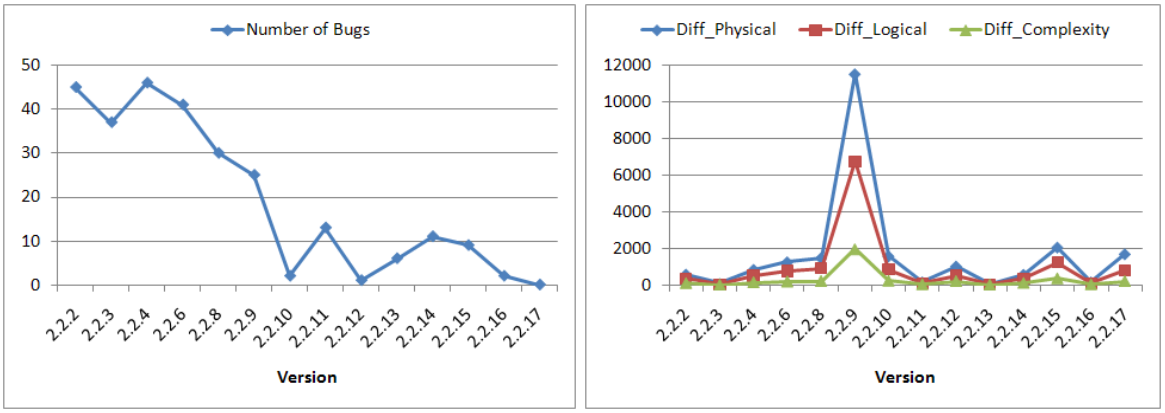
(a) (b)

Figure 2. (a) The number of bugs reported to each version; and (b) the size changes of each version in httpd branch 1.3.



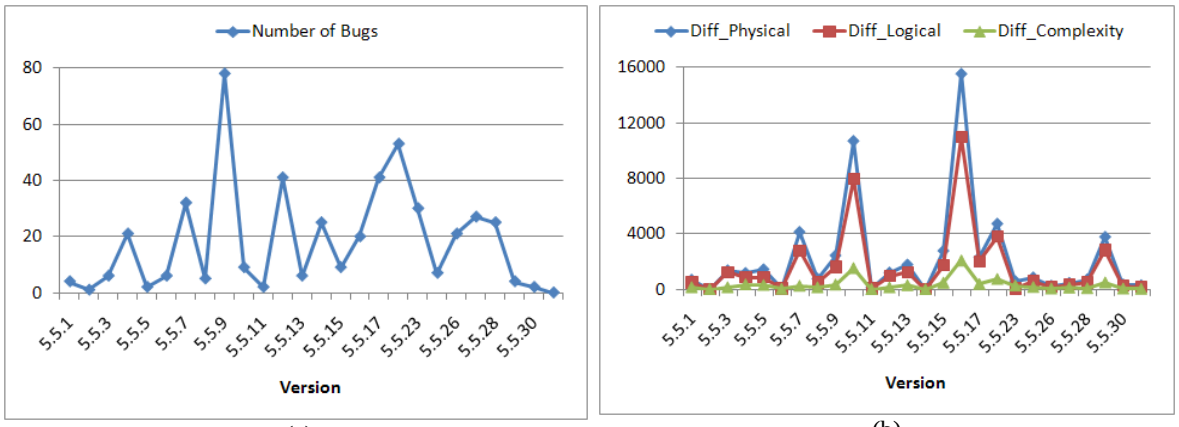
(a) (b)

Figure 3. (a) The number of bugs reported to each version; and (b) the size changes of each version in httpd branch 2.0.



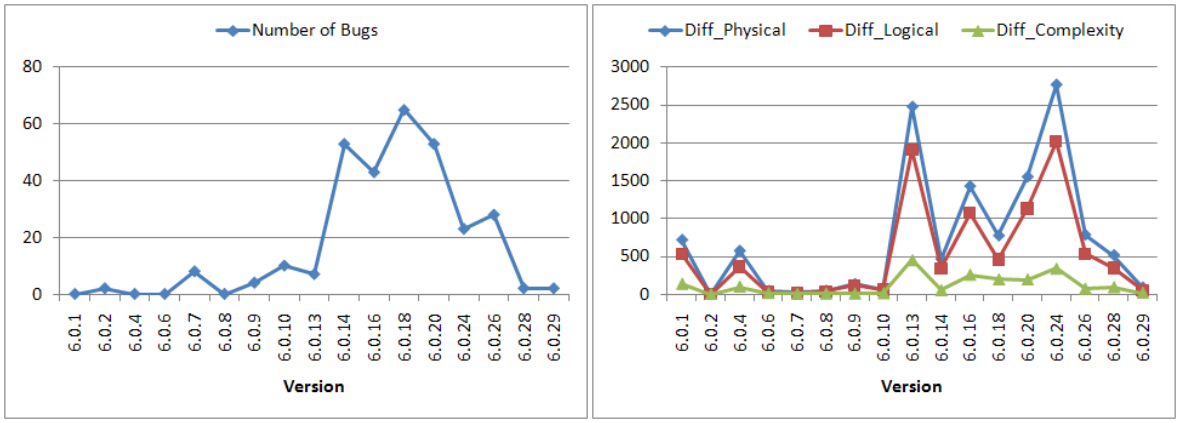
(a) (b)

Figure 4. (a) The number of bugs reported to each version; and (b) the size changes of each version in httpd branch 2.2.



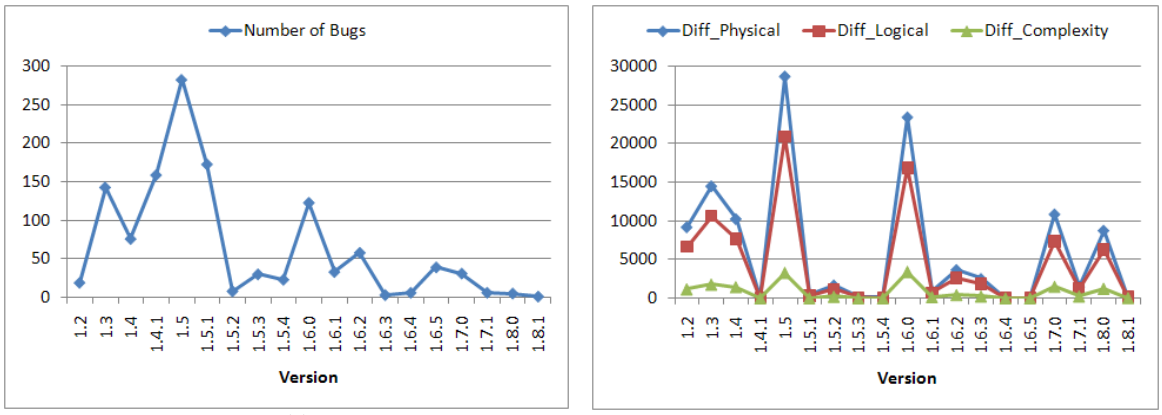
(a) (b)

Figure 5. (a) The number of bugs reported to each version; and (b) the size changes of each version in Tomcat branch 5.5.



(a) (b)

Figure 6. (a) The number of bugs reported to each version; and (b) the size changes of each version in Tomcat branch 6.0.



(a) (b)

Figure 7. (a) The number of bugs reported to each version; and (b) the size changes of each version in Ant trunk.

Table 1. Description of the six software branches

Branch	Number of releases	Version range
httpd 1.3	11	1.3.23 - 1.3.35
httpd 2.0	21	2.0.36 - 2.0.64
httpd 2.2	14	2.2.2 - 2.2.17
Tomcat 5.5	26	5.5.1 - 5.5.31
Tomcat 6.0	17	6.0.1 - 6.0.29
Ant trunk	19	1.2 - 1.8.1

To study the correlations between the number of bugs reported to each version and the size changes (Diff_Physical, Diff_Logical, and Diff_Complexity) of each version compared with its previous version, both Pearson's correlation tests and Spearman's rank correlation tests are performed. The results are summarized in Table 2 through Table 7, in which P represents Pearson's test, S represents Spearman's test, r indicates correlation coefficient and p indicates significance (two tailed). The correlations significant at the 0.05 level are bolded and highlighted. It can be seen that for all 36 tests, three correlations are significant at the 0.05 level (Table 7). The other 33 correlations are not significant, which means the number of bugs reported to a version has no correlation with the size changes to its previous version. Therefore, we cannot reject Hypothesis H01. Our hypothesis, i*n a software branch with n releases, there is no linear correlation between the number of changes to version* v_i *and the number of bugs reported in version* v_{i+1}*(*$i<n$*),* can not be proven to be false.

Table 2. The correlation tests of number of bugs and size changes in httpd branch 1.3

			*Diff_*		
			Physical	*Logical*	*Complexity*
Number of Bugs	P	*r*	0.334	0.395	0.365
		p	0.315	0.229	0.270
	S	*r*	0.566	0.566	0.502
		p	0.070	0.070	0.115
Number of dataset: 11					

Table 3. The correlation tests of number of bugs and size changes in httpd branch 2.0

			*Diff_*		
			Physical	*Logical*	*Complexity*
Number of Bugs	P	*r*	0.192	0.231	0.136
		p	0.404	0.314	0.557
	S	*r*	0.184	0.209	0.101
		p	0.425	0.363	0.664
Number of dataset: 21					

Table 4. The correlation tests of number of bugs and size changes in httpd branch 2.2

			*Diff_*		
			Physical	*Logical*	*Complexity*
Number of Bugs	P	*r*	0.063	0.081	0.066
		p	0.830	0.782	0.824
	S	*r*	-0.125	-0.046	-0.167
		p	0.669	0.875	0.568
Number of dataset: 14					

Table 5. The correlation tests of number of bugs and size changes in Tomcat branch 5.5

			*Diff_*		
			Physical	*Logical*	*Complexity*
Number of Bugs	P	*r*	0.121	0.128	0.114
		p	0.556	0.532	0.579
	S	*r*	0.366	0.342	0.345
		p	0.066	0.087	0.084
Number of dataset: 26					

Table 6. The correlation tests of number of bugs and size changes in Tomcat branch 6.0

			*Diff_*		
			Physical	*Logical*	*Complexity*
Number of Bugs	P	*r*	0.342	0.314	0.334
		p	0.179	0.219	0.190
	S	*r*	0.479	0.455	0.366
		p	0.052	0.066	0.148
Number of dataset: 17					

Table 7. The correlation tests of number of bugs and size changes in Ant trunk

			*Diff_*		
			Physical	*Logical*	*Complexity*
Number of Bugs	P	*r*	**0.621**	**0.628**	**0.568**
		p	**0.005**	**0.004**	**0.011**
	S	*r*	0.292	0.297	0.263
		p	0.225	0.218	0.276
Number of dataset: 19					

Some studies have found that number of bugs introduced to a new version of a product is directly correlated with the amount of changes made to its previous version. However, out study did not find significant correlations between reported bugs and size changes. We speculate the reasons could be twofold: (1) Bugs introduced in one version, say v5 might not be able to be detected in current version, it might be detected, reported, and fixed in later releases, say v8; and (2) Bugs detected and reported in one version, say v5 might not be introduced in modifying version v4, it might be introduced in earlier releases, say v2. Accordingly, the number of bugs reported to a specific version of a software product does not represent the quality of that specific product version.
In our 36 tests, 3 results indicate significant correlations, which are Pearson's tests on Ant trunk. Their corresponding scatter plots are in Figure 8. Although Spearman's rank correlations on the same data turned out to be insignificant, it is worth to study the difference between Ant trunk and other six branches and find out what could be the causes of this different behavior.

Apache Ant only contains one branch (its trunk). It does not release as frequently as other products, which could be the reason for its different bug reporting behavior. To further analyze this effect, we calculate the average time duration between two adjacent releases. The results are summarized in Table 8, which shows that Ant trunk has the longest average duration between two adjacent releases, 183 days. During this relatively longer period, about all the bugs introduced in modifying the previous version could be detected, reported, and fixed in current release. Therefore, the number of bugs reported to each version relatively represents the version quality. This is demonstrated in the correlation tests, as shown in Table 7.

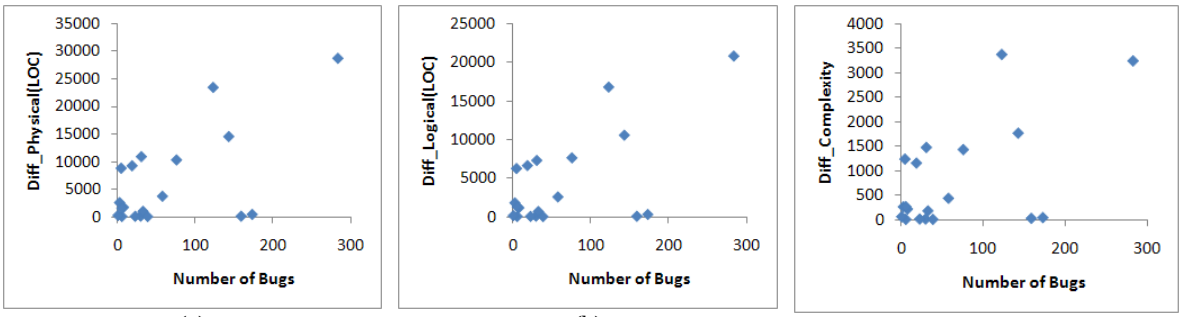
As discussed before, in a continually evolving software system, if the time duration between two adjacent releases is relative long, most of the bugs could be detected and reported and will not be carried onto later

releases. However, our study shows that most of the open-source product release occurs more frequently. It could take several releases to detect, report and fix a bug. Because open-source products are continually changed and released, what might affect the number of bugs reported might be the time duration between two adjacent releases. We accordingly test Hypothesis H02.

Again, both Person's correlation and Spearman's rank correlation tests are performed on these six branches. The result is summarized in Table 9, in which r indicates correlation coefficient and p indicates significance (two tailed). The correlations significant at the 0.05 level are bolded and highlighted.

From Table 9, we can see that three of the six branches show significant correlations between number of bugs reported and the time duration between current release and next release. They are httpd 2.2, Tomcat 5.5, and Tomcat 6.0. Figure 9 shows the scatter plots between number of bugs reported to each version and the time duration between the release dates of current version to next version. Although we cannot reject Hypothesis H02, we do find some significant correlations between the number of bugs reported and the time duration between adjacent releases. More specifically, 6 out of the 12 correlations turned out to be significant.

Combining the results obtained from Table 9 and Table 8, we can see that if the average time duration between two adjacent releases is short, such as Tomcat 6.0, Tomcat 5.5, and httpd 2.2, the number of bugs reported depends more on the time durations between adjacent releases. This is demonstrated in the correlation tests. In this case, some bugs introduced in current release might not be detected, reported, and fixed in next release and will be carried onto later versions. The longer a version is used, the more number of bugs will be reported. Accordingly, these bugs do not belong to a specific version—they belong to the entire branch.



(a) (b) (c)

Figure 8. The scatter plots of number of bugs versus (a) Diff_Physical; (b) Diff_Logical; and (c) Diff_Complexity, in Ant trunk.

Table 8. Average time duration between two adjacent releases

Branch	Number of releases	Date range	Average time duration between two releases (days)
Httpd 1.3	11	01/24/2002 - 04/24/2006	141
Httpd 2.0	21	05/01/2002 - 10/18/2010	147
Httpd 2.2	14	04/22/2006 - 10/18/2010	117
Tomcat 5.5	26	09/07/2004 - 09/16/2010	85
Tomcat 6.0	17	11/08/2006 - 07/22/2010	80
Ant trunk	19	10/24/2000 - 05/07/2010	183

Table 9. The correlation tests of number of bugs and time duration before next release

			Time Duration					
			httpd 1.3	httpd 2.0	httpd 2.2	Tomcat 5.5	Tomcat 6.0	Ant trunk
Number of Bugs	Pearson's test	*r*	-0.026	-0.401	**0.591**	**0.398**	**0.951**	-0.111
		p	0.938	0.071	**0.026**	**0.044**	**0.000**	0.651
	Spearman's test	*r*	0.143	-0.034	**0.722**	**0.524**	**0.931**	0.237
		p	0.676	0.884	**0.004**	**0.006**	**0.000**	0.329

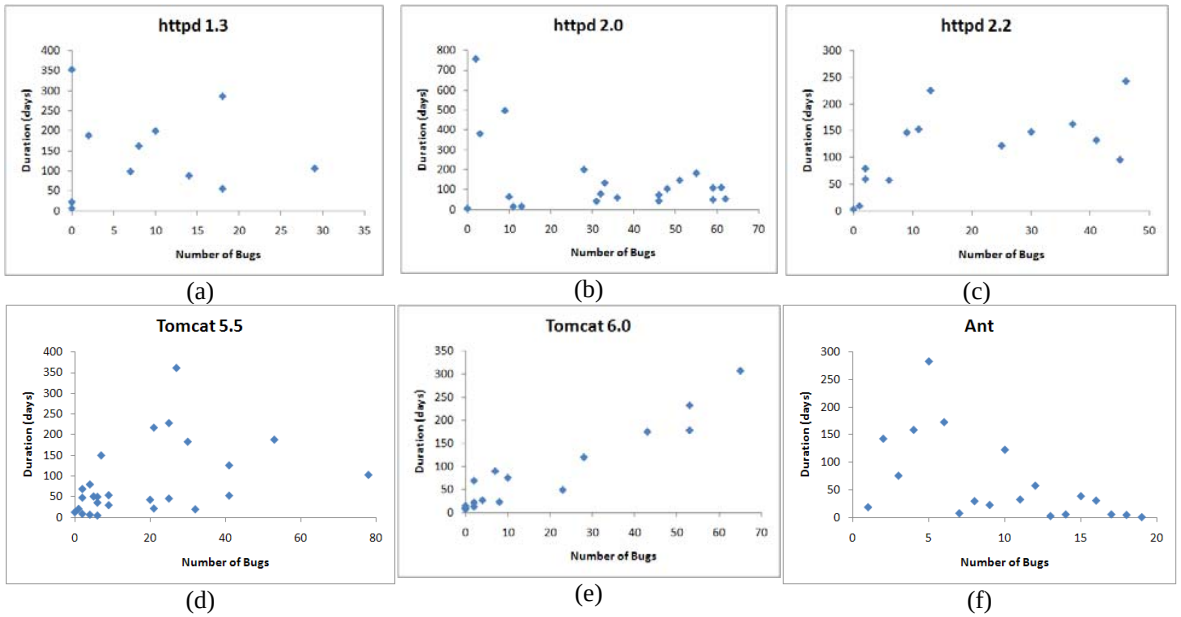


Figure 9. The scatter plot between number of bugs and time duration between two releases in (a) httpd 1.3; (b) httpd 2.0; (c) httpd 2.2; (d) Tomcat 5.5; (e) Tomcat 6.0; and (f) Ant trunk.

For httpd 2.0, httpd 1.3, because their average time duration between two adjacent releases is in not too short and not too long, their behaviors are not like Ant or Tomcat 5.5, Tomcat 6.0, and httpd 2.2: some bugs introduced in current release will be detected, reported, and fixed in next releases, some bugs will be carried on to later versions. Therefore, we did not find the number of bugs reported has significant correlations with either source code size changes or the time duration between adjacent releases.

Other factors that might affect the detecting and the report of bugs include the popularity of the product and popularity of the branch. For example, if more users are using one product, it is more likely to detect and report most of the bugs in a short period. However, for an unpopular product, even given a relatively long duration between adjacent releases, most of the bugs could still be hidden and carried onto later versions.

Although the number of bugs reported cannot be used to represent the quality of a specific product version, the accumulated number of bugs in the whole branch might be a good candidate to represent the quality of the entire product branch. Because no matter how frequently a new version is released, the bugs introduced in any version could be either reported in current release or future releases. In either case, they belong to the bug of this branch and the accumulated number of bugs (remove duplications) reported to all versions of this branch can be used to represent the quality of the entire branch quality. However, this representation awaits validations in future work.

CONCLUSIONS

In this paper, we studied the number of bugs reported to each version of a continually evolving software product. We found (1) the number of bugs reported to each version has no significant correlation with the size changes and that it cannot be used to represent the quality of a specific product version; (2) the number of bugs reported to each version has some correlations with the time duration between adjacent releases; and (3) time duration between adjacent releases has effect on the two correlations.

Although the number of bugs reported cannot be used to represent the quality of a specific product version, the accumulated number of bugs might be a good candidate to represent the quality of the entire product branch.

REFERENCES

[1] http://www.bugzilla.org/

[2] http://www.bugzilla.org/installation-list/

[3] Panjer, L. D. Predicting Eclipse bug lifetimes. In *Proceedings of the 4th International Workshop on Mining Software Repositories*, Minneapolis, MN (May 20-26, 2007), 29.

[4] Grammel, L., Schackmann, H., Lichter, H. BugzillaMetrics: an adaptable tool for evaluating metric specifications on change requests. In *Proceedings of the 9th International Workshop on Principles of Software Evolution*, Dubrovnik, Croatia (September 3-4, 2007), 35–38.

[5] Canfora, G. and Cerulo, L. Where is bug resolution knowledge stored? In *Proceeding of the 3rd International Workshop on Mining Software Repositories*, Shanghai, China (May 22-23, 2006), 183–184.

[6] Sliwerski, J., Zimmermann, T., and Zeller, A. When do changes induce fixes? In *Proceedings of the 2nd International workshop on Mining software repositories*, Saint Louis, Missouri (May 17, 2005), 1.

[7] English, M., Exton, C., Rigon, I., and Cleary, B. Fault detection and prediction in an open-source software project. In *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, Vancouver, British Columbia, Canada (May 18-19, 2009), Article No. 17.

[8] Li, Z., Tan, L., Wang, X., Lu, S., Zhou, Y., and Zhai, C. Have things changed now? An empirical study of bug characteristics in modern open source software. In *Proceedings of 1st Workshop on Architectural and System Support for Improving Software Dependability*, San Jose, California (October 21-21, 2006), 25–33.

[9] Canfora, G. and Cerulo, L. Supporting change request assignment in open source development. *In Proceedings of the 2006 ACM symposium on Applied computing*, Dijon, France (April 23-27, 2006), 1767–1772.

[10] Sahoo, S. K., Criswell, J., and Adve, V. An empirical study of reported bugs in server software with implications for automated bug diagnosis. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, Cape Town, South Africa (May 2-8, 2010), 485–494.

[11] Chowdhury, I. and Zulkernine, M. Can complexity, coupling, and cohesion metrics be used as early indicators of vulnerabilities? In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, Cape Town, South Africa (May 2-8, 2010), 1963–1969.

[12] Sun, C., Lo, D., Wang, X., Jiang, J., and Khoo, S. Discriminative model approach towards accurate duplicate bug report retrieval. In *Proceedings of the 32nd International Conference on Software Engineering*, Cape Town, South Africa (May 2-8, 2010), pp. 45–54.

[13] Zhang, H. and Kim, S. Monitoring software quality evolution for defects. *IEEE Software* 27(4) (2010), 58–64.

[14] Davies, J., Zhang, H., Nussbaum, L., and German, D. M. Perspectives on bugs in the Debian bug tracking system. In *Proceedings of 7th IEEE Working Conference on Mining Software Repositories*, Cape Town, South Africa (May 2-3, 2010), 86–89.

[15] Yu, L., Schach, S R., and Chen, K. Measuring the maintainability of open-source software. In *Proceedings of the 4th International Symposium on Empirical Software Engineering*, Noosa Heads, Queensland, Australia (November 2005), 287–293.

[16] Graves, T. L., Karr, A. F., Marron, J. S., and Siy, H. Predicting fault incidence using software change history. *IEEE Transactions on Software Engineering* 26(7) (2000) 653–661.

[17] Williams, C. C. and Hollingsworth, J. K. Automatic mining of source code repositories to improve bug finding techniques. *IEEE Transactions on Software Engineering* 31(6) (2005) 466–480.

[18] Livshits, B. and Zimmermann, T. DynaMine: finding common error patterns by mining software revision histories. *ACM SIGSOFT Software Engineering Notes* 30(5) (2005) 296–305.

[19] http://archive.apache.org/dist

[20] https://issues.apache.org/bugzilla/

[21] http://www.locmetrics.com