20th IMEKO TC4 International Symposium and
18th International Workshop on ADC Modelling and Testing
Research on Electric and Electronic Measurement for the Economic Upturn
Benevento, Italy, September 15-17, 2014

Measurements to assess the effort related to different kinds of software maintenance

Alessandro Murgia[1], Marco Ortu[2], Roberto Tonelli[2],
Giulio Concas[2], Michele Marchesi[2], Steve Counsell[3]

[1]*University of Antwerp, Antwerp - Belgium, alessandro.murgia@ua.ac.be*
[2]*Department of Electrical and Electronic Engineering (DIEE), University of Cagliari - Italy*
concas,michele,marco.ortu,roberto.tonelli@diee.unica.it
[3]*Department of Computer Science, Brunel University, Uxbridge - UK, steve.counsell@brunel.ac.uk*

Abstract – **The Issue Tracking System is a repository used to support the software development activity. Differently from Bug Tracking Systems, the Issue Tracking Systems are not bug-centric. Indeed, they record entries related to corrective, adaptive and perfective maintenance.**

This study tackles in the under-explored aspect of software maintenance related to the resolution of the issue-types new feature, task and improvement. We mine the issue tracking systems of five open source projects with more than 9000 issues, discovering that maintenance effort and priority are influenced by the issue-type handled.

I. INTRODUCTION

Bug tracking systems (BTS) are repositories historically used by software companies to handle activities mainly focused on corrective maintenance, namely bug fixing. Due to the necessity to cope with different maintenance activities, bug tracking systems are evolved in issue tracking systems (ITS). Indeed, ITS can keep track of corrective, adaptive and perfective maintenance requests reported as entry in its database. These entries are called issue and their workflow depends on companies policies. Despite the equal relevance of all the software maintenance activities, the bug fixing is the only one on spotlights [13][7][10][9].

Taking leverage of relatively recent adoption of ITS, we tackle in an under-explored aspect of software maintenance related to the handling of the issue-types *new feature*, *task* and *improvement*. We take into account also the *bug* issue-type to highlight similarities and differences with this well known type of issue. Our attention is focused on these types since they are commonly adopted by different software projects and are representative of Swanson's adaptive maintenance (new feature), perfective maintenance (improvement) and corrective maintenance (bug) [14] . The Swanson's classification can be found today incorporated into the ISO/IEC 14764 standard [1]. As far as the authors know, this is the first study exclusively devoted to consider how the maintenance activity is qualitatively and quantitatively influenced by the type of issue handled in open source projects.

This work follows the Goal-Question-Metric paradigm [15]. The *goal* of this study is to investigate the influence of issue-type on maintenance effort and priority. The *focus* is to evaluate how metrics, like fixing-time and priority, change for the issue-types new feature, improvement, task and bug. The *viewpoint* is that of issue triager and researchers. The first one, scheduling maintenance workload, is interested in evaluating how the maintenance activity is issue-dependent. The latter, building models for software maintenance based on data mined from software repositories, are interested on how to exploit the information related to the issue-type. The *environment* of this paper regards the ITS of five open source projects. To achieve our purpose, we pursue the following research questions:

- **RQ1:** Is the fixing-time dependent on issue-type?
- **RQ2:** Is the priority resolution dependent on issue-type?

Our findings highlight that maintenance effort and scheduling are not only qualitatively, but also quantitatively dependent on the type of issue handled. We believe that our introductory study can shed light on how the issue resolution process is influenced by the maintenance activities performed.

This paper is organized as follow. In section ii., we show the key-role played by BTS in literature for the analysis of maintenance effort and priority. In section iii. we present the projects used for this study. In section iv. we answer to the research questions. Finally, in section v. we wrap up the result and outline the future work.

II. RELATED WORKS

The bug-oriented repository Bugzilla played a key-role for the analysis of maintenance effort based on fixing-time. Demeyer et al. show that this BTS is the most common across mining software conferences [6]. Bugzilla has been widely studied since it stores bugs of Eclipse and Mozilla, two of the most common case studies [13][7][10][9]. Similar BTSs are used by Bougie et al. [3], which perform a replication study of [13] using the FreeBSD bug repository, and Bhattacharya et al. [2] which use the Google Code bug tracker.

Differently from previous studies, we mine the issue-oriented repository Jira and analyze the relationship between fixing-time and the issue-types: bug, task, new feature and improvement.

Jira is used by Weiss et al. to automatically predict the issue effort [16]. In their work the issue was represented by bug, task and new feature and they use only one open source project, JBoss. Moreover, they limited the analysis to 567 special issues where the developer manually filled the optional field *effort*, namely the issue fixing-time.

Bugzilla is used also by several studies related to bug priority on software maintenance [11][8][12]. The only exception is represented by Mockus et al. which use the bug reports in the BUGDB repository [11].

Differently from these works, we analyze the relationship between maintenance priority and maintenance issue types. As far as the authors know, we are there are no other works investigating these aspects.

III. DATASET

In April 2013, we mined the Jira repositories of five open source projects of the Apache Software Foundation (ASF)[1] : MyFaces Tobago (MFT), Solr, Struts 2, Tuscany and UIMA. Such repositories host 9323 entries and several issue-types. To have a statistically significant amount of data, we decide to keep those issue-types which presented at least 50 cases. We ended up with 4 issue-types:

- bug: a problem which impairs or prevents the functions of the product
- improvement: an enhancement to an existing feature
- new feature: a new feature of the product
- task: a task that needs to be done. Due to this fuzzy definition, we do not map its resolution to any specific maintenance type.

These types represent usually the 98% of issue-types hosted by our projects.

Priorities in Jira can be sorted according to their relevance in 5 categories: Blocker (highest priority), Critical, Major, Minor and Trivial (lowest priority). We use issue types and priorities presented by default by Jira since they are common across projects. Table 1 reports how issue-types are distributed numerically and perceptually across projects. Each issue of our dataset is characterized by the Jira's fields[2]: resolution, state, priority, data created and resolved. In our analysis we considered only the issues with resolution "fixed" and state "closed" since they have gone through the whole issue life-cycle. For the fixing-time we compute the difference between the Jira's issue fields resolved and created.

Table 1. Number and percentage of issue-types for each project.

Type / Project	Bug	Improvement	New Feature	Task
MFT	377 (46%)	276 (33%)	68 (8%)	92 (11%)
Solr	582 (49%)	394 (33%)	145 (12%)	55 (4%)
Struts 2	1377 (56%)	627 (25%)	185 (7%)	246 (10%)
Tuscany	1857 (71%)	465 (17%)	200 (7%)	64 (2%)
UIMA	1253 (57%)	591 (27%)	149 (6%)	185 (8%)
Total	5446 (58%)	2353 (25%)	830 (8%)	648 (7%)

IV. RESULTS AND DISCUSSION

In Table 1 we report the percentage of issues represented by each category. Considering all the projects together, 5446 (58%) represent bugs, 2353 (25%) represent improvements, 747 (8%) represent new features and 642 (7%) represent tasks.

Bug-issues are the majority, but there is still a 42% of issues which are not related to corrective maintenance.

Issue fixing-time. Figure 1 reports the box-plots of fixing-time, measured in days, for the different issue-types. Note that fixing-time is represented on a logarithmic scale, thus the differences in fixing-times appear squized. In Table 2 we report also some statistics belonging to the different issue-types: mean, median and 90th percentile, because the fixing-time distributions are rigth-skewed with a "fat-tail". Values much larger than the mean are quite common and the mean is not the only representative statistics [4].

RQ1: Is the fixing-time dependent on issue-type?

The data on fixing-time in Figure 1 show that the 50% of bugs are addressed in a couple of days, with the only exception of Struts 2 where it takes 2 weeks. This result is consistent with fixing-time of bugs in Eclipse and Mozilla [9]. The box-plots and the results in Table 2 indicate that fixing-time is issue-type dependent. The fixing-time of each issue type is quite regular across the different projects, since for the five projects analysed, fixing-time for bug is generally the lowest, while fixing-time for new features is generally the largest, as revealed by the box plot analysis and by data in Table 2. Tasks are an exception, since they display larger fluctuations with respect to other fixing types. The behaviour of task issue type may be due to the lower number of statistical samples available, which may enhance variability, but also to the fuzzy definition of task-issues. Indeed, task is defined in Jira as "something to be done", and may represent any maintenance operation.

The result on bug fixing-times is robust because it is consistent across all the examined projects and because the

[1]https://issues.apache.org/jira

[2]https://confluence.atlassian.com/display/JIRA/What+is+an+Issue

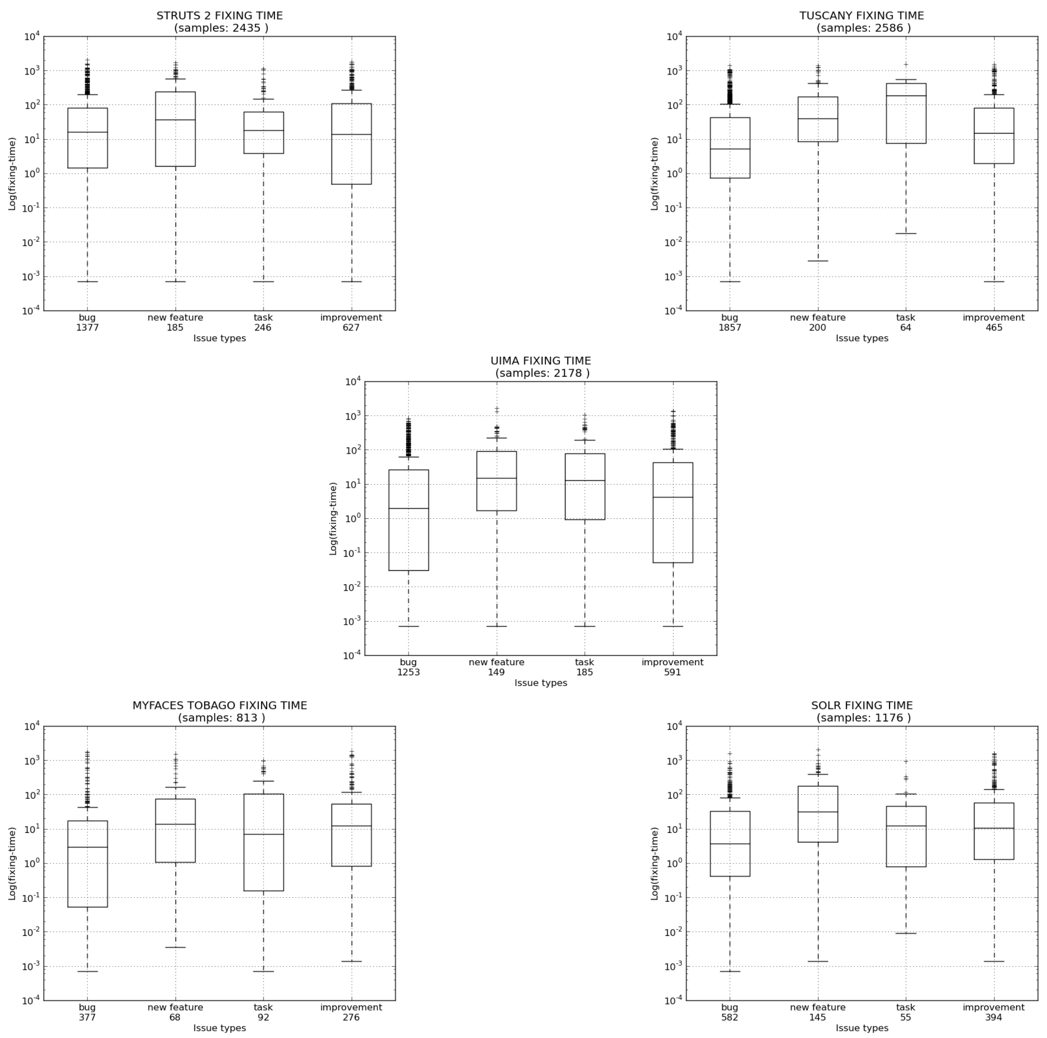

Fig. 1. Box-plot of fixing-time expressed in days

Table 2. Projects statistics.

	Bug			New Feature			Task			Improvement		
	Mean	Median	90th%	Mean	Median	90th%	Mean	Median	90th%	Mean	Median	90th%
MFT	56.93	2.95	77	137	13.94	455	115.60	7.02	470.33	106.21	12.13	234.96
Solr	45.70	3.68	124.55	150.51	30.97	461.68	55.26	12.4	104.92	84.71	10.70	185.51
Struts 2	89.01	15.95	227.26	185.35	36.19	643.35	65.30	17.96	129.22	132.13	13.90	360.40
Tuscany	53.59	5.19	144.53	149.61	39.04	423.94	229.26	184.32	430.50	99.49	14.56	256.74
UIMA	45.99	1.92	147.04	96.64	14.93	338.05	79.14	12.80	363.11	64.34	4.17	185.99

mean values are quite stable, being the difference among the larger and the smaller only a factor of two. The same holds for Improvement and New Feature, while the means for Task have slightly larger fluctuations.

Table 2 shows how mean, median and 90% have quite different values for every project and every issue-type. Thus, considering issue types, while for fixing a bug a rough estimate can be 50 days, the days for resolve and close an New Feature can easily be three times more, while for implementing an Improvement they can easily double. This may provide an underestimated fixing-time. For example, if we consider our first project, MFT, the estimate performed using bug issues alone would provide a total of about 45500 working days, while using all the issue-types with proper fixing-times the estimate would be about 76500 working days. Furthermore, the differences in mean, median and 90% indicate that the estimation of the total effort and the result for a smaller dataset can present large fluctuations. In fact, it is well known that the mean is not stable for power law distributions and grows with the number of samples considered [5]. On the other side, the median shows that half of the issues are solved in much shorter time than the average, and this result is particularly enhanced in bug-issues. Thus, while half of the bugs are solved in a very short time, the same is not true for the other issue-types, where the difference among mean and median is not so large. As a consequence of these properties, if one tries to estimate the effort using the fixing-time for a bug considering only a sub-sample of dataset, the resulting value can underestimate the correct value by even an order of magnitude. On the other hand, these data can be very useful for estimating the effort in the worst and in the best cases. In fact, the use of the median - which is for all issue-types much smaller than the average - would provide more reliable estimates than the mean for the issues less problematic issues, which are in the half bottom of the distribution. The worst cases, representing much problematic issues, can be handled using the 90th percentile, which is an order of magnitude larger than the median for all issue-types.

Among the implications of this analysis, at least two are immediate. First, any effort estimation obtained by the mere use of BTS data are strongly biased, and generally underestimate the true effort. Second, the detailed knowledge of fixing times for different issue-types can be very helpful for scheduling purposes, and assist a team manager in estimating the effort and the time needed for delivering the software product on time, balancing the scheduling among the different maintenance activities. Finally the analysis rises new questions on the reasons why the different maintenance activities deserve so clearly different efforts and times among for different issue-types.

Issue Priority. Table 3 reports the priority for each issue-type. *Major* and *Minor* priorities account for 76% - 98% of the choices made by the triager. Only 2 out 5 priority levels are normally selected for scheduling the issue priority level. Our hypothesis is that too many options, with no clear boundaries, inhibit their full adoption. Similar conclusion has been made by Herraiz observing bug reports in Bugzilla [8].

RQ2: Is the priority resolution dependent on issue-type?

Looking at priorities *Major* and *Minor* in table 3 we notice that the priority is, to some extent, dependent on the type of issue handled. Priority *Major* prevails in bug and task as well as in new features, even if in Solr the number of instances is relative small. On the other hand, this predominance is not exhibited by improvement, where on the contrary, the priority *Minor* sometime prevails (UIMA and Solr).

The results achieved on priority and fixing-time are consistent, indeed all suggest that for these companies the bug resolution has more relevance than the others maintenance activities.

V. CONCLUSION AND FUTURE WORK

There is little information on how maintenance effort and priority are quantitatively influenced by the type of maintenance performed. Many research activities on this topic, using data mined from BTS, are biased by corrective maintenance. In this study, we analyse data stored in ITS, which keep track of adaptive maintenance, perfective maintenance and corrective maintenance. We shed light on how the issue resolution process is affected by the maintenance activities performed, evaluating how fixing-time and priority are influenced by the issue-type handled.

Table 3. Number and percentage of issue-types for each project.

Type \ Priority	Blocker	Major	Critical	Trivial	Minor	Maj+Min
Struts2						
bug	52	877	101	49	298	0.85%
new feature	0	136	5	3	41	0.96%
task	40	164	12	8	22	0.76%
improvement	4	337	14	52	220	0.89%
UIMA						
bug	16	783	14	71	369	0.92%
new feature	1	92	0	4	52	0.97%
task	1	134	2	10	38	0.93%
improvement	0	225	2	83	281	0.86%
Tuscany						
bug	116	1363	103	21	254	
new feature	1	156	5	1	37	0.96%
task	2	50	4	2	6	0.88%
improvement	3	287	9	14	152	0.94%
Solr						
bug	15	316	25	46	180	0.85%
new feature	1	69	0	10	65	0.92%
task	2	30	1	7	15	0.82%
improvement	1	152	0	48	193	0.88%
MyFaces Tobago						
bug	3	236	16	30	92	0.87%
new feature	0	37	1	2	28	0.96%
task	2	50	0	15	25	0.82%
improvement	0	142	0	21	113	0.92%

Our results show that fixing-time and priority are issue dependent, namely they depend on the type of maintenance activity performed. The fixing-time of bug is generally lower than other issue-types, whereas new feature requires generally the highest fixing-time. Task's behavior is less predictable and seems project-dependent. For issue prioritization, bug and task are mainly labelled as major by the triager, whereas for improvement this trend is not confirmed.

These results are relevant from different point of views. Firstly, they point out that models for effort estimation based mainly on data extracted from bug repositories may provide biased estimation. Secondly, they can be exploited for scheduling purposes. Given a release deadline, the project manager can balance the maintenance activities according to their fixing-time. Finally, they arise intriguing questions on: why corrective maintenance appears to be faster than perfective or adaptive maintenance or why bug fixing is generally a major priority for the company.

We believe that our study has to be integrated considering companies with different marketing strategies and pre and post release maintenance activities. In future work, we will extend the analysis to projects targeting different customer markets. On one hand, we will consider projects that requires high level of reliability (e.g. health care devices), namely projects eager to ship bug-free software. On the other hand, we will examine projects where the reliability is sacrificed in order to frequently introduce new appealing features.

ACKNOWLEDGMENTS

This work has been sponsored by (i) Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) under project number 120028 entitled Change-centric Quality Assurance (CHAQ) and (ii) Regione Autonoma della Sardegna (RAS), Regional Law No. 7-2007, project CRP-17938 "LEAN 2.0"

REFERENCES

[1] International standard - ISO/IEC 14764 ieee std 14764-2006. *ISO/IEC 14764:2006 (E) IEEE Std 14764-2006 Revision of IEEE Std 1219-1998)*, 2006.

[2] P. Bhattacharya and I. Neamtiu. Bug-fix time prediction models: can we do better? In *MSR*, pages 207–210, 2011.

[3] G. Bougie, C. Treude, D. M. Germán, and M. D. Storey. A comparative exploration of freebsd bug lifetimes. In *MSR*, pages 106–109, 2010.

[4] G. Concas, M. Marchesi, A. Murgia, S. Pinna, and R. Tonelli. Assessing traditional and new metrics for object-oriented systems. 2010.

[5] G. Concas, M. Marchesi, A. Murgia, R. Tonelli, and I. Turnu. On the distribution of bugs in the eclipse system. *IEEE Trans. Software Eng.*, 37(6):872–877, 2011.

[6] S. Demeyer, A. Murgia, K. Wyckmans, and A. Lamkanfi. Happy birthday! a trend analysis on past msr papers. MSR '13, pages 353–362, 2013.

[7] E. Giger, M. Pinzger, and H. Gall. Predicting the fix time of bugs. RSSE '10, pages 52–56. ACM, 2010.

[8] I. Herraiz, D. M. Germán, J. M. González-Barahona, and G. Robles. Towards a simplification of the bug report form in eclipse. In *MSR*, pages 145–148, 2008.

[9] A. Lamkanfi and S. Demeyer. Filtering bug reports for fix-time analysis. In *CSMR*, pages 379–384, 2012.

[10] L. Marks, Y. Zou, and A. E. Hassan. Studying the fix-time for bugs in large open source projects. In *PROMISE*, page 11, 2011.

[11] A. Mockus, R. T. Fielding, and J. D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3):309–346, 2002.

[12] M. Nurolahzade, S. M. Nasehi, S. H. Khandkar, and S. Rawal. The role of patch review in software evolution: an analysis of the mozilla firefox. IWPSE-Evol '09, pages 9–18. ACM, 2009.

[13] L. D. Panjer. Predicting eclipse bug lifetimes. In *MSR*, page 29, 2007.

[14] E. B. Swanson. The dimensions of maintenance. In *Proceedings of the 2nd international conference on Software engineering*, ICSE '76, pages 492–497. IEEE Computer Society Press, 1976.

[15] R. van Solingen, V. Basili, G. Caldiera, and H. D. Rombach. *Goal Question Metric (GQM) Approach.* John Wiley Sons, Inc., 2002.

[16] C. Weiß, R. Premraj, T. Zimmermann, and A. Zeller. How long will it take to fix this bug? In *MSR*, page 1, 2007.