Towards a simplification of the bug report form in Eclipse

Israel Herraiz
Universidad Rey Juan Carlos
Madrid, Spain
herraiz@gsyc.es

Daniel M. German
University of Victoria
Canada
dmg@uvic.ca

Jesus M. Gonzalez-Barahona
Universidad Rey Juan Carlos
Madrid, Spain
jgb@gsyc.es

Gregorio Robles
Universidad Rey Juan Carlos
Madrid, Spain
grex@gsyc.es

ABSTRACT

We think that the bug report form of Eclipse contains too many fields, and that for some fields, there are too many options. In this MSR challenge report, we focus in the case of the severity field. That field contains seven different levels of severity. Some of them seem very similar, and it is hard to distinguish among them. Users assign severity, and developers give priority to the reports depending on their severity. However, if users can not distinguish well among the various severity options, they will probably assign different priorities to bugs that require the same priority. We study the mean time to close bugs reported in Eclipse, and how the severity assigned by users affects this time. The results shows that classifying by time to close, there are less clusters of bugs than levels of severity. We therefore conclude that there is a need to make a simpler bug report form.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement; K.6.3 [**Software management**]: Software maintenance; D.2.4 [**Software Engineering**]: Software/Program Verification—*Statistical methods*

General Terms

One-way unstructured comparison

Keywords

bug report, bug tracking system, Eclipse, MSR Challenge

1. INTRODUCTION

Bug reporting is an important task for the sustainability of a libre software project [3]. Libre software projects rely on their users for testing and verification: "With enough eyeballs any bug is shallow" (also known as Linus Law) [2]. Bug reporters (those submitting bugs) are important members of the communities behind libre projects. Hence it is important to constantly evaluate if the defect tracking system is satisfying the needs of both the users and the developers. In particular, we would like to know if it is possible and necessary to simplify the defect reporting form used in Eclipse without losing its effectiveness.

Software developers are usually unable to cope with all the bugs that are notified as they are submitted. They need to prioritize them. They need to determine, for any particular bug, how important it is, and allocate their time and resources according to such prioritization. Bug tracking systems allow reporters to select a severity for the bug they have found. This information is expected to be useful to developers, who do not have to sort through every single report to attend those with a greater severity first and those lesser one.

Bugzilla has become one the most pervasive bug tracking system in the libre software world. It was originally developed by the Mozilla project to help them manage and track defects. In addition to Eclipse, Bugzilla is used by other well-known projects such as GNOME and KDE. In Bugzilla every bug is labeled with a *severity* attribute which can take one of the seven types described in table 1. Those definitions are part of the default installation of Bugzilla, and are included in the bug reporting guidelines of Eclipse.

One of the major problems when assigning severity to a bug is that a reporter might not be the best qualified to de-

Severity	Description
Blocker	Blocks development and/or testing work.
Critical	Crashes, loss of data, severe memory leak.
Major	Major loss of function.
Normal	(no description given)
Minor	Minor loss of function, or other problem where easy workaround is present.
Trivial	Cosmetic problem like misspelled words or misaligned text
Enhancement	Request for enhancement

Table 1: Types of severity for bugs. Source: https://bugs.eclipse.org/bugs/page.cgi?id=fields.html

MSR '08 Leipzig, Germany

termine exactly what type the defect falls into [4]. Another issue is that to submit a report it is required to know the precise difference between the categories, and this is not always the case (e.g. a person who does not bother to read the description of each type of severity might find it confusing to report a defect as minor or trivial). We believe that these definitions are confusing, as having seven different levels for severity only confounds reporters.

The severity field is expected to be used by developers to classify bugs according to their importance, and to select what bugs are attended first. Developers would also use the field *priority* to specifically state the order in which bugs should be solved (there are five types of priorities: P1, P2, P3, P4 and P5).

We believe that if there are seven types of severities, and developers organize their work using them, then we should observe seven groups of defects (one per each type of severity) when they are classified according to the time that it takes between the report of the defect and its resolution. We should also expect to see a correlation between the seven severities and the five priorities.

The objective of this paper is to evaluate these two statements using the database of the Bugzilla tracking system of Eclipse.

The rest of this paper is organized as follows: the next section describes the data source used for this paper. Section 3 provides the statistical analysis used in this study, while Section 4 presents our results. The last section includes conclusions and some considerations for further work.

2. DATA SOURCE

For this study, we have used the database of the Bugzilla tracking system of Eclipse[1]. For each defect we collected its severity (as assigned by the user), priority (as assigned by the developer), developer id and time elapsed since the bug was reported till it was closed. We only considered closed bugs.

3. METHODOLOGY

Our main hypothesis can be stated as: when defects are classified according to the time it takes to solve them they fall into less than seven categories.

To test this hypothesis, we have divided closed bugs into seven categories (one for each degree of severity assigned by the user). For each bug in each category we have measured the time that it took to close the bug, as the time elapsed from the notification of the bug to when it was marked as closed.

After this step, we applied an statistical method called *one-way unstructured comparison* [1], that compared the mean time for each category. The method returns a value of a statistical parameter that can be used to determine if the means of two groups are different. If the difference among the means is smaller than this parameter, those means are not statistically different and therefore the two groups may be considered of the same kind.

We have applied this to the seven groups of bugs, classified by severity (assigned by the users). We have used the same method to the five groups of bugs grouped by priority (specified by the developers).

[1]Available from `http://archive.eclipse.org/arch/`

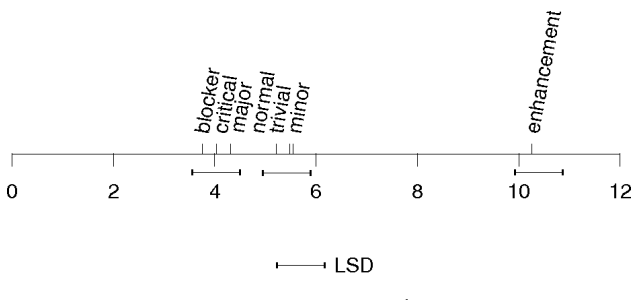

Figure 1: Bugs grouped by severity. Horizontal axis shows mean time interval to close a bug (in months). The straigth segment correspond to the test (LSD, Least Significant Difference) that determine if the means are statistically different. Using the LSD parameter, we obtain three different groups of bugs, as shown by the three segments just below the axis.

4. RESULTS

4.1 Groups by severity

Figure 1 shows the different groups of bugs, labeled by severity. The horizontal axis represents the time interval to close a bug, measured in months. The values are the mean of the times for each category. The plot shows the parameter LSD (Least Significant Difference). This parameter acts as a "yardstick". If two means are separated by more than the LSD value, those means are considered to be statistically different. Otherwise, the two means are obtained from samples that belong to a same population. In other words, bugs contained in the two groups are statistically of the same class.

When reporters submit new bugs, they are using three different degrees of severity ("blocker", "critical" and "major") for bugs that are attended with the same highest priority. On the other hand, they are using "normal", "trivial" and "minor" for bugs that are attended with a similar priority by the developers. Finally, requests for enhancements are attended with the lowest priority.

Therefore, attending to these criteria, we have three different classes of bugs. We have labeled these categories as *important*, *not important* and *request for enhancement*. Those classes contain following severities:

- **Important**: Blocker, critical and major.
- **Non-important**: Normal, trivial and minor.
- **Request for enhancement**: Enhancement.

4.2 Groups by priority

So far, we have analyzed how the seven groups of bugs (sorted by severity) can be simplified into three groups. The value of severity assigned by the user helps developers to assign the right priority to those bugs. Thus, high severity bugs should be attended with higher priority than low severe bugs.

Developers can assign an additional priority property to their bugs. This property has only five levels (compared to the seven levels of severity).

We believe that this asymmetry exists because bugs in a given severity class can be prioritized differently (some bugs labeled as critical should be solved before others).

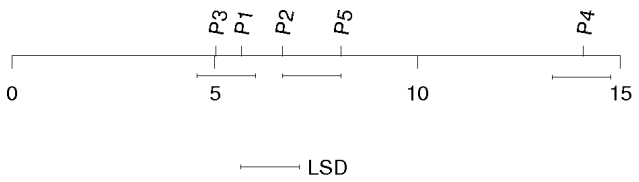

Figure 2: Bugs grouped by priority. Again, horizontal axis shows elapsed time to close the bug (in months). In this case, we have again three groups.

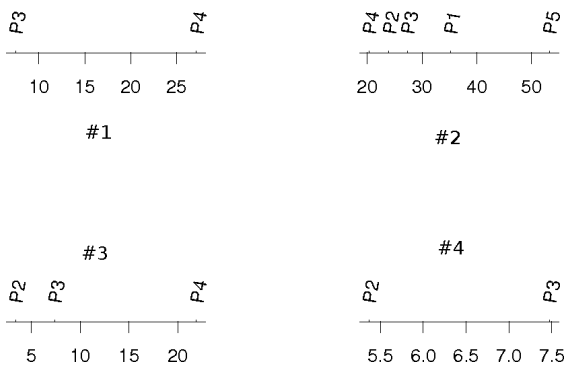

Figure 3: Mean time to close the bugs for the top four developers. The results are shown aggregated by priority. Horizontal axis shows time in months. Some developers use only a subset of the possible levels of priority.

However, the priority field is not used like this. The bug priority is used as a major field and classifies bugs according to the order in which they are solved: bugs with a lower priority are attended later than bugs with a higher priority.

We presume that the time to close a bug will be longer for lower priority bugs. The truth is that the time to close bugs depends on many factors. For instance, more complex bugs in spite of having a higher priority, may take much more time to be fixed than trivial bugs (that will have probably a very low priority). But overall, when considering the entire population of bugs, we can assume that the factor with the most impact on the time to close a bug is its priority.

With these assumptions, we classified bugs according to the time it took to close them and obtained five groups, classified according to the priority assigned by the developer.

Figure 2 shows the results. Again, the horizontal axis shows the time interval from the notification of the bug to when the bug is closed, measured in months. Each label in that axis corresponds to one of the levels of priority (from P1 to P5, in Bugzilla). The values shown are the mean times for each group of priority. Using the LSD parameter, we have three classes of bugs: the first class contains P3 and P1, the second P2 and P5, and the third class only P4. Again, we have found only three different classes of bugs. The mean time to close values does not match the values of the previous classes though.

It is peculiar that P3 has a shorter time to close than P1 and P2. We consider the possibility that some developers might be using priorities in different ways (some developers might never use P1 and P2, for example) and decided to plot the mean time to close for the four developers who have closed the most defects. Figure 3 shows the results: only the second top developer is using the five levels of priority. The rest are using only two or three levels.

5. CONCLUSIONS AND FURTHER WORK

We believe that the report forms of Bugzilla (the bug tracking system used by Eclipse) are too complex. There are too many fields, and for each field, too many options, and that some of these options can be removed without affecting the way bugs are handled.

In this study, we have focused in the fields for severity and priority. We demonstrate that the severity of defects can be reduced from seven options to three, and that priority can be reduced from five options to three.

When properly used the severity field provides valuable information to the developers. Unfortunately not every bug reporter is capable of using the current classification. Perhaps a solution is to leave this responsibility to a bug master: the report classifies the bug based on three categories (important, non-important, and request for enhancement), and the bug master further classifies the bug using the current seven.

While developers are using only three priorities, not all developers are using them in a consistent manner. Some never use priority 1 and 2, and some never use 4 and 5. This can lead to confusion. We strongly recommend that the number of priorities is reduced to 3: high, medium and low.

We have also observed that there are two different patterns of use of the priority field: few developers use them to subclassify the severity field (they will classify each of the critical types with P1, P2... etc; effectively ranking how each of the bugs in each severity should be handled); while others use them independently of the severity field (P1 or P2 will always be used for highly critical bugs and P4 or P5 will always be used for those with a very low severity). This lack of uniformity might be confusing if one defect has to be passed from one developer to another, and should be addressed.

There is at least another practical implication. Bugzilla designers tried to make Bugzilla universal. In order to fulfill this goal, they created many different levels of severity, priority, etc, and created many possible fields for the report forms. But people in Eclipse are not using them. Bugzilla designers might not be getting any feedback on how other projects use their system. For instance, if Bugzilla developers would find that most of the bugs severity may be labeled only with three categories, they would have changed it time ago (or at least make it customizable).

It is also interesting to mention that the defect tracking system of FreeBSD has only three levels for severity and three for priority. We would like to get access to the Bugzilla databases of several Mozilla Foundation projects (the original intended users of Bugzilla), and any other project using it as its defect tracking system to evaluate and compare how they use the bug severity and priority fields. However, bug tracking databases are not usually made available to researchers and third parties. This is unfortunate. Making those databases available would help understand how they are used, and, as this paper does, suggest improvements.

6. REFERENCES

[1] J. Maindonal and J. Braun. *Data Analysis and Graphics using R*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2006.

[2] E. Raymond. *The Cathedral & the Bazaar*. O'Reilly, 1999.

[3] L. Villa. How gnome learned to stop worrying and love the bug. In *Talk at the Otawa Linux Symposium*, Otawa, July 2003. http://tieguy.org/talks/OLS-2003-html/.

[4] L. Villa. Why everyone needs a bugmaster. In *Talk at linux.conf.au*, Canberra, April 2005. http://tieguy.org/talks/LCA-2005-paper-html/.