Operating System Compatibility Analysis of Eclipse and Netbeans Based on Bug Data

Xinlei (Oscar) Wang
xlwang@ucdavis.edu

Eilwoo Baik
ebaik@ucdavis.edu

Premkumar T. Devanbu
devanbu@cs.ucdavis.edu

Department of Computer Science
University of California, Davis
Davis, CA 95616

ABSTRACT

Eclipse and Netbeans are two top of the line Integrated Development Environments (IDEs) for Java development. Both of them provide support for a wide variety of development tasks and have a large user base. This paper provides an analysis and comparison for the compatibility and stability of Eclipse and Netbeans on the three most commonly used operating systems, Windows, Linux and Mac OS. Both IDEs are programmed in Java and use a Bugzilla issue tracker to track reported bugs and feature requests. We looked into the Bugzilla repository databases of these two IDEs, which contains the bug records and histories of these two IDEs. We used some basic data mining techniques to analyze some historical statistics of the bug data. Based on the analysis, we try to answer certain stability-comparison oriented questions in the paper, so that users can have a better idea which of these two IDEs is designed better to work on different platforms.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Performance, Reliability

Keywords

Eclipse, Netbeans, Bugs, Operating Systems, Stability

1. INTRODUCTION

Think of Integrated Development Environments (IDEs) for Java development, the two names that will come up are Eclipse and Netbeans. Both Eclipse and Netbeans provide support for a wide variety of development tasks such as refactoring and debugging on the three most commonly used operating systems, i.e., Windows, Mac OS and Linux. Both IDEs are programmed in Java and use a Bugzilla issue tracker to track reported bugs and feature requests. In this paper, we are going to look into the bug database of the Eclipse and Netbeans and try to answer the question of which IDE has better compatibility with different operating systems (Windows, Linux and Mac OS).

To answer the above question, we split it into a few sub-questions. The first sub-question is, which IDE is more stable and less failure-prone on each of the three operating systems. The second sub-question is, among all the issues on each operating system, which IDE has larger proportion of high-severity (major, critical and blocker) issues. Third, which IDE's bug fixes are easier to break again and finally which IDE's bugs require more time or more number of actions to finally get fixed.

Windows, Linux and Mac OS are the three most commonly used operating systems. Most of the time, users of Eclipse and Netbeans do their development work on these operating systems. Some users even use two or all of them together. We will answer each of the above questions based on these three different operating systems. By answering these questions from the users' perspective, we will see which IDE, on which operating system, there is lower probability to encounter a bug in general and a severe bug in particular. From the developers' perspective, we will see which IDE, on which operating system, the bugs are relatively more complex and require more work to get fixed. The architectural design of an operating system is different from one another. If a software engineer wants to design an application which runs on different operating systems, he/she must put the differences of the operating system designs into consideration. Therefore, from a general software engineer's perspective, we can see which IDE is designed better to run on the different operating systems.

2. DATA AND METHODOLOGY

The most important and probably also the most straightforward data to be extracted is the number of bugs existing in the database for each IDE. Therefore, we look into the number of bugs of each IDE on the three different operating systems respectively first. That is, we look at the tables that contain the bug records in the Eclipse and Netbeans bug database. We also examine when each of the bugs is raised in these bug-record tables, so that we know how the number of bugs changes over time for each IDE on different operating systems. In addition to the total number of bugs, we also try to get the number of bugs raised per reporter. For this, we count the distinct reporter records in the bug-record tables. Furthermore, we investigate the distribution of reporters who reported different amount of bugs.

MSR'11, May 21–22, 2011, Waikiki, Honolulu, HI, USA

Next, we investigate on the severity of the bugs in the database, in order to know what proportion of the bugs have higher severity (major, critical or blocker) for each of the IDEs on different operating systems. For this investigation, we extract the severity field in the bug-record table for Eclipse and priority field in the bug-record table for Netbeans. One problem is that the two IDEs have different notions of "severity levels", as in Eclipse has seven severity levels (blocker, critical, major, normal, minor, trivial and enhancement), but Netbeans only has five levels of priority (1-5). For simplicity, we categorize the blocker, critical and major issues as the high-severity issues for Eclipse and issues with priority 1 and 2 as high-severity issues for Netbeans.

Subsequently, we examine how reliable the bug fixes actually are, i.e., what percentage of the bugs was reopened after being fixed in each IDE on different operating systems. It gives an idea of the how many bugs were not fixed properly for the first one or more times. There could be some recent fixes which would be reopened which are not captured in the database we are looking at. We assume the number of these potential reopen bugs would be minimal compared to the other old fixes.

Finally, we look at the turnaround time for the bug fixes, i.e., how long it takes to finally get a bug fixed. Besides the time, we also consider the number of bug activities between when a bug is raised and when the bug is finally closed. For both the turnaround time and activity count, we get both the average and median results for every IDE on every operating system.

For all the questions, we investigate the bug data separately according to different operating systems. One problem is how to determine on which operating system the bug was reported. This is straightforward since bug reports have a field indicating the applicable operating system. Some bugs have "All" in the OS field, so we consider them as appeared across all operating systems. For other bugs that have a specific value in this field, we consider them as only appeared on that particular operating system. For all the analysis, we consider bugs with "All" or "Windows *" (where * can be XP, Vista and etc) as Windows bugs. Similarly, we consider bugs with "All" or "Linux *" as Linux bugs and bugs with "All" or "Mac *" as Mac bugs.

3. ANALYSIS AND DISCUSSION

In this section, we show the statistics we get from extracting and analyzing the bug data. Based on the statistics, we will discuss the underlying indications of our observations.

First, though it is straightforward to get the total number of bugs reported in the two IDEs, it is the most important indication of how stable the IDEs are on each of the operating systems. Here we categorize the bugs according to the year in which they were reported. The time period we looked at is from 2002 to 2008 because this is the overlapped period for the two IDEs in the bug dataset we are looking at. From the trend of bug numbers, we can analyze which IDE is getting better on which operating system.

For Eclipse, we can see from Figure 1 that the number of bugs on Windows gets sharply increased in 2004 and 2005 and then gets stabilized after that. On Linux and Mac, the number of bugs is fairly stable, but increasing slowly up to 2008. This suggests that Eclipse has reached a relatively stable stage on Windows. However, on Linux and Mac, more issues are coming up until 2008 and Eclipse is in the stage of getting stable. For Netbeans, Figure 1 shows us that the numbers of bugs on the three operating systems follow the same trend: keep increasing from 2002 to 2007 but drop

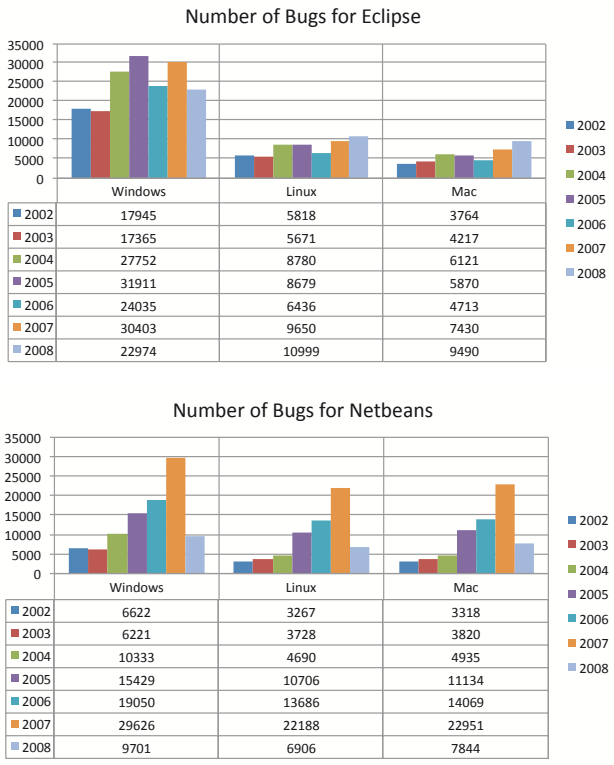

Figure 1: Number of Bugs Reported on Different OS's over Time

sharply in 2008. The reason of the drop in 2008 is that the data set does not have the data for the whole year of 2008. Instead, we only have the data up to May 2008. Therefore, we expect the number of bugs reported in 2008 would not decrease so much. Our observation suggests that the numbers of bugs on all three operating system for Netbeans increase quickly up to 2008 and hence it is not as stable as Eclipse from this perspective.

Alternatively, someone may argue that the total number of bugs reported depends much on the number of system tester or bug reporters who are working on each of the platforms. More reporters would mean that more bugs could be discovered and reported. Therefore, we also extract out the number of reporters worked on each platform in each year and averaged the data in Figure 1 to get the per-reporter results which is shown in Figure 2. First, we can see in this figure that Netbeans has more number of bugs reported per reporter on average, especially on Linux and Mac. Hence, again this suggests that Eclipse is preferable than Netbeans in terms of stability.

In addition to the above results that show the number of bugs, to compare the stability of the two IDEs on different platforms, we also examine the distribution of reporters who reported different amount of bugs. Assuming the reporters' average ability of finding bugs is the same for both IDEs, we argue that if more people discovered large amount of bugs, that means the IDE is less stable. The Figure 3 shows the percentage of reporters that reported large amount of bugs. We can easily see that the ratio of reporters who reported 100 to 500 bugs for Netbeans is larger than for Eclipse. More obvious same results can be observed for the ratio of reporters who reported more than 500 bugs. Within the IDEs

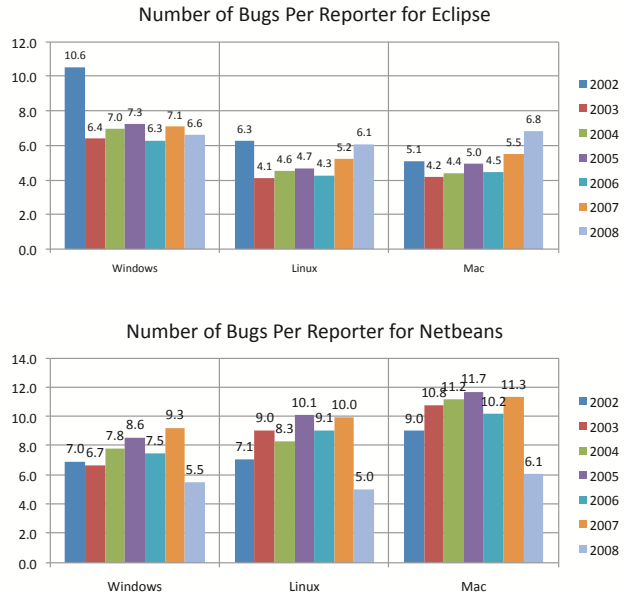

Figure 2: Number of Bugs Reported Per Reporter on Different OS's over Time

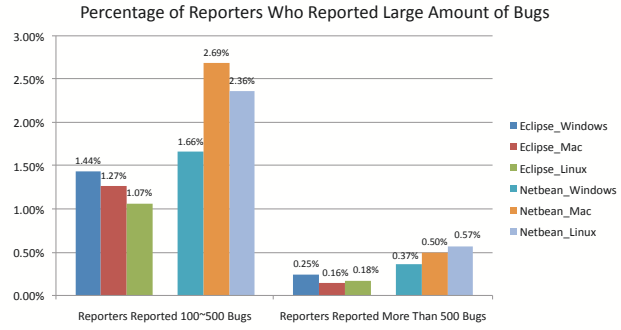

Figure 3: Percentage of Reporters Who Reported Large Amount of Bugs for Each IDE on Different OS's

themselves, Mac OS gets the best results for Eclipse whereas Windows gets the best results for Netbeans.

Next, we look at what percentage of bugs fall into the high severity category for each IDE on different operating systems. From Figure 4, we see the overall percentage of high-severity bugs in Netbeans is much higher than in Eclipse. But within each IDE, the difference between the operating systems is not significant. In particular, for Eclipse, Windows has the highest percentage and Mac has the lowest percentage of high-severity bugs. For netbeans, Mac has the highest percentage and Windows has the lowest percentage of high-severity bugs. Based on these observations, we think that Eclipse is more stable, as in high-severity bugs do not appear as frequently as in Netbeans, possibly because due to the different architectural designs of the two IDEs from a software engineer or a developer's point of view.

Subsequently, we compare the percentage of bugs that have been reopened after being fixed for each IDE on different operating systems. From Figure 5, again, we see that Eclipse outperforms Netbeans in that Eclipse has lower over-

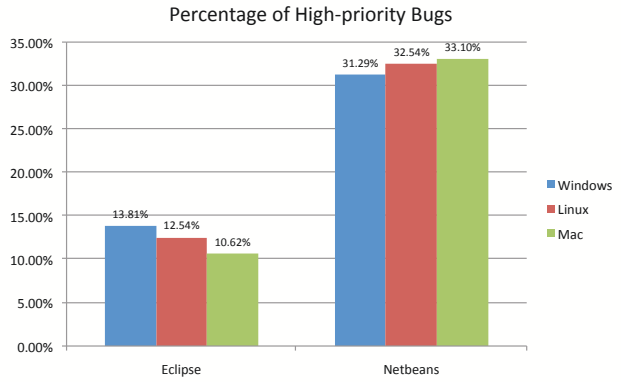

Figure 4: Percentage of High-severity Bugs for Each IDE on Different OS's

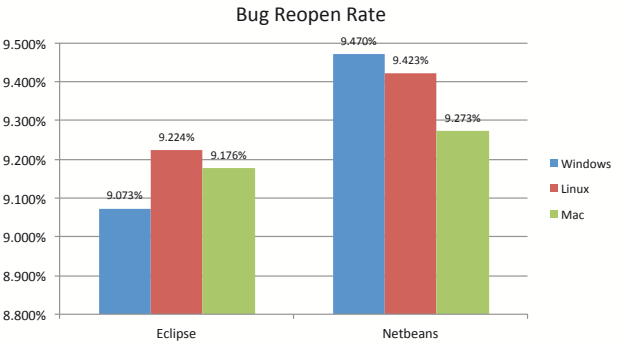

Figure 5: Bug Reopen Rate for Each IDE on Different OS's

all bug reopen rate. Looking at Eclipse alone, Windows has the lowest bug reopen rate and Linux has the highest. For Netbeans, Mac has the lowest bug reopen rate and Windows has the highest. Hence, overall speaking, Netbeans' bugs have more tendencies to break again after being fixed, especially on Windows. In a software development or maintenance process, most of time bugs get reopened due to new fixes on other parts of the system. Therefore, a reasonable inference from this is that Netbeans' design is not as sophisticated as Eclipse, so that fixes on one part has higher probability to break other parts of the system. However, the difference here is not significant based on the statistics.

Finally, we examine how much time and how many bug activities it takes to fix a bug for the two IDEs on each of the three operating systems. Figure 6 shows the average and the median of the number of activities it takes to fix a bug for the two IDEs on each operating system. Figure 7 shows the average and the median of the number of days it takes to fix a bug for the two IDEs on each operating system. From Figure 6, we can see both the average and median values are approximately the same. There is not much difference between the two IDEs and even among the operating systems. However, when we look at Figure 7, we observe that bugs of Eclipse need much less time (number of days) to be fixed compared to bugs of Netbeans. For both Eclipse and Netbeans, bugs on Windows take slightly less time to be fixed compared with the other two operating systems and Linux ranks as the second. Since these are

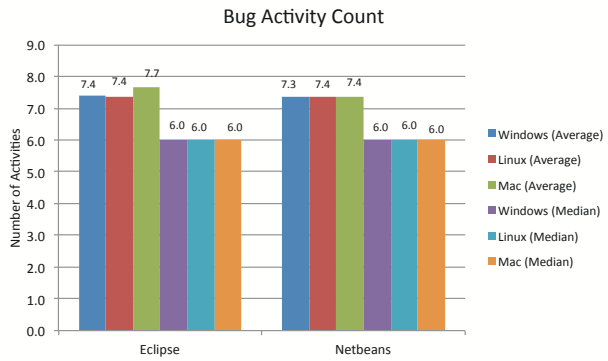

Figure 6: Bug Activity Count for Each IDE on Different OS's

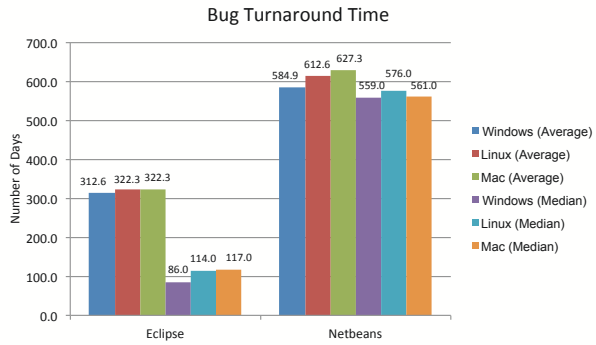

Figure 7: Bug Turnaround Time for Each IDE on Different OS's

comparable projects in terms of developer community (both quasi-commercial), these observations suggest that Eclipse is easier to maintain than Netbeans. To developers, on average, bugs on Netbeans are more complex to fix compared with bugs on Eclipse.

4. RELATED WORK

To improve software productivity and quality, software engineers are increasingly applying data mining algorithms to various software engineering tasks. Various algorithms have been presented to effectively mine sequences such as execution traces collected at runtime, static traces extracted from source code, and co-changed code locations, graphs such as dynamic call graphs collected at runtime and static call graphs extracted from source code, and text such as bug reports, e-mails, code comments, and documentation [3]. In this paper, we focus on bug reports only.

Bug reports are vital for the development and quality control of any software. They allow users to inform developers of problems that they encountered while using the software. A bug report typically contains a detailed description of a failure and occasionally pointers to the location that contains the fault (in form of patches or stack traces). However, according to the paper in [1], bug reports often provide inadequate or incorrect information and developers have to face bugs with descriptions such as "wqqwqw" (Eclipse bug 145133), just "layout" (Eclipse bug 52050) or "Login"(Eclipse bug 178041). It is no surprise that developers are slowed down by poorly written bug reports because identifying the problem takes more time. The authors put attention in making bug report quality from the point of view developers. From this paper, the Eclipse developers classified 100 bug reports on a scale from one (poor quality) to five (excellent quality). Based on the results from the survey, they constructed a measure of bug report quality that was able to classify 42% of the bugs correctly. In our paper, we do not consider the quality of bug itself but use bug statistics to measure the quality of the software.

In [2], the purpose is computing the bug-fix time ArgoUML (period 1/2002 - 3/2003) and PostgreSQL (period 07/1996-11/2000) and reporting bug-fix time statistics in order to figure out how the bug-fix time can be used as a factor for related analysis. In order to setup standard time from when the bug is introduced to when the bug is finally fixed and released, keyword-based change log search is used to identify bug fixes per file. The authors identify bug-introducing changes by applying fix-inducing change identification algorithms in order to obtain the commit time of the identified bug-introducing changes and their corresponding bug fixes from project histories. From the commit times, each bug-fix time and the average bug-fix time of each file are computed. From this, we introduced one of our basic concepts - turnaround time, meaning time from when a bug is reported and when it is fixed, which is computed in terms of both number of days and number of bug activities.

5. CONCLUSION

In this paper, we have examined the bug datasets of two open source Java IDEs, Eclipse and Netbeans, based on the three most commonly used operating systems, Windows, Linux and Mac OS. From all our findings, we can see Eclipse outperforms Netbeans in every comparison we have investigated. However, within each IDE themselves, their performance on different operating system does not have much difference. For Eclipse, the bug data on Mac OS shows a slightly preferable result than on the other operating systems. Therefore, in conclusion, our data suggests that from the point of view of bug report and repair history, Eclipse appears to be a better platform; it also suggests that users of Eclipse on Mac OS have a better experience than users on other platforms in terms of system stability. However, these findings have alternative explanations. It's possible that Netbeans users and Windows users are more finicky and prone to complain, where Eclipse and Mac OS users are loyalists, who "grin and bear it". Also, our work has only looked at the system stability aspect and there are other factors which affect the user experience which our study does not take into account. For example, due to Cocoa issues, Eclipse on Mac appears to be slower than other platforms, thus offering a worse user experience.

6. REFERENCES

[1] N. Bettenburg, S. Just, A. Schroter, C. Weiß, R. Premraj, and T. Zimmermann. Quality of bug reports in Eclipse. In *Proceedings of the 2007 OOPSLA Workshop on Eclipse Technology Exchange*, pages 21–25. ACM, 2007.

[2] S. Kim and E. Whitehead Jr. How long did it take to fix bugs? In *Proceedings of the 2006 International Workshop on Mining Software Repositories*, pages 173–174. ACM, 2006.

[3] T. Xie, S. Thummalapenta, D. Lo, and C. Liu. Data mining for software engineering. *IEEE Computer*, 42:35–42, 2009.