STUDENT PAPER: On Reliability Analysis of Open Source Software - FEDORA

Prasanth Anbalagan\(^1\) Mladen Vouk\(^2\)
Department of Computer Science, North Carolina State University, Raleigh, NC 27695, USA
\(^1\)panbala@ncsu.edu \(^2\)vouk@ncsu.edu

Abstract

Reliability analyses of software systems often focus only on the number of faults reported against the software. Using a broader set of metrics, such as problem resolution times and field software usage levels, can provide a more comprehensive view of the product. Some of these metrics are more readily available for open source products. We analyzed a suite of FEDORA releases and obtained some interesting findings. For example, we show that traditional reliability models may be used to predict problem rates across releases. We also show that security related reports tend to have a different profile than non-security related problem reporting and repair.

1 Introduction

Reliability and availability analysis of a widely deployed system typically requires data such as failure intensity, downtime, etc., for each system in the field [1]. This information is often not available for proprietary systems, and may be available in a different form for open source systems. In the case of FEDORA, one could glean some of the software quality information from the FEDORA problem report(PR) databases. Unique problem reports for FEDORA were collected from the REDHAT’s bug tracking system\(^1\). The unique problem reports associated with vulnerabilities were identified with the help of the Common Vulnerability Exposure\(^2\) database and National Vulnerability Database\(^3\). We estimated system usage based on the FEDORA downloads statistics. Assuming that a download also represents an operational system, we can estimate in-service time as the product of the number of systems and the time they have been operational [1].

Figures 1, 2, 3 and 4 show the Time-to-Problem-Report (TTPR) and Time-to-Problem-Correction (TTCP) for non-security problem reports in FEDORA releases 6 and 7 along with their release dates. The horizontal axis shows calendar time at which the problems were reported, and the vertical axis shows the TTPR and TTCP. The plots also show the burn in period (time period during which the software is developed and is not yet officially released).

From the figures, we see that, as the project progresses from the burn in period into the release date, TTPR decreases. This may be caused by an increased number of

\(^{1}\)http://bugzilla.redhat.com/
\(^{2}\)http://cve.mitre.org/
\(^{3}\)http://nvd.nist.gov/
users reporting more after an official release of the product. However, after several months into the release, there is an increase in TTPR, possibly indicating that product is getting better. The trend continues until the next release is made, and even for a small period after the next release. This may be due to further improvements, or because users may be moving to the new system. With problem corrections, we find that TTPC is initially longer and reduces as the project advances to the release time. After the release, we find that problems are corrected more rapidly. At the point of a next release, we find that problems in older releases appear to be almost completely resolved. Figures 5, and 6 show TTPR (for unique PRs) and TTPC respectively for security problem reports for FEDORA release 6. Interestingly time to security problem report appears to be relatively constant.

We can model problem reporting and repair trends along the lines discussed in [1]. Let the problem report rate ($\lambda$) be estimated by:

$$\lambda = \frac{n_o + n_f}{N \times \Delta t}$$  (1)

where $n_o$ represents number of open bugs from the previous week, $n_f$ represents number of problems reported in time interval $\Delta t$ (e.g., in minutes), and $N$ is the number of systems active in that interval (e.g., in a week expressed in minutes). It is important to note that this estimate may be lower than the true field failure or problem reporting rate since only unique problem reports are part of the TTPR count. Musa’s Logarithmic Poisson Execution Time (LPET) model was applied to FEDORA 6 data to model the problem report rate. Figures 7 and 8 indicate a growth in the quality of the releases over in-service time, and they also show that parameters obtained for FEDORA 6 can be utilised to predict the problem report rate for FEDORA 7. Similar results were obtained for other release chains and both security and non-security categories.

2 Conclusions

We have studied problem reports and corrections for FEDORA releases, identified trends across releases, and showed that it is possible to use traditional reliability models to predict problem report rates across releases. This may be useful when assessing user trust in open source software, assessing its possible availability and vulnerability, as well as in planning when to make a transition to a new system.

References