International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.3 (2014), pp.375-378
http://dx.doi.org/10.14257/ijmue.2014.9.3.36

An Entropy based Method for Defect Prediction in Software Product Lines

ChangKyun Jeon, Chulhoon Byun, NeungHoe Kim and Hoh Peter In

Department of Computer Science, Korea University,
Seoul, South Korea

{Curt_jeon, damby0, nunghoi, hoh_in}@korea.ac.kr

Abstract

Determining when software testing should begin and the number of resources that may be required in order to find and fix defects are complicated decisions. If we can predict the number of defects for an upcoming software product given the current development team, it will enable us to make better decisions. A majority of reported defects are managed and tracked using a defect life cycle, which tracks a defect throughout its lifetime. The process starts when the defect is found and ends when the resolution is verified and the defect is closed. Defects transition through different states according to the evolution of the project, which involves testing, debugging, verification. In paper, we presents defect prediction model for consecutive software products that is based on entropy.

Keywords: *Defect prediction, defect life cycle, Entropy, product line engineering, software engineering*

1. Introduction

Consumer electronics (CE) software products require high performance, stability, and multi-functional features. Thanks to software product lines, many CE products are able to satisfy these requirements. A software product line is a proactive and systematic approach to the development of software that allows for the creation of a variety of products [1]. Most software product lines are designed using a platform that serves as the basis for a family of products and they rely on an a priori architecture and artifacts from the platform products.

However, many development and test teams are needed in order to develop a consecutive product line. Inevitably, a large number of defects are found and are reported to the proper team during the stages of the development and testing process, such as the development phase, unit testing, acceptance testing, and the production readiness phase.

The reported and confirmed defects are fixed by the development team or are assigned a priority based on different reasons such as severity of the problem, strategy purpose, and high (rare) rate occurrence. Unfortunately, some of defects remain in an unresolved state until the end of the project owing to lack of clues or information.

We believe that it is possible to analyze the historical trends in defect tracking systems and to use the resulting information to predict the number of defects in upcoming software products and to obtain a concrete view of the life cycle of potentially unreported defects. This information will not only provide better understanding to project managers for making decisions at key milestones, but can also be used as one of the metrics for evaluating the performances of development teams and individual developers.

ISSN: 1975-0080 IJMUE

The model for predicting the life cycle of defects with occurrence rates and severities within specific domains could be used as a reference prediction model for the platform-based series of software products. In this paper, we present a defect prediction model using entropy after the analysis of repository data from a defect tracking system.

2. An Entropy based Defect Prediction Model

The defect life cycle is the cycle that a defect goes through during its lifetime. The cycle starts when a defect is initially found and ends when it has been retested, resolved and finally closed. Bugzilla, Git, and Jira are examples of defect tracking systems that are currently used for tracking the histories of defects [2]. There are some minor differences between defect tracking systems, but the majority of the defect transition statuses are shown in Figure 1.

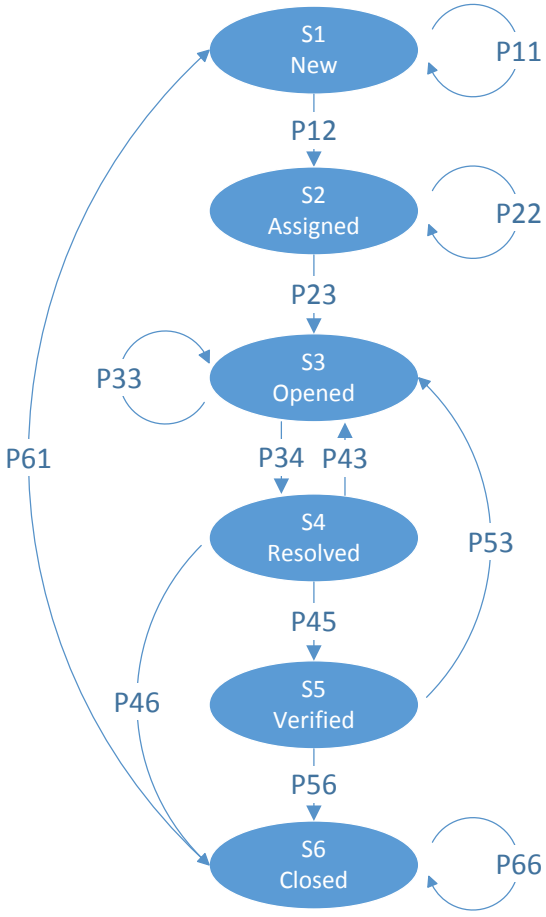

Figure 1. Defect life cycle transition diagram

The defects transit between the various statuses during their life cycles. When a bug is found for the first time, the tester needs to check to see if it is a valid defect or not. After the confirmation is completed, the defect is submitted to the defect tracking system with a status of "New". The defect is forwarded automatically to the designated person in the software development team and he/she changes it to the "Open" status in order to indicate that action is being taken to find a solution and to assign it to the right person. Once the defect has been assigned to the correct developer, it transitions to the

"Assign" state. When the developer has figured out the root cause and has found a solution, the status is changed to "Resolved" and the software is released in order to allow the tester to validate it. Once the solution has been verified, the tester closes the defect by changing it to the "Closed" status. However, in cases where the same defect reoccurs or closely related quality issues are discovered, the tester reopens the bug and moves it to the "Reopened" status. Following the transition statues, the impurity function measures the extent of purity for a region containing data points from possibly different classes. The information gain measurement is based on the entropy function derived on the basis of the information theory [3]. It may compute an entropy-based score quantifying the transition of defects statuses. The entropy to measure the probability of transition as follows (1):

$$entropy(D) = -\sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$
$$\sum_{j=1}^{|C|} \Pr(c_j) = 1, \tag{1}$$

3. Conclusions

In this paper, we presented a defect prediction model that predicts how many defects will occur during the development of consecutive software products. The Entropy that was used in our study has been used widely and has proven to be effective for various kinds of information processing. Defect prediction is a tool that can be used to improve the management of software development efforts. One of the most visible advantages of using defect prediction is the ability to predict the trends and directions of defect cycles in software projects. Therefore, we believe that the proposed model enables managers and project leaders to detect trends and to anticipate problems, thereby providing better control of resources, reducing risks, improving quality, and ensuring the success of development objectives, such as time to market. The experimental results demonstrate the effectiveness of the proposed model. We also believe that our model gives a high-level view of the software defects in the upcoming phases of development and how they need to be managed and handled.

In the future, we will carry out the case studies based on the proposed model using a series of software products that are based on the platform product.

Acknowledgements

This research was supported by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2012M3C4A7033345), and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012R1A1A2009021)

References

[1] P. Clements and L. Northrop, "Software Product Lines, Practices and Patterns", Addison-Wesley, (**2002**).
[2] Git: http://git-scm.com/ Bugzilla: http://www.bugzilla.org/ Jira : http://jira.dspace.org
[3] E. Shannon, "A mathematical theory of communication," Bell System Technical Journal," vol. 27, **(1948)**, pp. 379–423.