Local and Global Recency Weighting Approach to Bug Prediction

Hemant Joshi[+], Chuanlei Zhang[+], S. Ramaswamy[*], Coskun Bayrak[*]
[+]Applied Science Department, [*]Department of Computer Science
University of Arkansas at Little Rock, Little Rock AR 72204
{hmjoshi, cxzhang, srini, cxbayrak}@ualr.edu

Abstract

Finding and fixing software bugs is a challenging maintenance task, and a significant amount of effort is invested by software development companies on this issue. In this paper, we use the Eclipse project's recorded software bug history to predict occurrence of future bugs. The history contains information on when bugs have been reported and subsequently fixed.

1. Introduction

To remain competitive in the fast growing and changing world of software development, project managers must optimize the use of their limited resources to maximize the delivery of quality products on time and within budget [1]. Software bug repositories have been recognized as reliable data assets that can provide useful information to assist with the software development and project management process. Therefore, in this work, we have modeled the occurrence of bug patterns through the characterization of information stored within bug repositories. We use the proposed approach is to predict the number of bugs and types of bugs reported for the Eclipse project for a specified period, as required by the proposed challenge in [2].

The remaining part of the paper is organized as follows. In section 2, we discuss the information source for our input data. The model used for the bug predictions is described in Section 3. The prediction results obtained using our technique, are presented in Section 4. Section 5 concludes the paper.

2. Input Data

Although over 70,000 bugs associated with the Eclipse project over the past 6 years were downloaded, we were particularly interested in the 32 components of Eclipse, as specified in the challenging task list [3]. More specifically, the model was applied to Eclipse bugs repositories for the MSR challenge [2]. The bug-count for each of these 32 components was calculated for each month, from January 2001 to January 2007 (73 months).

3. Model for Prediction

The proposed model presented in this paper was built on two bug pattern assumptions. The first is based on the assumption that the number of bugs or bug-count reported for any given component in a given month *i,* is most strongly impacted by the bug-counts reported for the previous month *i-1*. This can be represented as:

$$\text{BugCount}_i \propto \text{BugCount}_{i-1}$$

This indicates that the bug prediction for any month is largely influenced by the bug-count of the preceding month. To model this, we introduce the stabilizing factor, α; which is a measure of how much the predicted bug-count deviates from the previous month. Hence,

$$\text{BugCount}_i = \alpha_i + \text{BugCount}_{i-1} \quad \text{-- (1)}$$

Here the factor α is used as an offset adjustment to predict how much the bug-count will vary in the month under consideration from its most recent predecessor. In order to determine the factor α, we take into account the bug-count history of all available years and use local corrections. Thus, we study this bug prediction problem by analyzing the historical bug data both locally and globally.

In our analysis, we refer this as 'recency-weighting' approach in which weights are arranged in a manner that boosts the weight of the most recently known monthly bug-count for that component. To model recency, we use a power decay function (with base 2) given by

$$f(x) = \frac{1}{2^x} \text{ where } x \geq 0 \quad \text{-- (2)}$$

Then, we calculate the average weighted function W(x) combined with generic bug-count trend (*slope, m*) for each month *i* from January 2001 until January 2007. Readers can refer to [4] for further details of this approach.

$$W_i = \frac{\sum_{i=72}^{0} m_i * \frac{1}{2^i}}{73} \quad \text{-- (3)}$$

where slope $m_i = \text{BugCount}_i - \text{BugCount}_{i-1}$

29th International Conference on Software Engineering Workshops(ICSEW'07)
0-7695-2830-9/07 $20.00 © 2007 IEEE

Although the above global analysis might shed significant light on understanding the overall bug-count trend, it does not include any information about trends (local) that might be specific to a particular month. As the bug-count is the 'reported' bug-count, it may be assumed that the bug-count gets affected by seasonal factors such as holidays, vacations, allergies, or some other local factors. So we add a local analysis factor in our model. Since we had to predict for the months of February, March and April 2007, we considered all the months of February, March, and April over the past 6 years to predict the number of bugs for February, March, and April in 2007. Hence, we considered six weight functions (Wj) for each component under consideration, to calculate the average local weight factor as shown in equation (4).

$$W = \frac{\sum_{j=local-2001}^{local-2007} W_j}{6}, l = Feb, Mar, Apr \text{ -- (4)}$$

Applying the previous year-month's bug count offset to predict current month-year bug-count for the given component is a viable approach, since the local trends specific to a month of the year may be expressed in this case. The implicit bug-count predicted also provided a higher weight for the previous two years than other prior (month) years.

4. Prediction Results

Table 1 shows the bug-count predictions for the 32 components of Eclipse dataset. The duration is 3-month period between February 2007 and April 2007.

Table 1. Bug Prediction Results for 32 Eclipse components [period Feb 2007 to Apr 2007]

Eclipse Component	Alpha (Global)	Beta (Global & Local)
Equinox.Bundles	36	36
Equinox.Framework	105	24
Equinox.Incubator	15	3
Equinox.Website	0	0
JDT.APT	15	15
JDT.Core	217	326
JDT.Debug	111	122
JDT.Doc	12	12
JDT.Text	66	105
JDT.UI	235	258
PDE.Build	27	37
PDE.Doc	6	2
PDE.UI	268	268
Platform.Ant	18	44
Platform.Compare	57	32
Platform.CVS	87	87
Platform.Debug	127	177
Platform.Doc	12	12
Platform.IDE	129	81
Platform.Releng	72	39
Platform.Resources	63	55
Platform.Runtime	57	57
Platform.Scripting	0	0
Platform.Search	3	6
Platform.SWT	315	288
Platform.Team	51	158
Platform.Text	72	76
Platform.UI	725	651
Platform.Update	57	57
Platform.UserAssistance	0	0
Platform.WebDAV	0	0
Platform.Website	3	3

At the time of competition, we had downloaded only 70,000 bugs but since then we have downloaded over 162,000 bugs for various components of Eclipse. We compared the prediction results with the actual bug-count for months of Dec 2006 and Jan 2007. The model performs better with 162,000 bugs than with 70,000 bugs for the two time periods.

5. Conclusion

In this work, we have presented an elegant approach to the prediction of bugs for a software project based upon the information contained in open-source bug repositories. We have illustrated the use of a local and global recency weighting technique for the design of a bug prediction algorithm. While we do encounter some extreme cases in the results, the overall results obtained are, in general, promising. In the future, we plan to address predicting for a more distant future by considering issues such as software versioning impacts. We also plan to study related issues such as change and stability analysis.

6. References

[1] A. E. Hassan and R. C. Holt, "The top ten list: dynamic fault prediction," Proceedings of the 21st International Conference on Software Maintenance, Budapest, Hungary, September 2005, pp. 263–272.

[2] MSR. Mining Challenge 2007. http://msr.uwaterloo.ca/msr2007/challenge/

[3] MSR. Mining Challenge 2007 Bug Prediction. http://msr.uwaterloo.ca/msr2007/challenge/#predict

[4] C. Zhang, H. Joshi, S. Ramaswamy, C. Bayrak, L. Yu, "A Dynamic Approach to Software Bug Estimation", *submitted to International Conference on Global Software Engineering*, Feb 2007.