

ANALYSIS OF OPEN SOURCE DEFECT TRACKING TOOLS FOR USE IN DEFECT ESTIMATION

Catherine V. Stringfellow, Dileep Potnuri
Department of Computer Science
Midwestern State University
Wichita Falls, TX U.S.A.

Abstract - There are several defect-tracking tools developed by the open source computing community, which are free and available to download from the internet. This paper evaluates seven free and open source defect tracking tools, based on how easy they are to install and use, the data they provide, and the ability to retrieve data from them. These tools may provide data that are also useful in estimating remaining defects in software to aid system test managers in making decisions regarding whether testing should continue or the software is ready for release. This paper determines which of these defect-tracking tools are best for the purpose of making defect estimates.

Results show that four open source defect tracking tools, besides being easy to use for defect tracking, also provide data that may be used with at least one defect estimation method.

Keywords: software tools, defect analysis, metrics

1.0 Introduction

In any software development process the software product should be tested before releasing it into the market. Exactly when testing should stop is very difficult to decide. Testing may stop when testing yield saturates, where yield is defined in terms of coverage elements found or number of faults exposed. Normally, however, marketing goals determine the release date. It would be helpful if testers could use defect data to determine the right point at which to stop testing and release software. Several researchers have developed a number of models and techniques to estimate the number of defects left in the software using defect data from industry's in-house defect reporting tools [1, 2, 3, 4, 5, 6].

Several defect-tracking tools developed by the open source computing community are free and available to download from the internet. There is a broad range in quality of code, documentation and testing of open source products [7]. A goal of this paper is to determine whether certain tools used to track defects are easy to install, learn and use. In addition, this paper analyzes these free and open source defect tracking tools to determine if the data they contain can also be used by a system test manager to determine whether testing should continue. The tools analyzed include Buggit, Bugzilla, Bugtrack, Roundup, Mantis, Incident Management Systems (IMS) and Bugs Online [8, 9, 10, 11]. Some of these tools are Linux compatible and some are Windows compatible. This study evaluates these defect tracking tools based on the effort to install and use the tool, the ability to extract data from the tool, as well as the type of data they contained.

2.0 Background

The ability to predict the remaining number of defects in a release has the potential to aid software test managers in making release decisions. Several approaches use defect data to estimate defect content.

2.1 Defect estimation using static models

Yu, et al. [12], construct a mathematical model that may be used to predict the number of defects found later in post-release, using the number of errors detected in earlier phases of the software development cycle. They found a correlation between the number of earlier defects and that of later ones. Biyani, et al. [13], examine the relationship between the errors discovered during development of a product release, as well as faults in previous releases, to those that escape to the field. Based on the actual data from four releases of a commercial application product consisting of several thousand modules, they show that the prior release is sufficient for predicting the number of errors during development or in the field and that modules with more errors in development have a higher probability of failure in the field.

Various statistical models [1, 2, 5, 14] use process data available only from the current project to estimate the defect content. Two types of models that can be used for this approach are the capture-recapture and curve fitting models [1,3, 5, 14]. Capture-recapture (CRC) models use the overlap and non-overlap of defects discovered between independent testers to estimate the remaining defect content [1, 2, 3, 15]. The various CRC models include the m0ml, mtml, mhjk, mtChao and mtChpm [1], which have different assumptions regarding testers' abilities and the probabilities of detecting a defect. The capture-recapture (CRC₁) model uses duplicate defect data, that is, errors found by more than one tester, to estimate the number of remaining defects [1, 2, 15]. Using data regarding the components in which an error exists and the group or person who found the error, the capture-recapture (CRC₂) model estimates the number of components with defects post-release that showed no errors in testing [3]. (A component is a file or group of files that implements the functionality of a particular feature of the system.) Several studies include experience in defect estimation methods [1, 15]. Experience-based methods take into consideration the data from previous projects and extrapolate the estimates of the present working project using the earlier results. Runeson, et al. [15], combine capture-recapture methods with a fault classification method and apply an experience-based multiplicative factor to faults found by a single reviewer to make them less sensitive to the composition of the inspection team.

Curve fitting models, such as the Detection Profile Method (DPM) and the Cumulative Method [5] plot test data and fit a mathematical function to estimate the remaining defect content. These methods make less restrictive assumptions. Wohlin and Runeson [5] evaluated the DPM and the Cumulative Method and compared them to capture-recapture models. Their results showed that the methods produced varying estimates. Hence, they conclude that the mean value of the estimates was the most sensible approach to estimate the number of remaining defects.

Stringfellow, et al. [15], apply several of these models. Their main focus is on using these methods to estimate the number of defective components in post-release. Their results show that the actual value turned out to be between the estimates provided by the m0ml and the mtml model, DPM and jackknife estimator (mhjk). This is good as it gives lower and upper limits. These estimates can be used as additional criteria to determine when it is suitable to stop testing and release software.

2.2 Defect estimation using software reliability models

Many dynamic models try to assess whether a software testing objective has been met to determine when to stop testing. These models are based on various sets of assumptions about the software and its execution environment. Software reliability growth models (SRGMs) use data about the times that failures occur to estimate the number of remaining failures in a system [6]. Generally, SRGMs estimate the number of failures in a system. However, existing empirical evidence shows that SRGMs can also be applied on error data to estimate the number of defects in a system [4, 6]. For the software practitioner, the assumptions or conditions for the various SRGMs are an open problem, because they are often violated in one way or another. Wood [6] and Stringfellow [4], however, demonstrate that SRGMs are usually robust, despite these assumption violations. In addition, Stringfellow, et al. [4], present an

empirical selection method that aids in choosing the appropriate SRGM model when assumptions are not met.

3.0 Approach

The approach for this project includes three phases. The first phase involves selecting, downloading and installing defect tracking tools that are freeware or open source. Several tools are installed on both Windows and Linux platforms. The second phase involves documenting the format of the data collected by the tools and implementing data extraction and conversion routines to transform data from the various defect tracking software tools into a consistent delimited format for a comprehensive defect estimation tool. The third phase involves analyzing the defect data for the tools to determine if any defect estimation method can be applied.

4.0 Results

Seven defect tracking tools were installed on either Windows or Linux platforms. Those tools include Buggit, Bugtrack, Bugs Online on the Windows platform and Bugzilla, Roundup, Mantis and Incident Management Systems (IMS) on the Linux side. Table 1 provides the details about each tool, such as the supporting software needed, effort to install, and the usability of the tool in terms of how easy it is to learn and use. All freeware is denoted by a superscript F. Roundup was immediately rejected, because the data could not be easily exported from the tool.

Table 1. Defect tracking tools analyzed.

	Buggit	Bugzilla	Bugtrack	Roundup	Mantis	IMS	Bugs Online
Platform	Windows 2000/XP	Linux (Fedora) ^F	Windows 2000	Linux (Fedora) ^F	Linux (Fedora) ^F	Linux (Fedora) ^F	Windows
Database	MS access	Mysql ^F	SQL server 2000	Mysql ^F	Mysql ^F	Mysql ^F	MS Access
Server (if any)		Apache ^F	IIS 5.0	Apache ^F	Apache ^F	Apache ^F	IIS 5.0
Other SW		Perl modules ^F		Python ^F	Php ^F	Php/zend optimizer ^F	
Effort to install	low	high	high	med	high	Med	med
Effort to learn and use	low	med	low	high	med	med	med
Ability to export data	Export from MSAccess	Export using PhpMyAdmin	Export from SQL Server 2000	Cannot export data	Export using PhpMyAdmin	Export using PhpMyAdmin	Export from MSAccess

Table 2 gives detailed information on the kinds of data reported for each defect by the tools. Analysis of the database formats of these defect tracking tools lead to several observations. All of the tools have an ID or a name for each bug in their databases, which is necessary to identify each defect uniquely. Four tools have data about the components in which a defect exists. This is useful in estimating fault-content by modules or components. The component may be a subsystem, a module, or a file.

Almost all defect tracking tools analyzed have data that give the date a defect report is created or added. For applying SRGMs, the add date or the open date of a defect is necessary. Almost all tools include severity levels in their data. Metrics by severity are very helpful in improving software reliability growth models (SRGMS) [4, 6].

Table 2. Defect data reported by defect reporting tools.

Data	Synonym(s)	Buggit	Bugzilla	Bugtrack	Roundup	Mantis	IMS	BugsOnline
GroupFound								
Name	ID / Bug Number	√	√	√	√	√	√	√
Component	CompName/Location	√	√	√				√
Release	Version	√	√	√		√		
Severity		√	√	√		√	√	√
Duplicates			√			√		
OpenDate	AddDate	√	√	√		√	√	√
CloseDate	EndDate	√					√	√
OriginID	Reporter		√			√		
OriginName	Reporter	√		√	√		√	√
Priority		√	√	√		√	√	√
Age								
Visibility		√						
Expandability								

None of the tools include data indicating the group that found the defect, but all of them have either a reporter ID or a reporter name from which one can deduce the testing group that found the defect. Data about origin ID, name or group that found a defect and duplicates are very useful for applying capture-recapture models to estimate number of defects in software or to estimate the number of defects in individual components. In order to apply the capture-recapture (CRC₁) model to estimate the number of remaining defects, the data about the duplicates are necessary. For the capture-recapture (CRC₂) model that estimates the number of remaining components with defects, the data regarding the components in which a error exists and the group or person who found the error are necessary.

Unfortunately, none of the defect tracking tools is expandable: they do not allow the user to add other fields to the defect report record to improve their usefulness.

Table 3 shows the types of defect estimation models that can be applied to the data from each defect tracking tool analyzed. Roundup was already rejected, because the data were not easy to extract. In any case, it does not provide enough information about defects to apply any defect estimation model. In comparing Buggit and BugsOnline, which are two Windows-based defect tracking tools, it is observed that Buggit is a more professional-looking, easier to install, and more usable tool to learn and use than BugsOnline. So, between these two defect tracking tools, Buggit is considered a better tool for both defect tracking and applying defect estimation techniques to its data. Between Bugzilla and IMS, two Linux-based defect tracking tools, IMS has some disadvantages over Bugzilla: IMS requires more supporting software and IMS does not provide data about the component in which a defect exists nor the version number, which are necessary to apply SRGMs and capture-recapture models. Therefore, IMS is rejected. Other tools that are considered useful include Bugtrack, a Windows 2000-based application, and Mantis, a Linux-based application as they provide necessary defect data for the estimation models. In addition, their interfaces are similar to commercially available products with graphical user interfaces that professional software engineers use.

Table 3. Models that can be applied on defect reporting tools.

Model	Buggit	Bugzilla	Roundup	Mantis	Bugtrack	IMS	BugsOnline
Capture-Recapture 1		√		√			
Capture-Recapture 2	√	√			√		√
SRGM	√	√		√	√	√	√

Future work involves the development of a Java application to import the data from these four defect tracking tools and to apply the appropriate defect estimation methods to aid software test managers in making release decisions. The application will prompt the user to select one of the four defect tracking tools, and to upload the delimited defect data file extracted from the defect tracking tool.

5.0 Summary

There are many free and open source defect tracking tools. Seven were selected for download and installation. They were evaluated based on the database, server and other software required to run them; the effort required to install, learn and use them; as well as the ability to export data from them. Four of these, Buggit, Bugzilla, Mantis and Bugtrack, proved to be relatively easy to install and learn. This study then examined the type of data reported for defects by these tools. Analysis showed these four defect tracking tools do provide the type of data necessary for applying one or two of the defect estimation models discussed in this paper. Bugzilla provides the type of data necessary for applying all three defect estimation models discussed. Future work will involve developing a comprehensive tool to use the data from these four defect tracking tools with defect estimation methods for the purpose of recommending release decisions.

6.0 References

- [1] Briand, L., El Emam, K. and Freimut, B., "A comparison and integration of capture-recapture models and the detection profile method," *Proceedings Ninth International Conference On Software Reliability Engineering*, Paderborn, Germany, 1998, 32 -- 41.
- [2] Briand, L., El Emam, K., Freimut, B., and Laitenberger, O., "A comprehensive evaluation of capture-recapture models for estimating software defect content," *IEEE Transactions on Software Engineering*, 26(6), June 2000, 518 -- 539.
- [3] Stringfellow, C., von Mayrhauser, A., Wohlin, C., and Petersson, H., "Estimating the number of components with defects post-release that showed no defects in testing," *Journal of Software Testing, Verification and Reliability*, 12 (2), June 2002, 93 -- 122.
- [4] Stringfellow, C., and Andrews, A., "An empirical method for selecting software reliability growth models," *Journal of Empirical Software Engineering*, 7(4), December 2002, 319 -- 343.
- [5] Wohlin, C. and Runeson, P., "An experimental evaluation of capture-recapture in software inspections," *Journal of Software Testing, Verification and Reliability*, 5, 4, 1995, 213 -- 232.
- [6] Wood, A., "Software reliability growth models: assumptions vs. reality", *Proceedings of the International Symposium on Software Reliability Engineering*, 23(2), 1997, 136 -- 141.
- [7] Cusumano, M., "Reflections on free and open software," *Communications of the ACM*, 47(10), October 2004, 25 -- 27.
- [8] Bugzilla installation guide. 1998 – 2004. (<http://www.bugzilla.org/docs/tip/html/installation.html>)
- [9] "Defect tracking tools." 2000 – 2004 (<http://testingfaqs.org/t-track.html>)
- [10] "Freeware bug tracking tools." 12 Nov 2003. (http://dmoz.org/Computers/Software/Configuration_Management/Bug_Tracking/Free)

- [11] "IMS installation guide." 24 Nov 2003. (<http://ims.sib3.ru/install.php>)
- [12] Yu, T., Shen, V. and Dunsmore, H., "An analysis of several software defect models," *IEEE Transactions. on Software Engineering*, 14(9), 1988, 1261 -- 1270.
- [13] Biyani, S. and Santhanam, P., "Exploring defect data from development and customer usage on software modules over multiple releases," *Proceedings Ninth International Conference on Software Reliability Engineering*, Paderborn, Germany, 4(7), November 1998, 316 -- 320.
- [14] Wohlin, C. and Runeson, P., "Defect content estimations from review data," *Proceedings International Conference in Software Engineering*, Kyoto, Japan, (1998), 400 -- 409.
- [15] Runeson, P. and Wohlin, C., "An experimental evaluation of an experience-based capture-recapture method in software code inspections," *Empirical Software Engineering: An International Journal*, 3(4), 1998, 381 -- 406.