Open source software reliability model: an empirical approach

Ying ZHOU
School of Information Technologies
The University of Sydney
Sydney, NSW 2006 Australia
Tel: 61 2 93513215
Email: zhouy@it.usyd.edu.au

Joseph DAVIS
School of Information Technologies
The University of Sydney
Sydney, NSW 2006 Australia
Tel: 61 2 9351 4291
Email: jdavis@it.usyd.edu.au

ABSTRACT
We collected bug tracking data from a few popular open source projects and investigated the time related bug reporting patterns from them. The results indicate that along its development cycle, open source projects exhibit similar reliability growth pattern with that of closed source project. Bug arrivals of most open source project will stabilize at a very low level, even though in comparison, no formal testing activities are involved. This stabilizing point would be viewed as the mature point for adoption consideration. The results also show that general Weibull distribution offers possible way to establish the reliability model; Also, popular measures such as page views and download are not highly correlated with the bug arrival rate and may not be suitable measures for a project's quality.

Categories and Subject Descriptors
D.2.5 [**Software Engineering**]: Testing and Debugging – *tracing.*

D.2.9 [**Software Engineering**]: Management – *Software quality assurance*.

General Terms
Management, Measurement, Reliability/

Keywords
Software reliability model, Weibull distribution, open source software

1. INTRODUCTION
Open source software has attracted significant attention in recent years. Report shows that a few major open source software products have surpassed their commercial counterparts in market share and quality evaluation [18]. They not only attract individuals who want high quality software and cannot afford expensive commercial version, but also become good candidates in many businesses or governments' Information Technology plan. Survey conducted by CIO in late 2002 revealed that the IT community is growing more comfortable with the open source development model and the majority (64%) of companies surveyed are using open source, most frequently as web server, server operating system and for web development [2]. The adoption rates are getting very substantial in server and operating system area, in which two dominant OS products Apache and Linux have already set up their brand names with proven quality. However, there are still many other areas that people are hesitant in picking up open source products. Recent empirical research published by Forrester suggests that although most European firms have clear open source adoption plans. There are still fears and unsolved questions especially for business people and project managers. Two common fears, which have also been outlined by Ray Lane, former Oracle executive in a keynote speaking in the open source conference 2004 are the lack of formal support and velocity of changes [4]. All these fears and concerns can be traced back to the quality and reliability of open source products.

Software reliability model has long been used as the most important and successful predictor of software quality when it hits the market. The widely used models in industry include Rayleigh model, which models the whole software life cycle as Rayleigh curve and has been used for projecting latent software defects when the development work is complete and the product is ready to ship to customers. Another widely used model is the exponential model, sometimes called reliability growth model, which has been used for modeling the defect arrival pattern at the backend of the development for the purpose of projecting failure pattern in field as well. Both models are supported by large body of empirical data. Most data are closely related with development process and are from large commercial software. Open source projects take a very different procedure with closed source commercial project. The contrast has been illustrated by Eric Raymond as "Bazaar" and "Cathedral" model [15]. The appropriateness of using these methods to project or evaluate open source project quality when making adoption decisions remains a open question. Other metrics developed in software industry to access a company's development environment and ability, such as CMM rating, are strongly with the process [8].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Open Source Application Spaces: Fifth Workshop on Open Source Software Engineering (5-WOSSE) May 17, 2005, St Louis, MO, USA.

Copyright 2005 ACM 1-59593-127-9 ... $5.00.

No mature method or metrics have yet been developed to evaluate the quality of open source projects. Except for a few big names like Gnome, Linux, Mozilla, Apache and so on, people make judgments on open source products based on relatively arbitrary criteria. Big open source projects hosts like sourceforge.net lists facts (such as projects size, developer number) and simple statistics (activity rank, weekly download, page view, CVS commits and so on) to assists potential adopters to assess the quality of an open source projects. The problem is there are too many factors involved and people simply don't know which one to examine, or when an open source software is mature enough for adoption. The purposes of this research is to build a general reliability model for open source software projects to see if it is possible to highlight a few key features and critical time points that can assist in making adoption decision. We will also check the correlations between bug arrival patterns and popular statistics such as page view and download patterns to see if these popular statistics would be used as alternative quality measures.

2. RELIABILITY MODELS

Weibull distribution family is perhaps the most widely used lifetime distribution model [9]. Its simplest form, the 2-parameter Weibull distribution, has long been used to model reliability pattern due to its ability in describing failure modes like initial, random and wear-out. [19]. Data from large commercial software suggests two special forms of Weibull distribution: Rayleigh distribution and exponential distribution have been applied in software reliability models[7].

The 2-parameter Weibull distribution has a probability distribution function of the form

$$f(t) = \lambda\beta(\lambda t)^{(\beta-1)} \exp(-(\lambda t)^{\beta}) \qquad (1)$$

Where t represents time; $\alpha = 1/\lambda$ represents the scale parameter of the distribution and β represents the shape parameter of the distribution. The Weibull probability density function is monotone decreasing if $\beta <= 1$ and becomes bell shaped when $\beta > 1$. The larger the β value the steeper the bell shape. Its special case Rayleigh distribution has $\beta = 2$; while exponential distribution has $\beta = 1$. Figure 1 shows several weibull probability density curves with varying values for the shape parameter β.

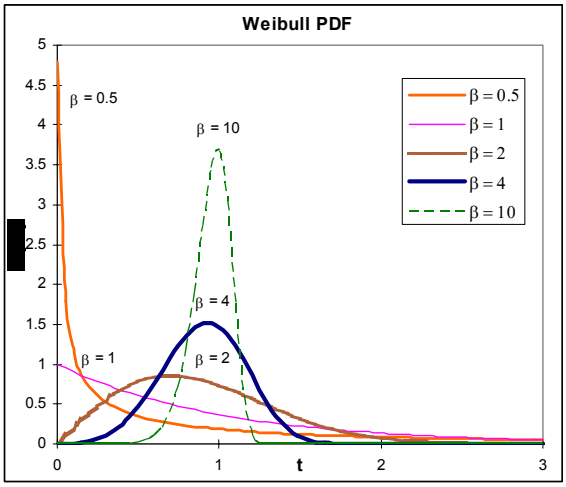


Figure 1 Weibull probability density

In software quality engineering, large body of empirical data supports the finding that software projects follow a life cycle pattern described by Rayleigh curve. This is considered as a desirable pattern since the bug arrival rate stabilizes at a very low level. In closed source software, the stabilizing behavior is usually an indicator of ending test effort and releasing the software to the field. The development cycle, from quality perspective, is divided into six phases, high-level design inspection, low-level design inspection, code inspection, unit test, component test and system test. The bug arrivals usually peak at the code inspection phase and get rather stabilized in the system test phase [7].

3. DATA COLLECTION

In the data collection stage, we first identified a few target open source projects according to product size. The main selection criteria include project duration and activity. New projects with less than one year history usually will not be able to contribute enough useful information. One typical phenonman associated with many open source projects is that they might get on and off during the project period. To rank projects according to activities, SourceForge.net devised an activity measure. It takes number of downloads, number of times mentioned in the forum and other measures into consideration and combine them to form an activity index. We picked our target projects following SourceForge's "Most active" rank list as of August, 2004. The projects identified are listed in table 1 along with their prominent features. Actual names of the projects were not revealed to follow standard software engineering ethics [3].

We mainly collected information regarding newly opened bugs per month. SourceForge provides simple statistics on monthly bug reported. However, these statistics show all bugs reported regardless whether they are valid or not. Each month, there are a few bugs reported and then deleted by the project owner or core developer members. They are considered as invalid bugs for various reasons; we need to exclude those bugs for accuracy. Besides, those statistics are collections of all software products under one project title, it is more sensible to differentiate among different products. Fortunately SourceForge keeps

detailed description for each reported bug, its status and the component it belongs to. We used the query function provided to isolate bugs from a certain component as well as each month's deleted report and rule them out from our data. Hence the original data we got for monthly opened bugs are slightly different from those reported in SourceForge statistics.

Table 1. Target Open Source Projects

Project Title	Starting time	Developer number
Project PrA	1999-11	14
Project PrB	2003-06-24	9
Project PrC	2003-04-13	40
Project PrD	2000-11-10	1
Project PrE	2001-03-18	8
Project PrF	2001-03-15	115
Project PrG	2000-02-09	40
Project PrH	2000-05-03	1

4. DATA ANALYSIS

4.1 Overall bug arrival trend

We plotted each month's reported bug numbers along the timeline in figure 2 to get an overall picture of a possible pattern. The eight projects were presented in three separate charts based on average monthly rate. The data show similar trend across all projects except for a few projects which are still in their infancies. The monthly bug arrival rate goes slowly upwards along until it reaches a peak; it then starts to decrease, stabilizes at a rather low level. This forms a bell shape curve. The trend is consistent in PrA, PrE, PrF, PrD, PrG and PrH projects. Two other projects, PrB and PrC are registered for short times (around 1 year). From the second chart in figure 2, it is clear that the bug arrival is still active in these two projects. Since the monthly bug rate keep increasing, it is hard to tell at this stage whether they are going to reach the peak and then decrease to a stabilizing state.

The monthly bug data reveal a clear pattern and there are families of models in the analysis of failure process data that fit this general distribution. However, we will concentrate on the Weibull distribution, which has long been demonstrated its appropriateness in reliability/failure time analysis [9].

4.2 Reliability growth model

The two-parameter weibull non-linear regression model we use to build the time and bug arrival rate relationship is:

$$y = K * \lambda * \beta * (\lambda * t)^{\beta-1} \exp(-\lambda t)^{\beta} \quad (2)$$

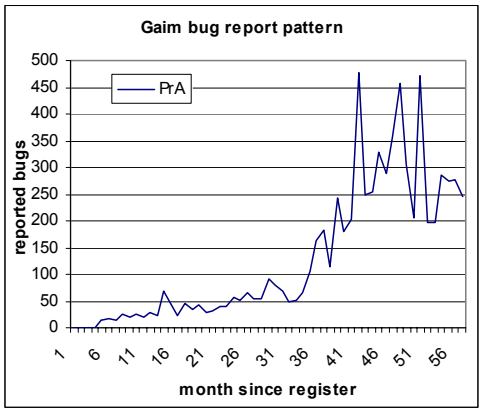
Where y is the bug arrival monthly rate β is the shape parameter and λ is the scale parameter. SPSS nonlinear regression procedure is employed to implement the general weibull model and to estimate the parameters. All data are grouped data with type I censoring [1]. Table 2 lists the estimations of the shape and scale parameters. The actual values and expected curves for all projects are plotted in figure 3 respectively.

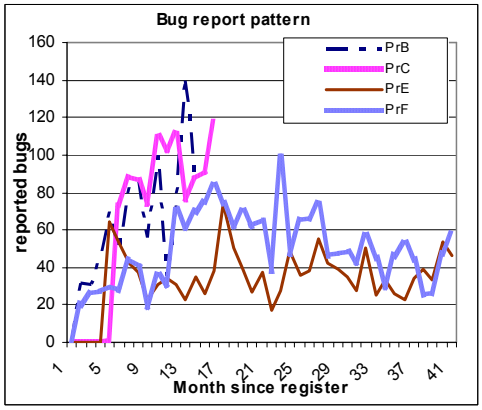
Table 2. Estimated parameters and R^2

Projects	λ	β	R^2
PrA	0.02	5.29	0.7938
PrH	0.04	2.31	0.4821
PrF	0.03	1.81	0.4982
PrE	0.01	1.27	0.1052
PrD	0.04	2.87	0.2705
PrG	0.007	1.85	0.5420

The R^2 indicates good fit for some projects such as PrA, PrF, PrH and PrG. The other two projects, PrE and PrD have very low R^2 value, indicating low correlations between time and bug arrival rate. We did not run regression on PrB and PrC . They were registered for less than two years and available data points are less than adequate for censored data analysis. Although in most cases, the estimated value of β is around 2, there is also case with much higher β value (5. 29 for PrA project). This suggests that general Weibull distribution is a possible candidate to model open source reliability pattern, although the special case Rayleigh distribution is not suitable in some case.

[1] It is not possible to get complete data set as most projects will keep going for a long period.





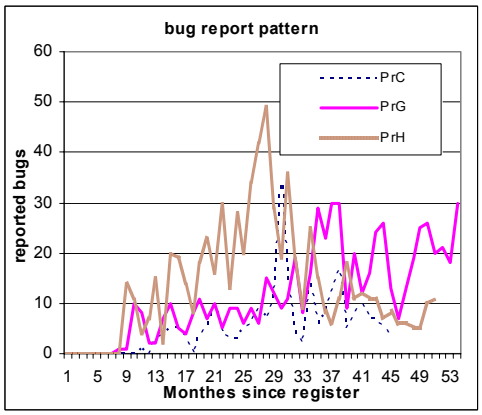
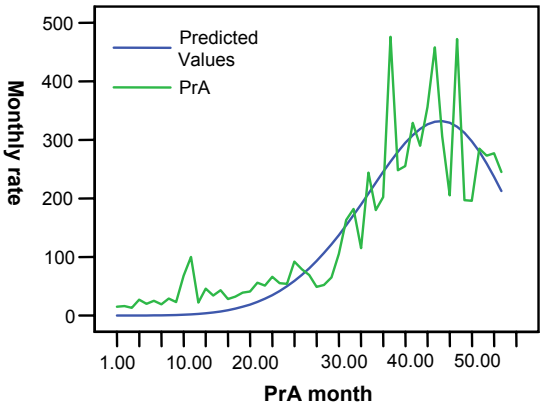
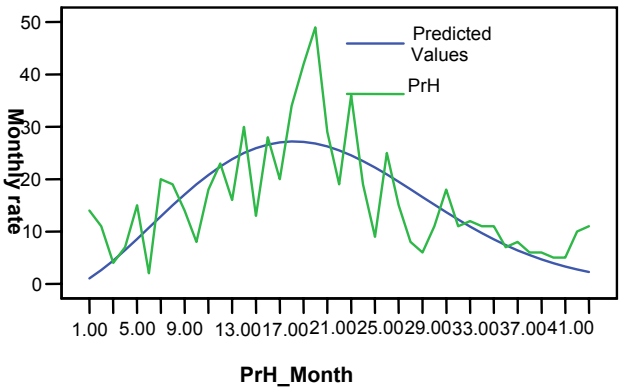
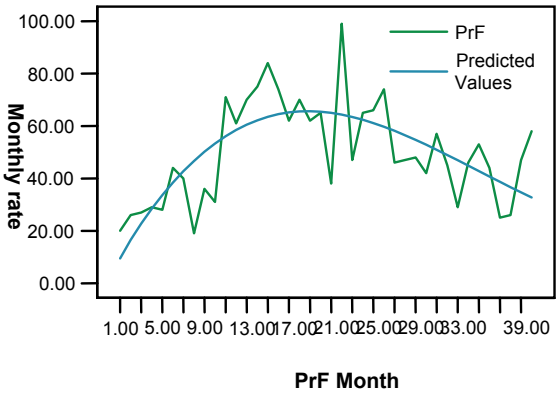


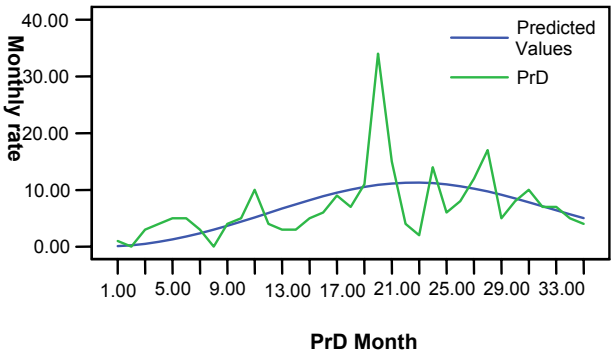
Figure 2 Bug arrival patterns of 8 projects

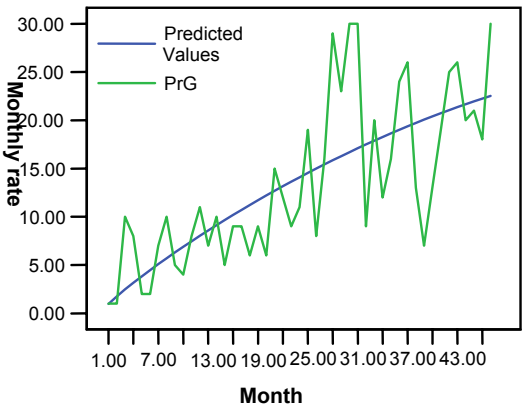
PrE and PrD are typical open source projects. They were initiated as Open source projects and continued to evolve with single code base. Both are rated as *stable* in SourceForge and have fairly large number of downloads per month. PrE has eight developers and has been active most of its life cycle. PrD has only one developer and its activity indexes have large variations along its life cycle. Neither of them is close to stable bug arrival level. A possible explanation as suggested by Samoladas et. al [17] is that sometimes open source project coordinators would make abrupt changes between subsequent releases; this might result in changes in code structure and hence bug arrival rate. To extract the reason behind such abnormal phenomena, detailed examination of individual project is required.











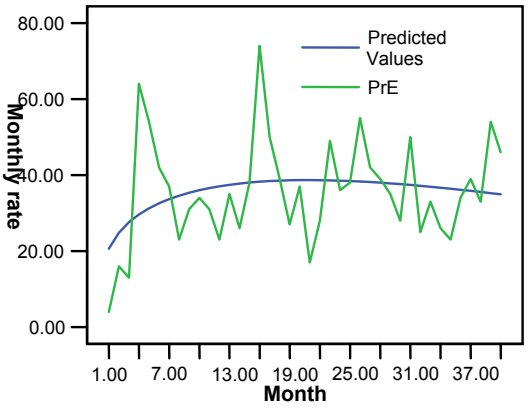


Figure 3. Reliability model estimation.

4.3 Bug arrival pattern vs. page view and download

We also collected data on page views and download for the six projects. The data are exported directly from SourceForge statistics. Table 3 lists correlation coefficients of monthly bug arrival rate vs. page views and monthly bug rate vs. download of 6 projects. Again, various projects told various stories. Among the six projects we examined, only one project demonstrate strong correlations between monthly bug arrival rates and both monthly page view and download numbers. All other projects reveal relatively low correlations among these variables. The relation between bug arrival and download, page view count is quite complex. A common view within open source community, as expressed by Eric Raymond, which states, "with enough eye balls, all bugs are shallow", implicitly suggests a positive relationship between user numbers and bug number. Our preliminary investigate does not attempt to build an overall picture of the interrelationship among those variables. It is rather an initial effort of examining the correlation between various possible quality indicators. The generally low correlation between download and bug arrival rate partially support observations made by Mockus et al. [11] that most bugs were reported by a relatively small developer community rather than end users. It also signifies that for most projects, number of page views and download might measure a project's user acceptance level, but they may not be used as quality measurements.

Table 3 . Correlation coefficient table

Project	Bug vs. page view	Bug vs. download
PrA	0.92172881	0.817904
PrH	0.049004	0.688673
PrF	0.536187	0.582681
PrD	0.566281	0.371755
PrG	0.637104	0.719103
PrE	0.382854	0.502188

5. CONCLUSITON

Our study involved only 8 popular open source projects; all of them are still under development. Hence, only provisionary conclusions could be drawn from the results. A few possible observations of the results of this study are:

1. Along their development cycle, open source projects exhibit similar reliability growth pattern with that of closed source projects. Bug arrivals of most open source project will stabilize at a very low level, even though no formal testing activities are involved.
2. General Weibull distribution is a possible way to establish the reliability model. Estimations of shape parameter from various open source projects are different, indicating that in contrast with closed source projects, it is unlikely to find a special case like Rayleigh curve to model all open source projects. It might be a better way to model individual open source project separately. Time series analysis would be appropriate for predicting latent bugs in individual open source projects. The evidence that most open source projects demonstrate seasonal ups and downs also makes time series analysis a favorable choice for predictive purpose.
3. Popular measures such as page view and download are not highly correlated with the bug arrival rate and are not suitable measures for a project's quality.

6. REFERENCES

[1]. ApacheWeek, "Apache 2 release history", 2004 [WWW document] URL http://www.apacheweek.com/features/ap2

[2]. Cosgrove, L. Confidence in open source growing, *CIO Research Report*, 2003 [WWW document] URL: http://www2.cio.com/research/surveyreport.cfm?id=51

[3]. El-Emam. Ethics and Open source. Empirical Software Engineering 4, 6 (2001), 291-292

[4]. Farber D. Six barriers to open source adoption, *ZDNet Tech Update*, March, 2004 [WWW document] URL:http://techupdate.zdnet.com/techupdate/stories/main/Six_barriers_to_open_source_adoption.html

[5]. Hisada Koji and Ikuo Arizino, Reliability Tests for Weibull distribution with varying shape-parameter, based on complete data, *IEEE transactions on Reliability*, Vol. 51/No. 3 Sep. 2002

[6]. Iannacci Federico "The linux Managing Model" *First Monday,* vol 8, no. 12 (Dec, 2003). [WWW document] URL: http://www.firstmonday.dk/issues/issue8_12/iannacci/

[7]. Kan H.S. *Metrics and models in software quality engineering*, 2nd edition, Addison-Wesley (2003)

[8]. Koch, C. Bursting the CMM hype, *CIO Magazine*, March 1, 2004.

[9]. Lawless J.F. Statistical Models and Methods for Lifetime Data, 2nd edition, New York: Wiley (2003)

[10]. McConnell S. (eds) Best Practice: Daily build and smoke test. *IEEE Software,* Vol. 13, No.4, July 1996

[11]. Mockus A, Fielding T.R., and Herbsleb, J.D. Two case studies of open source software development: Apache and Mozilla, *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 3 (July 2002) 309-346

[12]. Mozilla organization, "CVS tags for Mozilla major public release", July 9, 2004, [WWW document] URL: http://www.mozilla.org/releases/cvstags.html

[13]. Musa J.D. and Okumoto K. "A Logarithmic Poisson Execution Time Model for Software Reliability Measurement", *Proceedings of Seventh International Conference on Software Engineering*, 230-238 Mar. 1984

[14]. Peynot R and Metcalfe D. The business hole in open source support, *Forester Research*, August, 2004.

[15]. Raymond, E.S, The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, 2nd ed. Sebastopol, CA:O'Reilly, 2001.

[16]. Rodrigues G.N, Rosenblum D. and Emmerich W. A model driven approach for software systems reliability, *Proceedings of the 26th International Conference on Software Engineering*, 2004

[17]. Samoladas I., Stamelos I, Angelis L. And Oikonomou A. Open source software development should strive for even greater code maintainability, *Communications of the ACM*, Vol. 47/No. 10, Oct, 2004.

[18]. Wheeler D.A., Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers! 2003 [WWW document] URL: http://www.dwheeler.com/oss_fs_why.html

[19]. Yamada Shigeru, Jun Hishitani and Shunji Osaki, Software-Reliability Growth with a Weibull Test-Effort: A model & application, *IEEE Transactions on Reliability*, Vol. 42/No. 1, Mar. 1993