Using Software Reliability Growth Models for Developing a Software Quality Index

KASHIF BILAL
National University of Computer and Emerging Sciences Lahore
0923215106460
kbse2000@gmail.com

HAFIZ RIZWAN RASHEED
National University of Computer and Emerging Sciences Lahore
0923214795204
Kbse_2000@yahoo.com

ABSTRACT

Software reliability growth modeling is a mathematical technique to predict and estimate the reliability of software using reliability growth models. Defect data pattern is comprehensively modeled with this technique, therefore it is widely gaining acceptance as measure for software quality. Our experiment utilizes most popular open source project repository for gathering project statistics. This is a two pronged study for sorting out the quality related statistics and estimating the reliability .Currently these repositories apply acceptance measures for aiding the user judgment about particular software these measures are relatively arbitrary and based on project statistics related to developer, user and traffic of a particular project. Unlike arbitrary formulation adopted by opensource repositories we establish a relationship between software reliability and pile of project statistics to finally extract out only those factors which have high impact on software quality .Using Computer aided software reliability engineering tool [CASRE] a methodology is proposed to estimate and predict the reliability of a software under observation and regression analysis is performed to elaborate the impact of developer, user, web traffic statistics on software reliability. We pointed out that developer activities related data highly impacts the software reliability. Sifting through the available statistics we aim at developing a software quality index which can serve as minimum information to effectively determine the quality of software.

Categories and Subject Descriptors

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

General Terms

Management, Measurement, Reliability.

Keywords

Software reliability growth modeling, software reliability engineering, software quality index.

1. INTRODUCTION

Quantifying the quality of a software product is an issue of active research. The work of Ying [3] declares that with respect to the

Time related bug patterns the growth pattern of opensource project is similar to close source, therefore the data from opensource project archives can be utilized to develop an understanding for close source project as well. We start with gathering project statistical data from the largest opensource projects repository, sourceforge.net [1].This repository furnishes projects statistics related to developer, user and web-traffic data. The repository ranks the projects on criteria which is called ACTIVITY PERCENTILE (AP). AP determines the overall ranking of a project based on developer, user and traffic statistics. We investigate whether we can adopt AP as a quality index per se. If not, what modifications this AP index requires to be adopted as a comprehensive software quality or user acceptance index.

Software reliability is defined as the probability of failure free software operation in a defined environment for a specified period of time .Software reliability models are mathematical models which can be used to make predictions about a software system's failure rate, given the failure history of the system. The models make assumptions about the fault discovery and removal process. These assumptions determine the form of the model and the meaning of the model's parameters. The parameters determine the shape, size and location of the Probability Distribution Function (PDF) Plot. There are two types of models those that predict times between failures and those that predict the number of failures that will be found in future test intervals. Software tool CASRE [10] (Computer Aided Software Reliability Engineering) is used in our study due to a wide range of models support and friendly GUI.

Major models are examined and compared in [4]; applying different kind of failures and testing data. Quality Metrics for errors, faults and failures are discussed in [5] at Software Assurance Technology Center (SATC) at NASA. Reliability models are discussed in HANDBOOK OF SOFTWARE RELIABILITY [2] elaborating their usage for different projects and error data .Typically in software quality related research defect count , defect density or meantime between failure like variables are used as dependent variables. We believe that for consistency and longevity of product quality fault detection, correction and prevention mechanism must be consistent and durable. These phenomenon's' are correctly modeled through Software Reliability Growth Modeling. Therefore we are convinced to use it as a dependent variable in our research.

2. DATA COLLECTION

We picked our target 35 projects following Source Forge's most active rank list as of October, 2007 for three reasons. First reason is that most active projects provide rich enough data set for statistical justifications of our hypotheses. Second there can be unpredictable gaps in the activities of less active open source projects which cause the failure of reliability trend test. Third the software tool CASRE requires at least 50 bugs in a project and active response of the developer community in the recent past in fixing those bugs. The data available on sourforge.net is not in a form that can be used as a direct input for reliability assessment tool [CASRE]. The data computation required additional effort to bring it in a form usable for input to CASRE.

3. RELIABILITY MODELLING PROCEDURE

Every model is characterized by some assumption, limitations and selection criteria. The data set is scrutinized to ensure that model requirements are not violated. The data sets which cannot exhibit a certain level of trend for reliability growth is automatically discarded by CASRE tool [7] .The approach taken to perform the estimation includes following steps.

3.1 Apply Trend Test

The trend test implies that the Mean Time between Failures is increasing or fault count is decreasing.

3.2 Selection of Models

Not all the models are applicable to every data set therefore the tool automatically guides about data's validity for a particular model. Detailed mathematical descriptions of the models may be found in [9]. Based on our data set, the Times between Failures (TBF) Models are applicable.

3.3 Estimating the Model Parameters

The tool provides two means for parameter estimation: Maximum likelihood (ML) and Least Square (LS). For most of the data sets we have seen, ML produce valid parameter estimates and is sufficiently quick to be the preferred estimation method.

3.4 Model Evaluation

Model evaluation is the procedure in which we decide which particular model is best for the given data set. Mostly applicable techniques were Prequencial Likelihood Ratio and U Plot.

The models are ranked according to the evaluation result and the best one was chosen for reliability estimation. Estimated Reliability is used in following hypotheses as dependent variable. 20 out of initially selected 35 different projects could pass the reliability trend test. ANOVA significant F statistics for all the models discussed below are highly significant and less than 1%. Matlab statistical toolbox is used for computing regression statistics.

4. HYPOTHESIS 1

Projects AP is an aggregate measure of traffic communication and development Activities and its formula is available [1] .This hypothesis will study the relationship of AP and reliability, both are percentage values.

Null Hypothesis:

H0: Projects Activity Percentile has no effect on software reliability.

Alternate Hypothesis:

H1: Projects Activity Percentile has effect on software reliability.

4.1 Rationale

If there is a strong relationship between AP and software reliability we can adopt AP per se as a quality index. Project ranking is directly measured from AP, therefore project with AP=100 percent is ranked first. Similarly if a project's AP is 90 percent it means that out of 100 selected projects 90 are less active than this project. Project which exploits all the developer, user and web resources actively is considered best. This seems plausible because more customers buying some product or rush of customers to purchase a product can be an apparent justification of product's quality. This line of thinking can be illusive, because the apparent burst of activities can be short lived.

4.2 Analysis of Results

Table 1 Regression Statistics for Activity percentile

Model	R	R Square	Adjusted R Square	Std Error of estimates
Linear	0.8014	0.6422	0.62232	81.1541

Table 2 AP Model Equation Coefficients and p-values

	Coefficients	p-val
AP	1.538	0.567

AP can predict reliability up to 80.14 %, and explains 64.22% proportion of variance of reliability. This model explains the 62.23% of overall variation.

The significant value is >0.05 which shows that the null hypothesis will be accepted. It means that there is no significant relation between the AP and estimated reliability.

5. HYPOTHESIS 2

Traffic activities comprise of project downloads and web hits. It can be argued that more people downloading a project is a manifestation of its acceptance from user perspective.

Null Hypothesis:

H0: Project's Traffic statistics have no effect on software reliability.

Alternate Hypothesis:

H1: Project's Traffic statistics have an effect on software reliability.

5.1 Rationale

An empirical study [3] shows that popular measures such as page view and downloads are not highly correlated with the bug arrival rate and are not suitable measures for a project's quality. This previous study used only one reliability model for bugs data formulation and 5 projects download and page view statistics. We extended this work by applying more models and more web traffic statistics.

5.2 Analysis of Results

Table 3. Regression Statistics for Web Traffic

Model	R	R Square	Adjusted R Square	Std Error of estimates
Linear	0.7880	0.6209	0.5763	27.0329

Table 4.Web Traffic Equation Coefficients and p-values

Independen t variables	Coefficients	p-val
Downloads	22.5678	0.4454
Web hits	12.9866	0.4239

Output shows that, web traffic activities can predict reliability up to 78.80 %, and explains 62.09% proportion of variance of maintenance cost. This model explains the 57.63% of overall variation.

The significant value is >0.05 which shows that there is no significant relation between the two factors under consideration.

6. HYPOTHESIS 3

Development activities are read/write actions performed by the project participants. This data is collected from cvs repository. It can be believed that active and consistent developers' participation can have impact on reliability of a system. Four developer activities data is available Read Transaction, Write Transactions, Total Files Updated, and Total Read Write Transactions.

Null Hypothesis:

H0: Developers activities have no impact on software reliability.

Alternate Hypothesis:

H1: Developers activities have an impact on software reliability.

6.1 Rationale

Subversion activities are read/write activities performed on a projects source code. We believe that more reliable software is result of more development activities conducted by the developers over a specified period of time. This hypothesis is based on a very study of GNOME software [8] where developer effort data is computed for software quality.

6.2 Analysis of Results

Table 5. Regression Statistics for Developer Activities

Model	R	R Square	Adjusted R Square	Std Error of estimates
quadratic	0.8112	0.6580	0.5668	17.726

Table 6.Developer-Activities Equation coefficients and p-values

Independent variables	Coefficients	p-val
Read Activities	23.124	0.07123
Write Activities	1.6789	0.0854
Total Activities	5.777	0.023
Total files updated	10.987	0.013

The significant values for total activities and total files updated are < 0.05 which shows that these two factors have some impact on estimated reliability.

7. HYPOTHESIS 4

User feed back and developer user co-ordination activities can improve the quality of software product. An effective communication setup can result in quality software. This assumption leads us to understand the relationship between software reliability and communication statistics.

Null Hypothesis:

H0: Project's communication statistics have no effect on software reliability.

Alternate Hypothesis:

H1: Project's communication statistics have an effect on software reliability

7.1 Rationale

Communication related statistics effectively determine the quantity of developer/user communication. These activities include number of posts in projects mailing list, number of posts in projects discussion forums, and number of tickets (open and close) in the tracker system. This last attribute, i, e tickets comprises of data entered for patches, support and features.

7.2 Analysis of Results

Table 7. Regression Statistics for Communication Stats

Model	R	R	Adjusted	Std

		Square	**R Square**	**Error of estimates**
linear	0.6981	0.4873	0.3911	26.568

Table 8. Communication Stats Coefficients and p-values

Independent variables	**Coefficients**	**p-val**
Mailing Lists Posts	56.678	0.8842
Project-Discussion forums Posts	71.4509	0.1762
Project Tickets	11.222	0.0763

Communication activities also lack significant impact on estimated software reliability.

8. CONCLUSIONS

Starting with the current criteria of User Activity Percentile (AP) we suggest that it is not a comprehensive enough for user acceptance .An inactive project i.e. marked low through AP criteria can be of high quality, if it is also being distributed by some media other than web, like Linux red hat distribution through cd. Similarly active projects marked high by AP criteria can be of poor quality as it may have latent bugs which burst out as the community of its user grows. We instigate to devise a strategy in which we can transform this user activity percentile to a user acceptance percentile, which can serve as quality criteria.

Based on the data analysis in our study we can conclude that the software ranking criteria developed by source forge has no significant relationship with software quality and this criterion is quite arbitrary. Even though some of the statistics underlying this criteria, show strong relationship with the computed reliability such as the development activities. Coefficients of the model [table 6] from these parameters can be used to form a quality index equation .In order to develop a quality index some other formula can be suggested which

(i) Assigns more weight to factors strongly impacting software reliability like total developer activities, Total source files updated.

(ii) Drop factors not impacting the quality like communication and download activities.

(iii) Instead of only utilizing estimated reliability compute additional factors based on predicted values of Time between failure, Failure intensity and reliability.

9. FUTURE RECOMMENDATIONS

This research was conducted by considering only 20 different projects of large to medium size. In future, a research can be conducted by broadening the data collection. This study only pointed out suitable statistics, proposing a refined quality index requires further work. A software system can be developed which keeps CASRE in bank end and compute the reliability and defect data related estimates from online statistical data available at source forge. As the current results are showing impact of developer activities to software reliability it will be required to compute further developer's activity data like developer time and effort. Currently there is no availability of developer time and developers' effort related data on source forge. In order to compute these measures additional source code repository analysis tools like casually [6] will be required.

10. ACKNOWLEDGMENTS

Our thanks to sourceforge.net for providing the data repository for experiments.

11. REFERENCES

[1] www.sourceforge.net

[2] Handbook of software Reliability. IEEE computer society and McGraw Hill Book Company.1996

[3] Ying Zhou, Joseph Davis. 1995 Open source software reliability model: an empirical approach. ACM SIGSOFT Software Engineering Notes. ISSN:0163-5948

[4] Software Reliability Model Study by Michael Grottke.

[5] Software Metrics and Reliability. Dr. Linda Rosenberg
Unisys/NASA GSFC Bld 6 Code 300.1 Greenbelt,MD
2077, USA301 286 0087Linda.Rosenberg@gsfc.nasa.gov

[6] http://cvsanaly-web.tigris.org/

[7] User-Manual Computer aided software reliability engineering tool version-3.

[8] Results from Software Engineering research into opensource Development projects using public data. Stefan Koch and George Schneider. Department of information business Vienna university of economics and BA,Augasse 26 A 1090 Vienna, Austria

[9] The NavSWC technical report TR 82 171, "A Survey of Software Reliability Modeling and Estimation", prepared by Dr. William H. Farr [NSWC83].

[10] http://www.openchannelfoundation.org/projects/CASRE_3.0