Software Analytics - Analysis and prediction of defect data using projection methods

Sarat Kiran Andhavarapu, *A01960723*

***Abstract*—In spite of diligent planning, good documentation and proper process control during software development, appearance of defects are inevitable. These software defects may lead to degradation of the quality and even sometimes extremely serious problems. it is important to make conscious efforts to control and minimize these defects by using techniques to allow in-process quality monitoring and control. In this paper, I compared Weibull, Rayleigh and Gamma models to check the goodness of the fit for the defect curses in the taken datasets and prediction accuracy with defect curves of prior release and using defect curves of other projects. [4]**

***Keywords*—Defect data analysis, Software metrics, Evaluation, Machine Learning Algorithms, Weibull model, Rayleigh model, Gamma model, defect curves and projection.**

I. Introduction

Due to increasing extent and complexity of software systems, it is common to have large teams, communities of developers working on the same project at the same time this may cause bugs. One example that burned up a 327.6 million project in minutes is in 1998, when the Mars Climate Orbiter built by NASA's Jet Propulsion Laboratory approached the Red Planet at the wrong angle. The biggest problem was that different parts of the engineering team were using different units of measurement. One group working on the thrusters measured in English units of pounds-force seconds; the others used metric Newton-seconds and this is not been checked, Even these small bugs can collapse the whole system.

Quality assurance activities, such as tests or code reviews, are an expensive, but vital part of the software development process as we can see from the above mentioned example. Any support that makes this phase more effective may thus improve software quality or reduce development costs. [2]

Defect-occurrence projection is necessary for the development of methods to mitigate the risks of software defect occurrences. The occurrence of defects is not only problem with the users, but also cause problems in maintenance planning for software makers. The costly consequences of defect occurrences have increased interest in insuring software consumers against the associated risks. [3]

In this experiment we aim to answer three questions:.

Question 1: How are defect curves of the same product similar to each other?.

Question 2: What are the modeling quality (i.e. the goodness of fit) of the defect curves using Weibull, Rayleigh, and Gamma models? To measure modeling quality, you should use R2 and AIC score. What is the best model?

Question 3: What are the prediction accuracy of the defect volume (for each time unit and for the whole reporting time) when using i) only defect curves of prior releases as training data, and ii) only defect curves of other projects?

So we will look at each question, discuss the method we tried to solve the problem and finally results that we obtained.

II. Datasets

A. What I understand from the datasets ?

Product	Releases	Software type	Cycle Time	License
Eclipse PDE	3	Programming IDE	3 months	Open Source
Eclipse TPTP	3	Agent Controller	3 months	Open Source
Apache httpd 1.3	3	HTTP server	3 months	Open Source

We work on three datasets for this project. Datasets are the bugs reported on software after their release date. we need to get reliable bug reports for this project. So I looked into two major software foundations - eclipse and apache. So I took two datasets from apache and one dataset from the eclipse. I downloaded eclipse dataset from bugs.eclipse.org and apache dataset from issues.apache.org. Each dataset contains three versions (for example 1.0,1.1,1.2) of one project example Tomcat,apache httpd,etc. From the date of release version in a particular project, we look at the number of bugs reported for each quarter (3 months) for 16 quarters after the release date. We repeat this for all the datasets to get the number of bugs for each quarter.

1) Apache Httpd 1.3 - It contains versions 1.3.26,1.3.27, 1.3.29.
2) Eclipse TPTP Agent Controller - It contains versions 4.0, 4.1, 4.2.
3) Eclipse PDE - It contains versions 3.2, 3.3, 3.4.

Each project contains 3 files that have of number of bugs per quarter for 16 quarters.

III. Analysis

A. Question 1: Similarities between the defect curves

Once I have the datasets formed, we plot them into graphs called defect curves. So we will be left with 3 defect curves

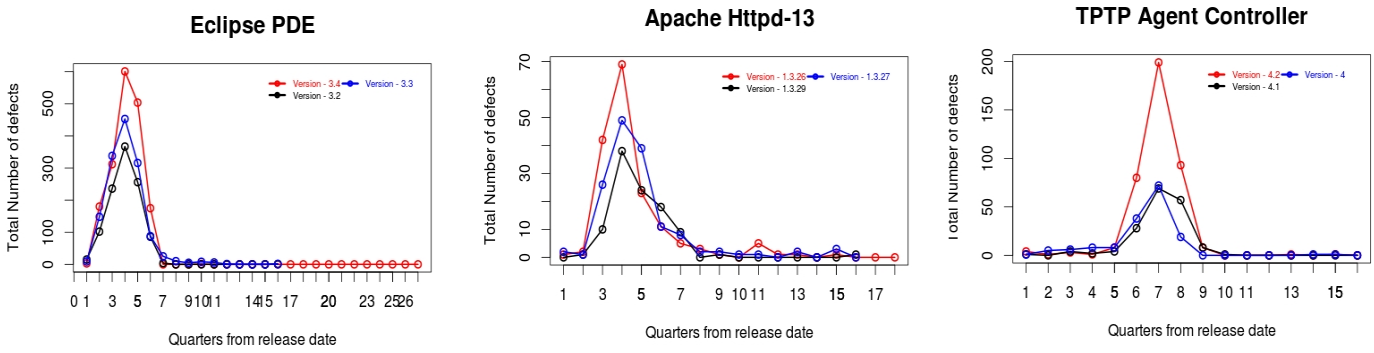

Fig. 1: Defect curves of three projects with respective versions.

for each project. Now I want to find the similarities for each version of a project. In order to find the similarities between two defective curves, I follow the following method

1) Scan the inputs and normalize both the curves.
2) After normalizing the defect curves we do 1 minus of difference of normalized values over 2.

we normalized the actual defect curves(so they become distributions of defects over time) and compared them to the other defect curves. To compare two distributions, We use the following similarity measure:

$$Sim\,(X,Y) = 1 - Diff\,(X,Y) = 1 - \frac{1}{2}\sum |X(t) - Y(t)|$$

This measure first computes the total area of difference between two distributions. If they are identical then $\sum t|X(t) - Y(t)| = 0$ If they are entirely disjoint then $\sum t|X(t) - Y(t)| = 2$, since the area of each distribution is 1. Therefore, we divide this sum by 2 to normalize their difference to the range of [0,1]. The output will give us the percentage of similarities between the defective curves. Figure 1 gives us the graphs for bugs vs different versions in a project. The x-axis is the number of the quarter and y-axis is the sum of number of bugs in that particular quarter. [5] All the projects have similar kind of graphs suggesting a trend is being established here. Curve similarity for the versions within the project is calculated by above mentioned method and are tabulated as follows

TABLE I: Defect curves similarities for versions in project

Projects	Versions	%	Versions	%	Versions	%
Eclipse PDE	3.4 3.2	93.1%	3.4 3.3	88.7%	3.3 3.2	93.6%
Apache Httpd 13	1.3.29 1.3.26	72.2%	1.3.29 1.3.27	81.2%	1.3.27 1.3.26	79.8%
TPTP Agent Controller	4.0 4.1	74.0%	4.1 4.2	84.6%	4.2 4.0	81.4%

B. Question 2: Goodness of fit

Here we use different models to get the modeling quality or the goodness of fit.

1) Weibull distribution : The probability density function of a Weibull random variable is

$$f\,(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & x \geq 0, \\ 0 & x < 0 \end{cases}$$

where k >0 is the shape parameter and lambda >0 is the scale parameter of the distribution. Its complementary cumulative distribution function is a stretched exponential function. The Weibull distribution is related to a number of other probability distributions; in particular, it interpolates between the exponential distribution (k = 1) and the Rayleigh distribution (k = 2).

2) A Rayleigh distribution is often observed when the overall magnitude of a vector is related to its directional components. The probability density function of the Rayleigh distribution is

$$f\,(x;\sigma) = \frac{x}{\sigma^2} e^{-x^2/\left(2\sigma^2\right)}, \qquad x \geq 0,$$

$where\, \sigma > 0\, is\, the\, scale\, parameter\, of\, the\, distribution$

3) In probability theory and statistics, the gamma distribution is a two-parameter family of continuous probability distributions.

$$f\,(x;k,\theta) = \frac{x^{k-1}e^{-\frac{x}{\theta}}}{\theta^k \Gamma(k)} \qquad x > 0\, and\, k,\theta > 0$$

$Here\, \Gamma\,(k)\ is\, the\, gamma\, function\, evaluated\, at\, k$

4) AIC score - The Akaike information criterion (AIC) is a measure of the relative quality of a statistical model, for a given set of dataset.AIC deals with the trade-off between the goodness of fit of the model and the complexity of the model. It is founded on information entropy: it offers a relative estimate of the information lost when a given model is used to represent the process that generates the data. AIC does not provide a test of a model in the sense of testing a null hypothesis; i.e.

TABLE II: Top metrics that are correlated with bugs in each software system

Projects		R^2			AIC		
Product	Version Number	Weibull	Gamma	Rayleigh	Weibull	Gamma	Rayleigh
TPTP Agent Controller	3.2	***0.99***	0.97	0.73	***105.61***	144.74	181.85
	3.3	***0.99***	0.98	0.76	***115.45***	148.42	187.91
	3.4	***0.99***	0.95	0.73	***233.73***	271.21	320.43
Apache Httpd13	1.3.26	0.96	***0.98***	0.56	107.90	***93.17***	145.94
	1.3.27	0.97	***0.98***	0.60	85.15	***75.13***	122.74
	1.3.29	0.88	***0.94***	0.58	94.50	***83.60***	114.05
Eclipse PDE	4.0	***0.98***	0.97	0.29	***84.04***	91.42	193.88
	4.1	***0.99***	0.14	0.27	***67.98***	148.94	182.74
	4.2	***0.99***	0.05	0.27	***88.50***	178.59	182.74

AIC can tell nothing about the quality of the model in an absolute sense. If all the candidate models fit poorly, AIC will not give any warning of that. In the general case, the AIC is:

$$AIC = 2k - 2ln(L)$$

where k is the number of parameters in the statistical model, and L is the maximized value of the likelihood function for the estimated model.

5) Coefficient of determination (R^2) : the coefficient of determination, denoted R2 and pronounced R squared, indicates how well data points fit a statistical model sometimes simply a line or curve. It is a statistic used in the context of statistical models whose main purpose is either the prediction of future outcomes or the testing of hypotheses, on the basis of other related information. It provides a measure of how well observed outcomes are replicated by the model, as the proportion of total variation of outcomes explained by the model.

The goodness of fit of a statistical model describes how well it fits a set of observations. Measures of goodness of fit typically summarize the discrepancy between observed values and the values expected under the model in question. Such measures can be used in statistical hypothesis testing, e.g. to test for normality of residuals, to test whether two samples are drawn from identical distributions, or whether outcome frequencies follow a specified distribution. In the analysis of variance, one of the components into which the variance is partitioned may be a lack-of-fit sum of squares.

After we find the defect curves we need to fit the defect curve into the three models we discussed above - Weibull, Rayleigh, and Gamma models. For this we are going to use non-linear least square. Non linear least square is the form of least squares analysis used to fit a set of m observations with a model that is non-linear in n unknown parameters (m $>$ n). It is used in some forms of non-linear regression. The basis of the method is to approximate the model by a linear one and to refine the parameters by successive iterations. There are many similarities to linear least squares, but also some significant differences.

I try to fit the defect curve with the already available models by changing the shape and scale parameters. With the non linear least squares we can assess r square,total number of bugs, AIC estimate values. I repeated this for all three models, all versions, and on all the projects and are plotted on the table 2.

The better the r square value the better is the model, the lower the AIC value the better. According to the table some interesting facts are bold. I observe for most versions the Weibull models is best in both r square and AIC values (though r square and AIC values are related). But for the project Apache httpd 13 this is not the case, interestingly gamma model takes over the Weibull model, the margin between Weibull and gamma for the project is not much but gamma outruns Weibull. One other interesting thing I can observe from the table is Weibull is always performs in a good way, while gamma does not do good most cases, but in one project outforms other two models. Rayleigh performs average in all cases which is good in few cases if we need consistency.

For the projects I took, the best model is Weibull.

C. Question 3: Prediction accuracy

One of the main questions we have to ask while dealing with software analytics is can we use the present data as training set and predict future values. This is main part we will looking forward to, because it helps the software developers to get to know things ahead of time and try to prevent or other ways to make the customers stress free usage. [1] The prediction accuracy of the defect volume when using only defect curves of prior releases as training data changes when you change how many quarters we are taking as training data. as one can guess, the more number of quarters we take, the better the prediction is. Figure 2 gives the error we got when we used prior versions to predict the latest version. for eclipse version 3.2 gives better prediction for version 3.4. similarly others. Similarly we find the errors for other two projects using their prior versions. By plotting the graphs I observed that immediate prior version predict better than other versions. I can say this by looking into the graphs error axis(y-axis).

The next question we are looking into is rather very interesting. Till now we are tying to predict the defect volume

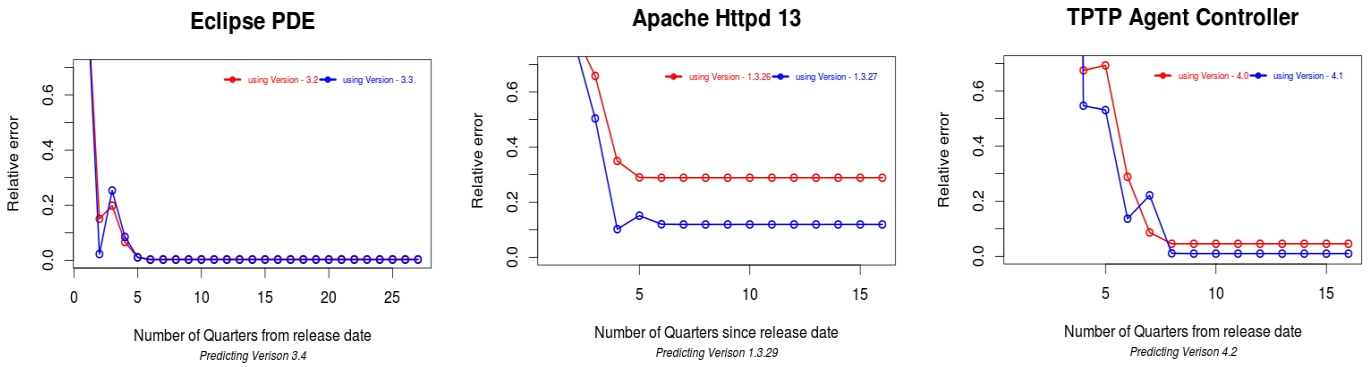

Fig. 2: Predicting defect value using prior versions.

by looking into the prior versions, but now can we predict the defect volume by using different product estimates. We

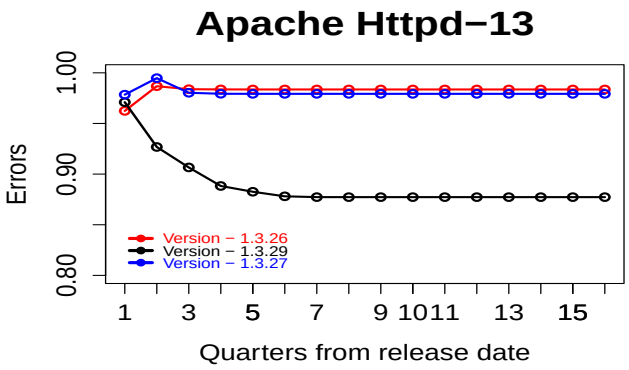

Fig. 3: Predicting using other projects

use different parameters like cost,license,shape,scale of the other project values to predict the current version. I used the project versions of Eclipse PDE and eclipse TPTP as training sets. These training sets forecast the values of product Apache httpd-13. Surprisingly, only version 1.3.29 is effected when we increase the number of quarters for the training. This should be observed for other projects also, since if we increase the number of quarters then we can predict with more accuracy but for the version 1.3.26, 1.3.27 it didnot matter how many quarter we send.

IV. Conclusion

In this Projects, we have taken an empirical, data-driven approach to get the defects reported in the open source software products using a large collection from issues.apache.org and bugs.eclipse.org. Using models such as Weibull, Rayleigh, and Gamma, we are able to fit the defect curse accurately to get the shape and scale parameters. With the help of scale, shape, cost and other parameters we can predict the defect volume of the present version. Both prior releases and to my surprise other project versions can also be used to predict the defect volume of the current version. With the datasets I took, Weibull is the best model when compared to Rayleigh, and Gamma models.

V. Future Work

This project can be extended to attain accurate results. Taking a much larger training set, using other models with accurate parameters can be helpful projecting the defect curves. This can be extended to other projects to train the models better and get definite parameters for forecasting.

Acknowledgment

I would Like to thank Dr Tung for his excellent guidance for this project.

References

[1] W.D. Jones. Reliability models for very large software systems in industry. In *Software Reliability Engineering, 1991. Proceedings., 1991 International Symposium on*, pages 35–42, May 1991. doi: 10.1109/ISSRE.1991.145351.

[2] F. Khomh, T. Dhaliwal, Ying Zou, and B. Adams. Do faster releases improve software quality? an empirical case study of mozilla firefox. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pages 179–188, June 2012. doi: 10.1109/MSR.2012.6224279.

[3] Paul Luo Li, Mary Shaw, Jim Herbsleb, Bonnie Ray, and P. Santhanam. Empirical evaluation of defect projection models for widely-deployed production software systems. *SIGSOFT Softw. Eng. Notes*, 29(6):263–272, October 2004. ISSN 0163-5948. doi: 10.1145/1041685.1029930. URL http://doi.acm.org/10.1145/1041685.1029930.

[4] Shwetha Rameshan. Overview of Rayleighs Defect Prediction Model. http://ig.obsglobal.com/2013/12/overview-of-rayleighs-defect-prediction-model/, 2013. [Online;].

[5] Tim Klinger P. Santhanam Tung Thanh Nguyen, Evelyn Duesterwald and Tien N. Nguyen. Characterizing defect trends in software support. June 2012.