SOFTWARE TESTING DEFECT PREDICTION MODEL-A PRACTICAL APPROACH

Shaik Nafeez Umar

Lead-Industry Consultant, CSC Company, Hyderabad, Andhra Pradesh, India
nshaik5@csc.com*, *nafishaik123@gmail.com

Abstract

Software defects prediction aims to reduce software testing efforts by guiding the testers through the defect classification of software systems. Defect predictors are widely used in many organizations to predict software defects in order to save time, improve quality, testing and for better planning of the resources to meet the timelines. The application of statistical software testing defect prediction model in a real life setting is extremely difficult because it requires more number of data variables and metrics and also historical defect data to predict the next releases or new similar type of projects. This paper explains our statistical model, how it will accurately predict the defects for upcoming software releases or projects. We have used 20 past release data points of software project, 5 parameters and build a model by applying descriptive statistics, correlation and multiple linear regression models with 95% confidence intervals (CI). In this appropriate multiple linear regression model the R-square value was 0.91 and its Standard Error is 5.90%. The Software testing defect prediction model is now being used to predict defects at various testing projects and operational releases. We have found 90.76% precision between actual and predicted defects.

Index Terms: *Software defects, SDLC, STLC, Multiple Linear Regression*

----------***----------

1. INTRODUCTION

In the past thirty years, many software defect prediction models have been developed. In the software testing/development organization, a need for release/project wise better defect prediction models. Predicting the defects in testing projects is a big challenge. Software development organizations have been working on making good plans to achieve better development, maintenance and management processes by predicting the defects. Companies spend huge amount of money in allocating resources to testing the software systems in order to find the defects. If we can have a model to predict the defects in the release/project, the schedule variance can be minimized and can be received excellent customer satisfaction. Evaluation of many software models were presented in [1, 2, 3 and 27]. Statistical based models of software defects are little help to a Project Manager who must decide between these alternatives [4].

Software defects are more costly if discovered and fixed in the later stages of the testing and development life cycles or during the production [5]. Consequently, testing is one of the most critical and time consuming phase of the software development life cycle and accounts for 50% of the total cost of development [5]. Defect predictors improve the efficiency of the testing phase in addition to helping developers evaluate the quality and defect proneness of their software product [6]. They can also help managers in allocating resources, rescheduling, training plans and budget allocations. Most defect prediction models combine well known methodologies and algorithms such as statistical techniques [7, 8 and 9] and machine learning [10, 11, 12 and 13] they require historical data in terms of software metrics and actual defect rates, and combine these metrics and defect information as training data to learn which modules seem to be defect prone.

Recent research on defect prediction shows that AI based defect predictors can detect 70% of all defects in a software system on average [14], while manual code reviews can detect between 35 to 60% of defects [15] and inspections can detect 30% of defects at the most [16]. A number of authors, for example [17, 18 and 19] have newly used Bayesian Networks models in software engineering management. Bayesian Networks models can useful to predict number of software defects remaining undetected after testing [20], this can be used project managers in particularly help to decide when to stop testing and release software, trading-off the time for additional testing against the likely benefit.

2. OBJECTIVE AND METHODOLOGY

- Defect prediction improves efficiency of the testing phase in addition to helping developers evaluate the quality and defect proneness of their software product.
- Help managers in allocating resources, rescheduling, training plans and budget allocations.
- Depending on the forecasted trends:
    - Resources can be efficiently ramped up or down
    - Gaps in Skills and trainings can be plugged
- Predicts defect leakage into production.

3. OBJECTIVE AND METHODOLOGY

The objective of this paper is to predict software testing defects using statistical models and evaluate the accuracy of the statistical defect prediction model. To determine potential of the statistical models to capture the number of defects on the basis of past data and their metrics, we proceed as follows. To identify the best predictors in the available set of 18 parameters, we calculate product moment correlation between the number of defects and the predictors. Then we proceed with more advanced statistical models to deal with normal distribution of the target variable and the specifics of the historical data and check multicollinerity within predictor parameters.

The key variables we analyzed were the descriptive parameters of target variable. As a result for the analysis, we used a dataset of 20 software releases which is correlated with among the influence variables. The parameters of the generalized Multiple Linear Regression are first estimated using the method of ordinal least squares (OLS). The application of the OLS method for fitting the parameters of the generalized linear regression model is justified if error distribution is assumed to be normal. In this case OLS and the maximum likelihood (ML) estimates of β are very close the same for the linear model [21 and 27]. We use Multiple Linear Regression (MLR) for the estimation of the defect using the five predictors with correlation coefficients. In the predictive modeling, the multiple regression models are used for predicting defects in software field.

$$Y=\beta 0+ \beta 1X1+ \beta 2\ X2+ \beta 3\ X3+ \beta 4\ X4+\ldots\ldots\ldots\ldots..+ \beta n\ Xn,$$

Where Y=Dependent parameter (Defects),
β1, β2, β3, β4,……. βn Coefficient values and X1, X1, X1, X1,…. X1 are independent parameters (Total number of test cases executed, Test team size, Allocated development effort, Test case execution effort and Total number of components delivered)

4. RESULTS AND DISCUSSION

Out of total 20 software data items, we have identified total number of Test Cases executed, Test Team size, Allocated Unit testing effort, Test case execution Effort and Total numbers of components delivered are independent variables and number of defects as dependent variable. We have identified dependent variables based on the past data and experience. We have been observing the same pattern in many of our projects. Total number of defects depends on total number of test cases and is directly proportional. If number of test cases are high and critical to requirements the chances are getting defects is high. This is one of the strongly influencing parameter. There is strong correlation between these two parameters. Number of defects in the software release/project is directly proportional to Test team size. If test team size is more, then probability of getting defects is high. Number of defects in the software release/project is indirectly proportional to allocated unit testing effort. If the allocated unit testing effort is more, then developers will be able to find more number of defects in the unit testing phase only and leakage of the defects to the next phase will be minimized. This is one of the strong parameter negatively influencing on the number of defects. If the allocated unit testing effort is less then probability of getting defects are very high in the testing phase. Number of defects in the software release/project is directly proportional to test case execution effort. If the test case execution effort is more, then probability of getting defects is high. Number of defects in the software release/project is directly proportional to total number of components delivered. If the total number of components delivered is more, then probabilities of getting defects are high.

In literature, various authors have also used other type of metrics like KLOC for prediction model, while this for simple liner regression [14]. It was strongly associated between defects. Of the 20 releases, the mean defects were 28 and its standard deviation is 16 while the mean total number of test cases executed 264. The proportion of test cases executed and defects rate are 9:1. The average allocated unit testing effort was 433 and SD is 226, correlation coefficient a value is 0.2441, it is not significant at 0.05 levels. It means it was no associated with number of defects. Product moment correlation coefficient is most widely used for calculating correlations and its significance level, except it assumes normality in the distribution of analyzed variables. As our statistical predictive model variables were clearly normally distributed. There is highly significant difference between defect and test team size, it was 0.7665($p<0.01$) and next followed by total number of components delivered i.e. 0.6485 ($p<0.01$) [table 1].

The coefficient values of multiple linear regressions presents β values were positive and negative and low its standard errors i.e. Total number of test cases executed, test team size, allocated unit testing effort, test case execution effort and total number of components. We observed test team size only significantly at 0.05 levels [table 2].

It seems to graphical representation of actual and estimated defects, R1 and R2 releases were vary from actual and estimated defects. While next to continued R3 to R15 were same pattern, it was no variation among the trend lines. In the statistical defect prediction model shows next five releases data point to be predicted and it will be given number of defects bases on release requirements [graph 1].

We used Analysis of variance (ANOVA) for good fit of model also its r-square, the predictive model coefficient of determination is 0.91, and it means 91% of the variation in defects is associated with number of predictors. The F-ratio is

26.37 ($p<0.01$) which is highly significant at 0.01 levels. Finally the standard error (SE) was 5.90% it is very low error in this prediction model. In this model we compare the actual defects and estimated defects, both of the same patterns and its precision was 90.76%. We used error analysis for validation of the model. We assumed that the errors followed by normality and mean and variance are constant.

CONCLUSION

We analyzed an extensive historical dataset of software releases to identify factors influencing parameters of defect prediction model. We found strong correlation between defects and test team size, total number of test cases executed, total number of components delivered. In this analysis we use multiple regression analysis for estimating software defects. These models is account for the typical problems for identifying and influence parameters as well as metrics data, such as inconsistent and heterogeneity of data. Overall these models indicate good prediction for upcoming software defects as well as quality improvement. By using the predictive model identified project manager's key parameters that require controlling and improve both the development and testing process. The R-square value was 0.91 (91%) which is highly significant at 0.01 level and low standard error (SE) was 5.90%.

We calibrated our model based on its prediction accuracy and discovered that it is possible to detect on 84% of number of defectives using a defect predictors.

We will continue this work by:

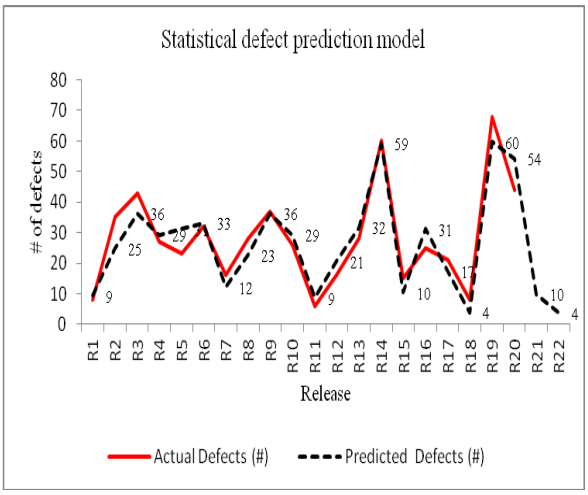
- Extending the analysis by analyzing the impact of different combinations of factors on proportion of defect types.
- Extending the model to predict the Defect Severity Index (DSI).
- Extending the model to predict the defect leakage into the production or next phases.
- Extending the model to take a decision like GO or No-GO into the production.
- Extending the model with the results of newer analysis using advanced statistical models.
- Validating the model against historical data and assumptions in software engineering.

Table 1 Illustrates the descriptive statistics like mean and standard deviation values of final model parameters.

Multiple Linear Regression (MLR)	Coefficients values	Standard Error (SE)	Significant P-value
Intercept	-5.36813	5.641756	p>0.05
Total number of Test Cases executed (#)	0.019694	0.013713	p>0.05
Test Team size (#)	6.238706	0.91904	p<0.01
Allocated Unit testing effort (%)	-0.02611	0.007791	p<0.01
Test case execution Effort (%)	-0.07895	0.013876	p<0.01
Total number of components delivered (#)	-0.15725	0.324758	p>0.05

Table 2 significant values of Multiple Linear Regression (MLR)

Parameter	Mean	Standard Deviation (SD)	Correlation	p-value
Total number of Test Cases executed (#)	264	169	0.5425	p<0.05
Test Team size (#)	10	4	0.7665	p<0.01
Allocated Unit testing effort (Hrs)	433	226	0.2441	p>0.05
Test case execution Effort (Hrs)	252	188	0.0593	p>0.01
Total number of components delivered (#)	11	11	0.6485	p<0.01
Number of defects (#)	28	16	-	-



Graph 1 Trend pattern of Actual and estimated software defects

ACKNOWLEDGEMENT

I want to thank Narendra singh Rajput, Lead Associate Manager, for his valuable feedback and guidance and Prof Balasiddamuni, Department of Statistics, S V University, Tirupati and Raghav Beeram, Global Head - Test Management Consulting, Independent Testing Services, CSC, Hyderabad for guiding me and supporting me in my research paper.

I would like to express my love affection to my mother Smt. Shaik Zaibunnisa, My Father Sri. Late S. Mahaboob Basha, Brothers and Sisters, My spouse Ballary Ameen, Children's Shaik Mohammed Wafeeq, Shaik Waneeya, my Niece Samreen Aaleia for enthusing me throughout the work

REFERENCES

[1] M. Boraso, C. Montangero, and H. Sedehi, "Software cost estimation: An experimental study of model performances," tech. report, 1996.
[2] T. Menzies, D. Port, Z. Chen, J. Hihn, and S. Stukes, "Validation methods for calibrating software effort models," in ICSE '05: Proceedings of the 27th international conference on Software engineering, (New York, NY, USA), pp. 587–595, ACM Press, 2005.
[3] O. Benediktsson, D. Dalcher, K. Reed, and M. Woodman, "COCOMO based effort estimation for iterative and incremental software development," Software Quality Journal, vol. 11, pp. 265–281, 2003.
[4] Fenton, N. E. and Neil, M. A Critique of Software Defect Prediction Models, IEEE Transactions on Software Engineering, 25(5), 675-689, 1999.
[5] Brooks, A. The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley. Eds. 1995.
[6] Neil, M., Krause, P., Fenton, N. E., Software Quality Prediction Using Bayesian Networks in Software Engineering with Computational Intelligence, (Ed Khoshgoftaar TM), Kluwer, ISBN 1-4020-7427-1, Chapter 6, 2003.
[7] Nagappan.N., Ball, T., Murphy, B. Using Historical In Process and Product Metrics for Early Estimation of Software Failures, In Proceedings of the International Symposium on Software Reliability Engineering. NC. 2006.
[8] Ostrand. T.J., Weyuker E.J., Bell, R.M Predicting the Location and Number of Faults in Large Software Systems, IEEE Transactions on Software Engineering (31), 4:340-355, 2005.
[9] Zimmermann, T., Weisgerber, P., Diehl, S., Zeller, A. Mining version histories to guide software changes. In proceedings of the 26th International Conference on software Engineering, 563-572, IEEE Computer Society, DC, USA, 2004.
[10] Munson, J.C., Khoshgoftaar, T.M. The detection of fault-prone programs. IEEE Transactions on software Engineering (18), 5: 423-433. 1992.
[11] G. Succi, M. Stefanovic, W. Pedrycz Advanced Statistical Models for Software Data, Proceedings of the 5th World Multi-Conference on Systems, Cybernetics and Informatics, Orlando, Florida, 2003.
[12] Lessmann. S., Basens. B., Mues., C. Pietsch. Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings. IEEE Transactions on Software Engineering (34), 4: 1-12. 2008.
[13] Moser, R., Pedrycz, W., Succi. G. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction, In Proceedings of the 30th International Conference on software Engineering, 181-190. 2008.
[14] Menzies, T., Greenwald, J., Frank, A. Data mining static code attributes to learn defect predictors. IEEE Transactions on Software Engineering (33), 1: 2-13. 2007.
[15]BogaziciUniversity, http://code.google.com/p/prest/. Shull, F., Boehm, V.B., Brown, A., Costa, P., Lindvall, M., Port, D., Rus, I., Tesoriero, R., and Zelkowitz, M. 2002. What We Have Learned About Fighting Defects. In Proceedings of the Eighth International Software Metrics Symposium, 249-258, 2002.
[16] Fagan, M. Design and Code Inspections to Reduce Errors in Program Development. IBM Systems Journal (15), 3. 1976.
[17] Bibi, S. and Stamelos, I. Software Process Modeling with Bayesian Belief Networks In Proceedings of 10th International Software Metrics Symposium (Metrics2004) 14-16 September 2004, Chicago, USA.
[18] Fan, Chin-Feng, Yu, Yuan-Chang. BBN-based software project risk management, J Systems Software, 73, 193-203, 2004.
[19] Stamlosa, I., Angelisa, L., Dimoua, P., Sakellaris, P. On the use of Bayesian belief networks for the prediction of software productivity Information and Software Tech, 45 (1), 51-60, 2003.
[20] Fenton, N.E., Krause, P., Neil, M., Software Measurement: Uncertainty and Causal Modeling, IEEE Software 10(4), 116-122, 2002.

[21] Fenton, N.E. and Neil, M.A Critique of Software Defect Prediction Models, IEEE Transactions on Software Engineering, 25(5), 675-689, 1999.
[22] Cameron A.C. and Trivedi P.K. Econometrics models based on count data: comparisons and applications of some estimators and tests, Journal of Applied Econometrics, 1, 29-93, 1986.
[23] Lambert D. Zero-inflated Poisson regression with an application to defects in manufacturing, Technimetrics, 34, 1-14, 1990.
[24] Long J.S. Regression Models for Categorical and Limited Dependent Variables, Advanced Quantitative Techniques in the social Sciences, No 7, Sage Publications.1997.
[25] Graves T., Karr A.F., Marron J.S., Siy H. Predicting Fault Incidence Using Software Change History, IEEE Transactions on Software Engineering, Vol.26, No.& July. 2000.
[26] Stamelosa, I., Angelisa, L., Dimoua, P., Sakellaris, P. On the use of Bayesian belief networks for the prediction of software productivity Information and Software Tech, 45 (1), 51-60, 2003.
[27] Nafeez Umar Shaik, Bathineedi Koteswara Rao and Beeram Raghav, Software defect prediction model: A statistical approach, Software Testing Conference (STC-2010), 10th Annual International Software Testing Conference in India 2010, November, 22-23, Bangalore.

BIOGRAPHIES

Working as Lead Industry Consultant in Test Management Consulting group, AppLabs – a Computer Science Corporation Company, Hyderabad, India. He has 9 years in IT experience as Statistician and Statistical modeller. He has published 14 papers in National/International journals. He is expert in Statistical and Econometric modelling