Volume 4, Issue 9, September 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Predicting Changeability in Software Components Using Bisquare Method for Optimization

Ankita Urvashi [1], Anamika Chhabra [2]
[1]Computer Science, Indo Global College of Engineering, India
[2]Computer Science, Indo Global College of Engineering, India

Abstract : In this research paper, we have current scientific datasets which are used for prediction may consists data points which may not following a particular trend due to large variability within short intervals and, for finding a mathematical relation between the two random variables becomes a challenge as it does not fit into a particular distribution equation to really represent the mathematical relationship . Therefore a need arises to have a custom mathematical equation which can help us to define a trend analysis between the two variables.Hence there arises a need to optimize iteratively the datasets to finally arrive at best fit line. A similar scenario was observed when change-proneness has a function of factors include[WMC,DIT,NOC,CBO,RFC,LOCM,Ca,NPM] and then bisquare method used for optimization, which produces a valid data model that has been proved by using multiple statistical tests like coefficient of goodness(s),error measure, slope of rate of change(e),R-square.. Previous work have assumed that all trends relationship are linear in nature, while dataset collected for shows it is highly non-linear. Therefore, in our current research work we have found non-linear data fitting algorithm bi-square robust that gives best possible results as it more accurate with realistic data with trend and pass through most of statistical test of significances ,thus we get a promising approach to measure change-proneness of the software development process which has not been used previously.

Keywords - Application Development, Change-Proneness, Software Metrics, Software Stability

I. INTRODUCTION

. Changeability is the ease with which a source code can be modified. It is assessed through metrics calculated from the history of changes made. These metrics reflect how well or bad is the change for the project in terms of it degree however, the choice of these metrics is matter of real concern as they must be chosen in such a manner that they must measure the true image and state of the software components with respect to the basic principles of software stability with openness for further change .Therefore, in next section of this research paper we discuss how previous and contemporary researchers have address this issue.

Which gives us function equation to define the relationship of each metric with respect to change-proneness?, There is an urgent need for presenting the of various statistics for measuring the change-proneness for knowing what is wrong with our project in progress so that there is no burn out between the stakeholder working in the life cycle of the application. Since, components of software are like organic compounds that change internally and externally with multiple environmental and business reasons, for usage of software to continue, and for it to remain non obsolete it needs to remain constant state of change to remain in sync with the real business life. The major concern is maintenance and further development of the software without conflicts, issues and bugs , therefore , when software components undergo adaptation , enrichment and feature additions there is always a risk of too much change leading to change is overall structure and form of the software itself that it may lead to the huge burnout between the stake holders of the project in progress .in fact, changeability is defined as a measure of impact of changes made to a module on the rest of the system

II. RELATED WORK

Many change-proneness measures for the class-objects has based on the size and complexity. Some of the case studies been carried out for linking change-proneness and complexity so as to know if ay correlation exists between change-proneness and metrics. It is difficult to determine how far an individual study can be generalized and moreover the data gathered over longer project duration may reveal different patterns as a decreasing number of components are left unchanged. [2] Many other methods are described to identify and visualize class interactions that are the most change-prone. That method was object-oriented software developed using design patterns [3]. Even design patterns that are actually used to develop software designs that are less prone to lot of changes. Besides these many other case studies even concluded that change-proneness can be detected between the components with high or with low values for metrics providing an area beneficial for further studies. [4] based on the systematic literature review. We have formulated scope of work in section.

III. PROBLEM FORMULATION

It is difficult to calculate the change prone ness of the software due to inherent nature the software itself. The software continuously under goes evolution and is not a static product in itself. The software continuously needs adapt to the real life business scenarios to remain in realm of usage or else it will be obsolete and of no use for current uses , this of course leads to change in structure and surface of software landscape such that it may endanger the very purpose of the software components. Therefore, a clear understanding of change proneness is required and we need to find a way to identify the highly change prone areas of the software to as keep the software live and relevant to the current users. Now, this can done we analysis the classes /routines involved in the interaction of change which may include redesign , refactoring and change in levels of intimacy between the class routines . Do doing this proper selection of metrics that can measure the change proness is critical and then the nature of the dataset produced out of that , previous studies have assumed that this dataset can be considered and understood in linear space model but it seems the changeproness metric calculation might not follow a linear fitting solution to find the intercepts of corresponding function of changeproness [f(y)].Therefore , the solution is find an non linear method of finding the values of function of changeproness for better understanding of changeproness and fitting model that can find real trends in the non linear space model which is more real and closer to the nature of dataset of metrics used to measure changeproness .

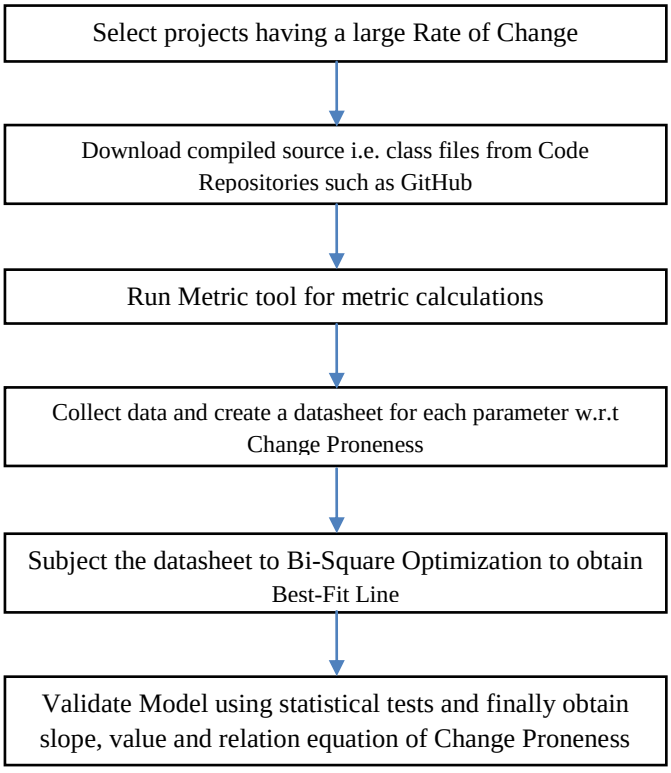
IV. PROPOSED METHOLOGY

The proposed methodology here tries to fulfil the basic objective of measuring the change-proneness in terms of its rate in change influenced by the number of contributors to the code production and the code metrics that significantly influence the change in the code with respect to the classes authored, furthermore, it attempts to find the degree of change-proneness [low, medium and high] degree.

F(y) = {(number of addition+ number of deletion/ total commits}*number of contributors

Here F(Y)=measure of chage-proneness

Workflow



V. RESULTS AND DISCUSSION

When there are multiple factors contributing to a particular function *f(y)* and the nature of the data is nonlinear , few assumptions can be really taken to drive some mathematical expression , therefore , in this research work we have analyzed each factor contributing to the change-proneness in such a manner that the data is fitted with multiple combination of data fitting techniques to finally arrive at particular logical mathematical expression which would represent the true relation representing the independent and dependent variable . The basic approach is to curve fitting, which is the process of constructing a curve, or mathematical function, that has the best fit to a series of data points, [the change-proneness metrics] possibly subject to constraints.[6] [Conformance, standards and best practices in software development] Curve fitting can involve either interpolation, where an exact fit to the data is required, or building a smoothing, in which a "smooth" function is constructed that approximately fits the data. A related topic is regression analysis, which focuses more on questions of statistical inference such as how much uncertainty is present in a curve that

is fit to data observed with random errors. [7] Therefore, to completing this purpose we conducted many experiments which would help find the best suited combinational optimization method for finding the relationship as shown in the Table 5.2

It is apparent from the various experiment conducted that bi-square optimization method is the best possible combination which can really help us to get the mathematical expression representing the change-proneness as function of the metrics, However, it must be noted that, graphical measures are more beneficial than numerical measures because they allow you to view the entire data set at once, and they can easily display a wide range of relationships between the model and the data. The numerical measures are more narrowly focused on a particular aspect of the data and often try to compress that information into a single number. In practice, depending on your data and analysis requirements, you might need to use both types to determine the best fit.

5.1 Interpretation of Statistical Test conducted by the above experiments

a) Coefficient of Goodness: This is value which shows how much is the model successful in fitting or building the model, Sometimes, it is possible that none of fits can be considered suitable for the dataset in question, based on these methods. In this case, it might be that you need to select a different model or some other combination, that why we designed so many combinational experiments. Hence, the goodness-of-fit measures indicate that a particular fit is suitable have been measured here whose value range between 0.9116 to 0.9130 that shows relation is best for NPM.

b) Error Measures: The Sum of squares due to error is statistic measures that totals the deviation of the response values [change-proneness vs. metrics] from the fit to the response values. It is also called the summed square of residuals and is usually labeled as SSE, from this we are able find how far is the data fitting, it is acceptable for us or not, for each metric the value has been calculated and it lies between 2.718e+05 to 2.763e+05 that shows minimum error is there for NOC.

5.2 Slope (Rate of Change)

With reference to table no 5.2 we can measure how the x axis changes with respect to the y-axis, from which the rate of change per metric can be known and understood. The slope of a regression line (*b*) represents the rate of change in *y* as *x* changes. Because *y* is dependent on *x*, the slope describes the predicted values of *y* given *x*. When using the ordinary least squares method, one of the most common linear regressions, slope, is found by calculating *b* as the covariance [8].Its range lies between -3.471 to 0.1208.

$$\frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sum_{i=1}^{n}(x_i-\overline{x})^2} \quad (1)$$

a) Of *x* and *y*, divided by the sum of squares (variance) of *x*, the slope must be calculated before the *y*-intercept when using a linear regression, as the intercept is calculated using the slope. The slope of a regression line is used with a t-statistic to test the significance of a linear relationship between *x* and *y*. But in our case , we had to make valid data model and had to further optimize the slope curve using bi-square optimization method to best fit the line and get the appropriate slope *'m'* representing the rate of change in software methods .

b) R-Square : This measures helps how far good the model is able to explain the variations in the datasets fitted by the algorithm , it should be equal to 1 ideally for it to explain the complete the complete variation, and it also explains the correlation between the predicted values (change-proneness) against the response values (metrics) .

VI. TABLES

6.1 Table: Relationship between Metrics and Change-Proneness

S No.	Metric Name	Relationship with Change Proneness
1.	WMC (Weighted Methods per Class)	High WMC decreases reusability and Code Quality
2.	DIT (Depth of Inheritance)	More the classes are inherited deeply makes them interdependent and poor Code Quality
3.	NOC (Number Of Children)	High NOC states high reusability thus less proneness to change
4.	CBO (Coupling Between Object Classes)	High CBO straightaway makes code interdependent thus minute changes too affect coupled classes thus more proneness to change
5.	RFC (Response For a Class)	It measures the number of different methods which may be executed when a method invocation takes place for a specific object and hence increases Change Proneness
6.	LCOM (Lack Of Cohesion Of Methods)	In cases where no common fields are accessed by class methods lead to new class thus Change Proneness occurs
7.	Ca (Afferent couplings)	High Ca leads to Change Proneness
8.	NPM (Number Of Public Methods)	If NPM is high it makes class highly accessible and hence susceptible to Change Proneness

RESULT USING BISQUARE METHOD

6.2 Table: Results

Method	**Mathematical Equation model**	**Coefficient of Goodness**	**Error**	**Slope (Rate of Change)**	**R-square**
WMC	y=m*x + c	0.9126	2.734e+05	-2.179	0.912047
DIT	y=m*x + c	0.9116	2.763e+05	-1.168	0.911306
NOC	y=m*x + c	0.9131	2.718e+05	-3.471	0.911014
CBO	y=m*x + c	0.9119	2.754e+05	0.04461	0.911999
RFC	y=m*x + c	0.9123	2.743e+05	-0.4156	0.912475
LCOM	y=m*x + c	0.9126	2.733e+05	-1.008	0.912402
Ca	y=m*x + c	0.9127	2.731e+05	-2.179	0.911669
NPM	y=m*x + c	0.913	2.72e+05	0.1208	0.911978

RESULT USING NON-BISQUARE METHOD:

Method	**Mathematical Equation model**	**Coefficient of Goodness**	**Error**	**Slope (Rate of Change)**	**R-square**
WMC	y=m*x + c	0.00016	3126853	-2.179	0.00688
DIT	y=m*x + c	0.00047	3125855	-1.168	0.00656
NOC	y=m*x + c	0.00045	3113016	-3.471	0.00243
CBO	y=m*x + c	0.00017	3116242	0.04461	0.00346
RFC	y=m*x + c	0.00016	3122228	-0.4156	0.00539
LCOM	y=m*x + c	0.00306	3117764	-1.008	0.00395
Ca	y=m*x + c	0.00546	3110250	-2.179	0.00153
NPM	y=m*x + c	0.00038	3126160	0.1208	0.00666

REASONS TO CHOOSE BISQUARE METHOD:

- Efficiency of an estimator may change significantly if the distribution changes, often dropping. This is one of the motivations of bisquare based robust statistics method – an estimator such as the sample mean is an efficient estimator of the population mean of a normal distribution, for example, but can be an inefficient estimator of a mixture distribution of two normal distributions with the same mean and different variances. For example, if a distribution is a combination of 98% N(μ, s) and 2% N(μ, 10s), the presence of extreme values from the latter distribution (often "contaminating outliers") significantly reduces the efficiency of the sample mean as an estimator of μ. By contrast, the trimmed mean is less efficient for a normal distribution, but is more robust (less affected) by changes in distribution, and thus may be more efficient for a mixture distribution like used in our software metrics dataset.
- Many of the software metrics are "long-tailed" (relative to the normal distribution) is,and distributions of that sort,and researchers need small p-values, if you use methods that make stronger distributional assumptions but in our we cannot , hence we need methods like bisquare. In other words, non-bisquare method gives more of false positives.
- On the other hand, other approaches in our reseach are not robust because they are dedicated to one single type of model like normal model but this is not the case when we apply iteratively bisqure method
- It is based on roboust regression and least absolute deviation from performace goal in optimization .
- Fitting for finding rate of change requires a parametric model that relates the response data to the predictor data with one or more coefficients. The result of the fitting process is an estimate of the model coefficients.To obtain the coefficient estimates, the least-squares method minimizes the summed square of residuals, hence , This method minimizes a weighted sum of squares, where the weight given to each data point [software metric value] depends on how far the point is from the fitted line [slope or trend line giving use rate of change]. Points near the line get full weight. Points farther from the line get reduced weight. Points that are farther from the line than would be expected by random chance get zero weight, which is done best by bisquare method , hence we get best fitted trend line

VII. GRAPHS MADE USING BISQUARE METHOD OF OPTIMIZATION

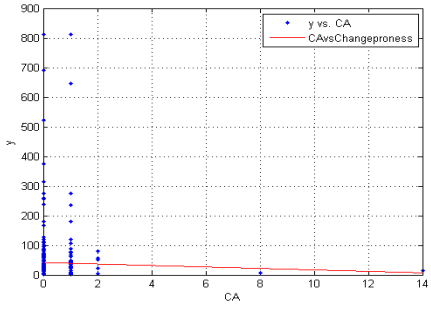
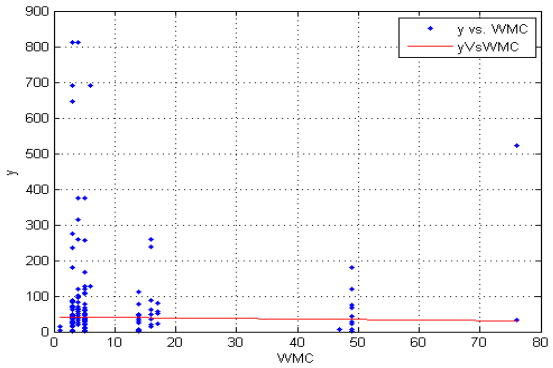


Figure 7.1: Coupling Afferent

It is apparent from the graph that this metric directly proportional to change-proneness, however, here we can see the value is in decreasing as more and more components are found in representative five projects. This shows that with passage of time when more components are added the metric is getting mature and stable, hence there is not much change occur and it is more or less not steady or downwards in nature rather than directly upward proportional.



The data points basically represent the 'weighted methods per class, which class to be given more priority' with respect to the change-proneness. It is apparent from the graph that this metric directly proportional to change-proneness, however ,here we can see the value decrease as more and more components are found in representative five projects. This shows that with passage of time the metric is getting mature and stable, hence there is not much change occurs and it is more or less not steady or plateau in nature rather than directly upward proportional. However, this trend line has been only possible if ignore the high variability which is done iteratively by the bisquare which makes the

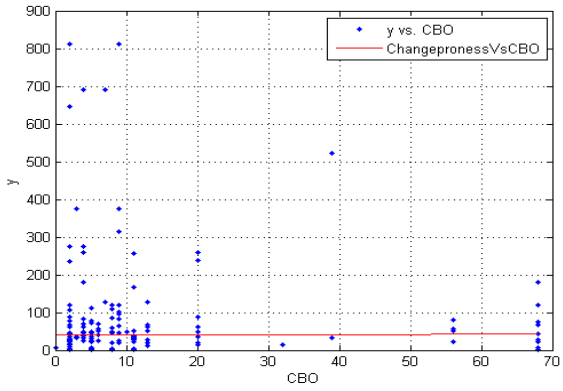


Figure 7.3

Here data points basically represent the 'code interdependency' with respect to the change-proneness.it is apparent from the graph that this metric directly proportional to change-proneness, however ,here we can see the value stable as more and more components are found in representative five projects. This shows that with passage of time the metric is getting mature and stable, hence there is not much change occur and it is more or less not steady or plateau in nature rather than directly upward proportional, this also reflects that the metric value even if it pass through high erratic values .

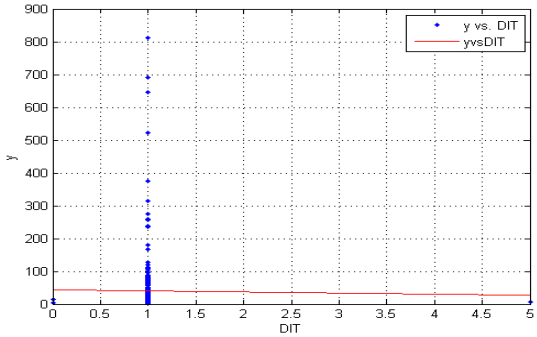


Figure 7.4

These data points basically represent the 'more the classes are inherited deeply makes the interdependent and poor quality' with respect to the change-proneness.it is apparent from the graph that this metric directly proportional to change-proneness, however ,here we can see the value stable as more and more components are found in representative five projects. This shows that with passage of time the metric is getting mature and stable, hence there is not much change occurs and it is more or less not steady or plateau in nature rather than directly upward proportional.

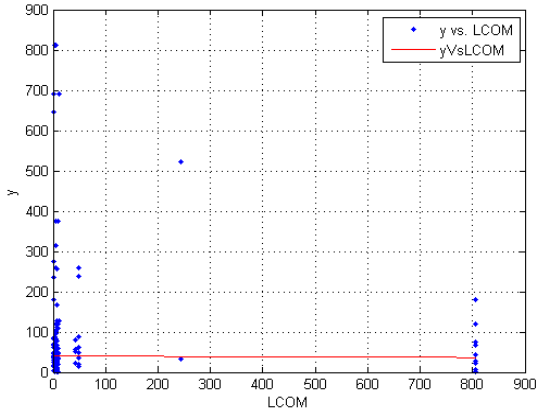


Figure 7.5

These data points basically represent the 'lack of cohesion of methods, when no common fields are accessed by class methods lead to new classes with respect to the change-proneness. It is apparent from the graph that this metric directly proportional to change-proneness, however ,here we can see the value stable as more and more components are found in representative five projects. This shows that with passage of time the metric is getting mature and stable, hence there is not much change occurs and it is more or less not steady or plateau in nature rather than directly upward proportional.

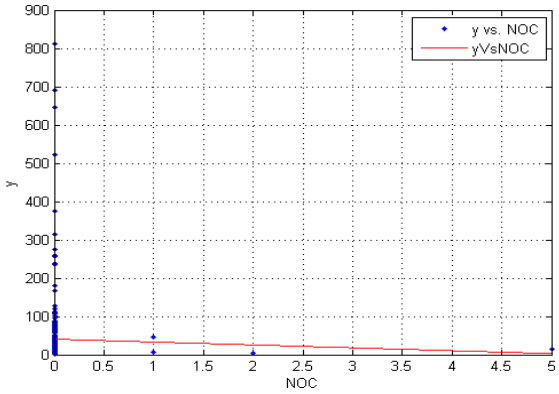


Figure 7.6

Here data points basically represent the 'how many number of children' with respect to the change-proneness.it is apparent from the graph that this metric indirectly proportional to change-proneness, however ,here we can see the value decrease as more and more components are found in representative five projects. This shows that with passage of time the metric is getting mature and stable, hence there is not much change occurs and it is more or less not steady or plateau in nature rather than directly upward proportional

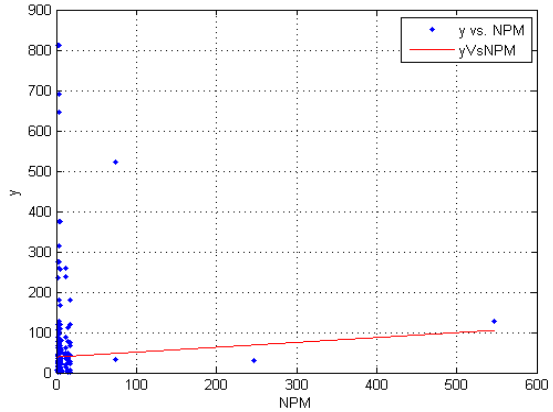


Figure 7.7

These data points basically represent the 'how many methods are used publically' with respect to the change-proneness.it is apparent from the graph that this metric directly proportional to change-proneness, however ,here we can see the value increases as more and more components are found in representative five projects. This shows that metric is getting unstable, hence there is too much changes started occurring with the passage of time and it is more or less not steady or plateau in nature rather than directly upward proportional.

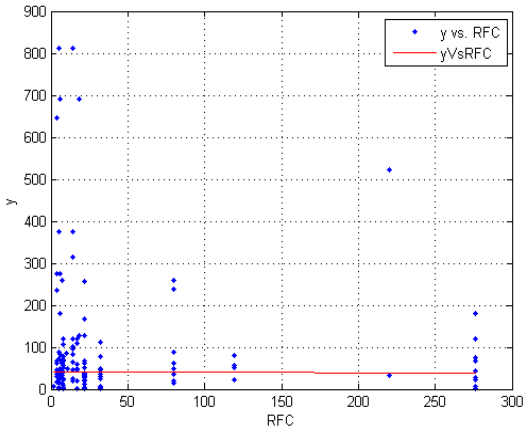


Figure 7.8

These data points basically represent the 'how many different methods which may be executed when a method invocation takes place for specific object' with respect to the change-proneness.it is apparent from the graph that this metric directly proportional to change-proneness, however ,here we can see the value stable as more and more components are found in representative five projects. This shows that with passage of time the metric is getting mature and stable, hence there is not much change occurs and it is more or less not steady or plateau in nature rather than directly upward proportional.

VIII. CONCLUSION

In apparent from the statistical test conducted to valid the optimized the dataset for getting smooth curve and hence proper slope value for finding rate of change of the component (java classes) are producing correct and valid results . Thus, in this process we have been able to contribute in a novel way as this combinational algorithm has not been used in previous work to find the change-proneness using such a strategy. In this research paper, we have done a statistical study on the factors that influence the rate of change of piece of code (class or groups of class of working together).After conducting systematic literature survey with stake holders of a various software development. It was found that during the life cycle of the application development ,it undergoes many organic changes due to change in objectives of building the software, and it is now normal that the software remains in continuous change and measuring change in the software process is critical for the software to remain in use as well as for its relevant, hence ,Moreover, in this research we have being able to find the trend on which the change-proneness can be modeled as function, a relationship equation that can help us to predict what will happen [good or bad] if a particular metric value changes. Previous work have assumed that all trends relationship are linear in nature, while dataset collected for shows it is highly non-linear. Therefore, in our current research work we have found non-linear data fitting algorithm bi-square robust gives best possible results as it more accurate with realistic with trend and pass through most of test of significances ,thus we get a promising approach to measure change-proneness of the software development process.

IX. FUTURE SCOPE

In this current research a mathematical function representing the relationship of metrics that influence the refactoring opportunities leading to lot of changes have been developed using real time dataset collected and made of open source project taken from Github repository , for future scope , we suggest that the nature of distribution of the metrics with respect to the change-proneness must also be understood with the help of calculation of probability density function ,since , a probability density function (pdf), or density of a continuous random variable, is a function that describes the relative likelihood for this random variable to take on a given value , this would give better insight into the nature of mathematical relationship between the change-proneness and the factors/random variables that influence it . It is limited to five projects so develop for more java based projects.Also it ignores outliers that may also be required sometimes.

REFERENCES

[1] Ingram, C.; Riddle, S., "Using early stage project data to predict change-proneness," Emerging Trends in Software Metrics (WETSoM), 2012 3rd International Workshop on, vol., no., pp.42, 48, 3-3 June

[2] Selvarani, R.; Nair, T.R.G.; Prasad, V.K., "Estimation of Defect Proneness Using Design Complexity Measurements in Object-Oriented Software," 2009 International Conference on Signal Processing Systems , vol., no., pp.766,770, 15-17 May 2009

[3] Elish, M.O.; Rine, D., "Investigation of metrics for object-oriented design logical stability," Software Maintenance and Reengineering, 2003. Proceedings. Seventh European Conference on, vol., no., pp.193, 200, 26-28 March 2003

[4] Cheng Zhang; Budgen, D., "What Do We Know about the Effectiveness of Software Design Patterns?" Software Engineering, IEEE Transactions on, vol.38, no.5, pp.1213, 1231, Sept.-Oct. 2012

[5] Okamura, H.; Dohi, T.; Osaki, S., "Software reliability growth model with normal distribution and its parameter estimation," *Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), 2011 International Conference on* , vol., no., pp.411,416, 17-19 June 2011

[6] Dwyer, D.; D'Onofrio, P., "Improvements in estimating software reliability from growth test data," *Reliability and Maintainability Symposium (RAMS), 2011 Proceedings - Annual* , vol., no., pp.1,5, 24-27 Jan. 2011

[7] Debbarma, M.K.; Kar, N.; Saha, A., "Static and dynamic software metrics complexity analysis in regression testing," *Computer Communication and Informatics (ICCCI), 2012 International Conference on* , vol., no., pp.1,6, 10-12 Jan. 2012

[8] Srikanth, H.; Cohen, M.B., "Regression testing in Software as a Service: An industrial case study," *Software Maintenance (ICSM), 2011 27th IEEE International Conference on* , vol., no., pp.372,381, 25-30 Sept. 2011

