IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. VIII (Mar-Apr. 2014), PP 45-48
www.iosrjournals.org

Change-Proneness of Software Components

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 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 ,In this paper we have found 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 .*

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

I. Introduction

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 .

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.

II. Related Work

Many change-proneness measure 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, so Claire Ingram has suggested techniques (linear regression) relies on developing an accurate method for predicting change-proneness [1]. Subversion2 project from CARMEN system was used to obtain the list of code files which was representative of datasets. After eliminating the useless data there were 254 unique files left for study.

Metrics used in this were cccc packages consisting of 6 separate metrics(LOC,NOC,DIT,WMC,CBO,MVG).data sets has been analyzed using Krushkal-Walis and Mann Whitney tests followed by linear regression, that finds only limited evidences exists between predictors(LOC,NOC,DIT,WMC) tested for change proneness then some metric values can differentiate groups of components. The data gathered over a 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]

III. Proposed Methodology

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] as well as follows:

1) Define the change-proneness with the mathematical expression using factors that influence its rate of change.
2) Define the factors or the metrics with mathematical expression using the relations with respect to the best practices in building code and the application, conformance to the coding standards and designs and with respect to rate to changes the project itself can bear for finally to remain stable in the final release.
3) Select the project open source repositories which are undergoing large number of additions and deletion on same piece of code multiple times.
4) Run the tool that can for extraction of the metric values for each compiled classes
5) Group of each class object, which is found by finding the numeric ranges of the conformance ranges of each metric which lead to best practices in software engineering. However, for finding the violation of these conformance ranges meant either, there is huge opportunity in refactoring the class object in question or it is undergoing too many changes, therefore standard deviation was also considered for calculating the change-proneness slope.

Next, step was to find the trend and mathematical function which are best represent the relationship between the change-proneness as function of each metric. Therefore, in this step, we got find the rate of each metric using slope method and data fitting algorithms[5] to arrive at the function representing the mathematical relation between the change-proneness and each metric measured refer Table 5.1

IV. 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.

4.1 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.

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.

4.2 Slope (Rate of Change)

This measure give 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]

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

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) .

V. Tables

5.1 Table: Relationship between Metrics and Change-Proneness

S No.	Metric Name	Description	Relationship with Change-Proneness
1.	WMC (Weighted Methods per Class)	It is the measure of number of methods defined in a class.	High WMC in a class means it is highly application specific and cannot be reused thus increase in WMC ∝ Bug density and 1/Code Quality
2.	DIT (Depth of Inheritance)	It is the measure of maximum Inheritance path from the class to the root class.	Increase in DIT ∝ Increase in density of bugs and 1/Code Quality
3.	NOC (Number Of Children)	It is the number of immediate child classes derived from base class.	High NOC indicates high reuse of base class thus indicates fewer faults. High NOC ∝ Less susceptibility for change
4.	CBO (Coupling Between Object Classes)	It is the number of classes to which a class is coupled i.e. Two classes are coupled when methods declared in one class use methods or instance variables defined by the other class.	High CBO is not desirable hence it makes the design monolithic and dependent and hence much more susceptible to changes since either of the coupled classes face changes others will face too. High CBO ∝ High susceptibility for change.
5.	LOC (Lines Of Code)	It measures the size of a computer program by counting the number of executable statements.	High LOC value may or may not affect certain aspects of the software.
6.	RFC (Response For a Class)	The response set of a class is a set of methods that can potentially be executed in response to a message received by an object of that class. RFC is simply the number of methods in the set.	High RFC ∝ High susceptibility of changes, Density of bugs and 1/Code Quality
7.	LCOM (Lack Of Cohesion Of Methods)	Take each pair of methods in the class. If they access disjoint sets of instance variables, increase P by one. If they share at least one variable access, increase Q by one. *LCOM = P - Q , if P > Q LCOM = 0 otherwise* LCOM = 0 indicates a cohesive class. LCOM1 > 0 indicates that the class needs or can be split into two or more classes, since its variables belong in disjoint sets.	High LCOM value ∝ class more susceptible to errors and might be disaggregated into two or more classes It can help to track whether the cohesion principle is adhered to in the design of an application and advice changes.

8.	Ca (Afferent couplings)	A class's afferent couplings is a measure of how many other classes use the specific class.	High Ca ∝ High susceptibility for change
9.	NPM (Number Of Public Methods)	The NPM metric simply counts all the methods in a class that are declared as public. It can be used to measure the size of an API provided by a package.	High NPM means class is highly accessible to other parts of software thus High NPM ∝ High susceptibility for changes and complexity.

5.2 Table : Results

Method	Mathematical Equation model	Coefficient of Goodness	Error	Slope (Rate of Change)	Adjusted R-square
WMC	y=m*x + c	0.9126	2.734e+05	-2.179	0.912
DIT	y=m*x + c	0.9116	2.763e+05	-1.168	0.911
NOC	y=m*x + c	0.9131	2.718e+05	-3.471	0.9125
CBO	y=m*x + c	0.9119	2.754e+05	0.04461	0.9113
RFC	y=m*x + c	0.9123	2.743e+05	-0.4156	0.9117
LCOM	y=m*x + c	0.9126	2.733e+05	-1.008	0.912
Ca	y=m*x + c	0.9127	2.731e+05	-2.179	0.912
NPM	y=m*x + c	0.913	2.72e+05	0.1208	0.9124

VI. 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 algorithms has not been used in previous work to find the change-proneness using such a strategy.

VII. 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 .

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