



Review of Defect Prediction through Variants of COQUALMO Model

Karmvir Kaur*, Monika**

*Research Scholar, University Institute of Engineering and Technology,
Panjab University, Chandigarh, INDIA

**Assistant Professor, University Institute of Engineering and Technology,
Panjab University, Chandigarh, INDIA

Abstract—Defect Handling is one of the most important activities involved in the various software projects. Defect Handling includes identification in Software projects will make defect removal and prevention much easier. So defect handling is the most important aspect of software engineering. So we will work on defect prediction that is one of the most important stages of defect handling. This paper gives the review over defect prediction through variants of COQUALMO Model used so far for various software systems. The paper gives review of various parameters, models, methods and techniques used for defect prediction of software systems by collecting datasets from various fields including industries, research centers and student projects.

Keywords: Defect, Defect Analysis, Defect Prevention, Defect Prediction, Root Cause Analysis.

I. Introduction

A defect can occur in any phase of software development life cycle. So defect handling is the most important aspect of software engineering. Software Defect is termed as "imperfections in the software development process that would cause that software to fail to meet the desired requirements". Basically there are three types of defects in this research paper Requirement defects, Design defects, Code defects. Defects occur in all the phases of the Software development life cycle so that defect prevention is very essential part of Software development Life cycle for improving the Software Quality. For improving the Software Quality it is necessary to identify the defects from the given set of projects at the first step, then it involves classification and analysis of the pattern and after that it involves elimination for prevention of defects. Defect handling having stages Defect Identification, Defect Classification, Defect Analysis, Defect Prediction Technique, Defect Prevention, and Process Improvement. We will work on defect prediction that is one of the most important stages of defect handling. Defect Prediction Technique we mean to identify and prevent the defect causing failures before occurring. Some Defect predictions models are COQUALMO model which we have taken and expanded further in this research paper and the second one is mining defects using ODC.

II. Implementation of COQUALMO model:

COQUALMO is an extension of the COCOMO II software project effort and schedule estimation model. COQUALMO estimates a software project's quality in terms of Delivered Defect Density (DDD). This is the number of nontrivial delivered defects per unit of size, measured in either function points (DDD/FP) or thousands of source lines of code (DDDKSLOC). This specifies which COCOMO II model is being used in this data submission. If this is a "historical" data submission, select the Post-Architecture model or the Applications Composition model.

- Application Composition: This model involves prototyping efforts to resolve potential high risk issues such as user interfaces, software/system interaction, performance, or technology maturity.
- Early Design: This model involves exploration of alternative software/system architectures and concepts of operations. At this stage of development, not enough is known to support fine-grain cost estimation.
- Post-Architecture: This model involves the actual development and maintenance of a software product. This stage of development proceeds most cost-effectively if a software life-cycle architecture has been developed; validated with respect to the system's mission, concept of operation, and risk; and established as the framework for the product.

COQUALMO is a well-researched model that is useful in prediction the number of defects, or errors, per thousands of lines of code or function points.

III. Prediction using size and complexity metrics:

A wide range of prediction models have been proposed. Complexity and size metrics have been used in an attempt to predict the number of defects a system will reveal in operation or testing. The source code metrics we parsed each source code release using the Imagix-4D C/C++ analysis tool. This tool finds the lines of code

(LOC), Halstead's Theory of Software Science metrics and McCabe's Cyclomatic Complexity. Productivity was measured in terms of "lines of code per time unit." Quality was measured in terms of "defects per KLOC" where "K" was the symbol for 1000 lines of code.

The equation (1) involving lines of code L (LOC) was so that, for example, a 1000 LOC (i.e. 1 KLOC) module is expected to have about 23 defects:

$$D = 4.86 + 0.018 L \quad (1)$$

Other equations had the following dependent metrics: Number of decisions C ; Number of subroutine calls J ; and a composite metric $C+J$.

Halstead proposed a number of size metrics, which have been interpreted as 'complexity' metrics, and used these as predictors of program defects. Most notably, Halstead asserted that the number of defects D in a program P is predicted by (2):

$$D = V/3000 \quad (2)$$

where V is the (language dependent) volume metric (which like all the Halstead metrics is defined in terms of number of unique operators and unique operands in P ; for details see. The divisor 3000 represents the mean number of mental discriminations between decisions made by the programmer. Each such decision possibly results in error and thereby a residual defect. Thus, Halstead's model was, unlike Akiyama's, based on some kind of theory. Interestingly, Halstead himself "validated" (1) using Akiyama's data.

Cyclomatic complexity is a methodology to measure the complexity at a certain area of code. In other words it is another way to reach at defect clustering.

Cyclomatic complexity is to measure the number of independent paths in a program's source code.

The formula is as follows: $M = E - N + 2P$

where

M = cyclomatic complexity

E = number of edges of the graph

N = number of nodes of the graph

P = the number of connected components.

IV. ODC COQUALMO Models and Tools:

There are different implementations of ODC COQUALMO. Initially we created our static model in a spreadsheet and then transitioned to a web-based version as part of the integrated COCOMO Suite tool. A dynamic simulation version models the defect generation and detection rates over time for continuous project usage, and provide continuous outputs as shown in the next section. ODC is a method for classifying and analyzing software defects. It bridges the gap between statistical defect modeling and causal analysis. ODC classifies each defect based upon the semantics of the defect correction and links the defect distribution to the development progress and maturity of the product. It provides an in-process measurement paradigm that extracts key properties from defects and enables measurement of cause-effect relationships as opposed to a mere taxonomy of defects for descriptive purposes. ODC COQUALMO decomposes defects from the basic COQUALMO model using ODC [5]. The top-level quantities for requirements, design and code defects are decomposed into the ODC categories per defect distributions input to the model. With more granular defect definitions, ODC COQUALMO enables tradeoffs of different detection efficiencies for the removal practices per type of defect.

In the table shown below it has been seen that lots of factors affect software defect prediction, resulting in inconsistencies among learning methods. Hence, there is a need to develop methods that could remove some of the randomness i.e. complexity/uncertainties in the data, leading us to a more definitive explanation of the error analysis. One of the steps in this direction is to capture the dependency among attributes using probabilistic models, rather than just using the "size" and "complexity" metrics. COQUALMO predicts the defect density of the software under development where defects conceptually flow into a holding tank through various defect introduction pipes and are removed through various defect removal pipes. COQUALMO consists of 2 sub-models, namely the 'Defect Introduction (DI)' and the 'Defect Removal (DR)' models. The DI model is formulated using product, process, computer and personnel attributes (based on COCOMO II, USC-CSE, 1999) and predicts the number of requirements, design and coding defects that are introduced during various activities of the development life cycle. The DR model captures the effects of 3 relatively orthogonal profiles of defect removal techniques, namely Automated Analysis, People Reviews, Execution Testing and Tools, and predicts the number of requirements, design and coding defects that are eliminated. The residual number of defects is the difference between the number of defects introduced and the number of defects removed.

Year	Paper	Description	Advantages	Disadvantages
1992	Orthogonal Defect Classification Using Defect Data to Improve Software Development	This paper will present a framework developed by IBM for classifying and analyzing defect data collected during software development.	To improve the quality in developed software project.	It is based on the semantic fixes must have at least three requirements. Orthogonality, consistency across phases, and uniformity across products.
1999	A Critique of Software Defect Prediction Models	Norman E. Fenton study the critique of software defect prediction models using size and complexity.	To offer many advantages of using a univariate decision criterion such as McCabe's metric.	They are not predictions, but Descriptions of existing trends in defect inflow.
2008	ODC COQUALMO - A Software Defect Introduction and Removal Model using Orthogonal Defect Classification	In this paper, predicting software defect introduction and removal rates, ODC COQUALMO is useful for identifying appropriate defect reduction strategies.	ODC COQUALMO is a significant time, cost and quality.	It is based on the semantic fixes must have at least three requirements. Orthogonality, consistency across phases, and uniformity across products.
2010	Defect Analysis and Prevention for Software Process Quality Improvement	This paper will focus on finding the total number of defects that has occurred in the software development process for five similar projects and aims at classifying various defects using first level of Orthogonal Defect Classification (ODC), finding root causes of the defects and use the learning of the projects as preventive ideas.	The advantage of Popularity and low cost, and also improves the accuracy of tracking the identified defects.	It is a hard decision to inspect a large batch that has been unit tested and there may be the view that there is no longer time to inspect
	A Value-Added Predictive Defect Type Distribution Model Based on Project Characteristics	This Paper discussed a new approach for predicting the distribution of defects and their types based on project characteristics. For this approach, the maximum likelihood estimation was used in fitting the probability density function to the actual defect data, and the regression analysis was used to identify the relationship between the project characteristics and the parameters.	The advantages of utilizing lower memory and reducing the test.	Cannot easily deal with missing or uncertain data.
2011	Defect Prediction and Analysis Using ODC Approach in a Web Application	This paper provides a detail report of a software process model which will help in predicting the defects by classification and evaluation.	Its help in predicting the defects by classification and evaluation.	Its evaluation is not a simple task
2012	Defect Handling In Software Metrics	Arpita Mittal & Sanjay Kumar Dubey first studied about the various types of defect techniques and then they have undergone through the survey of COQUALMO cost constructive model which is a two-step software defect prediction model for improving the software quality.	Producing high quality software through systematic	Not supporting object-oriented Features.
	Software Defect Prediction Models for Quality Improvement	The paper showcases on how the various defect prediction models are implemented resulting in reduced magnitude of defects.	To improve the software quality.	It is hard from a decision making Standpoint.
	Evaluating Stratification Alternatives to Improve Software Defect Prediction	This paper study the software defect prediction problem, i.e. predicting which modules will experience a failure during operation based on software metrics.	Their insensitivity to class distributions in data.	Imbalanced distribution between classes in SDP data is a main cause of its learning difficulty.

V. Conclusion and Future Work

Software estimation models can and should play an important role in facilitating the right balance of activities to meet quality goals. By predicting software defect introduction and removal rates, COQUALMO variants are useful for identifying

appropriate defect reduction strategies. The extension for ODC defect types provides more granular insight into defect profiles and their impacts to specific risks. We have shown that the ODC COQUALMO model can be used in different ways to reason about and optimize quality processes.

VI. References

- [1] Arpita Mittal, Sanjay kumarDubey, "Defect Handling In Software Metrics" International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2012
- [2]Norman E. Fenton," A Critique of Software Defect Prediction Models" IEEE transactions on software engineering,VOL.25, NO. 5, September/October 1999.
- [3]Raymond Madachy, Barry Boehm, "ODC COQUALMO - A Software Defect Introduction and Removal Model using Orthogonal Defect Classification"university of southern California center for systems and software engineering 941 W. 37th Place, Los Angeles, CA, USA, USC-CSSE-2008-817
- [4] SakthiKumares, R Baskaran," Defect Analysis and Prevention for Software ProcessQuality Improvement" International Journal of Computer Applications (0975 – 8887)Volume 8– No.7, October 2010.
- [5]Mrinal Singh Rawat, "Software Defect Prediction Models for Quality Improvement"IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2, September 2012
- [6]Lourdes Pelayo and Scott Dick, "Evaluating Stratification Alternatives to Improve Software Defect Prediction"IEEE transactions on reliability, VOL. 61, NO. 2, JUNE 2012
- [7]Chillarege R., Bhandari I., Chaar J., Halliday M., Moebus D., Ray B., and Wong M.: Orthogonal Defect Classification - A Concept for In-Process Measurements, IEEE Transactions on Software Engineering, Vol. 18, No. 11, pp. 943-956 (1992)
- [8]Youngki Hong, JongmoonBaik, In-Young Ko, Ho-Jin Choi, "A Value-Added Predictive Defect Type Distribution Modelbased on Project Characteristics"IEEE/ACIS International Conference on Computer and Information Science, March 16,2010
- [9]MiroslawStaron, Wilhelm Meding, "Defect Inflow Predictionin Large Software Projects"e-Informatics Software Engineering Journal, Volume 4, Issue 1, 2010
- [10] J. Pipitone and S. Easterbrook,"Assessing climate model software quality: a defect density analysis of three models" Geosci. Model Dev., 5, 1009–1022, 2012
- [11]Stefan Wagner and Tilman Seifert "Software Quality Economics for DefectDetectionTechniques Using Failure Prediction"TechnischeUniversit "at M'unchenBoltzmannstr. 385748 Garching, Germany 1999
- [12]Pranayanath Reddy Anantula, RaghuramChamarthi, "Defect Prediction and Analysis UsingODC Approach in a Web Application" International Journal of Computer Science and Information Technologies, Vol. 2 (5), 2011
- [13]Mukeshsoni, "Defect Prevention: Reducing cost and improving quality" published in IEEE Computer,Volume 30, Issue 8, August 1997.
- [14]B. Steece, S. Chulani, and B. Boehm, "Determining Software Quality Using COQUALMO," in Case Studies in Reliability and Maintenance, W. Blischke and D. Murthy, Eds.: Wiley, 2002.
- [15]Muhammad Dhiauddin Mohamed Suffian and SuhaimiIbrahim,VOLUME "A Prediction Model for Functional Defects in System Testing using Six Sigma": 1 NO. 6 september2011.