Asian Journal of Information Technology 13 (10): 657-660, 2014
ISSN: 1682-3915
© Medwell Journals, 2014

Review and Development of Software Testing Defect Corrctive Model (STDCM) for Software Projects

K. Karnavel and R. Dillibabu
Department of Industrial Engineering, Anna University, Chennai-25, India

Abstract: There have been a number of methods used to estimate the number of defects remaining in software. In this study, researchers present an analysis for the method of COQUALMO Model and it is difficult to increase the quality without increasing the cost and time. The COQUALMO Model usually tries to predicts the residual defects in the software product. This model has pipes in stage by stage of various phases. They are injecting the defects and removing the defects which involves more computation time, cost and man power to predict the residual defects. We developed a new static model for estimating the number of remaining residual defects in the software product have following assumptions: the existing model can produce the minimum quality. But STDCM achieves the maximum customer satisfaction, since peer reviews have been conducted to find the bug in earlier for improving the quality and STDCM produces maximum quality of the software without increasing the cost and time and there is a chance for getting very less residual errors at the testing stage. Software Testing Defect Corrective Model (STDCM) shows its attraction in its applicability to a broader scope of circumstances.

Key words: Reviews, inspections, residual defects and cost estimation, scope, quality

INTRODUCTION

Software has become critical to advancement in almost all areas of human endeavor. The art of programming only is no longer sufficient to construct large programs. There are serious problems in the cost, timeliness, maintenance and quality of many software products. Software engineering has the objective of solving these problems by producing good quality, maintainable software, on time, within budget. To achieve this objective, we have to focus in a disciplined manner on both the quality of the product and on the process used to develop the product. The definition of software engineering by Fritz Bauer in 1968 stated as "The establishment and use of sound engineering principles in order to obtain economically developed software that is reliable and works efficiently on real machines". Stephen Schach defined the same as "A discipline whose aim is the production of quality software, software that is delivered on time, within budget and that satisfies its requirements" (Schach, 1990). Both the definitions are popular and acceptable to majority. However, due to increase in cost of maintaining software, objective is now shifting to produce quality software that is maintainable, delivered on time, within budget and also satisfies its requirements.

SOFTWARE TESTING

Software testing is a research that is performed to help with the necessary information about the quality of the software and the user can be sure that the product meets its predefined objectives.

SOFTWARE QUALITY

Kitchenham and Pfleeger (1996)'s on software quality gives a succinct exposition of software quality. They discussed five views of quality in a comprehensive manner is shown in Fig. 1. The concept of softwarre quality and the efforts to understand it in terms of measurable quantities date back to the mid-1970s. McCall *et al*. (1977) were the first to study the concept of software quality in terms of quality factors and quality criteria. A quality factor represents a behavioral characteristic of a system. Some examples of high-level quality factors are correctness, reliability, effeciency, testability, maintainability and reusability.

LITERATURE REVIEW

Cai (1998) applied new static model for estimating the number of remaining defects and use a set of real data to test the new model. The new model coincides with the

Corresponding Author: K. Karnavel, Department of Industrial Engineering, Anna University, Chennai-25, India

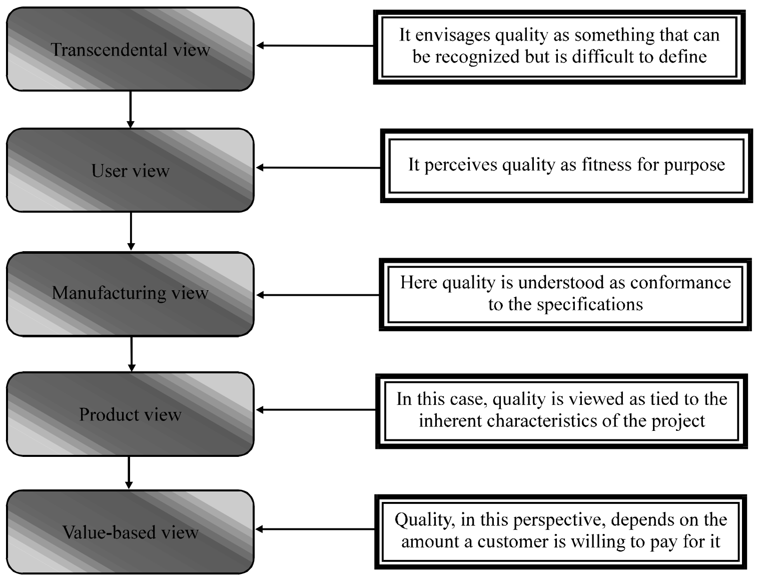
Transcendental view
User view
Manufacturing view
Product view
Value-based view
It envisages quality as something that can be recognized but is difficult to define
It perceives quality as fitness for purpose
Here quality is understood as conformance to the specifications
In this case, quality is viewed as tied to the inherent characteristics of the project
Quality, in this perspective, depends on the amount a customer is willing to pay for it

Fig. 1: Views of quality

Mills Model in a particular case and shows its attraction in its applicability to a broader scope of circumstances. They given a practical example shows that the new model can offer good estimates for the number of remaining software defects. It is also applicable to statistical problems other than software reliability modeling. They are not give systematic review and doesn't seem applicable to estimating the number of remaining defects.

Chulani (1999) applied COCQUALMO Model for 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. In this model it is difficult to increase the quality without increasing the cost and time. They are injecting the defects and removing the defects but it involves more computation time, cost and man power to predict the residual defects.

Biffl (2003) compared and investigated the performance of objective and subjective defect content estimation techniques. For validation of the techniques they conducted a controlled experiment with 31 inspection teams with consisted of 4-6 persons. They reported on data from an experiment with software engineering defect data on the number of (major) defects in a requirement document. They are used the relative error, a confidence interval and the correctness for deciding on a reinspection as main evaluation criteria but it has to provid major defect in the requirement document and less number of accuracy.

Westland (2004) analysed that the short software development life cycles appear to favor higher rates of detection but that for any reasonable development cycle, most errors will go uncorrected. Short life cycles are likely to force constrained software development cycles and are likely to exacerabate the risk from post-release defects. It has to defined uncorrected defects become exponentially more costly in each phase.

Chun (2006) applied a new method (Capture-Recapture Model) that estimates the number of undetected errors in complex software design documents. This idea use the correlation matrix of multiple inspectors and to formulate the estimation problem as a goal program. Capture-Recapture Model intially used by biologists to estimate the size of wildlife populations has been widely used to estimate the number of software design errors. It shows the undetected errors will be present and leads to software fault or failure.

Turakhia *et al*. (2006) used statistical testing to isolate the embedded outlier population, test conditions and test application support for the statistical-testing framework and the data modeling for identifying the outliers. The identification of outliers that correlate to latnet defects critically depends on the choice of test reponse and the statistical model's effectiveness in estimating the healthy-die response but it provides low efficiency, less reliability and cost is very high.

Jacobs *et al*. (2007) studied on Defect Injection (DI) and Defect Detection (DD) influencing factors and their grouping, resulting for use in devlopment projects. To decrease the number of injected defects in a development project, the DI factors could be used as areas of attention. While quality of documentation is expected to be poor and lack of product of quality.

Ravishanker *et al.* (2008) applied non-homogeneous Poisson Process Model and multivariate model that apply Markov switching to characterize the software defect disovery process. While the process remains complex and increased in failure rate also.

Zuo *et al.* (2009) studied quality prediction model for there have been number of faults is negatively correlated with the workload deviation which indicates that quality is decreasing due to unsatisfied workload budget. Although, the problems occurred in testing phase for overheads workload and decrease in quality of the product.

Quah (2009) studied defect tracking is used as a Proxy Method to predict software readiness. They developed defect predictive model is divided into three parts: Prediction model for presentation, Logic Tire Prediction Model for business tier and Prediction Model for data access tier. While evaulating the software readiness is very complex.

Catal (2011) studied software engineering discipline contains several prediction approaches such as test effort prediction, correction cost prediction, fault prediction, reusability prediction, security prediction, effort prediction and quality prediction. They investigated 90 software fault prediction papers published between year 1990 and year 2009. They given roadmap for research scholars in area of software fault prediction.

A detailed literature study was conducted in the area of software quality and in particular software testing. The abstract of the literature papers was presented above. The following conclusion can be drawn from the review of literature:

- None of the alternatives is better than the others in all aspects
- The waterfall and COQUALMO Models are not produce satisfactory quality
- The strength and weakness of other techniques are complementary

LIMITATION OF COQUALMO

- The effort to fix the defect introduced and removed is not quantified directly by the model
- COQUALMO does not associate time aspect to the defect removal and fixing process
- Defects are not given weights and classifications in terms of software artifact they originated from

NEW MODEL DEVELOPMENT

The STDCM has been developed based on the two important models namely, Waterfall and COQUALMO (Chulani, 1999). The integration of these models were carried out in a stage wise manner. The validation process is also considered in the integration of these model in stage by stage wise fashion. The framework of these model is shown in Fig 2. The following steps are considered for the construction of STDCM:

- The first phase of the model is requirement analysis the customer requirement of the project are collected in the requirements phase of STDCM. The requirements are analyzed using SoftQFD(SQFD), in which House of Quality (HoQ) matrix was used to validate the requirements of the given project

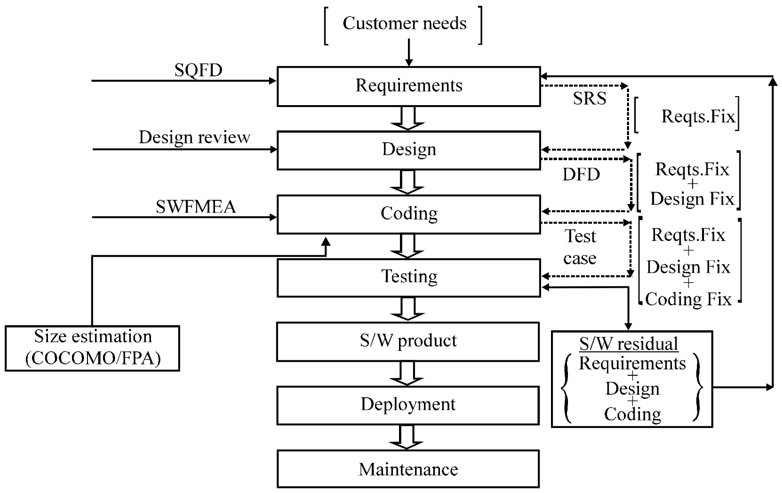

Fig. 2: Framework of Software Testing Defect Corrective Model (STDCM); SWFMEA = Software Failure Mode Effect Analysis; SQFD = Software Quality Function Deployment; SRS = System Requirements Specifications; DFD = Data Flow Diagram; $\Rightarrow$ Represents flow of process; $\longrightarrow$ Represents validation tools and reviews; $\dashrightarrow$ Represents inputs and outputs

- The second phase of the model is design: here the output of this phase is Data Flow Diagram (DFD). The input of this phase is designed review corrected DFD will be obtained
- The third phase of the model is coding: to estimate the lines of code using COCOMO/FPA. The SWFMEA is validated using statistical tools
- The fourth phase of the model is testing: the input is output of the SWFMEA and test cases

CONCLUSION

This study presents a new model namely STDCM for finding residual defects in developing software projects in IT industry. In this study or review is presented, analysis shows that the COQUALMO and WATERFALL Model based method doesn't seem applicable to estimating the number of remaining residual defects. But the new static model for estimating the number of remaining residual defects in the software product. STDCM achieves the maximum customer satisfaction, since peer reviews have been conducted to find the bug in earlier for improving the quality and produces maximum quality of the software without increasing the cost and time and there is a chance for getting very less residual errors at the testing stage. The construction shows the application of peer-review process and the size estimation computation of COCOMO/FPA.

ACKNOWLEDGEMENTS

This research was supported from many people, including R&D Lab, Department of Industrial Engineering, the family members and our friends and colleagues. First, I would like to thank the supervisor who gave us much professional guidance and patiently answered the various queries and the Industrial Engineering Department professors, research scholars and PG students gave us numerous suggestions to improve the presentation; we thank him for all his effort and encouragement.

REFERENCES

Biffl, S., 2003. Evaluating defect estimation models with major defects. J. Syst. Software, 65: 13-29.

Cai, K.Y., 1998. On estimating the number of defects remaining in software. J. Syst. Software, 40: 93-114.

Catal, C., 2011. Software fault prediction: A literature review and current trends. Expert Syst. Appl., 38: 4626-4636.

Chulani, S., 1999. Constructive quality modeling for defect density prediction: COQUALMO. IBM Research, Center for Software Engineering, http://www.chillarege.com/fastabstracts/issre99/99120.pdf.

Chun, Y.H., 2006. Estimating the number of undetected software errors via the correlated capture-recapture model. Eur. J. Oper. Res., 175: 1180-1192.

Jacobs, J., J. Van Moll, R. Kusters, J. Trienekens and A. Brombacher, 2007. Identification of factors that influence defect injection and detection in development of software intensive products. Inform. Software Technol., 49: 774-789.

Kitchenham, B. and S.L. Pfleeger, 1996. Software quality: The elusive target. IEEE Software, 13: 12-21.

McCall, A.J., P.K. Richards and G.F. Walters, 1977. Factors in software quality. Technical Report No. RADC-TR-77-369, U.S. Department of Commerce, Washington, DC., USA.

Quah, T.S., 2009. Estimating software readiness using predictive models. Inform. Sci., 179: 430-445.

Ravishanker, N., Z. Liu and B.K. Ray, 2008. NHPP models with Markov switching for software reliability. Comput. Stat. Data Anal., 52: 3988-3999.

Schach, S., 1990. Software Engineering. Vanderbilt University Press, USA.

Turakhia, R.P., W.R. Daasch, J. Lurkins and B. Benware, 2006. Changing test and data modeling requirements for screening latent defects as statistical outliers. IEEE Des. Test Comput., 23: 100-109.

Westland, J.C., 2004. The cost behavior of software defects. Dec. Support Syst., 37: 229-238.

Zuo, X., Y. Liang and H. Lei, 2009. Study on the quality prediction model of software development. Proceedings of the International Conference on E-Business and Information System Security, May 23-24, 2009, Wuhan, China, pp: 1-4.