International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.4, July-Aug 2012 pp-1540-1542 ISSN: 2249-6645

Design of Model For Incremental Development Of Software Modules To Evaluate The Quality Of Software Modules

Dr C.S. Lamba [1], Sanjay Kumar[2]
1.Professor, Rajasthan Institute of Engineering
Jaipur, Rajasthan, India
2. Lecturer, University Campus School
Rohtak, Haryana (124001), India

Abstract: This paper is introducing a model, which are used to evaluate the expected quality of software modules during incremental development of software system. This is not only focus on modification of existing system, but also focus to find the methods for developing high quality of software products at reasonable cost. As we know that software modules or software systems are being used in more and more critical areas of industries as well as public or private sectors, then quality of software modules becomes a key factor of business success and human safety. The goal of this paper within the subject areas are the identification of the methods and modules that were used throughout the design and verification of the use and the quality of these methods with respect to the analysis products.
Keyword: Software development approach,

I. Introduction

It is very difficult to build high quality software with limited quality assurance budgets. The proposed model can be used to learn fault predictors from software metrics. Fault prediction prior to software release can guide Verification and Validation (*V&V*) activity and allocate scarce resources to modules which are predicted to be fault-prone.
One of the most important goals of proposed model is to detect fault prone modules as early as possible in the software development life cycle. Design and code metrics have been successfully used for predicting fault-prone modules.

Through this model we can introduce fault prediction from software requirements and analysis. Furthermore, we investigate the advantages of the incremental development of software fault prediction models, and we compare the performance of these models as the volume of data and their life cycle origin (design, code, or their combination) evolution during project development. We confirm that increasing the volume of training data improves model performance. And that, models built from code metrics typically outperform those built using design metrics only. However, both types of models prove to be useful as they can be constructed in different phases of the life cycle. We also demonstrate that models that utilize a combination of design and code level metrics outperform models which use either one metric set exclusively.

II. Problem Description

The software industry is currently entering a period of maturity, in which particular informal approaches are specified more precisely and are supported by the appropriate standards. Quality characteristics of software products are defined in ISO/IEC9126 [1]. For each characteristic, a set of attributes which can be measured is determined.Such a definition helps in evaluating the quality of software, but gives no guidance on how to construct a high quality software product.Therequirements for a quality management system are defined in ISO 9001 [2].All the requirements are intended for application within a software process in order toenhance the customer satisfaction, which is considered the primary measure of thesoftware product quality. The quality management system, as defined by the standard,can be subject to a certification.This paper describes a model, which we used to evaluate the expected as well asthe actual quality of a huge software system that was developed. There are a few methods for selecting the metrics and collecting data that are relevantfor a particular purpose, described in the literature. The best known examples are Goal Question Metric approach [3,4,5] and the Quality Function Deployment approach[6,7]. Both of the two methods represent the viewpoint of the software development organization. Our approach was based on a modification to Goal Question Metric approach. The modification was needed, because our assessment was done on behalf of the customer, and not of the development company, and it had no other goal in mind than just to evaluate the expected quality of the developed software.

Important characteristics

i. Software requirements are the foundation from which quality is measured
ii. Specified standards define development criteria that guide the manner in which the software is engineered.
iii. If the software meets only the explicit requirements, and does not meet the implicit requirements, the software quality is suspect.

Software Quality factors:

i. Operational characteristics:-
a. Correctness - does it do what I want?
b. Reliability - does it do it accurately?
c. Efficiency - will it run efficiently on my hardware?
d. Integrity is it secure.
e. Usability - is it designed for the user?

ii. Product revision:-
   a. Maintainability - can I fix it ?
   b. Flexibility - can I change it
   c. Testability - can I test it?
iii. Product transition:-
   a. Portability will I be able to use it on another machine?
   b. Re usability - will I be able to reuse some of the software.
   c. Interoperability - will I be able to interface it with another system.

III. Related Works

One of the oldest software development tools is *flowcharting,* which developed since the 1920s. The software development methodology has emerged since the 1960s. The oldest formalized methodology for building information systems is the *Systems development life cycle*. The traditional Systems development life cycle originated in the 1960s to develop large scale functional business systems in an age of large scale business conglomerates. Information systems activities resolved around heavy data processing and number crunching routines. [8]

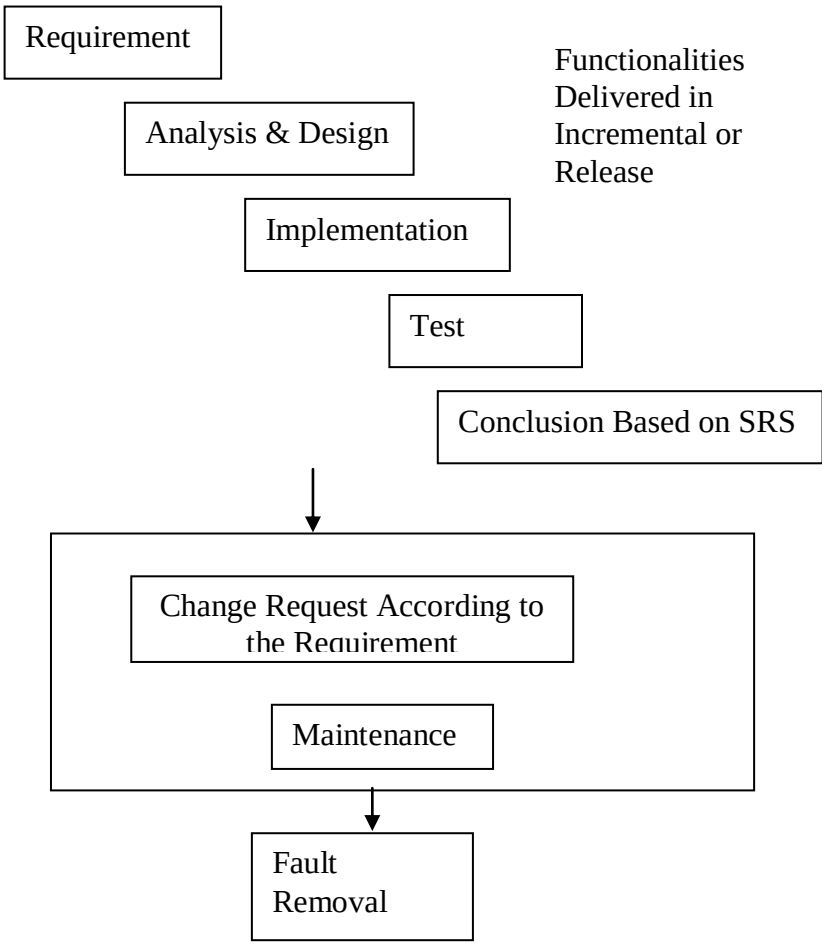
Software development approaches.

Every software development methodology has more or less its own approach to software development. There is a set of more general approaches, which are developed into several specific methodologies. These approaches are [9]:

i. Waterfall: linear framework type
ii. Prototyping: iterative framework type
iii. Incremental: combination of linear and iterative framework type
iv. Spiral: combination linear and iterative framework type
v. Rapid Application Development (RAD): Iterative Framework Type

IV. Proposed Model

The proposed model is designed in such a way that it not only used for defect prevention rather than it also be used for defect removal. It will evaluate the software quality through the representative user testing at the system level.
During incremental development of software modules, this model not only help to indicate the software changes to correct failure found, but also helpful to untested new software that will be added to the software already under test.



'Fault Prediction Model'

In the above proposed model there are three major task that are very beneficial to increase the quality of software modules during incremental development of software system.

These tasks are given below:

i. The functionalities delivered in increment or release
ii. Parts of the system that have to be changed
iii. Traceability that will create a links between above two tasks.

This model is used for each increment during incremental development:

i. To determine the requirements elicitation or refinement, analysis and design, implementation, integration and the system testing for current increment
ii. While the software is developed in increment j is being tested, increment j+1 has started. Therefore each phase of incremental development includes fault removal activities for previous increment (n). Fault should be removed both from the previous increments or releases and the current one and fault correction may introduce new faults
iii. Changes to the requirement according to the request based on today's may lead to deviation from the original increment plan, when it comes to effort and time plan
iv. Several development teams' works in parallel for implementing the software modules, some may finish their works before others and start working on next increment, that should be synchronized with each other and dependencies should be released.
v. Through each increment quality will be increases from starting to end of software modules

Although basic idea behind this is to deliver the final system in smaller parts and much more functionality is delivered at the end of each increment. During the each release of software modules many correction should also be corrected through the proposed model, which made integration and testing simpler.

Therefore proposed model made integration and testing simpler and increases the quality of software module throughout its life cycle.

Goal of Proposed Model

The major goal of proposed model is to detect faults as early as possible in the development life cycle.

This model helps us to better designing of modelling algorithm in incremental development, towards improving

i. The information contents of the training data
ii. Evaluation of software modules which would be inject the additional knowledge regarding context in which software is used into modelling processes.

V. Conclusion

The key factors of the proposed model is monitoring the functionalities and parts of the system that have to be changed, through this controlling the quality of software modules.

It will predict the probability of presence of faults and estimating and preventing the faults that will reduce the testing effort. Through works model fault proven model is designed that are known in advance for review, analysis and design, testing for incremental development.

References

1. ISO/IEC 9126-1: Software engineering – Product quality. ISO/IEC (2001)
2. ISO 9001: Quality management systems – Requirements. ISO (2001)
3. Basili V.R., Weiss D.M.: A Methodology for Collecting Valid Software Engineering Data,IEEE Transactions on Software Engineering, Nov. (1984)
4. Basili, V.R., Caldiera, G., Rombach, H.D.: The Goal Question Metric Approach. In: Encyclopedia of Software Engineering, Wiley-Interscience, New York (1994)
5. Solingen, R., Berghout, E.: The Goal/Question/Metric Method, McGraw-Hill (1999)
6. Fenton, N: Software Metrics: A Rigorous Approach, Chapman and Hall (1993)
7. Haag, S., Raja, M.K., Schkade, L.L.: Quality Function Deployment Usage in Software Development. In: Communications of the ACM, 1 (1996) 41-49
8. Edsger Dijkstra, "Notes on Structured programming", http://www.cs.utexas.edu/users/EWD/ewd02xx/EWD249.PDF
9. R. Pressman, "Software Engineering: A Practitioner's Approach", 6th Edition, McGraw publishing, 2007.