International Journal on Recent and Innovation Trends in Computing and Communication ISSN 2321 – 8169

Volume: 1 Issue: 3 144 – 147

IMPERFECTION DETECTION MANAGEMENT IN SOFTWARE METRICS

[1]N. Vijayaraj (M.Sc., M.Phil), [2]S. Henry Leo Kanickam (M.Sc., M.Phil)
[1]*Assistant Professor (Department of Computer Science)*
[2]*Assistant Professor (Department of Information Technology)*
[1,2]*St Joseph's College (Autonomous), Trichy – 2.*
[1]*vijay_sjctni@yahoo.com*, [2]*Henryleo.msc@gmail.com*

ABSTRACT: **This paper involved in the software and various software projects. Imperfection Management basically includes identification and taking actions for fault prevention for improving the software quality. Earlier Imperfection identification in software projects will make Imperfection removal and prevention much easier. Software engineers generally take experiences from the past and prevent the Imperfection s from re-occurring. In this research paper our first section is the introduction of software Imperfection s, second section is the introduction of Imperfection Management life cycle models, third section is use of Cost Constructive COQUALMO model for Imperfection Management, fourth section is conclusions and last section is future scope.**

***Keywords*: Imperfection Analysis, Imperfection Prevention, Imperfection Prediction, Root Cause Analysis**

*****

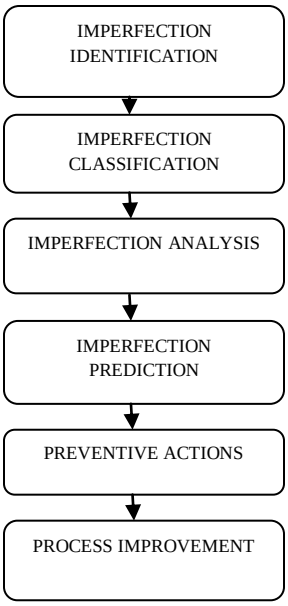
I. INTRODUCTION

Software Imperfection is termed as **"imperfections in the software development process that would cause that software to fail to meet the desired requirements".** An Imperfection generally represents the undesirable aspects of the software quality. Generally we are concerned with three types of Imperfection s artifacts in this research paper and they are: Requirement Imperfection s, design Imperfections and coding Imperfections. Imperfections occur during all the phases of the Software development life cycle. Hence Imperfection prevention is very essential part of Software development Life cycle for improving the Software Quality. Imperfection prevention firstly involves identification of Imperfection, and then modification and changing the relevant processes, preventing the re-occurring of the Imperfection s in the development process. As early as Imperfections are identified in the development process, the more smoothly the development process progresses. For improving the quality of the Software process it is necessary to identify the Imperfection s 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 Imperfections.

II. WORK FLOW STAGES IMPERFECTION MANAGEMENT

2.1 Imperfection Identification OR Imperfection Detection in Software Process

Imperfection Identification is the first activity involved for improving the quality of the Software Process. It is widely used in many of the Software projects, for discovery of the Software Imperfections, then documenting them for improving the quality of the Software product.



Variety of testing techniques used to identify the various types of the Software Imperfection s involved in the Software development process, will may involve either functional or non-functional domains. Hence in software development process firstly Imperfections are discovered and then are documented. Output of Imperfection

document information will be the input for Imperfection analysis technique.

2.2 Imperfection Classification [2]

ODC classifies Imperfection at two different points in time: One is Opener Section, where the Imperfection was firstly investigated and second one is Closer Section, where the Imperfections are fixed. For Small Sized and Medium Sized Projects Imperfection s are classified to first level of ODC to save efforts and time.

For larger projects Imperfection s are deeply understood and analyzed. Firstly for all types of projects Imperfections are firstly investigated and then they are fixed. Action planning and tracing helps in achieving the degree of Imperfection reduction and cross learning.

2.3 Imperfection Analysis [1]

By the term analysis we meant the identification of the root cause of the Imperfection and then further devising the solution to overcome the Imperfection in further development process which will be further useful in improving the software quality and productivity of the software project. Some of the Imperfection analysis techniques such as Fish Bone Analysis, Imperfection Classification and using Imperfection taxonomies and the Root Cause Analysis (RCA). RCA goal is to first identify the root cause of the Imperfection s and then initiating actions for the Imperfection elimination.

2.4 Imperfection Prediction Technique

By the term Imperfection prediction technique we mean to identify and prevent the Imperfection causing failures before occurring. This is done by first gaining the experiences from the earlier Software Projects by Software Engineers and then identifying the root cause for the Imperfection s and then eliminating the causes. Some Imperfection predictions models are COQUALMO model which we have taken and expanded further in this research paper and the second one is mining Imperfection s using ODC. Identification of Imperfection s at the early stages so that wastage in process and product development can be eliminated. And Cost efficiency which involves meeting the deadlines and leading to process improvement for many organizations.

2.5 Imperfection Prevention

By the term Imperfection prevention we mean that gaining the experience from the past projects by the software engineers and identifying the cause of Imperfection s and further taking the corrective measure to prevent the Imperfection re-occurrence. Imperfection Prevention is one very important activity in the Software project thereby further improving the quality of the software project.

2.6 Process Improvement

By the term Process improvement we mean the continuously working for improvement for the quality of the software process. Process Improvement meant that following preventive actions for software improvement and then further taking actions for further improvement of quality.

III. CASE STUDIES

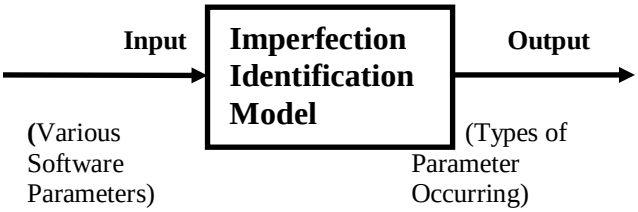
COQUALMO-Imperfection Prediction Model

COQUALMO is a Constructive quality model for the Imperfection prediction density of the Software development. COQUALMO which basically predicts the Imperfection density where Imperfection s conceptually flows from one phase to another thereby making larger Imperfection and leading to project chaos [2]. Imperfection flow from one tank to another tank, thus making larger Imperfection introduction pipes and then Imperfection s are removed with various Imperfection removal pipes.

COQUALMO which is an Imperfection prediction model is basically a two-step process which involves in Imperfection identification (DI) at the first step and the Imperfection Removal (DR) model in the second step thereby leading to the Imperfection improvement of the software quality. Hence two steps of COQUALMO are as follows:

3.1 Imperfection Identification OR Imperfection Introduction Model (DI) [3]

This model typically results in the development of various phases of software development life cycle models. There are basically three types of Imperfections which mainly occurs during the development of various phase of software development life cycle which are as: Requirement Imperfections, Design Imperfections, and coding Imperfections. For Imperfection identification we can use Imperfection. Introduction Range which is particularly the ratio between the highest and the lowest Imperfection identification range. If the Imperfections are identified in the earlier phases of the software development, the better will be the Software Development.



3.2 Imperfection Removal (IR)

By the term Imperfection removal we mean removing the Imperfection s such as requirement Imperfections, design Imperfections and coding Imperfections which would typically results in the software failures. Imperfection removal basically results in the estimation of Imperfection removal by the Imperfection removal activity.

There are basically six orthogonal profiles for the Imperfection removal which are very low, low, nominal, high, and very high and very very high. Very low, low

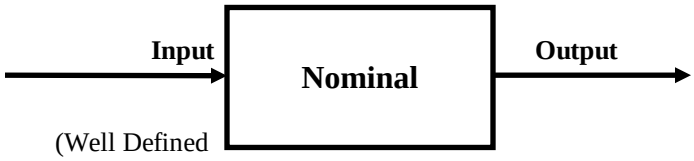
profiles basically requires no peer reviews, no testing and simple compilers, nominal profile requires well-defined sequences, unit and integration and system testing and some more extensive compilers, High,very high and very-very high typically requires formal reviews and procedures, highly advanced testing tools and more formalized compilers.

For Very Low Profiles:

(No testing;
Simple compilers;
No Peer Reviews)

Fig. 3A

For Nominal Profiles:



(Well Defined Sequences;
Basic unit;
Integration and System test;
More extensive Compilers)

Fig. 3B

For High, Very High, and Very Very High Profiles:

(Formal Review Roles And Procedures; Highly Advanced Testing tools and more Formalized compilers)

Fig. 3C

(Various Software Parameters)

Fig. 3D

Basically three types of profiles occurs in the Imperfection prediction in software modules

Prop Minor = minor Imperfection s/total Imperfection s delivered;

Prop Major = major Imperfection s/total Imperfection s delivered;

Prop Extreme = extreme Imperfection s/total Imperfection s delivered.

Minor: It usually means small, a minor Imperfection occurring in software projects does not affect the software much and doesn't make software unusable.

Major: It means large or more major Imperfection occurring in software projects and making the application become unusable.

Extreme: It causes software totally unstable. A failure in any part of the application totally results in unstable software.

IV. CONCLUSION

In this research paper we have first studied about the various types of Imperfection techniques and then we have undergone through the survey of COQUALMO cost constructive model which is a two-step software Imperfection prediction model for improving the software quality. Earlier we identify the Imperfection introduced in the software, the better software will results. Hence, identifying Imperfection s at the earliest software development is very useful and leading to the prevention of software Imperfection s which will lead to failure.

The life cycle of Imperfection identification consists of Imperfection identification, Imperfection classification, Imperfection analysis, Imperfection prediction, preventive actions and process improvement. We have studied three techniques of Imperfection Management i.e. Imperfection Detection Technique, Second Imperfection Analysis Technique, and Imperfection Prediction Technique. We have work flow stages which consist of Imperfection Identification, Second action is Imperfection Classification, and Third action is Imperfection Analysis and Imperfection Prevention. But in this research paper we have used a Process Improvement model which includes Imperfection Identification, Second Action is Imperfection Classification, Third Action is Imperfection Analysis, Fourth Action is Imperfection Prediction and fifth action is Imperfection Prevention, and final Action is Process Improvement. Using these six work flow stages of Imperfection Management provided us the added advantage for improving the quality of software projects, and follows a systematic approach, as in first phase we are able to identify the Imperfection involved in the software, second action classifies Imperfection s into opener section and closer section, third action that is Imperfection Analysis which identifies the root cause of the Imperfection and then further devise the solution to overcome the Imperfection in further development process, fourth action is Imperfection Prediction

technique we mean to identify and prevent the Imperfection causing failures before occurring, and is Imperfection Prevention Technique which means that gaining the experience from the past projects by the software engineers and identifying the cause of Imperfections and further taking the corrective measure to prevent the Imperfection re-occurrence and finally action is for process improvement. After that we have used COQUALMO i.e. cost constructive model as the case studies for demonstrating the fact of Imperfection prediction technique.

But our Work Flow stages that we used in this paper are more complex, further increasing the number of stages of Imperfection Management.

5. Future Scope

Since Imperfection can cause malfunctioning in the software projects leading to software failures, so Imperfection prediction is mainly necessary ,which is one of the important phase in the development of software development life cycle(SDLC).In this research paper we have used COQUALMO which is a two-step software prediction model, thereby improving the software quality. In future scope we can add various aspects for Imperfection Management in various software projects. We can compare Imperfection s with various other techniques such as Genetic algorithm, neural network, fuzzy logic, decision tree for adding feature for handling Imperfection s in software projects.

REFERENCES

1. A Imperfection Prediction and Analysis Using ODC Approach in a Web Application: *Pranayanath Reddy Anantula and Raghuram Chamarthi (Tejoraghuram), ISSN: 0975-9646.*

2. Imperfection Analysis and Prevention for Software Process Quality Improvement-*Sakti Kumaresh and R Baskaran, VLOUME 8-NO.7, OCTOBER 2010.*

3. Constructive Quality Modeling for Imperfection Density Prediction: COQUALMO-*Sunita Chulani, FAST ABSTRACT ISSRECOPYRIGHT 1999.*

4. A Prediction Model for Functional Imperfection s in System Testing using Six Sigma: *Muhammad Dhiauddin Mohamed Suffian and SuhaimiIbrahim, VOLUME 1 NO. 6 SEPTEMBER 2011.*

5. Predicting Imperfection Types in Software Projects: *dr Lukasz Radlinski, Institute of Information in Management, Faculty of Economics and Management, University of Szczecin.*

6. A Machine Learning Based Model for Software Imperfection Prediction: Onur *Kutlubay, Mehmet Balman and Dogu Gul, Ayse B.Bener, BogaziciUniversity, Computer Engineering Department.*

7. A Critique of Software Imperfection Prediction Models, *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL.25, NO. 3, MAY/JUNE 1999.*

8. Butcher (2002), “Improving software testing via ODC: Three Case Studies”, *M.Butcher, H. Munro, T.Kratschmer, IBM Systems Journal, Vol.41, No.1*

9. Brad (2001),” How good is the software: A Review of Imperfection prediction Techniques”, Brad Clark, Dave Zubrow, Carrnegie Mellon University.

10. Mullen (2002) “Orthogonal Imperfection Classification at CISCO”, T.Mullen,D.Hsiao, Proceedings ASM conference.

11. Ram, “Orthogonal Imperfection Classification”. www.Chillaregte.com/odc.

12. Ram (1992), “Adapting ODC to improve software quality: A case Study”, Yang Gu, Software Engineer, IBM http://www.ibm.com.

13. Yang (1992), “Orthogonal Imperfection Classification A Concept for In- Process Measurements”, Ram Chillarege, IEEE Transactions on software Engineering, Vol 18, No.11, November.

14. Paulk (1993), “Capability Maturity Model for Software”, Version 1.1, Mark C.Paulk, Bill Curtis, Mary Beth Chrissis, Charles V.Weber,Software Engineering Institute.

15. Chillarege (2002),”Test and development process retrospective 0a case study using ODC triggers”, Chillarege, R.; Ram Prasad, K.'.

16. Ajit Ashok Shenvi, 2009,”Imperfection Prevention with Orthogonal Imperfection Classification”, In Proc-ISEC ’09, February 23-26, 2009.