Modelling and Analysing of Software Defect Prevention Using ODC

Prakriti Trivedi[1] Som Pachori[2]
[1]HOD, Computer Department, [2]M Tech IV Semester (SE),
Govt Engineering College Ajmer
[1]hod.ceit@rediffmail.com [2]sompachori@in.com

***Abstract*— As the time passes the software complexity is increasing and due to this software reliability and quality will be affected. And for measuring the software reliability and quality various defect measurement and defect tracing mechanism will be used .Software defect prevention work typically focuses on individual inspection and testing technique. ODC is a mechanism by which we exploit software defect that occur during the software development life cycle. Orthogonal defect classification is a concept which enables developers, quality managers and project managers to evaluate the effectiveness and correctness of the software**

***Keywords*— Defect Prevention, ODC, Defect Trigger**

I. INTRODUCTION

Software defect prevention is an important part of the software development. The quality; reliability and the cost of the software product heavily depend on the software defect detection and prevention process. In the development of software product 40% or the more of the project time is spent on defect detection activities. Software defect prevention research has proposed new inspection and testing methods and has studied and compared different inspection and testing methods.

In the paper the basic idea is to provide implementation of ODC in real world application. It begins with and overview of various defect classification schemes followed by ODC concepts. The latter part will describe how will we adopt ODC in software development.

The end of this paper describes the Improvement in software project after implementing ODC.Page Layout

An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

II. DEFECT CLASSIFICATION SCHEME

Since 1975, a number of classification schemes have been developed by different organizations, such as HP and IBM, to classify software defects and to identify common causes for defects in order to determine corrective action

A. Hewlett-Packard - "Company-Wide Software Metrics[9][10]

It classifies defects from three perspectives in three area

(1) Identifying where the defect occurred (e.g., in the design or the code).

(2) Finding out what was wrong (e.g., the data definition or the logic description may be incorrect).

(3) Specifying why it was wrong, missing or incorrect.

B. The IBM Orthogonal Defect Classification Scheme[1]

The IBM Orthogonal Defect Classification (ODC) was originally described in the paper by Chillarege et al. in 1992 [1][4]. As described by Chillarege, the goal of ODC is to provide a scheme to capture the key attributes of defects so that mathematical analysis is possible. The software development process is evaluated based on the data analysis. According to ODC, the defect attributes that need to be captured include: defect trigger, defect type, and defect qualifier. The "defect type" attribute describes the actual correction that was made. For example, if the fix to a defect involves interactions between two classes or methods, it is an interface defect. The "defect trigger" attribute represents the condition that leads the defect to surface. For example, if the tester found the defect by executing two units of code in sequence, the defect trigger is "Test sequencing". "The defect qualifier" indicates whether the defect is caused by a missing or wrong element

III. ODC CONCEPTS

ODC is a defect classification scheme by which we characterize and capture defect information. ODC is a measurement system for software processes based on the semantic information contained in the defect stream. And it can help us evaluate the effectiveness and efficiency of testing, enable error tracking, and evaluate customer satisfaction via an analysis mechanism behind the scheme

A. Defect Trigger

It provides surface to the fault and results in a failure. It just provides a measurement for the development process.

It is very hard for the developer to find the fault during testing Process. For this purpose various verification and testing activities are conducted to find that fault

DEFECT TRIGGER	CLASSIFICATION
REVIEW/INSPECTION TRIGGER	DESIGN CONFORMANCE LOGIC/FLOW BACKWARD COMPATIBILITY LATERAL COMPATIBILITY CONCURRENCY INTERNAL DOCUMENT LANGUAGE DEPENDENCY SIDE EFFECTS RARE SITUATION
FUNCTION TEST TRIGGER	SIMPLE PATH COMPLEX PATH COVERAGE VARIATION SEQUENCING INTERACTION
SYSTEM TEST TRIGGER	WORKLOAD STRESS RECOVERY STARTUP/RESTART HARDWARE CONFIGURATION SOFTWARE CONFIGURATION BLOCKED TEST/NORMAL MODE

B. Defect Type

ODC becomes more understandable when we discuss the
defect type. The defect types are chosen so as to be general enough to apply to any software development independent of a specific product. [2] ODC provides a framework for identifying defect types and the sources of errors in a software development effort. Using the feedback provided by analysis of the defects it inserts in its systems.

- FUNCTION: Affects significant capability, end-user
  interfaces, product interfaces, and interface with hardware
  architecture or global data structure.
- LOGIC: Affects the functionality of a code module and can be fixed by re-implementing an algorithm or local data structure without a need for requesting a
  high level design change.
- INTERFACE: Affects the interaction of components via macros, call statements and/or parameters.
- CHECKING: Affects program logic that would properly validate data and values before they are stored or used in computation.
- ASSIGNMENT: Requires change in a few lines of code, such as initialization of control blocks or data structures.
- TIMING/SERIALIZATION: Affects the proper management of shared and real-time resources.
- BUILD/PACKAGE/MERGE: Affects productversion or configuration; requires formal changes to reconfigure or rebuild the product.

IV. ADOPTING ODC IN SOFTWARE DEVELOPMENT

ODC has its own life cycle, which we can integrate into the Iterative software development lifecycle .With this integration, we can monitor software quality status at each development phase[7]. And if we find some abnormalities in our result we

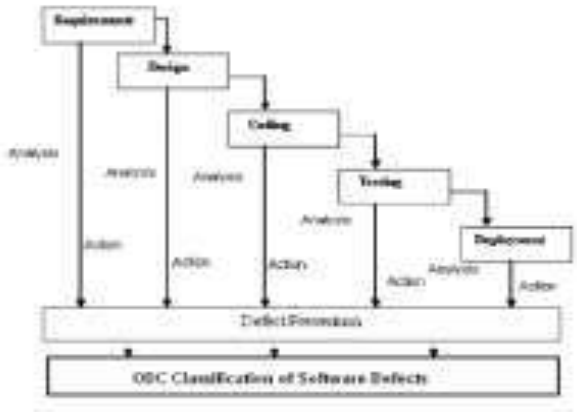


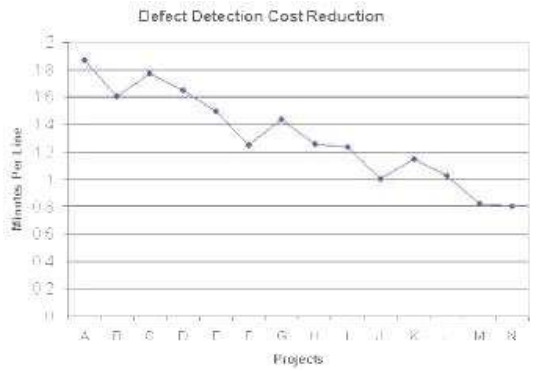
Figure 1:Relation between SDLC and ODC

Therefore, for any given phase, defect detection measures should be taken. The measures must be appropriate to the typical type of defects injected and the information or work product produced. The goal is to minimize the amount of defects that propagates to the subsequent phases.

V. IMPROVEMENT AFTER IMPLEMENTING THE ODC IN SOFTWARE PROJECTS

After implementing ODC in various software project at college level we see that the defect detection rate will be reduced. These projects are very different in terms of the languages (procedure language VB 6 and Object-Oriented language VB .Net ,Web Application develop through ASP.Net, Php), the architectures (client-server and multi-tier), the databases (as simple as SQL and as complicated as Orcale 8i), the resources the complexity (from one week for a single person to several months for eight people), and the characteristics (adding new functionalities to an old system, fixing the bugs in an old system, and developing a new system). With only these projects and the large variations, it is too early to draw a statistical conclusion on what was improved as experience was gained.

The cost of defect detection has dropped dramatically,



although there are some fluctuations

Figure .2: Defect Detection Cost Reduction v/s per project

VI. CONCLUSION AND FUTURE WORK

In this paper we have presented an approach for defining, introducing orthogonal defect classification.ODC can help improve efficiency and effectiveness of development and testing. These are all critical for quality improvement. This paper builds some fundamental work that demonstrated the existence of a relationship between the type of defects and their effect on the software development. By predicting software defect introduction and removal rates, ODC is 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. The use of value-neutral software engineering methods often causes software projects to expend significant amounts of scarce resources on activities with negative returns on investment.. The ODC defect detection efficiency functions are being evaluated for different domains and operational scenarios. This research investigated the software defect detection process to address: how to conduct the process better, how to evaluate and control the process better, and how to continuously improve the process.

REFERENCES

[1] Ram Chillarege (www.chillarege.com)

[2] R. Chillarege, W.-L. Kao, andR. G. Condit, "Defect Type and its Impact on the Growt h Curve," in Proceedings of The 13th International Conference on Software Engineering, 1991

[3] Chillarege, R; Bhandari, I; Chaar, J; Halliday, M; Moebus, D; Ray, B; Wong, M; Orthogonal defect classification - A concept for in- process measurements, IEEE Transactions on Software Engineering, vol. 18, pp. 943-956, Nov. 1992

[4]. IBM Research Center for Software Engineering (http://www.research.ibm.com/softeng/ODC/ODC.HTM)

[5] "Software Triggers as a function of time -ODC on field faults", Ram Chillarege and Kathryn A. Bassin, DCCA-5: Fifth IFIP Working Conference on Dependable Computing for Critical Applications, Sept 1995.

[6] "Improving software testing via ODC: Three case studies", M. Butcher, H. Munro, T. Kratschmer, IBM Systems Journal, Vol 41, No. 1, 2002.

[7]. "Identifying Risk using ODC Based Growt h Models", R. Chillarege, S. Biyani, Proceedings, 5th International Symposium on Software Reliability Engineering, IEEE, Monterey, California, pp 282- 288, November 1994

[8] Butcher, M., Munro, H., Kratschmer, T., Improving Software Testing via ODC: Three Case Studies - Orthogonal Defect Classification, IBM Systems Journal, March 2002.

[9] Kan, S. H., Parrish, J., Manlove, D., In-process Metrics for Software Testing, IBM Systems Journal, 40(1), pages 220- 241, 2001.

[10] Grady, Robert B., Caswell, D., L. Software Metrics: Establishing a Company-Wide Program, Prentice Hall, Englewood Cliffs, NJ, 1987.