Defect Prevention Technique Used In Test Case for Quality Improvement

Abhiraja Sharma
Assistant Professor

Virendra Kumar
Assistant Professor

Som Pachori
M.Tech(S.E.)

Suresh Gyanvihar University
Jaipur-302017

ABSTRACT

To produce high quality software both software developers and testers need continuous improvement in their work methodologies and processes. In this paper, we develop the test case which drives from use case and applying defect classification scheme (ODC) at every test for classifying the defects. For this we conduct an exploratory study on two large web projects to identify a fault classification that is representative of and supported by real world faults. Through our study we provide support to several categories of an existing web application fault classification, and identify new fault categories. Researchers and experimenters will find the proposed fault classification useful when evaluating techniques for testing web applications

General Terms

Experimentation

Keywords

Test case, ODC, defect prevention technique,

1. INTRODUCTION

Defect prevention is a process to find the reason or cause by which defect occurs. SEI gives the definition that “Defect prevention is a process whose purpose is to change to relevant process to prevent that type of defect from recurring”. ODC is a technique that characterizes the different types of defects. The main purpose of this whole process is to produce quality software and this will be achieved when software is defect free.ODC, can improve software quality by offering testing teams a more detailed look at the defects they uncover throughout the software development lifecycle. Idea behind this approach is that instead of generating test cases from traditional specification document, use case model can be used as an effective tool for the generation of Test cases and produce the defect free environment for similar type of projects. In this paper we study two similar types of projects first is web project of hospital and second is web project of college. First we produce test case and after that applying ODC scheme and categories the defect. After this we use of defect data and similar type of test case for another project.

2. STEPS OF WORKING

The first project is ARPAN HOSPITAL. The second GURU CHARYA academic project

2.1 Derive Test Case from Use Case

Advantage of test case generation from use cases is that normally in organizations stakeholders for test cases and use cases belong to different groups. Like, requirement engineers and system developers are responsible to develop and manage use cases while software testers write test cases and test scripts. Although one way elicitation of test cases from use cases is possible but it does not define relationships among test cases and fulfills least traceability requirements between test cases and use cases. It is possible that a change in one test case may have its impact on other test cases; in the absence of relationships among test cases; a state can be reached where a state of disorder among test cases can exist. Similarly based on the interaction among test cases a possible impact is possible on use cases as well.

2.2 Defect Dictation

Defects are found by preplanned activities specifically intended to uncover defects. In general, defects are identified at various stages of software life cycle through activities like Design review, Code Inspection, GUI review, function and unit testing. Once defects are identified they are then classified using first level of Orthogonal Defect Classification.

2.3 Categorize Defect

The classification scheme is ODC. Orthogonal Defect Classification (ODC) is a methodology used to classify software defects. When combined with a set of data analysis techniques designed to suit the software development process, ODC provides a powerful way to evaluate the development process and software product.

2.4 Defect Fix

Find the Root Cause Analysis of defect and fix it. The goal of RCA is to identify the root cause of defects and initiate actions so that the source of defects is eliminated.

2.5 Defect Prevention

Defect prevention is an important activity in any software project. The purpose of Defect Prevention is to identify the cause of defects and prevent them from recurring. Defect Prevention involves analyzing defects that were encountered in the past and taking specific actions to prevent the occurrence of those types of defects in the future. Defect Prevention can be applied to one or more phases of the software lifecycle to improve software process quality

3. DEFECT PREVENTION ON PROJECT

To study of the defect area in software project two similar type of project are taking. These selected projects were developed under **Microsoft .net platform**. Information like number of lines of code (KLOC) produced by the software. Here four Test case Table and four defect table.

3.1 Steps of Working

Generate the TEST CASE the table of “test case” shown blow. Collect the defect from every “test case”. Apply defect prevention technique. Defect density is a measure of the total number of defects in a project divided by the size of the software being measured.

Defect Density (DD) = Number of defects / size (KLOC) – (1)
Defect density is calculated to track the impact of defect reduction and to judge the quality improvement on the project that has implemented defect preventive action with the project that did not follow any preventive action

Table 1: Use Case 1

Use case id	Use Case Description	Test Case ID	Test Case Description
UC1	Successfully login into the system	TC1	To verify that login page is successfully displayed to the user. User can
		TC2	To verify that user with valid login id and password is able to successfully login to the system.
		TC3	To verify that user with invalid login id or password is not allowed to login in the system.
		TC4	To verify that application home page is displayed on successful login into the system.
		TC5	To verify that valid alert message is displayed to the user on un successful attempt to login in the system.
		TC6	To verify that by pressing 'Enter' key of key board user is successfully logged into the system provided that valid user id and password are given.

Table: 2 Defect Table 1 from Use Case Table 1

Test case no.	REQ	Design	LOG	GUI	Doc	Total
TC1	0	1	3	0	0	4
TC2	1	0	6	0	0	7
TC3	0	2	2	0	0	4
TC4	0	0	1	1	0	2
TC5	1	1	5	0	0	7
TC6	1	0	6	1	0	8
TOTAL	**3**	**4**	**23**	**2**	**0**	**32**

Table 3: Use Case 2

Use case id	Use Case Description	Test Case ID	Test Case Description
UC2	Secretary can schedule patient for nurse, physician and lab technician on Scheduling grid.	TC7	To verify that secretary is able to successfully access the scheduling grid.
		TC8	To verify that secretary can select scheduling duration with time unit 5 having any time length within defined duration for time.
		TC9	To verify that by double clicking on the selected time slot duration, 'New Appointment' screen is displayed to the secretary.
		TC10	To verify that secretary can select nurse, physician or lab technician on scheduling main page and schedule patients for these roles.
		TC11	To verify that secretary is unable to save patient data without providing mandatory field values of SEX, NAME
		TC12	To verify that secretary is not allowed to select an already scheduled slot on scheduling grid page.
		TC13	To verify that appropriate alert message is displayed when secretary attempts to schedule a patient on already reserved slot.
		TC14	To verify that alert message is displayed when secretary attempts to schedule a patient on break slots.
		TC15	To verify that secretary is able to schedule a patient on break slots.
		TC16	To verify that secretary is not allowed to schedule a patient on blocked slots.
		TC17	To verify that links and buttons displayed on 'New Appointment'

			screen are functionally active.
		TC18	To verify that secretary can provide remarks for patient on 'New Appointment' screen.
		TC19	To verify that secretary is able to search patient's data on 'New Appointment' screen.
		TC20	To verify that secretary can view patient's history on 'New Appointment' screen.
		TC21	To verify that 'Save' button is shown disabled when secretary first time access 'New Appointment' screen.
		TC22	To verify that calendar widget is opened when secretary clicks 'Calendar' link to provide patient's date of birth.
		TC23	To verify that proper alert message is displayed when secretary provides invalid format for date of birth value.

Table 4: Defect Table 2 from Use Case Table 2

Test case no.	REQ	Design	LOG	GUI	Doc	Total
TC7	1	1	2	0	0	4
TC8	1	0	3	1	1	6
TC9	0	2	1	2	0	5
TC10	0	1	2	1	0	4
TC11	1	0	4	1	0	6
TC12	1	0	2	0	0	3
TC13	0	3	1	0	0	4
TC14	0	1	3	1	0	5
TC15	0	0	1	1	0	2
TC16	1	0	2	0	0	3
TC17	1	1	3	0	1	6
TC18	1	0	2	0	0	3
TC19	0	0	1	2	0	3
TC20	1	1	1	2	0	5
TC21	2	0	1	0	0	3
TC22	0	0	3	0	2	5
TC23	0	2	2	0	0	4
Total	**10**	**12**	**34**	**11**	**4**	**71**

Table 5: Use Case 3

Use case id	**Use Case Description**	**Test Case ID**	**Test Case Description**
UC3	Secretary can update scheduling Grid duration.	TC24	To verify that secretary is able to update already saved scheduling duration.
		TC25	To verify that secretary is not allowed to update scheduling grid duration if already scheduled appointments are affected.
		TC26	To verify that secretary is allowed to update break timings on scheduling grid.
		TC27	To verify that secretary is not allowed to specify break timing outside the available scheduling grid duration.
		TC28	To verify that any update in scheduling grid only affects the grids of specified dates.

		TC29	To verify that any update in scheduling grid doesn't affect the history appointments.

Table 6: Defect Table 3 from Use Case Table 3

Test case no.	REQ	Design	LOG	GUI	Doc	Total
TC24	2	2	3	1	0	8
TC25	1	2	4	0	1	8
TC26	1	0	4	0	0	5
TC27	2	1	3	1	0	6
TC28	2	2	3	1	0	8
TC29	1	0	4	0	0	5
Total	**9**	**7**	**21**	**3**	**1**	**40**

Table 7: Use Case 4

Use case id	Use Case Description	Test Case ID	Test Case Description
UC4	Secretary can update patient Schedules (like reschedule appointment, cancel appointment, update appointment etc.) on scheduling grid.	TC30	To verify that secretary is able to re-schedule a patient appointment to a new slot.
		TC31	To verify that secretary is able to cancel a patient's appointment.
		TC32	To verify that secretary is able to update a patient's appointment.
		TC33	To verify that by double clicking a scheduled slot 'Edit Appointment' screen is displayed.
		TC34	To verify that secretary is not allowed to save data on 'Edit Appointment' screen without providing the mandatory fields data.
		TC35	To verify that secretary is able to re-schedule a patient from one physician grid to another.
		TC36	To verify that links and buttons displayed on 'Edit Appointment' screen are functionally active.
		TC37	To verify that secretary can provide remarks for patient on 'Edit Appointment' screen.
		TC38	To verify that secretary is able to search patient's data on 'Edit Appointment' screen.
		TC39	To verify that secretary can view patient's history on 'Edit Appointment'

Table 8: Defect Table 4 from Use Case Table 4

Test case no.	REQ	Design	LOG	GUI	Doc	Total
TC31	2	2	3	1	0	8
TC32	1	2	2	0	0	5
TC33	1	0	6	2	0	9
TC34	2	0	7	1	0	10
TC35	1	2	6	0	0	9
C36	1	1	4	2	0	8
TC37	1	1	4	0	0	6
TC38	0	2	2	0	2	6
TC39	3	0	3	1	0	7
TOTAL	**15**	**10**	**37**	**7**	**2**	**68**

- **Now we calculate the total no. of defect and defect density**

Table 9: Defect Densities

USE CASE(UC) No	KLOC	No of Defects	Defect Density(Approx.)
1	5	32	0.006
2	8	71	0.009
3	3	40	0.013
4	14	68	0.005

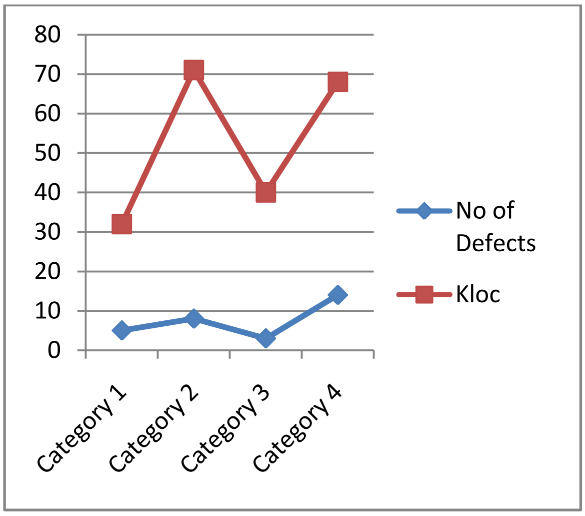

Fig 1: Plotting a graph of KLOC and no. of defect

The project size can be measured either in terms of kilo lines of code (KLOC) produced or in terms of Function Point (FP). For the projects that are taken for study, the project size is measured in terms of KLOC. Comparison is then made between KLOC and number of defect produced by the project. This comparison is depicted in the above figure. From (fig 1), it is evident that, the number varies.

Table 10: Code Descriptions

Code	Name	Description of defect type
REQ	Requirements	Error in understanding the requirements, or inadequate Requirements definition.
DSN	Design error	Error in developing design, or inadequate design, or technical Inadequacy in design.
LOG	Logical error	Logical Error
GUI	Graphical error	Error in screen/report layout and design
DYP	Documentation error	Typographical error in documentation or in code, including spelling errors, mistyped words, and missing delimiters in code.
TC AND UC	Test case and Use case	

Table 11: Observed defect pattern across projects

Use case no	REQ	Design	LOG	GUI	Doc	Total
1	3	4	23	2	0	32
2	10	12	34	11	4	71
3	9	7	21	3	1	41
4	15	10	37	7	2	71
Total	37	33	115	23	7	215

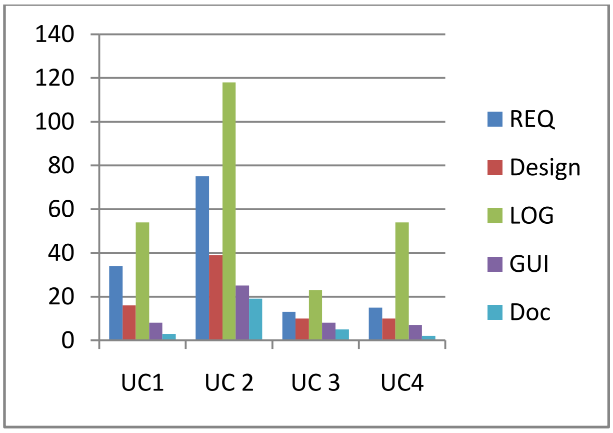

Fig 2: From table 11 we got a chart representation and this chart present no. of defect in classified way.

4. DEFECT PREVENTION

From above tables and graph we collected the defect data now we applying prevention action on second project (similar like first project). Preventive action is implemented in the next set of similar project, and the process improvement was observed in terms of average defect density.

Table 12: Use Case (GURU CHARYA)

Use case id	**Use Case Description**	**Test Case ID**	**Test Case Description**
UC1	Login page	TC1	To verify that login page is successfully displayed to the user
		TC2	To verify that user with valid login id and password is able to successfully login to the system.
		TC3	To verify that user with invalid login id or password is not allowed to login in the system
		TC4	To verify that application home page is displayed on successful login into the system
		TC5	To verify that valid alert message is displayed to the user on un successful attempt to login in the System. Successfully login into the system.
		TC6	To verify that by pressing 'Enter' key of key board user is successfully logged into the system provided That valid user id and password are given.
UC2	Admin and student data	TC7	To verify that admin is able to successfully access the scheduling grid.
		TC8	The student data show properly to admin
		TC9	New changes and up dates shows to admin
		TC10	To verify that by double clicking on the selected Student, 'New Screen' is displayed to the Admin
		TC11	To verify that admin can select Subject, year and result
		TC12	To verify that 'Save' button is shown disabled when admin first time access 'New Student' screen.
		TC13	To verify that do not save same data of student
		TC14	To verify that secretary is unable to save student data without providing mandatory field values of SEX, NAME and AGE.
		TC15	To verify that appropriate alert message is displayed when admin attempts to schedule a student on already reserved slot
		TC16	To verify that links and buttons displayed on 'New Student' screen are functionally active
		TC17	To verify that admin is able to search student data on 'New Student' screen.
		TC18	To verify that proper alert message is displayed when admin provides invalid format for date of birth value
UC3	Admin updates for students	TC19	To verify that admin able to access updating grid
		TC20	To verify that updating shows on page
		TC21	To verify that alert message shows to

			student.
		TC22	To verify that click of student work properly
		TC23	To verify that full page view to student
UC4	STUDENT RESULT	TC24	To verify that login page is successfully displayed to the student
		TC25	To verify that user with valid login id and password is able to successfully login to the system.
		TC26	To verify that user with invalid login id or password is not allowed to login in the system.
		TC27	To verify that result page is displayed on successful login into the system.
		TC28	To verify that the right result shows to right student
		TC29	To verify that DOB and SEX
		TC30	To verify that full page view to student
		TC31	To verify that proper alert message is displayed when admin provides invalid format for date of birth value

We can see that the similar type projects have the similar type of test cases for example the "LOGIN PAGE" is same for both of projects. So the defect prevention can apply similar type of project. Here we implement DP in next project.

Table 12: After D.P

Use case no	Kloc	No of defects	Defect density(ap
1	6	20	0.003
2	8	42	0.004
3	3	35	0.012
4	25	110	0.004

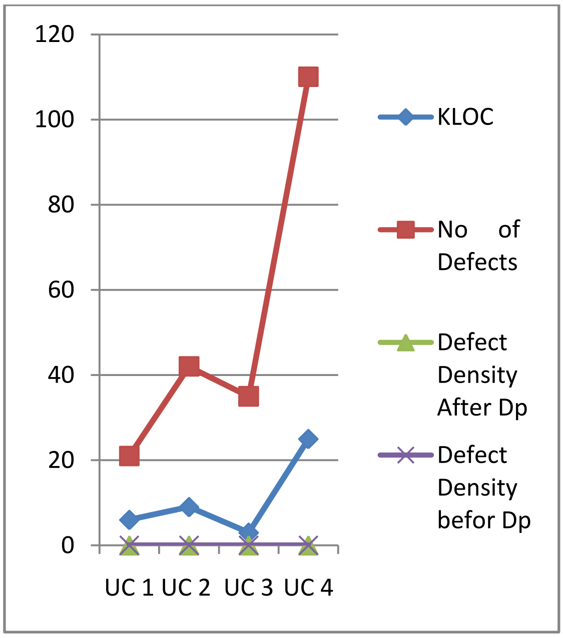

Fig 3: Compression of Defect Density after D.P

The Defect Prevention as provided in the table 1 and table7 shows that the defect density after implementing DP is well below that of defect density before the DP implementation. The average defect density has gone down from 0.0108 (first set of projects-Table 1) to 0.0074 (second set of project – Table 7). By implementing the defect preventive action, not only reduces the defect density, rework effort is also reduced due to which effort involved in various processes is also reduced considerably.

5. CONCLUSION

Implementation of defect prevention action not only helps to give a quality project, but also a valuable investment. Defect prevention practices enhance the ability of software developer to learn from those errors and, more importantly, learn from the mistakes of others. The benefits of adopting defect prevention strategy would be enormous and to list a few.

- Defect prevention reduces development time and cost.
- Increases customer satisfaction.
- Reduces rework effort, hereby decreases cost and improves product quality.

Our work describes a study carried out in a graduate Engineering course in order to identify the patterns and root causes of defects detected in course projects. The root causes were validated through a student survey. From the analysis of these patterns and their root causes, we derived the improvement actions that are useful to design better course projects

6. REFERENCES

[l] Chillarege, Bhandari, et al, "Orthogonal defect classification – A concept for in-process measurement".. IEEE Bansactions on Software Engineering 18, 11 (Nov 1992), 943-956.

[2] Ram Chillarege, Kothanda Ram Prasad "Test and Development Process Retrospective - A Case Study using ODC Triggers" June 2002 DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks

[3] Chillarege, R., and Biyani, S., 1994, "Identifying Risk-using ODC Based Growth Models", Proc. of the fifth International Symposium on Software Reliability Engineering, November 6-9, Monterey, California, 282-288

[4] Bhandari, I.S., Halliday, M.J., Tarver, E.D., Brown, D.D., Chaar, J.K., Chillarege, R "A Case Study of Software Process Improvement During Development," ., IEEE Transactions on Software Engineering, vol. 19, no. 12, December 1993, pp. 1157-1170.

[5] Guide to Software Quality Assurance "Prepared by: ESA Board for Software Standardisation and Control (BSSC) ESA PSS-05-11 Issue 1 Revision 1 (March 1995)

[6] Michal Lyu "Software reliability and system reliability" Orthogonal defect classification April 1996 Handbook of Software Reliability Publisher: McGraw-Hill, Inc.

[7] Pan Tiejun ,Zheng Leina, Fang Chengbin."Defect Tracing System Based on ODC," 2008 International Conference on Computer Science and Software Engineering.

[8] Pankaj Jalote, Naresh Agarwal, 2007, "Using Defect Analysis Feedback for Improving Quality and Productivity in Iterative Software Development" In proc-ITI 3rd International Conference on Information and Communications Technology, pp. 703-713.

[9] Suma V and T R Gopalakrishnan Nair , 2008, " Effective Defect Prevention Approach in Software Process for Achieving Better Quality Levels" Proceedings of World Academy of Science, Engineering and Technology Volume 32 August 2008

[10] Sunita Chulani "Constructive Quality Modeling for Defect Density Prediction: COQUALMO", International Symposium on Software Reliability Engineering (ISSRE'99), Boca Raton, November 1-4, 1999.