Volume 4, Issue 6, June 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Enhancing Software Quality Using Defect Prevention Techniques at Design Phase of SDLC

Ms Suman[1]
M-Tech (CSE), Department of Computer Science & Engineering, Maharshi Dayanand University, Rohtak, India

Dr. Manoj Wadhwa[2]
Professor & HOD, Department of Computer Science & Engineering, Maharshi Dayanand University, Rohtak, India

Abstract- Defect Preventation is one of the important parts in Software Engineering. To improve software quality, the defects will find out at earlier stages of Software Development Life Cycle (SDLC). The purpose of defect preventation is to identify those defects in beginning or earlier phases of SDLC and prevent their reoccurrence .The defects found are generally recorded in database, which is used for tracking and prioritizing defects. The cost of finding and fixing bugs or defects is single largest expense of software. Defect Preventation. Various approaches used for defect detection and preventation. Inspection is required at each phase to achieve the minimal post deployment defects to next phase. The focus of this paper is an approach to software safety analysis based on a combination of two existing fault removal techniques. A comprehensive software safety analysis involving a combination of Design Failure Modes and Effects Analysis (DFMEA) and Design Fault Tree Analysis (DFTA) are conducted on the functions of the critical system during design phase to identify potentially hazardous design faults. A prototype safety-critical system - Lift Door Control System (LDCS), is described here and DFMEA and DFTA technique is applied on a component of LDCS.

Keywords: Defect, Defect Prevention, Root Cause Analysis software safety, LDCS

I. INTRODUCTION

A defect is defined as: *"*An incorrect step, process, or data definition in a computer program.*"* or *"*imperfections in the Software development process that would cause that software to fail to meet the desired requirements" Software defects can be injected during any phase of the software development life cycle. Jones lists the following as sources of defects: requirement errors, design defects, coding defects, documentation defects and incorrect corrections. Hence defect prevention is very essential part of Software development Life cycle for improving the Software Quality.

Defect prevention [8] firstly involves identification of defect, and then modification and changing the relevant processes, preventing the re-occurring of the defects in the development process. As early as defects 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 defects 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 defects

Lyu proposes four techniques which are used in the software development life cycle in order to reduce the amount of defects:

i) Defect prevention takes aim to reduce the number of defects introduce while producing the software in the software development life cycle. This is done directly in any software engineering activity.
ii) Defect removal is to detect defects by software verification or software inspection. The main goal is to eliminate introduced defects. Strategies to achieve this may be dynamic analysis, or formal inspections of code.
iii) Defect tolerance is to provide continuous software service which satisfies given requirements despite a defect having occurred in the software.
iv) Defect forecasting is to estimate where new defects are likely to emerge in the software [1].

II. 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

III. NEED FOR DEFECT PREVENTION

Analysis of the defects at early stages reduces the time, cost and the resources required. The knowledge of defect injecting methods and processes enable the defect prevention. Once this knowledge is used appropriately, the quality is improved. It also enhances the total productivity.

IV. BENEFITS OF DEFECT PREVENTION

There is need in all IT companies to reduce as many numbers of defects as possible. Existences of defect prevention strategies reflect a high level of test process maturity. Detection of errors in the development life cycle helps to prevent the increment of errors from requirement specifications to design and from design into code. Defect prevention decreases the cost and time. Thus, it significantly reduces the number of defects, down the cost for rework, makes it easier to maintain port and reuse, makes the system reliable, and offers reduced time and resources required for the organization to develop high quality systems. Defects that traced in later stages of the life cycle increase product cost and time

V. TECHNIQUES FOR DEFECT PREVENTATION IN DESIGN PHASE OF SDLC

1) Cleanroom

Cleanroom is a theory based process of defect prevention which certifies the high level of reliability of software system under statistical quality control. The whole activity performed by a team. The main objective of clean room technique is to produce software system that shows zero defects in use. In this technique, more efforts put on designing. If any error occurs during the process, the backward traceability performs to fix it at initial level[8].

Benefits:

i) It assures defect free design before testing that causes to save time during testing.
ii) It involves peer reviews also ensures to have well defined requirements with less complicated design.
iii) Simple process, focus on incremental development, easy to understand and maintain software.

2) Failure Modes and Effects Analysis (FMEA)

FMEA stands for Failure Modes and Effects Analysis. FMEA is a step-by-step approach for identifying all possible failures in a design, a manufacturing or assembly process, or a product or service. "Failure modes" means the ways, or modes, in which something might fail. Failures are any errors or defects, especially ones that affect the customer. "Effects analysis" refers to studying the consequences of those failures. Failures are prioritized according to how serious their consequences are, how frequently they occur and how easily they can be detected. The purpose of the FMEA is to take actions to remove or reduce failures, starting with the highest-priority defects. Design Failure modes and effects analysis (DFMEA) is a defect prevention technique used to identify potential failure modes in a product design phase, assess the risk of each potential failure, and then implement appropriate actions to eliminate or mitigate those failure modes. Once identified, this information can be persisted and used in future projects to help avoid defects.

The purpose of DFMEA is to identify possible failure modes of the system components during design phase, evaluate their influences on system behavior and propose proper countermeasures to suppress these effects[4].

FMEA Objectives:

The primary objective of an FMEA is to improve the design.

- For System FMEAs, the objective is to improve the design of the system.
- For Design FMEAs, the objective is to improve the design of the subsystem or component.
- For Process FMEAs, the objective is to improve the design of the manufacturing process. identify and prevent safety hazards
- Minimize loss of product performance or performance degradation
- Improve test and verification plans
- Improve Process Control Plans (in the case of Process FMEAs)
- Develop Preventive Maintenance plans for in-service machinery and equipment
- Develop online diagnostic techniques

Benefits:

i) Identify and remove the possible failure modes in early stage of development.
ii) Focus on prevention measures.
iii) Reduce the reoccurrence of same types of failures in future.

iv) Prioritization of potential failures based on risk helps support the most effective allocation of people and resources to prevent the.
v) Improved software quality and reliability result in an improved customer experience and greater customer satisfaction

Procedure

Step 1: Identify and Describe the Target Product Focus Area
Step 2: Create a DFMEA Worksheet and Enter Initial Data
Step 3: Determine Failure Modes and Add to DFMEA Worksheet

A *failure mode* is a type of failure that could occur. In software systems, this is evidenced by symptoms such as a blue screen, system hang, incorrect output, and data corruption. Identifying Failure Modes Potential failure modes can be identified from many different sources:

- Brainstorming
- Root cause analysis
- Defect taxonomy
- Managing Time

Step 4: Rate Failure Mode Impact and Detectability
Step 5: Calculate the Risk Priority Number for Each Failure Mode.

The *risk priority number (RPN)* is a very simple calculation. It is simply product of the impact rank, likelihood rank, and delectability rank:
*RPN = Impact Rank * Likelihood Rank * Delectability Rank*.

Step 6: Identify the Failure Modes with the Highest Potential Risk with qualitative and quantitative methods
Step 7: Define an Action Plan to Eliminate or Minimize the Causes
Step 8: Reassess the Risk Priority after the Actions performed

3) Fault Tree Analysis (FTA)

In fault tree analysis, Boolean logic is used to describe the combinations of events that may cause the failure in the software product. It takes a single failure and analyze with possible all causes for that failure. FTA finds the opportunities to identify the causes of specific failure and prevent those causes to happen .FTA is a top-down failure analysis used for discovering the root causes of failures or potential failures. It uses Boolean logic to combine a series of lower-level events. The symbols used in a single FTA Logic Diagram are called Logic Gates and are similar to the symbols used by electronic circuit designers[3][4].

Benefits:

i) Identify the combination of possible causes for a specific failure
ii) Graphically presents the combination of causes
iii) Complex and multiple failure modes can be analyzed
iv) The fault tree provides a framework for thorough
v) qualitative and quantitative evaluation of the top event
vi) The fault tree explicitly shows all the different relationships that are necessary to result in the top event.
vii) In constructing the fault tree, a thorough understanding is obtained of the logic and basic causes leading to the top event..

Procedure

Step 1: Select and Define the Failure to Analyze
Step 2: Create the Fault Tree
Step 3: Analyze the Fault Tree
The mathematical concept of cut sets originated in graph theory and has been used in the context of fault trees to mean the unique combinations of basic events that, should they all occur, will cause the failure or undesired event
Step 4: Review the Matrix Rows to Identify Minimal Cut Sets As a reminder, a minimal cut set is a cut set where no events can be removed and still cause the failure if they all occur at the same time.
Step 5: Interpret the Result

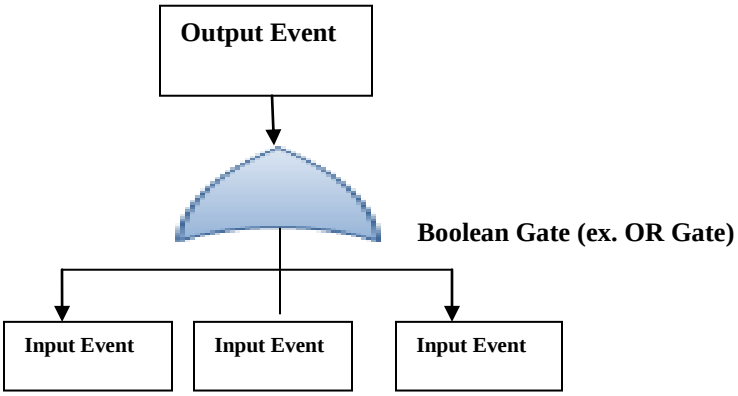

Fig. 1 Basic Fault Tree Constructs

DP Techniques in SDLC Design Phase

- **Design Phase**
  - Cleanroom approach emphasizes on strict verification procedure of design and implementation. It is also helpful for those companies which outsource their development. So for the other companies it will be much easy to implement if the design is defect free and easy to understand.
  - Failure mode and effects analysis (FMEA) technique can be profitable due to different benefits and it is recommended techniques for the design phase. FMEA is a technique to analyze the potential defects in the early development cycle where it is uncomplicated to prevent these issues, ultimately increasing product quality. This technique is beneficial since it design out defect; failure and we got the safe and customer satisfying products. Despite different benefits of other techniques in the design phase, FMEA and FTA approaches increase customer satisfaction, documents risk and actions taken to reduce risk reduce delayed changes and related cost as well as it works as medium for teamwork with exchanging ideas between functions.

VI. LIFT DOOR CONTROL SYSTEM (LDCS)

Lift or Elevator doors protect riders from falling into the shaft. The most common configuration is to have two panels that meet in the middle, and slide open laterally. In wider entryways within limited space, the doors run on independent tracks so that while open, they are overlapped behind one another, and while closed, they form cascading layers on one side. This can be configured so that two sets of such cascading doors operate like the centre opening doors, allowing for a very wide lift cab. Some buildings have lifts with the single door on the shaft way, and double cascading doors on the cab. During a failure of an ingress-egress control system, *e.g.*, a user operate a door open somewhere in a building, a fail secure lock will close, lock, and remain locked even when a user attempts to unlock it with the key that the user usually employs. In such a case, an independent release, such as a reboot or alarming of the securing mechanism, is required. In contrast, a component may be considered failsafe even if its failure does not secure the system. For example, if a door locked from the inside is left unlocked or is unlocked at the wrong time, it has failed (in some cases, along with the entire system), the door may be (but is not necessarily) fail-safe if its being unlocked does not open it or attract additional attention to its unlocked state[4].

VII. SAFETY ANALYSIS OF LDCS

The safety analysis of LDCS software functions takes place in three sequential steps.

A. *Design Failure Mode and Effects Analysis (FMEA*)

A Design FMEA is an analytical technique utilized first by a design responsible engineer/team as a means to assure that, to the extent possible, potential Failure Modes and their associated Causes/Mechanisms have been considered and addressed. End items, along with every related system, subassembly and component, should be evaluated. The severity consequences of a failure, as well as the observability requirements and the effects of the failure will be used to define the criticality level of the function and thus whether this function will be considered in further deeper criticality analysis.

B. *Design Fault Tree Analysis (DFTA)*

After determining the top-level failure events, a complete Design Fault Tree Analysis shall be performed to analyze the faults that can cause those failures in design phase. This is a top down technique that determines the origin of the critical failure. The top-down technique is applied following the information provided at the design level, descending to the code modules. DFTA will be used to confirm the criticality of the functions (as output from DFMEA) when analyzing the design (from the software requirements phase, through the design) and to help:

- Reduce the criticality level of the functions due to software design fault-related techniques used (or recommended to be used)
- Detail the test-case definition for the set of validation test cases to be executed.

C. Simulation of Result

The simulation of the results will be performed after the above two steps in order to highlight the potential discrepancies and prepare the recommended corrective measures.

1) DFMEA Analysis of LDCS

The DFMEA, a sample of which is shown in the Table 1 below presents some software failure modes defined for LDCS. The origin and effects of each failure mode are analysed identifying the top level events for further refinement, when the consequence of this failure could be catastrophic for this system. The top events that were singled out for further analysis of failure mode are Improper Functionality, Inadequate timing of elevator door, incorrect selection of data structure and Resource management. Focus on defect prevention by identifying and eliminating defects in the software design stage helps to drive quality upstream

2) DFTA Analysis of LDCS

Fault tree analysis (FTA) is a technique that uses Boolean logic to describe the combinations of intermediate causal events that can initiate a failure ("unintended event"). FTA starts with a specific failure and strives to enumerate all the causes of that event and their relationships. The FTA is a graphical representation of the conditions or other factors causing or contributing to the occurrence of the so-called top event, which normally is identified as an undesirable event. A systematic construction of the fault tree consists in defining the immediate cause of the top event. These immediate cause events are the immediate cause or immediate mechanism for the top event to occur. All applicable fault types should be considered for applicability as the cause of a higher level fault. This process proceeds down until the limit of resolution of the tree is reached, thereby reaching the basic events, which are the terminal nodes of the tree.

Table 1. Example of DFMEA table for software in design phase of LDCS

Failure Mode	**Causes of Failure**	**Effect**	**Severity of Risk**	**Preventative Solutions**
Poor functionality	• Incomplete requirement • Incorrect selected alternative for final solution • wrong technology used • Lack of info. with wrong estimation objectives of project	Improper working of software which can lead to major failures	Critical	Software should be designed to work in proper functional mode
Improper Timing	• delays • Bad decision making	Unpredictable sequence of operations leading to hazards	Critical	Designing should be such that it runs in proper order
Bad selection of Data Structure	• Incomplete logic • Incorrect Algorithm	Out of memory errors	High	Algorithm logic is Verified for accuracy. Data Structures and Memory overflow is checked.
Resource management	• Less availability of funds • Inadequate time management • Improper Team harmony	Project can be delayed or it can lead to failure of project	High	Proper planning and execution of available resources

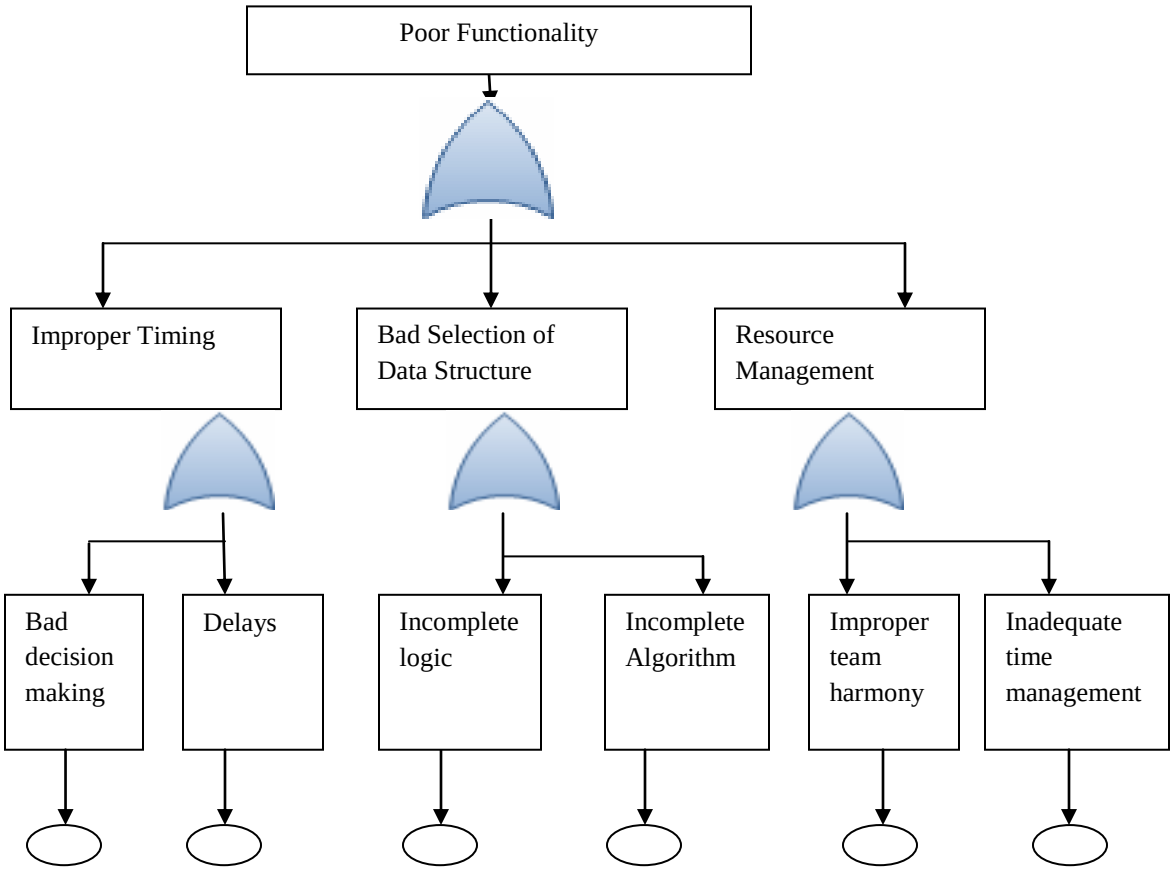

Figure 2. Design Fault Tree sample for top event

VIII. RESULTS

In view of the comprehensive safety analysis, and specification and implementation the safety properties during LDCS design and development, the expected result was that safety-specific LDCS development would produce a software system with fewer latent safety-critical faults than traditional non safety specific techniques alone. This is due to the belief that the safety-specific techniques will prevent safety critical faults in the specifications and designs that the traditional techniques have a tendency to miss. During the operation of LDCS, the safety specific development version of LDCS clearly demonstrated the fulfilment of the safety properties. For example, if the functionality of door of the elevator is not proper then it can lead to various catastrophic hazards, so control program should be designed to work in proper functional mode. If the timing of door open/close is not adequate then it can lead to some unpredictable operations so designing should be such that it runs in proper order. Likewise, in the safety-version of LDCS, if the selection of data structure in design mode is improper then whole software program can go wrong so algorithmic logic must be verified for accuracy before implementing it.

IX. CONCLUSION

Implementation of defect preventive techniques not only helps to give a qualitative project, but it is also a valuable to save cost. Defect prevention practices enhance the knowledge of software developers 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, thereby decreases cost and improves product quality. This paper discussed a FMEA and FTA defect prevention techniques in design phase of software and applied this approach to software safety analysis for critical systems. A comprehensive software safety analysis involving a combination of DFMEA and DFTA techniques was conducted on a component of the critical system to identify potentially hazardous design faults. The safety properties of the prototype elevator door control system were identified as part of the safety critical requirements. We also briefly compared safety-specific and non-safety specific techniques at developing LDCS. The non-safety version of LDCS broadly focused on achieving the functional behavior of the system. The safety-specific version clearly demonstrated that the software safety properties identified in LDCS specification were fully met in the working system.

References

[1] FMEA and FTA analysis for application of the reliability centered maintenance methodology: Case study on Hydraulic turbines ABCM Symposium Series in Mechatronics - Vol. 3 - pp.803- 812

[2] Improvising the Security of Software Application by the Use of Fault Tree Analysis in Decision Making Special Issue of International Journal of Computer Applications on Advanced Computing and Communication Technologies for HPC Applications - ACCTHPCA, June 2012

[3] Software Tool for Distributed Elevator Systems Scientific Publications of the State University of Novi pazar Ser. a: Appl. Math. Inform. And Mech. Vol. 3, 1 (2011).

[4] FMEA and Fault Tree based Software Safety Analysis of a Railroad Crossing Critical System (Global Journal of Computer Science and Technology Volume 11 Issue 8 Version 1.0 May 2011 ISSN: 0975-4172 & Print ISSN: 0975-4350

[5] Defect Analysis and Prevention for Software Process Quality Improvement International Journal of Computer Applications (0975 – 8887)Volume 8– No.7, October 2010

[6] Failure knowledge diagnosis model based on the integration of FMEA and FTA. (IEEE) Print:-ISBN: 978-1-4673-1689-7 DOI: 10.1109/ICCIAutom.2011.6183941 Date of Current Version: 16 April 2012 Issue Date: 27-29 Dec. 2011

[7] Contemporary Trends in Defect Prevention: A Survey Report I.J. Modern Education and Computer Science, 2012,3, 14-20 Published Online April 2012 in MECS DOI: 10.5815/ijmecs.2012.03.02

[8] B. Robinson, P. Francis, and F. Ekdahl "A defect-driven process for software quality improvement," USA: New York, ACM, 2008

[9] P.K. Suri, Rajni Rana, "Defect Analysis and Preventation Techniques for improving Software quality", IJARCSSE, vol 3, issue 7, July 2013

[10]

