Software Quality Assurance and Errors

Wasim Abbas

Abstract— despite the years of effort to ensure the quality of software, software companies still not able to release the software which is completely error free. This paper is focused on the reasons of errors in the software, what kind of errors or bugs in the software, and how can improve the quality of software by minimizing the errors. Software defects have a major impact on software development life cycle. Finding and correcting errors activities are very expensive. It is not possible to avoid errors completely in the software, errors may be inevitable but we can minimize their numbers and impact on our projects. To do this development teams need to implement a defect management process that focuses on preventing defects, catching defects as early in the process as possible, and minimizing the impact of defects.

Index Terms— Error, Failure, Impact, inevitable, Prevention, Software Quality, Testing.

———————————— u ————————————

1 INTRODUCTION

Quality is an important issue in software industry, - good quality is necessary to
Condition of staying in the market – therefore most software companies try to do some improvements in order to deal with this challenge. Software bugs have a major impact on the software quality. Software quality can be improved by focusing on to reduce the errors of the software.
Software development cost is increasing day by day, Software development companies are under more pressure than ever before. These companies have immense pressure to get software products to the marketplace quickly due to their competitors, which pressures to reduce development schedules and also create the pressure to cut costs to reduce development cost of the software. And also software applications are more complex now a day. All of these factors can make it difficult to maintain code quality while managing costs. However, it is always cost effective to minimize defects in the software as early as possible because the cost of fixing defects increases exponentially as software progresses through the development lifecycle, that's why it is important to catch defects as early as possible. The cost of discovering defects after release increases up to 30 times more than if errors are caught in the design and architectural phase.
This paper will discuss the reason of the occurrence of the errors, type o errors; this paper will also provide the approaches and techniques to minimize the error and bugs.
Remaining of this paper is structured as:

Section 2: What is software quality?
Section 3: Types of errors which affect the quality of software
Section 4: Classification of errors of coding
Section 5: Reasons of errors
Section 6: Approaches to minimize the errors.
Section 7: Conclusion.
Section 8: References:

Wasim Abbas is currently pursuing MS (CS) degree program in Department of computer science, Virtual university of Pakistan,PH-0800-880-88, +92-42-111-880-880,Student Id- MS140400134,wasimabbasjoyia@gmail.com

2 SOFTWARE QUALITY

It is hard to define the concept of software quality because it is always difficult to find the adequate quality-criteria, attributes and proper quality-measuring methods and tools.
What does software quality really mean?"
Here are Garvin's 5 quality definitions:

1. The transcendent definition says that quality is absolute and universally recognizable, despite the fact that it cannot be defined precisely. Only experience can teach to recognize quality.
2. According to the user-based definition, quality is "fitness for use". This definition starts from the assumption that the individual customers have different needs and those goods that best satisfy their needs are considered to have the highest quality.
3. The product-based definition regards quality as a precise and measurable variable, as an inherent characteristic of goods.
4. Manufacturing-based definition identifies quality as conformity with specifications. It focuses on the supply side.
5. Value-based definition defines quality in relation to costs. A quality product provides Performance at an acceptable price or conformance at an acceptable cost.

One way to measure the quality of the software product is based on number of bugs or error in the software product. A quality Software would have the minimum number of errors. Remaining of this paper will discuss the errors and error avoidance methods. [2].

3 WHAT IS ERROR?

An error is any flaw or imperfection in a software work product or software process. The term error refers to a Defect, fault or failure. The IEEE / Standard defines the following terms as

3.1 Errors

Human actions that lead to incorrect result. An error can also be defined as an issue or situation calling software change request i.e. if something is broken or not properly built or generated with a reason for not usable in certain cases, it can be error.

3.2 Fault

Incorrect decision taken while understanding the given information to solve problems or in implementation of process. A single error may lead to a single or several faults. Various errors may lead to one fault.

3.3 Failure

It is inability of a function to meet the expected requirements. All above definitions have a causal relationship with each other. Thus an error can be referred to as defect or fault or failure.

4 TYPES OF ERRORS

Following are the most common software errors. This topic surely helps in finding more bugs more effectively. Also, you can use this as a checklist while preparing test cases and while performing testing.

4.1 User interface and procedure errors

Missing functions or incorrect functions that don't fulfill the user expectations, incomplete information, misleading and confusing information, Wrong contents in help text, inappropriate error messages are not properly formed. Performance issues like poor responsiveness can't redirect output, inappropriate use of key board etc. [3]

4.2 Error Handling

Not fully provide protection against corrupted data, tests of user input, version control; Ignores – overflow, data comparison, Error recovery – aborting errors, recovery from hardware problems.

4.3 Loop Boundary related errors

Most of the time during writing the loops programmers usually make mistakes like write wrong boundary conditions i.e. wrong outcome or programmer's assumptions about loop terminations are wrong like infinitely running loop.An infinite loop or endless loop is a sequence of instructions in a computer program which loops endlessly, either due to the loop having no terminating condition, having one that can never be met, or one that causes the loop to start over [6].

4.4 Arithmetic and logical errors

Not good logic, bad arithmetic, outdated constants, calculation errors, and incorrect conversion from one data type to another, wrong formula, incorrect approximation.

4.5 Initial and Later states

Not to set the data item to zero, incorrect initialization of loop-control variable, incorrect initialization of a pointer, to clear a string or flag, Incorrect initialization [3].

4.6 Control flow Errors

Returning state is not correct, Exception handling based exits, Stack underflow / overflow, not able to block or un-block interrupts, Comparison sometimes yields wrong result, Missing / wrong default, and Data Type errors [3].

4.7 Errors in Handling or Interpreting Data

Un-terminated null strings, overwriting a file after an error exit or user abort [3].

4.8 Synchronization error

Assumption that one event or task finished before another begins, resource races, tasks starts before its prerequisites are met, Messages cross or don't arrive in the order sent.

4.9 Load Conditions

Required resources are not available, No available large memory area, Low priority tasks not put off, Doesn't erase old files from mass storage, Doesn't return unused memory.[3]

4.10 Hardware

Inappropriate hardware device, hardware Device unavailable, Underutilizing device intelligence, Misunderstood status or return code, Wrong operation or instruction codes. [3]

4.11 Sources, Version and ID Control

No Title or version ID, Failure to update multiple copies of data or program files. [3]

4.12 Testing Errors

Failure to notice / report a problem, Failure to use the most promising test case, Corrupted data files, Misinterpreted specifications or documentation, Failure to make it clear how to reproduce the problem, Failure to check for unresolved problems just before release, Failure to verify fixes, Failure to provide summary report.

5 REASONS OF ERRORS

There are many reasons for software errors. In most cases humans do mistakes in software design and coding.

Once we know the causes for software defects it will be easier for us to take corrective measures to minimize these defects.

5.1 Human Factor

Soft wares are developed by human. Humans do mistakes. Human beings are not perfect. Then how can we expect perfect software product by human without any defects in it? Hence there are errors in software. Yet, we have not discovered any non human agent who would be able to make software better than human. So we have to rely on the human intelligence to develop software, so in his way we cannot avoid errors. [5]

5.2 Overconfident People

Usually overconfident people prefer to say things like:
'no problem'
'piece of cake'
'I can whip that out in a few hours'
'it should be easy to update that old code'
instead of saying,
'that adds a lot of complexity and we could end up making a lot of mistakes'
'we have no idea if we can do that; we'll wing it'
'I can't estimate how long it will take, until I take a close look at it'
'we can't figure out what that old spaghetti code did in the first place'
If there are too many unrealistic 'no problem's', the result is software bugs.
This overconfidence can cause for errors in the software. [5]

5.3 Communication Failure

Miscommunication is another reason for defects in software. Success of any software application depends on communication between customer, development and testing teams. Unclear requirements and misinterpretation of requirements are two major factors causing defects in software. The communication failure can happen at requirement gathering stage, requirement interpretation / documentation stage, requirement-to-implementation translation stage etc. Miscommunication or erroneous communication creates the situation where programmer has to deal with the problem which is not completely understood by him. In this situation programmer would definitely make the mistakes in the software. Another example of miscommunication is that when a programmer tries to modify code developed by another programmer, he would do the mistakes. [4][5]

5.4 Unrealistic Development Schedule

Due to competition software developing originations are in great hurry to market their software products as soon as possible. This is the reason soft wares are developed under unrealistic release schedules, with limited / insufficient resources. This situation will likely to introduce errors. [4][5]

5.5 Poor Design Logic

Software systems are very complex now-a-days, Lack of time and an urge to complete it as quickly as possible may lead to errors. Without proper understanding of the technical feasibility before designing the architecture of the software would invite errors. People don't have time to think or brain storming.[4][5]

5.6 Poor Coding Practices

Programmers can make mistakes like other human beings. All developers are not domain experts. Programmers without proper domain knowledge can make mistakes during coding. Sometimes errors are occurred into the code due to simply bad coding techniques. Bad coding practices included no use of exception handling or error handling, lack of proper validations (data types, field ranges, boundary conditions, memory overflows etc.) may lead to introduction of errors in the code.

5.7 Poorly Documented Code

it is not easy to modify code that is badly written or poorly documented. The result of this poorly documented code is software errors. In many organizations management provides no incentive for programmers to document their code or write clear, understandable code. In fact, it's usually the opposite; they get points for quick coding.
The new programmer starting to work on this code may get confused due to complexity of the project and poorly documented code. Many times it takes longer to make small changes in poorly documented code as there is huge learning curve before making any code change.[4][5]

5.8 Changing Requirements

changes invites the errors, the customer may not understand the effects of changes, or may understand and request them anyway. Changes affects in the way like redesign, rescheduling of engineers, effects on other projects, work already completed that may have to be redone or thrown out etc. If there are many minor changes or any major changes, known

and unknown dependencies among parts of the project are likely to interact and cause problems, and the complexity of keeping track of changes may result in errors. Enthusiasm of engineering staff may be affected. [5]

5.9 Use of Third-Party Tools

Third party software are used in the software development rottenly, these third party tool can have bugs in them. These tools could be tools that aid in the programming (e.g. class libraries, shared DLLs, compilers, HTML editors, debuggers etc.) .A bug in these tools may become the cause of bugs in the software that is being developed by using these tools. [4][5]

5.10 Poor Testing

Testing is very important in finding the errors but unfortunately, there are some shortcomings in the testing process like lack of seriousness for testing, lack of skilled tester, testing activity conducted without much importance given to it etc. due to poor testing errors are not fully eliminated in the software.

6 APPROACHES TO MINIMIZE THE ERRORS.

The strategies to minimize the errors in the software products are as follows.

6.1 Defect Prevention

Analysis of the defects at early stages reduces the time cost and the resources required. Techniques like defect injecting methods enable the defect prevention. Defect preventive techniques improve the quality of the software product. Defect prevention can be achieved with automation of the development process. There is several tools available right from the requirements phase to testing phase. Tools used at this phase include requirement management tools, requirements recorders tools, requirement verifier's tools etc. the design tools include database design tools, applications design tools, visual modeling tools like Rational Rose and so on.[1]

6.2 Train and Educate the Programmer

Human is biggest source of the errors by providing them training these error can be avoided. Train the people and educate them in product and domain specific knowledge is a preventive approach to minimize the errors in the software product. Developers should improve the development process knowledge and expertise in software development methodology.

6.3 Defect Identification

There are several approaches to identify the defects like inspections, prototypes, testing and correctness proofs.

6.3.1 Formal Inspection

It is the most effective and expensive quality assurance technique for identifying defects at the early stages of the development? Through prototyping several requirements are clearly understood which helps in overcoming the defects.

6.3.2 Testing

It is one of the least effective techniques? Those of the defects, which could have escaped by identification at the early stages, can be detected at the time of testing. Testing phase can be automated by the use of tools like code generation tools, code testing tools, code coverage analyzer tools. Several tools like defect tracking tools, configuration management tools and the test procedures generation tools can be used in all phases of development

Correctness proofs are also a good means of detecting errors especially at the coding level. [1]

6.4 Categorize The Errors

After identification of errors first, categorized causes, defects, defect type, defect severity and defect priority in following category: Cause category: -Coding / Logic -Inconsistent with requirements -Inadequate Error Handling -Test Coverage -DB Issue -Network Issue -Software Issue -Hardware issue Type: -Cosmetic / UI -Functional Error Severity 1 – Critical 2 – High 3 – Medium 4 – Low Priority 1 – Critical 2 – High 3 – Medium 4 – Low Using this categorization, weights can be assigned and each organization can use this weight to prioritize the defects. Once we know the propriety of defects, preventive measures like training and education of programmer, mentoring, per reviews can be taken to prevent the defects. [1]

6.5 Reuse Of Tested Code

Errors and bugs can also be reduced by reusing the code which is already tested and proved to be error free. However, reusing code is not always a good approach, particularly when business logic is involved. Reuse in this case may cause serious business process bugs

6.6 Avoid Legacy problems

Before reusing old code, libraries, APIs, configurations etc. it must be considered if the old work is valid for reuse, or if it is likely to be prone to legacy problems.

Legacy problems are problems which inherent when old designs are expected to work with today's requirements, especially when the old designs were not developed or tested

with those requirements in mind like 32 bit architecture based source code would not work with 64 bit architecture.

7 Conclusion

The overall goal of my paper is to discuss the errors due to which software quality suffers. What kind of error is common in software products? This paper also covered the reasons of the occurrence of the errors in the software products and paper also provided the identification method of errors and preventive measures of identified error so that these errors would be avoided in the software products to ensure the quality of the software product.

8 References

[1] "On Minimizing Software Defects during New Product Development Using Enhanced Preventive Approach" Khaleel Ahmad, Nitasha Varshney.

[2] Product Quality by Dr. GARVIN (1984).

[3] Padmini C (Author of Beginners guide to Software Testing).

[4] Softwaretestingtricks.com / 2008 / 12 / why-are-bugsdefects-in-software.html.

[5] Softwaretestinghelp.com / why-does-software-have-bugs.

[6] Softwareengineering.stackexchange.com