Software Testing and Quality Assurance

DCAP503

Editor
Manpreet Kaur

LOVELY
PROFESSIONAL
UNIVERSITY

LOVELY
PROFESSIONAL
UNIVERSITY

SOFTWARE TESTING AND QUALITY ASSURANCE

Edited By
Manpreet Kaur

ISBN: 978-93-87034-85-3

Printed by
EXCEL BOOKS PRIVATE LIMITED
Regd. Office: E-77, South Ext. Part-I, Delhi-110049
Corporate Office: 1E/14, Jhandewalan Extension, New Delhi-110055
+91-8800697053, +91-011-47520129
info@excelbooks.com/projects@excelbooks.com
internationalalliance@excelbooks.com
www.excelbooks.com

for
Lovely Professional University
Phagwara

CONTENTS

SYLLABUS

Software Testing and Quality Assurance

Objectives: The objective of this course is to impart understanding of techniques for software testing and quality assurance. To help students to develop skills that will enable them to construct software of high quality - software that is reliable, and that is reasonably easy to understand, modify and maintain.

S. No.	Description
1.	***Introduction to Software Testing:*** Introduction, Definition of a Bug, Role of a Software Tester, Software Development Model, Software Testing Axioms, Software Testing Terms and Definitions.
2.	***Fundamentals of Software Testing:*** Testing Strategies and Techniques, Structural and Functional testing, Static Black Box and Dynamic Black Box Testing Techniques.
3.	***White Box Testing:*** Static White Box Testing, Dynamic White Box Testing.
4.	***Special Types of Testing:*** Configuration Testing, Compatibility Testing, Graphical User Interface Testing.
5.	***Documentation and Security Testing:*** Documentation Testing, Security Testing.
8.	***Web site Testing:*** Web Page Fundamentals, Black Box Testing, White Box Testing and Gray Box Testing, Configuration and Compatibility Testing.
9.	***Testing Tools:*** Benefits of Automation Testing, Random Testing, Bug Bashes and Beta Testing.
8.	***Test Planning:*** Test Planning, Test Cases, Bug life cycle.
9.	***Software Quality Assurance:*** Definition of Quality, Testing and Quality Assurance at Workplace, Test Management and Organizational Structure, Software Quality Assurance Metrics, Quality Management in IT.
10.	***Organizational Structure:*** CMM (Capability Maturity Model), ISO 9000, Software Engineering Standards.

Unit 1: Introduction to Software Testing

CONTENTS

Objectives

After studying this unit, you will be able to:

- State the importance of testing in the software development life cycle
- Describe the evolution and types of software testing
- Define a bug and illustrate the reason of occurrence of a bug
- Describe the software development models

Introduction

The success of any software product or application is greatly dependent on its quality. Today, testing is seen as the best way to ensure the quality of any product. Quality testing can greatly reduce the cascading impact of rework of projects, which have the capability of increasing the budgets and delaying the schedule. The need for testing is increasing, as businesses face pressure to develop sophisticated applications in shorter timeframes. Testing is a method of investigation conducted to assess the quality of the software product or service. It is also the process of checking the correctness of a product and assessing how well it works.

The process of testing identifies the defects in a product by following a method of comparison, where the behavior and the state of a particular product is compared against a set of standards which include specifications, contracts, and past versions of the product. Software testing is an incremental and iterative process to detect a mismatch, a defect or an error. As pointed by Myers, "Testing is a process of executing a program with the intent of finding errors".

According to IEEE 83a, "Software testing is the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies specified requirements."

1.1 Software Testing

Software testing is an integral part of the software development life cycle which identifies the defects, flaws or the errors in the application. It is incremental and iterative in nature. The goal of testing as described by Millers states that, "The general aim of testing is to affirm the quality of software systems by systematically exercising the software in carefully controlled circumstances". Let us now list out the objectives of software testing:

1. It ensures if the solutions meet the business requirements, thereby enhancing customer confidence.
2. It catches the bugs, errors and defects.
3. It ensures if the system is stable and ready for use.
4. It identifies the areas of weakness in an application or product.
5. It establishes the degree of quality.
6. It determines user acceptability

The scope of testing can be comprehensive and could examine components like business requirements, design requirements, programmer's code, hardware configuration, and systems administration standards and constraints. The ambit of testing could also involve testing with respect to industry standards and professional best practices. Testing thus provides an opportunity to validate and verify all aspects of software engineering.

Thus, software testing brings many benefits to an organization. It saves time and money, since defects are identified early. Testing ensures that the product is stable with less downtime, thereby gaining customer satisfaction.

Did you know? In the year 2003, social security checks for 50,000 people were mailed by the U.S. Treasury Department without any beneficiary name. It was later found out that the missing names were due to a software program maintenance error.

1.1.1 Evolution of Software Testing

During the early days, software was developed by small groups of people, but had to be maintained by a different set of people. It was at this time that people who were maintaining the software had nightmarish experiences, as the software would throw up a lot of errors. Due to this, it was difficult to handle large projects and thus, tasks were not completed on time and within budget. As a result, projects were delayed and some of them were abandoned halfway.

Software testing evolved over time. Advances in software development brought in new changes in the way testing was perceived, which led to systematic improvement in testing.

Deve Gelperin and William C. Hetzel classified software testing based on the goals and phases of testing, which are as follows:

1. ***Until 1956 - Debugging Oriented:*** Until 1956, testing and debugging were not differentiated, and testing connoted debugging.
2. ***1957–1978 - Demonstration Oriented:*** During this period, the goal of testing was to make sure that the software satisfies its specifications. The period of 70s also witnessed the acceptance of the idea that software could be tested exhaustively.
3. ***1979–1982 - Destruction Oriented:*** During this period, testing grew in importance with contributions from Myers (author of "The Art of Software Testing"). Test cases that had the intent of finding errors were designed and verification and validation activities were initiated.

4. ***1983–1987 - Evaluation Oriented:*** During this period, Guidelines for Lifecycle Validation, Verification and Testing of Computer Software was created by the National Bureau of standards in 1983. This methodology enabled testing to be integrated with analysis, design and implementation of the software lifecycle.

5. ***1988-2000-Prevention Oriented:*** During this period, the goal of testing metamorphed into a mechanism to prevent errors rather than just detecting errors. Hetzel, in 1991, gave the definition of Testing as "Testing is planning, designing, building, maintaining and executing tests and test environments". There was greater emphasis of early test design and a process, consisting of three phases -- test planning, test analysis, and test design evolved.

1.2 Definition of Bug

A bug, also known as a software bug, is an error in a software program that may produce incorrect, undesired result or prevent the program from working correctly. In software testing, a bug not only means an error, but anything that affects the quality of the software program. Software bugs take different names such as – defect, fault, problem, error, incident, anomaly, failure, variance and inconsistency and so on.

The following are certain conditions that result in a bug in a software:

1. If the software does not respond or act in the way as stipulated in the product specification.
2. If the software behaves in a way that is stipulated in an opposite way in the product specification.
3. If the software responds or reacts in a way that is not mentioned in the product specification.
4. If the software does not behave in the mandatory way as expected --perhaps, this might not be mentioned in the product specification.
5. If the software is difficult to understand or has cumbersome steps to follow, or if it is slow in its reaction.

The above rules can be applied to a calculator to understand it better. We are aware that the calculator should perform the operations of addition, subtraction, multiplication and division. If the + key does not work, the bug follows the rule specified in first condition, namely, no response in the way as stipulated in the specification. The calculator is not expected to crash and freeze, and if it does, the bug follows the rule specified in condition 2, that is behaving in the opposite way as stipulated in the specification. If there are additional features found in the calculator, namely a function to find the squares of a number, but is not mentioned in the specification, it falls in the rule specified in the third condition, where there is no mention in the product specification. Let us assume that the battery condition is weak, and at this stage, the calculator does not give the right answers. This condition is a typical situation which follows condition four. Condition 5 brings the customer perspective. It occurs whenever the performance of the software is not in an acceptable way to the user – a sample case where the tester finds the space allotted for the button very small or if the lights are too flashy that the user is unable to see the answer, it follows condition 5. The feedback of the tester becomes important, as it brings out the negatives even before the product reaches the customer.

Example: During the test phase of a mobile phone application, it was detected that when the numerical key, zero, is clicked, the value is not displayed on the screen. This might have occurred due to many reasons- hardware or software related issues. However, though no error occurs in such a condition, the quality parameter of the mobile application is compromised, and hence it is considered as a bug.

1.2.1 Types of Bugs

Software bugs, which occur irrespective of the size of the program, are generally encountered when different groups of developers work to develop a single program. We will now list the common types of bugs and their causes.

Some of the common types of software bugs include the following:

1. Bugs due to conceptual error

Example: Incorrect usage of syntax in the program, misspelled keywords, using wrong or improper design or concept.

2. Math bugs

Example: Divide by zero error, overflow or underflow, lack of precision in arithmetic values due to incorrect rounding or truncation of decimal values.

3. Logical bugs

Example: Infinite loops, infinite recursion, applying wrong logic, incorrect usage of jump or break conditions.

4. Resource bugs

Example: Stack or buffer overflow, access violations, using variables that are not initialized.

5. Co-programming bugs

Example: Concurrency errors, deadlock, race condition.

6. Team working bugs

Example: Out of date comments, non-matching of documentation or files, linking the program to incorrect files.

Reasons for Occurrence of a Bug

Bugs can occur in a program due to many reasons. Some of the common reasons that may cause bugs in software are:

1. ***Human Factor:*** The developer who develops the software is human and there is always a chance that some error, incorrect usage of logic or syntax or improper linking of files might occur. Since the entire software development process involves humans, it is prone to errors.

2. ***Communication Failure:*** The lack of communication or communicating incorrect information during the various phases of software development will also lead to bugs. Lack of communication can happen during various stages of Software Development Life Cycle (Requirements, Design, Development, Testing, and Modification). Many bugs occur due to lack of communication when a developer tries to modify software developed by another developer.

   Example: The requirements defined in the initial phase of the Software Development Life Cycle may not be properly communicated to the team that handles the design of the software. The design specifications that reach the development team may be incorrect, which could lead to occurrence of bugs in the software.

3. ***Unrealistic Development Timeframe:*** Sometimes developers work under tight schedules, with limited or insufficient resources, and unrealistic project deadlines. Compromises are made in the requirements or design of the software in order to meet the delivery requirements. Most of the time, it leads to wrong understanding of the requirements, inefficient design, improper development, and inadequate testing.

4. ***Poor Design Logic:*** During complex software development, some amount of R&D and brainstorming has to be done to develop a reliable design. It is also important to choose the right components, products and techniques for software development and testing during the design phase. If there is any lack of proper understanding in the technical feasibility of the components, products, or techniques in the development process, it may cause bugs in the software that is being developed.

5. ***Poor Coding Practices:*** Poor coding practices such as inefficient use of codes, misspelled keywords, or incorrect validation of variables will lead to bugs. Sometimes, faulty tools such as editors, compilers, and debuggers generate wrong codes, which cause errors in the software and it is difficult to debug such errors.
6. ***Lack of Skilled Testing:*** If there are any drawbacks in the testing process of the software, bugs go unnoticed.
7. ***Change Requests:*** If there are last minute changes on requirements, tools, or platform, it could cause errors and can lead to serious problems in the application.

Task

Browse the web and collect information pertaining to some of the famous bugs encountered during a project and the reasons for their occurrence.

http://www.wired.com/software/coolapps/news/2005/11/69355
http://www5.in.tum.de/~huckle/bugse.html
http://www.cds.caltech.edu/conferences/1997/vecs/tutorial/Examples/Cases/failures.htm

1.2.2 Cost of Bugs

Any bug that exists in the software delivered to the customer can lead to huge financial losses. It will bring down the reputation of the organization developing the software. In case the bug causes any damage to life or property of the customer, it can also lead to huge penalty on the organization that developed the software. Bugs that are detected during the testing stage will take some time to get fixed, affecting the timeline and increasing the project cost.

If the bugs are detected and fixed at the earliest in the software development life cycle, it can result in higher return on investment for the organization with respect to time and cost. The cost of fixing a bug differs depending on the development stage at which it is detected. Let us analyze the impact when bugs are detected at different stages of project life cycle.

1. ***Requirement Stage:*** It would cost the organization some time to rewrite the requirements and can be easily detected and fixed.
2. ***Coding Stage:*** It would make the developer spend some time to debug. However, the time required to fix the bug will vary depending on the complexity of the bug. A bug detected by the developer at this stage can be resolved easily, since the developer understands the problem and will be able to fix it.
3. ***Integration Stage:*** It will require more time to be resolved. Since the problem occurs at a higher level, it requires time to check the part of the code or configuration which is wrong and additional time to fix the bug. The developer and other system engineers will be involved in detecting and resolving the bug.
4. ***Testing Stage:*** It will involve the developer, system engineer, and project manager for detecting and resolving the bug. It is an iterative process, which takes a lot of time and effort with respect to man hours and cost in order to detect and fix the bug. Since the bugs have to be tracked and prioritized for debugging, it will increase the project cost in terms of man power and tools required and causes delay in the delivery times as well.
5. ***Production Stage:*** It will take huge time and investment for the organization, as it involves developers, system engineers, project managers, and customers for detecting and resolving the bug. It demands prioritizing and detailed planning when compared to a bug detected in testing stage. Such bugs not only cause a huge loss, but also bring down the reputation of the organization.

A bug found and fixed during the early stages – say, while defining the specification or requirements, might cost the company a very small amount, for example, consider it to be one rupee per bug. If the same bug is found during the coding and testing stage, it costs the company reasonably, for example, 10 to 100 rupees. If the same bug is found by the customer, then the cost would be very high, that is, the cost the company has to pay can be from a few thousands to a few lakhs. Along with it, the reputation of the company will also be damaged.

Caselet

Costly Bug that Caused the Death of Soldiers

The Patriot missile defense system of U.S. Defence forces was first put to use during the Gulf War (1990/1991) as a counter attack against the Iraqi Scud missiles. Though the Patriot is considered to be a very successful anti-missile system, it failed to defend against the enemy missiles attacks several times. A well known incident is the killing of 28 U.S. soldiers in Dhahran, Saudi Arabia. This incident took place since the anti-missile system had failed to defend against the enemy missile attack.

Later it was found that the failure had occurred due to a bug. Due to a simple timing error in the anti-missile system's clock, the tracking system was not accurately tracking the anti-missiles after 14 hours and the missile system was operational in Dhahran for more than 100 hours.

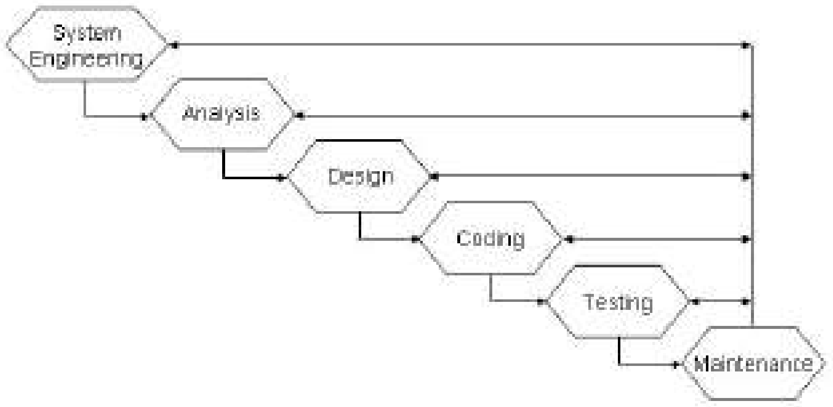
1.3 Software Development Models

A software life cycle development model describes the different phases or activities of a project from its conception. Various software development models are used based on the requirement of the project. We shall now describe some of the most popular and widely used software development models, which include the waterfall model, V model, spiral model, RAD model, prototyping model, and Agile model. According to the nature of the model, the testing approach also differs.

1.3.1 Waterfall Model

This is one of the oldest software lifecycle models. The process starts at the system level and is followed by various phases like the analysis, design, coding, testing and maintenance as depicted in figure 1.1.

Figure 1.1: Waterfall Model



1. ***System Engineering:*** In this phase, system requirements that are essential for the development of the software are defined. These requirements mainly define the software and the hardware requirements relevant for the software development process.
2. ***Analysis:*** In this phase, the developers conduct feasibility studies to define the goals of development. The performance and interfacing requirements for the software are listed out.

3. ***Design:*** In this phase, the requirements described during the analysis phase are defined in terms of the software structure – for example, the database design is completed. These representations help in determining the logical flow of the software and also help in quality assessment. These specifications help the developers to provide inputs during the actual coding process.

4. ***Coding:*** This is the programming phase where the design is developed into a machine-readable form. The developer writes the source code using a software language as per the design specification.

5. ***Testing:*** In this phase, the software is tested against the documented test methods. Testing detects the possible bugs and makes the necessary corrections. These tests also enable the developers to know whether the software is performing according to the requirements.

6. ***Maintenance:*** As new requirements arise, there is a need to upgrade the software. There are instances when problems which need to be solved during the live production environment arise. These are done during the maintenance phase.

The main drawback of this model is that the errors and defects that are present in one phase of the lifecycle are passed on to the other, which results in longer delays and additional costs as the problems need to be solved at each stage.

1.3.2 Spiral Model

The drawbacks of the waterfall model are overcome in the spiral model. The model consists of four phases - planning, risk analysis, design engineering and customer evaluation. The four phases are iteratively followed till the problems are rectified. Two to three prototypes are developed before the final product is delivered. Each prototype follows the entire cycle to solve problems. This method of software development enables to understand the problems associated with a particular phase and deals with those problems when the same phase is repeated again. The figure 1.2 is the pictorial representation of spiral model. The various phases involved in each cycle are,

1. ***Plan:*** In this phase, the specifications, objectives, constraints, and alternatives of the project are listed in logical order as per the project requirements. The objectives and specifications are defined in order to decide the strategies to be followed during the project life cycle.

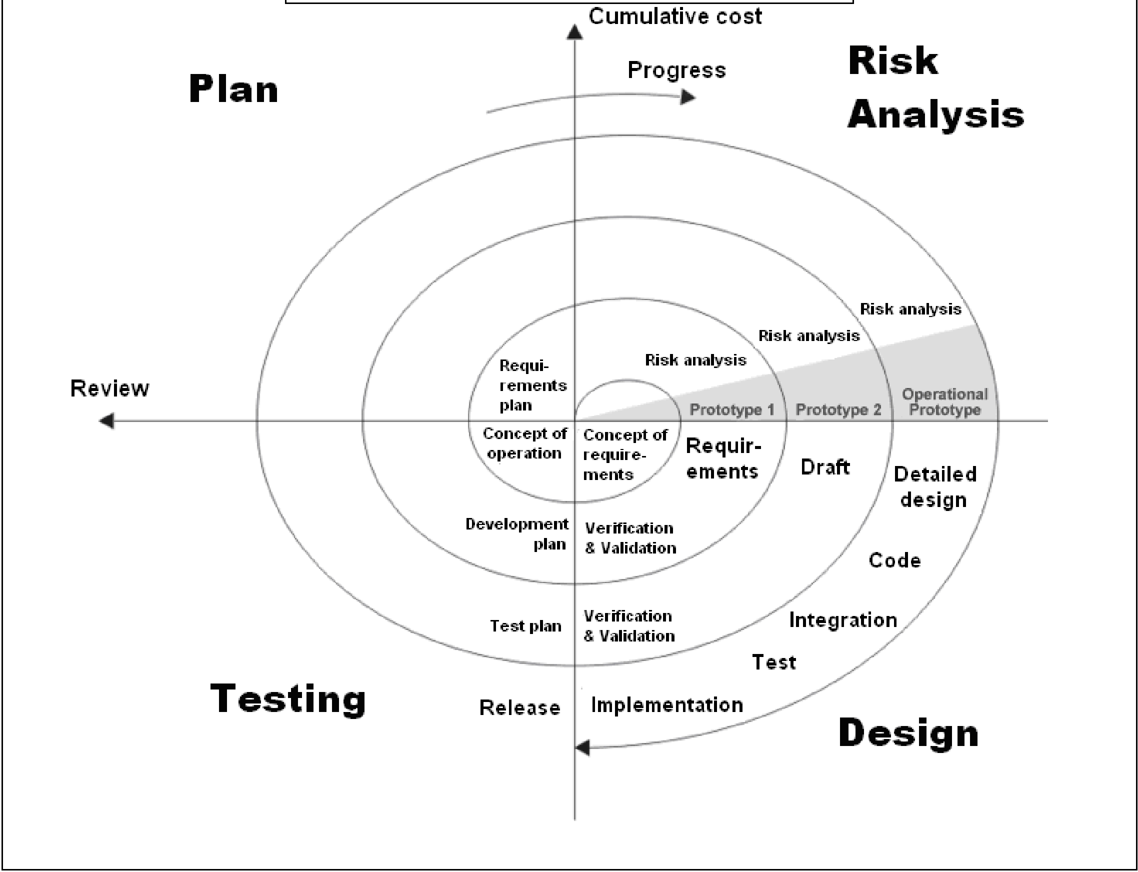
2. ***Risk Analysis:*** This is a very crucial phase of spiral model. During this phase, all the alternatives that are helpful in developing a cost effective project are analyzed and all possible risks involved in the project development are identified.

3. ***Design Engineering:*** This is the phase where the actual development of the software takes place. The software product is developed iteratively and passed on to the next phase.

4. ***Testing:*** In this phase, the customer receives the product and gives comments and suggestions which can help in identifying and resolving potential problems in the developed software. During this cycle all the phases concentrate on the feedback received from the customer and the testing team to resolve the drawbacks and bugs found in each prototype of the product. The main drawback of the model is the amount of time taken to complete the iterations which can increase costs. Testing at the customer’s end and fixing of bugs might require higher cost and time.

Figure 1.2 shows the Spiral model of software development.

Figure 1.2: Spiral Model



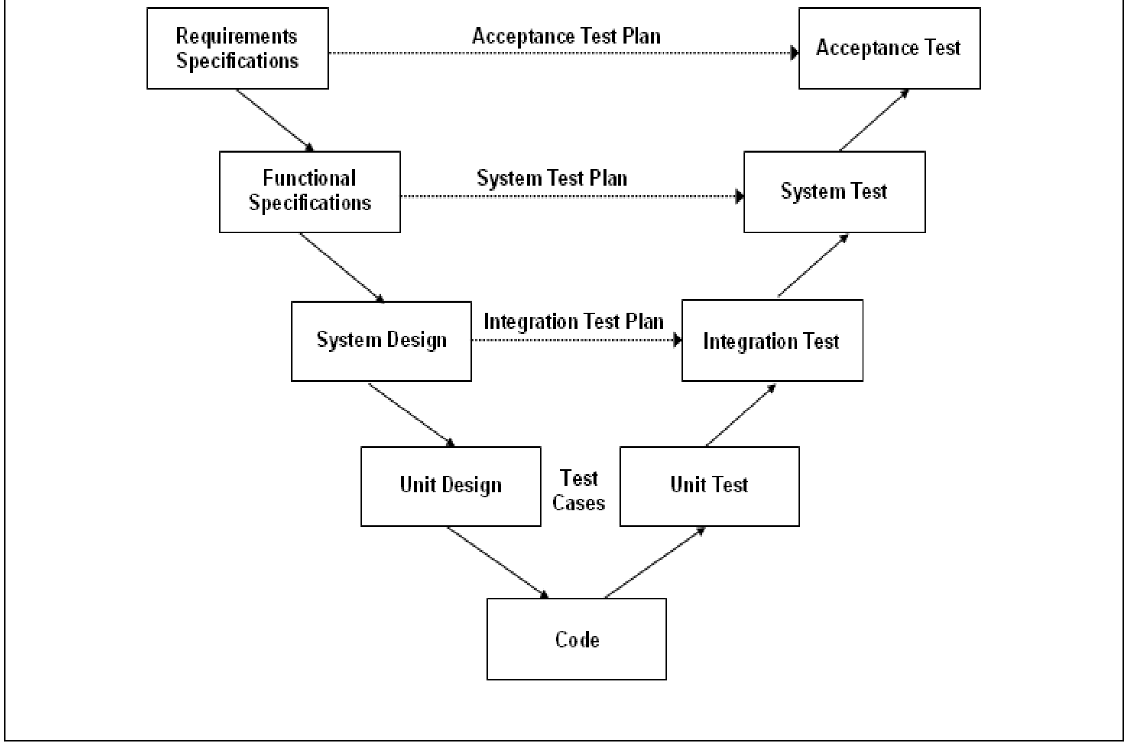
1.3.3 V-Model

The drawbacks of other models are overcome if testing starts at the beginning of the project. V model is a popular method since it incorporates testing into the entire development life cycle. This model ensures proper quality checks throughout the project lifecycle.

The salient aspect of V model is that it portrays distinct testing levels and illustrates how each level addresses a different stage of the lifecycle. The development activities begin with defining the business requirements, where it moves from the high level to low level design, whereas the testing cycle moves from the low level to the higher level. The customer provides the requirement specifications that define the business requirements, which is followed by the functional specification phase. In this phase the process, structural, and event models are developed to understand the requirement specifications carefully. It is very important to carefully define the specification during this stage, since it decides the effectiveness of testability during the test phase, i.e., during system test stage.

Figure 1.3 shows the pictorial representation of the V model. The left half of the V model is the software development phase and the right side of the diagram shows the test phase.

Figure 1.3: V Model



During the unit design stage, the individual programs or modules of the entire software are specified, based on which the test cases for unit testing are developed. Test cases are constructed to check various aspects of the software. These checks are carried out to test the actual program structure or to test whether the software functions as per the specification.

Example: While developing test cases to check the software developed for a calculator, we could check the actual code logic used to perform certain calculations, memory required by each module, techniques followed to link various modules of the software, and the overall efficiency.

Unit testing focuses on the types of faults that occur when writing code. Integration testing focuses on low-level design, especially those errors that occur in interfaces. System test evaluates if the system effectively implements the high level design, specifically the adequacy of performance in a production setting. Acceptance tests are performed by the customers to ensure that the product meets the business requirements. The powerful benefit of this model is that testers are able to verify and validate the specification at an earlier point in the project. This reduces the defects and builds in quality significantly. The only drawback that is seen in this model is that it is not suitable when the requirements are not fully documented and is not applicable to all aspects of development and testing.

1.3.4 Rapid Application Development Model

Rapid Application Development (RAD) is a software development model which is created from the business requirements, project management requirements and software requirements specifications (SRS). In this model, a prototype is created and matched against the requirements. If there is a gap, there is another model created and prototype developed. This model follows a linear sequential software development process, where an extremely short development cycle is adopted and a re-usable component is used for development. When the requirements are well understood and defined, the RAD process enables the development team to develop the final product in a shorter period.

The RAD model consists of the following four phases:

1. ***Requirements Planning:*** During this phase, the project requirements are gathered and the project outline is planned.
2. ***Design Phase:*** The RAD team prepares a design or model (prototype) of the system required. While developing the design or model, all the requirements as well as the changes in the previous phase are listed.
3. ***Construction Phase:*** This is the phase where the various RAD tools are used to develop the first prototype of the model. The prototype developed is based on the design phase of RAD model. The prototype is checked to make sure that it meets the customer requirements. In case, the customer is not satisfied, then changes are made in the model and one more prototype is built. This modification process is carried out until the customer is completely satisfied with the product.
4. ***Testing and Handover Phase:*** The testing and implementation phase relies heavily on the re-use of software components. Since the software components would have been already tested while being developed for other projects, the time spent on testing the software is very less. Once the testing has been completed the software product can be implemented and used by the customer.

Notes

Disadvantages of RAD model

1. It has to be made sure that no reusable component is missing, since it can lead to failure of the entire project.
2. This method is suitable for small projects only and cannot be applied for complex projects.

1.3.5 Agile Model

The Agile software development model is the most popular model used today. The traditional software development models follow either sequential waterfall model or iterative spiral model. Such software models cannot be efficiently adapted to complex software development projects that have continuous and multiple changes. Therefore, the Agile software development model was developed, which responds to changes quickly and smoothly. The control mechanism used in Agile is the feedback from people. The feedback helps to break work into small pieces which is tested and further refined with user input.

Did you Know? The disadvantages that the developers faced in sequential models were overcome by iterative methodologies. However, iterative methodologies still follow traditional waterfall approach.

The Agile development model follows incremental method of software development rather than sequential, i.e., software is developed in incremental rapid cycles which results in small incremental releases. Every release is built on the functionalities of the previous release. Each release is thoroughly tested to ensure that all the bugs are detected and resolved before the product is released.

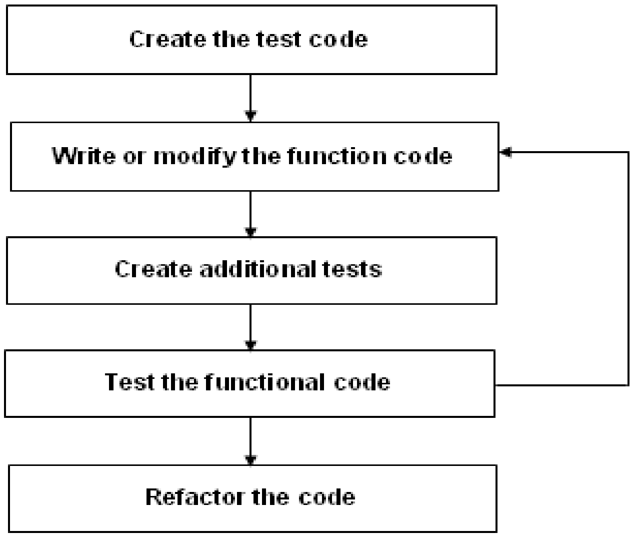
Another important aspect of the Agile model is that customers, developers, and testers constantly interact with each other during the entire development process. The tester is aware of the requirements being developed and can easily find out any gaps that exist between the developed software and requirements. This is carried out for every release that is made during the software development life cycle.

Many approaches are used to achieve agile methodology, such as Dynamic Systems Development Method (DSDM), SCRUM, and Extreme Programming (XP). Among all these approaches, Extreme Programming (XP) is the most popular and widely used approach.

In XP software development life cycle, programmers usually work in pairs. One programmer takes the responsibility of writing the code for the software and the other programmer reviews the code to ensure that it uses simple solutions and follows the best design principles and coding practices. This means the second programmer acts as a tester and tests the software as and when it is developed to find the bugs.

Test-driven development is one of the core practices of XP. It adopts feedback approach of software development where test cases are developed before the actual code is developed. The figure illustrates the various phases of test-driven development.

Figure 1.4: Agile Model



1. ***Create the Test Code:*** The developer creates the test code using an automated test framework even before the actual code is written and these test codes are submitted to the test team. The functionality of the software is developed based on the test code.
2. ***Write or Modify the Function Code:*** The functional code is written for the test codes that have cleared the test case. The actual functional code is not written until the test case requirements are met with. Once the functional code is written it is again checked using the test case. The functional code module is completed only after it clears the test cases.
3. ***Create Additional Tests:*** In this step the tester tests the module for various types of input. The tester develops various test conditions depending on the complexity of the module.
4. ***Test Functional Code:*** The functional code is tested based on the test case developed in step 3 and step 1. The steps 2 to 4 are repeated until the code clears all the test cases.
5. ***Refactor the Code:*** In this step, some changes to the code are made to make the code easy to maintain and extensible. This enables the developer to make changes to a particular module without affecting the entire application. Any new feature can be added to the existing application easily, without major modifications. It also removes any duplicate code or unused code and reduces the code complexity.

Thus, in Agile method testers have a strong role to play in development of efficient software.

Case Study

Error in Intel's Pentium Microprocessor

A Mathematics professor at Lynchburg College, USA, Dr.Thomas Nicely, found a floating point division error in Intel Pentium microprocessor in the year 1994. He was computing on his Pentium-based computer to find the sum of the reciprocals of prime numbers.

He noticed that the computation result was significantly different from the theoretical values. When the same computation was carried out on another computer with a different microprocessor (486 CPU) he was able to get the correct values that obey the theoretical values. The worst and well-known case was division of 4195835 by 3145727. The correct value is 1.33382, however the Pentium's floating point unit computed and generated a value 1.33374 (Only 6 places after decimal is mentioned here), which has an error of 0.006.

Dr.Thomas Nicely reported the bug to Intel, but Intel ignored it and Nicley did not receive any proper response from Intel. Later, Nicley took the issue to public with the help of the Internet and media. Following these events, Intel publicly announced that "an error is only likely to occur [about] once in nine billion random floating points". They also mentioned in their announcement that "an average spreadsheet user could encounter this subtle flaw once in every 27,000 years of use." However, it was noted that the Intel Pentium processors output was wrong every time when such division was performed for such values.

On November 28, 1994, Intel publicly admitted the problem and issued a statement saying that it would replace pentium chips only for those who could explain the need of high accuracy in complex calculations. Industry experts, public, and media criticized this attitude of Intel. Later, in the month of December, Intel announced that it would freely replace the processor for any owner who asked for one.

The bug problem made Intel to issue an apology to its customers and the public for the way it handled the bug and issues related to it. It had to spend more than $400 million for replacing bad chips. Learning a lesson from this, Intel now reports all known problems on its official website and it also monitors customer feedbacks carefully and regularly.

Questions

1. Is Intel right in their approach to customers? How did they win customers' confidence?
2. Explain the kind of bug faced by Intel. What method would you suggest to overcome the bug?

Adapted from (http://www.willamette.edu/~mjaneba/pentprob.html)

1.4 Summary

- Software testing demonstrates that a program performs the intended functions correctly. The ultimate goal of software testing is to ensure that the product is error free.
- Detecting bugs are the most important part of a software testing process. These bugs can be an error in the program or issues that affect the quality of the software.
- Depending on the kind of error or the reason for error the bugs are classified into various types such as, bugs due to conceptual error, math bugs, logical bugs, resource bugs, co-programming bugs, and team working bugs.
- Bugs in software can occur due to various reasons such as human error, lack of communication, tight time lines, improper design logic, inefficient coding practices, and unskilled testers.
- Bugs can prove to be very costly. They not only take time to resolve affecting the project time lines and project cost, but also result in losses as the company may have to compensate the customer by

way of money or by way of replacing the defective product. It can also damage the reputation of the company.

- The Waterfall model of software development is a traditional model which follows sequential method of software development. V and Spiral models, which employ testing as an integral part of software development, are more efficient than the Waterfall model. Agile is considered to be the most advanced and efficient type of software development model. Extreme Programming (XP) is a method developed based on the agile model, which uses test driven development to develop highly efficient software.

1.5 Keywords

Confined: Within bounds or limits

IEEE: IEEE is the acronym of Institute of Electrical and Electronics Engineers. It is the world's largest professional association dedicated to advancing technological innovation and excellence for the benefit of humanity. They foster development of national and international standards.

Race Condition: A race condition occurs when a program doesn't work as it is supposed to, because of an unexpected call or ordering of events that produce contention or a clamor over the same resource.

Recursion: Recursion is a process of defining or expressing a function or procedure in terms of itself.

1.6. Self Assessment

1. State whether the following statements are true or false:
   (a) Software testing identifies the areas of weakness in an application or product.
   (b) The year 1988-2000 followed prevention oriented approach.
   (c) Testing is a process of executing a program with the intent of finding errors -- was pointed out by Myers.
   (d) Some compromises will be made in the requirements or design of the software to meet the delivery requirements.
   (e) Sometimes efficient tools such as editors, compilers, and debuggers generate wrong codes which cause errors in the software.
   (f) The analysis phase defines the software and the hardware requirements relevant for the software development process.
2. Fill in the blanks:
   (a) Software testing is an _______________________ and _______________________ process to detect a mismatch, a defect or an error.
   (b) Many bugs occur due to lack of _______________________ when a developer tries to modify software developed by another developer.
   (c) The _______________________ of fixing a bug differs depending on the development stage at which it is detected.
   (d) In Test Driven Development (Agile), the _______________________ is not written until the test code does not clear the test case requirements test.
3. Multiple Choice Questions
   (a) Demonstration oriented testing was followed during which of the following period?
      (i) 1957-1978 (ii) 1979-1982 (iii) 1983-1987 (iv) 1988-2000
   (b) Which of the following factors causes errors due to incorrect usage of logic or syntax?
      (i) Communication failure (ii) Human (iii) Lack of skilled testing (iv) Unrealistic timeframe

7. "Automated testing uses testing tools to perform the test." Does this mean that there is no human intervention needed for automated testing?
8. Is quality assurance and quality control mutually exclusive quality initiatives? Which activity is closer to testing?
9. "Software testing is a risk-based exercise." Explain.

2.9 Further Readings

Books

Ron Patton, Software Testing-Second Edition, SAMS Publishing, USA
Hutcheson, Marnie L, Software Testing Fundamentals, Wiley Publishing, USA
Kassem A. Saleh, Software Engineering, J.Ross Publishing, 2009, US

Online link

http://qastation.wordpress.com/2008/04/21/static-testing-vs-dynamic-testing/
http://www.adager.com/vesoft/automatedtesting.html
http://www.scribd.com/doc/2453259/Testing-Techniques-and-Strategies
http://www.testinggeek.com/index.php/testing-articles/137-equivalence-partitioning-introduction
http://www.cc.gatech.edu/classes/cs3302_98_summer/7-02-unittest/sld009.htm
http://www.slideshare.net/nworah/types-of-software-testing
http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/
http://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/

Unit 3: Black Box Testing

CONTENTS

Objectives

After studying this unit, you will be able to:

- Illustrate structural and functional testing strategies and techniques
- Explain static black box testing techniques
- Explain dynamic black box testing techniques
- Discuss test to pass and test to fail
- Explain equivalence partitioning, data testing, and state testing
- Explain random testing and mutation testing

Introduction

We are aware that the Testing Technique specifies a strategy that is used in testing select input test cases and analyze test results. There are various testing aspects that are revealed through the Structural and Functional Testing. When the features and operational behavior of the product needs to be tested, Functional Testing or Black Box Testing can be approached. The advantage of this kind of testing is that they totally ignore the internal workings of the system.

Organizations have to make the right choice between Structural and Functional testing. With increasingly complex applications, Total Cost of Ownership (TCO) and Return on Investment (ROI) are two criteria that favor the Black Box testing technique. However, if the strength of the application needs to be introspected, or if the application has to be checked for stability or needs to be ascertained for thoroughness, it would have to undergo white box testing.

3.1 Structural and Functional Testing

Structural and functional testing are two important types of software testing. Structural and functional testing are also called as white box and black box testing.

3.1.1 Black Box Testing

Black box testing, also termed as behavioral testing, checks if the software works as per the desired requirements or specifications. It is called black box testing because the tester performs the tests without knowing the internal logic of how exactly the software works. He/she focuses on the outputs generated in response to selected inputs and execution conditions.

Did you know? In neural networking or Artificial Intelligence simulation, a black box is used to describe the constantly changing section of the program environment which a programmer cannot test easily.

Developing efficient test cases is very essential during black box testing. Since the tester has no knowledge of the internal working of the software, they need to rely completely on the analysis of the transformation of the inputs to the outputs based on which they find bugs in the software. This test enables the tester to know whether or not the software does what it is supposed to do. The functional specifications or requirements of the software provide the information about the software functionalities.

Example: Testing search engine is a good example for black box testing. You are not aware of the processes that work behind the search engine to provide the desired information. While testing a search engine you provide input in the form of words or characters, and check for output parameters such as relevance of the search result, time taken to perform the search or the order of listing the search result.

Advantages of Black Box Testing

Black box testing has many advantages, which include the following:

1. Testers do not have to understand the internal working of the software, and it is easy to create test cases with the perspective of an end user.
2. The testers mainly deal with the Graphic User Interfaces (GUI) for output, and they do not spend time analyzing the internal interfaces. Therefore, test cases can be developed quickly and easily.
3. As soon as the specification of the product is complete, the test cases can be designed.
4. Black box testing helps to expose any ambiguities or inconsistencies in the specifications, and tests are carried out from a user's perspective.

Example: Testing the functions of an ATM is a good example of black box testing. The tester acts as a customer who is using the ATM and checks the functions of the machine. He/She does not know the internal working of the logic. The test cases are developed to check the functions through the GUI of the ATM such as change in display of the GUI when card is detected, masking the password or navigating from main menu to a specific function.

Disadvantages of Black Box Testing

1. A tester can test only a small number of possible inputs and it is highly impossible to test every possible input stream.
2. It is very difficult to design test cases if specifications are not clear and concise.
3. Situations, such as unnecessary repetition of test inputs, can occur if the tester is not informed of test cases that the programmer has already tested.

4. This type of testing cannot be focused on specific segments of function that may be very complex, therefore bugs can go undetected.

Example: When there is a complex system to be tested like the online Indian railways booking system, it is difficult to identify tricky inputs and write test cases to cover all possible scenarios. .

Notes **Performing a black box test, the tester attempts to find the following errors based on the behavior of the software:**

Missing or incorrect functionality.

1. Errors in interface.
2. Data structure errors.
3. Performance errors.
4. Initialization and terminal errors.

Caution As a part of black box testing strategy, it is very important to use test monitoring tools. This is needed to track the tests that have already been executed, to avoid repetition and to aid in the software maintenance.

3.1.2 White Box Testing

White box testing is also called as glass box testing. In this test, the tester focuses on the structure of the software code. The tester develops test cases to check the logical working of the software code.

Black box testing helps to answer the validation question "are we building the right software?", but white box testing helps to answer the verification question "are we building the software right?"

Example: White box testing helps the tester to check how exactly the calculator performs the addition of two numbers and if it is the efficient way to perform the addition operation.

In white box testing, each software module is tested independently. The tester has to develop test cases not only to test the individual module of the software, but also to test how exactly the modules interact with each other when software is executed. All the tests are carried out at the source code level. The tester checks all the parameters of the code such as efficiency of the code written, branching statements, internal logic of the module, interfaces between external hardware and internal module, memory organization, code clarity, and so on. Therefore, the test cases must be carefully designed in order to cover the internal working of the application.

Caution The tester who writes the test cases to perform white box testing has to be very well aware of the language and logic used to develop the test software. He/she needs to know programming concepts as well.

Advantages of White Box Testing

1. As the tester has the knowledge of internal coding, it is very easy to develop test cases to test the software effectively.
2. Testing is carried out at the code level; hence it helps in optimizing the code.
3. Unnecessary or extra lines of code which can generate hidden bugs can be removed.

3.3 Summary

- Structural testing techniques check the occurrence of bugs in the test software using the actual codes of the software. The tester works with the source code of the software while performing the test.
- In a functional test, the tester is not aware of the actual working of the software. The test analysis is performed based on the outputs that the software generates for various inputs. The bugs are detected by comparing the expected output with the obtained output.
- Black box testing is a functional testing technique. The tester performs the test to check the behavior of the software by providing pre-defined inputs and analyzing the outputs.
- A tester performing white box testing knows the actual code level working of the software. The test cases target to find the bugs associated with the code's logic, structure, module interface and memory organization.
- In order to make the software bug free, a certain level of both black and white box testing has to be performed on software.
- A static black box testing involves checking for bugs in the specification document. Any mistakes or incorrect information present in the specification is considered as a bug.
- Dynamic black box testing refers to testing for bugs by executing the software.
- Test to pass and test to fail are dynamic black box testing techniques. Test to pass involves providing normal inputs to the software to check whether it works without any bugs. During test to fail, the tester provides erratic inputs to check the software.
- Equivalence partitioning involves grouping similar test cases and performing the test where, a test case from each class is used to perform the test.
- Dynamic black box testing techniques like the data testing is carried out to check for occurrence of bugs in the input data provided to the software. State testing focuses on the transitions of internal software state.
- Other testing techniques such as random and mutation testing are some of the popular dynamic testing techniques to perform efficient software testing.

3.4 Keywords

Encapsulation: A technique where the internal representation of an object is generally hidden from view outside of the object's definition.

Neural Networking or Artificial Intelligence Simulation: This is a branch of computer science where intelligent machines are created through software programs that simulate or reproduce the creative functions of the human brain.

Source Code: It refers to a collection of statements or declarations that are written in a computer programming language. Source code needs the compiler or interpreter to translate the file into the object code before execution.

State Diagram: It is an illustration of the states an object can attain as well as the transitions between those states.

3.5 Self Assessment

1. State whether the following statements are true or false:
   (a) The testing strategies and techniques are developed to address a particular type of need or to test certain required parameters of software.
   (b) Static Black box testing consists of viewing the specification at the high and low level.
   (c) The testing team must strike a balance depending on the project requirement to adopt both black and white box testing to test the software.

(d) While performing software test the tester should first begin the test with test to fail and check whether the software works fine without any bugs.

(e) The tester enters erratic or irrelevant data and checks how the response of the software while performing mutation testing.

2. Fill up the blanks:

(a) During functional testing the tester checks only the ________________ of the software and will not check the actual code.

(b) Static black box testing is performed to check the specification using ___________ and ___________ techniques.

(c) Developing efficient_____________ is very essential during testing.

(d) _____________ testing is used for high level black box testing.

(e) The main objective of ________________ is to identify the test cases that perform same kind of testing and similar output.

3. Multiple Choice Questions

(a) Identify the testing technique that is used to test how the actual code works.

(i) Structural testing

(ii) Functional testing

(iii) Static testing

(iv) Black box testing

(b) Which of the following testing will help to expose any ambiguities or inconsistencies in the specifications and are carried out from a user's perspective?

(i) White box testing

(ii) Automation testing

(iii) Manual testing

(iv) Black box testing

(c) Which dynamic testing technique's main focus is to push the software to its limit and check the bugs that occur when the software is operated under extreme conditions?

(i) Test to pass

(ii) Test to fail

(iii) Data testing

(iv) State testing

(d) Which testing is also called as Adhoc testing?

(i) Equivalence Partitioning

(ii) Data testing

(iii) Random testing

(iv) Test to pass

Answers: Self Assessment

1. (a) True (b) True (c) True
   (d) False (e) False
2. (a) Behavior (b) High Level and Low Level (c) Test cases
   (d) State based (e) Partitioning
3. (a) Structural testing (b) Black box testing (c) Test to fail (d) Random testing

3.6 Review Questions

1. Do you agree with the fact that the logical flow of the software in its different forms (states) can be tested? If so, which type of testing will you apply?
2. "Static black box testing is more research oriented and the research helps to understand how the specification is organized and the reason behind the organization of the specification." Justify that high level and low level static black box testing improves quality.
3. What makes you think that Test to pass is different from Test to fail? Explain.
4. "Selecting the right testing method for testing the software is very important, since both black and white box testing methodologies have their merits". Could you list the merits of both black box and white box testing techniques?
5. Do you believe that there is a difference between boundary condition and sub-boundary condition? Explain.
6. Do you agree that Equivalence partitioning reduces the number of test cases without compromising the quality of the test being carried out? Explain
7. "Developing efficient test cases is very essential during black box testing." Why do you think so?

3.7 Further Readings

Books

Ron Patton, Software Testing-Second Edition, SAMS Publishing, USA
Hutcheson, Marnie L, Software Testing Fundamentals, Wiley Publishing, USA
Kassem A. Saleh, Software Engineering, J.Ross Publishing, 2009, US

Online link

http://qastation.wordpress.com/2008/04/21/static-testing-vs-dynamic-testing/
http://www.adager.com/vesoft/automatedtesting.html
http://www.scribd.com/doc/2453259/Testing-Techniques-and-Strategies
http://www.testinggeek.com/index.php/testing-articles/137-equivalence-partitioning-introduction
http://www.cc.gatech.edu/classes/cs3302_98_summer/7-02-unittest/sld009.htm
http://www.slideshare.net/nworah/types-of-software-testing
http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/
http://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/

Unit 4: White Box Testing

CONTENTS

Objectives

After studying this unit, you will be able to:

- Explain static white box testing
- Explain dynamic white box testing

Introduction

The IEEE definition of software lists four components which are needed in order to assure the quality of software applications: computer programs or the code which is the brain behind any application; procedures that define the flow of the program, its methods and the way of functioning; documentation needed for developers and users; the data that includes parameters. The computer programs or the source code of the software is an important artifact that needs to be tested for ensuring quality of the software product. The testing that encompasses the verification of the computer programs and the logic of the application is known as White Box testing.

IEEE defines White box testing as "The testing that takes into account the internal mechanism of a system or component". White box testing takes care of the intricacies of the product and evaluates it for accuracy and precision, to meet the requirement specifications.

White box testing verifies the designs and codes involved in the development of a software product. It involves validating whether the code has been implemented as per design specifications and also validating the security concerns of the software's functionality. Thus skilled testers with knowledge of programming are required to conduct white box testing.

Determine the length of a supplied string.	strlen wcslen	RtlStringCbLength RtlStringCchLength
Create a formatted text string that is based on a format string and a set of additional function arguments.	sprintf swprintf _snprintf _snwprintf	RtlStringCbPrintf RtlStringCbPrintfEx RtlStringCchPrintf RtlStringCchPrintfEx
Create a formatted text string that is based on a format string and one additional function argument.	vsprintf vswprintf _vsnprintf _vsnwprintf	RtlStringCbVPrintf RtlStringCbVPrintfEx RtlStringCchVPrintf RtlStringCchVPrintfEx

Source: http://www.osronline.com/ddkx/kmarch/other_9bqf.htm

Notes It is the responsibility of the programmer to use the inbuilt string function of C or C++ languages properly to overcome any overruns problem. Proper logic and correct usage of strings can overcome the overrun problem.

7.2.4 Computer Forensics

Computer forensics is also called as cyber forensics. This is a technique of computer investigation and analysis that is used to gather substantial evidence against a cyber crime for presenting it in a court of law. The main aim of computer forensics is to conduct a structured investigation of a cyber crime to find out what happened and who was responsible for it. This is done to protect the security of software.

Computer forensics deals with identifying and solving crimes that are carried out by using computer technology. The governments across the globe have imposed many laws to check cyber crimes. However, lack of evidence has made it difficult to prosecute the people responsible for the crimes. Computer forensics helps to overcome such difficulties. It helps to gather evidence to take legal actions against those who carry out such crimes.

Did you know? International Data Corporation (IDC), in the year 2005, reported that "the market for intrusion-detection and vulnerability-assessment software will reach 1.45 billion dollars in 2006" (US-CERT, 2005).

Notes **Some of the major reasons for criminal activities in computer are:**

Unauthorized use of username and password.

1. Accessing other users' computer via the internet.
2. Releasing virus to other computers.
3. Harassment and stalking in cyberspace.
4. E-mail Fraud.
5. Theft of company documents.

The tester should check for security vulnerability issues related to test software from a computer forensic perspective. Sometimes, hackers do not really need to break into your system to steal the data, since some data can be easily accessed. If the hacker knows where exactly to look for a particular data then he/she can easily get the data from the software.

Example: When you download any file or picture from the internet-- by default, all the files and pictures are saved in "Temporary Internet Files" folder on Windows operating system. In case you have accessed any confidential information on the Internet, the same would also be saved in the "Temporary Internet Files", and a hacker can easily view and use this file.

Any unprotected data that is available for other users is called latent data. It is the responsibility of the tester to assess whether or not any such latent data can cause security vulnerability. If yes, then necessary measures have to be taken to prevent it from occurring.

Notes A tester who is very well aware of the security vulnerabilities of the system can provide vital information to computer forensics to know exactly how the software security could be breached. This helps to find out the possible ways an attacker could have carried out the attack.

Case Study

National Widgets Website Security Problem

National widgets wanted to build a Web site for its users, and it approached Front End Associates to develop a Web site for them. Front End Associates developed a highly impressive Web site for National widgets.

National widgets deployed the Web site developed by Front End Associates and for about 18 months the Web site operated without any problem. However, some of the employees of National Widgets raised security concerns in the Web site. National Widgets brought the issue to Front End Associates' notice and asked them to fix the problem for free. However, the Front End Associates were not ready to fix the problem for free and did not respond to National widgets' request properly.

Later, Front End replied to National Widgets saying that there were no security problems in the software that they had developed. It also justified its stand by saying that they had hired Web site security testing experts to carry out security tests on the software and provided a detailed report. National widgets decided to verify the test reports that the team of testing experts had prepared after testing the Web site. National widgets had to take the support of its lawyers to view the test report which was with the Front End. After analyzing the report it was noticed that they had not conducted an effective testing of the software. The company that Front End hired to perform security testing had simply run a few scanning tools to check for minor issues in the software. They did not perform an effective testing to find security vulnerabilities in the software.

National widgets planned to conduct an independent testing of the software and assembled a group of experts to carry out the test. For this, National widgets asked Front End to provide the source code of the Web site that Front End had developed for them. However, Front End refused to give the source code to National Widgets, saying it will cause copyright issue. National widgets, with the help of law, was able to decompile the code and perform a code review. After a thorough testing of the software, more than six serious problems related to security were found. These problems were due to poor design that Front End had adopted during the initial stages of Web page development. The problems were so severe that it required both time and cost to make necessary changes. Therefore, National widgets raised a request to Front End to fix the problem without any additional cost. However, Front End partially acknowledged the defects in the Web page, but was not ready to accept the mistake completely. This became a legal issue and both the companies went to court to solve the dispute.

This problem had a direct impact on both the companies. The companies lost lakhs of rupees by paying legal fee, productivity was hindered, and reputation was damaged. Along with this, both the companies had to spend huge time and money for answering each other's queries, producing documents, re-testing of the software, and trial. Even though Front End was the most affected in terms of loss of revenue and reputation, National widgets also saw some setbacks due to this issue.

Questions

1. What was the problem that National faced and what was the reason behind it?
2. Do you think Front End was responsible for developing such a Website? Justify.

Adapted from
http://www.owasp.org/index.php/Secure_software_contracting_hypothetical_case_study#Conclusions

7.3 Summary

- Product documentation provides users the information about the product specifications. This helps the users to know about the product and its features. Thus, it enables the customers to use the product easily.
- The tester performs documentation testing to check for any errors in the document. Since documentation errors will not only convey incorrect or wrong information to the users it will also bring down the reputation of the company.
- The various software components of documentation are Packaging Text and Graphics, Marketing Material, Ads and Other Inserts, Warranty/Registration, End User License Agreement, Labels and Stickers, Installation and Setup instructions, User's Manual, Online Help, Tutorials, Wizards, and Computer Based Training (CBT), Samples, Examples, and Templates, and Error Messages.
- Documentation testing helps to improve the usability and reliability of a software product. It also helps the organization to reduce the product support cost.
- Security testing is the most important aspect of software testing. This enables the tester to find the system's vulnerability to security risks.
- Security threat modeling helps to analyze the system in a structured way, so as to find the threats that the system faces with respect to security. This model not only detects the threats, but it also documents the threats found and rates them based on the severity of the threat.
- Buffer overrun is one of the most popular bugs that the hackers use to attack the system. It is a major security threat for any software product.
- The usages of safe string function have enabled the developers to overcome the problem of buffer overrun. The tester has to make sure that the developers use these functions to develop their programs.
- Testers must test the software for any latent data available in it, since this data can cause issues related to software security.

7.4 Keywords

Crypto System: Any computer system that involves cryptography is called as crypto system. Cryptography is an art of studying hidden, coded, or encrypted information.

Unicode: Binary codes that are used to represent text or script characters in computer programming languages.

Virus: A computer program that can copy itself and infect a computer.

Vulnerability: Susceptibility to attack.

Warranty: A written assurance that some product or service will be provided or will meet certain specifications.

7.5 Self Assessment

1. State whether the following statements are true or false.

   (a) Documentation meets its objective only if it provides necessary and complete information to the end users or customers.

   (b) The details of the license will sometimes be printed on the envelope or package of software CD.

   (c) Today, many organizations provide the entire information about a product using printed manuals.

   (d) Threat modeling is a highly structured and organized approach of threat correction.

(e) The tester has the knowledge of the entire system architecture and potential vulnerabilities of the system.

(f) The main aim of computer forensics is to conduct a structured investigation of a cyber crime to find out what happened and who was responsible for it.

2. Fill in the blanks

(a) __________________ material creates interest in the customer or end user to buy the product.

(b) In many software products, __________________ is done when the user tries to install the software.

(c) The software displays the __________________ when it encounters unusual or exceptional events.

(d) ______________ make use of weak codes in the software to carry out an attack on the software.

(e) The tester will use a common __________________ to record all the threats that he/she has detected in the system.

(f) The __________________ perform extra processing of the input data for proper handling of buffers in the software.

3. Select a suitable choice for every question:

(a) Identify which among the following is not documentation.

(i) Labels and stickers (ii) Tutorials and wizards

(iii) End User License Agreement (iv) User feedback report

(b) What is a legal document?

(i) Warranty (ii) End User License Agreement

(iii) Registration form (iv) Error messages

(c) What is called as short version of a user manual?

(i) Tutorials (ii) Wizards (iii) Online help (iv) Installation guide

(d) What is the most important aspect of software security?

(i) Cost (ii) Time (iii) Information (iv) Quality

(e) Which is the step that follows soon after identifying the threats in software threat modeling?

(i) Identify assets (ii) Decompose the application

(iii) Rate the threats (iv) Document the threats

7.6 Review Questions

1. Do you believe that documentation is a window that provides user a complete view of the product?
2. Documentation testing is a crucial element of any software testing process. Justify
3. Do you think software components can be called as documentation? If yes, explain with examples.
4. “Software security testing tests the software behavior when the software is attacked by some external element.” What do you consider as external element and how would you ensure testing the same?

4. ***Stress:*** Stress testing is a process in which the software is tested for its effectiveness in providing steady or satisfactory performance under extreme and adverse conditions. These may include heavy network traffic, heavy process load, under or over clocking of underlying hardware, and working under maximum requests for resource utilization of the peripheral or in the system. Stress testing helps to estimate the level of robustness and reliability, even when the limits for normal operation of the system are crossed. Stress testing is considered vital with respect to software that operates in critical or real time situation.

Example: Consider a browser window. Users can open multiple browser windows to navigate between different pages at the same time. However, all these windows are dependent upon one another, and in case one browser crashes, all of them crash. Stress testing is carried out in this scenario to test the browser.

Load2Test is a stress, performance, and load testing tool from Enteros Inc.

5. ***Performance:*** Performance testing is a testing process used to determine the speed or effectiveness of software. This process involves performing quantitative tests in a laboratory such as measuring the response time, the number of instructions executed per second at which a system functions. The qualitative attributes like reliability, scalability, and interoperability are also assessed. Performance testing is often conducted with stress testing. Performance testing is used to confirm that a system meets the specifications stated by its manufacturer. Performance testing can compare two or more devices or programs in terms of factors like speed, data transfer rate, bandwidth, efficiency, or reliability. Performance testing can also be used as an analytical aid in locating communication blocks. It has been noticed that often a system works in a better way if a problem is resolved at a single point or in a single component.

Example: Even the fastest computer would not function properly on today's Web if the connection occurs at only 40 to 50 Kbps (kilobits per second).

6. ***Load:*** Load testing is a process of subjecting a computer, peripheral, server, network, or application to a work level that is approaching the limits of its specifications. The load testing process can be performed under controlled laboratory conditions to compare the capabilities of different systems or to precisely measure the capabilities of a single system. Load testing can also be performed in a field to obtain a qualitative idea of the performance of a system in real world. Load testing is considered as a part of a more general process known as performance testing.

Example: Few examples of load testing are:

(a) Transferring a number of tasks to a printer at a time.

(b) Loading a server with a large amount of e-mail traffic.

(c) Operating multiple applications on a computer server.

The above mentioned testing types can be automated.

In nearly all software development models, the code-test-fix loop gets repeated many times before the software is released. If the tests are carried out for a particular feature, they are to be performed several times. The tests are performed to check if the bugs found previously are fixed and that no new bugs are introduced.

Example: Hewlett Packard offers a suite of solutions called HP Quality center which automates testing and quality assurance for client/server software and systems. They can be used for e-business applications and enterprise resource planning applications. The following are the different products which are part of the suite.

Quality center – This product can be used for tests and defect tracking.

Load Runner – This product helps in enterprise load testing,

WinRunner – This product helps in test automation for the enterprise.

Example: If a software project has many test cases to run, there may be hardly enough time to execute them just once. Running them several times might be impossible. In such a situation, software test tools and automation can help solve the problem by providing a more efficient means to run the tests, when compared to manual testing.

Did you know? Automated testing involves higher upfront costs and should be looked at as a long-term investment, where the pay-offs come anywhere between 2-4 years down the road.

Task Read on how ROI on Test automation is carried out from the following link

www.keane.com/resources%2Fpdf%2FWhitePapers%2FWP_ROIforTestAutomation.pdf.

9.1.1 Test Tools

While performing automation software testing, a tester comes across various testing tools. The types to be used are based on the type of software that a tester is testing.

The advantage of these test tools is that the tester need not be an expert on how they work or what exactly they do in order to use the test tools.

Example: Consider a tester testing networking software that permits a computer to communicate with up to 1 million other computers at the same time. It can be difficult to carry out a controlled test with 1 million real connections. But, with a special tool that replicates those connections, the tester can adjust the number from one to a million and perform the tests without having to set up a real-world scenario.

It is not necessary to understand how the tool works. The only requirement to test a particular application would be the knowledge of white box skills and awareness of the low level protocol to effectively use this tool.

In this section, we will discuss the major classes and uses of testing tools. Some instances are based on tools included with most programming languages, whereas others are commercial tools that are sold individually. In many cases it is seen that a new tool has to be developed to satisfy the tester's needs.

Viewers and Monitors

A viewer or monitor test tool allows the tester to view details of the software's operation, which otherwise would not have been possible.

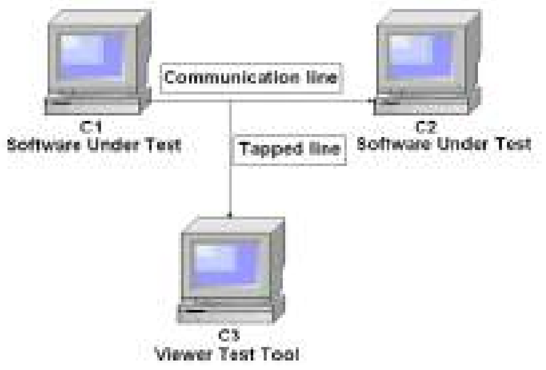
Example: An example of the viewing tool is the coverage analyzer tool.

Most code coverage analyzers are intrusive tools since they need to be compiled and linked into the program to access the information they provide.

A communications analyzer is another type of viewer. This tool enables the tester to see the raw protocol data moving across a network or other communications cable. It taps into the line, pulls off the data as it passes by, and displays it on another computer.

Figure 9.1 illustrates the working of a communication analyzer.

Figure 9.1: A Communications Analyzer



Here, a user can enter a test case on C1, confirm that the resulting communications data is correct on C3, and then check whether the appropriate results occurred on C2. A user can also use this system to investigate why a bug occurs. By looking at the data pulled off the wire, the user can determine whether the problem occurred in creating the data (C1) or interpreting the data (C2).

The code debuggers are also known as viewers since they allow programmers or white-box testers to view internal variable values and program states.

A viewer test tool is classified as a tool which enables a user to look at data that is not visible to an average user.

Drivers

Drivers are tools used to control and operate the software being tested.

Example: A batch file or a simple list of programs or commands executed sequentially, is an example of a driver tool.

In the earlier days, MS-DOS was a popular means for testers to execute their test programs. The testers would create a batch file containing the name of the test program and would start running the batch of programs. But with the upcoming new operating systems and programming languages, there are many sophisticated methods of executing test programs.

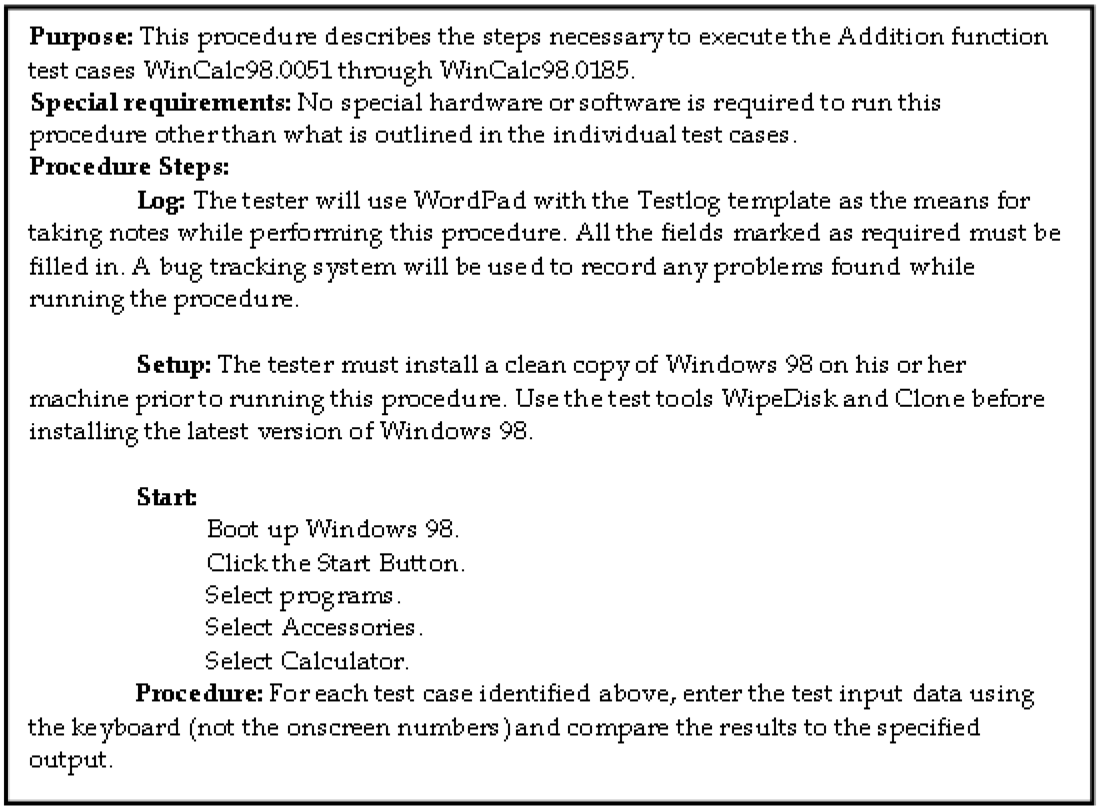
Example: Java or a Perl script can replace an old MS-DOS batch file, and the Windows Task Scheduler can be used to execute various test programs during any time of the day.

13. ***Wrap Up:*** Tells how to restore the environment to its pre-test condition.
14. ***Contingencies:*** Tells what to do if things do not go as planned.

It is not enough for a test procedure to direct a tester to try all the test cases and get back with the observations. The test procedure should instruct a new tester on the way to perform the testing. A detailed procedure helps a tester know what exactly needs to be tested and how.

Example: Figure 11.3 shows an excerpt from a fictional example of a test procedure for Windows Calculator.

Figure 11.3 Sample Test Procedure



Purpose: This procedure describes the steps necessary to execute the Addition function test cases WinCalc98.0051 through WinCalc98.0185.
Special requirements: No special hardware or software is required to run this procedure other than what is outlined in the individual test cases.
Procedure Steps:

Log: The tester will use WordPad with the Testlog template as the means for taking notes while performing this procedure. All the fields marked as required must be filled in. A bug tracking system will be used to record any problems found while running the procedure.

Setup: The tester must install a clean copy of Windows 98 on his or her machine prior to running this procedure. Use the test tools WipeDisk and Clone before installing the latest version of Windows 98.

Start:
Boot up Windows 98.
Click the Start Button.
Select programs.
Select Accessories.
Select Calculator.

Procedure: For each test case identified above, enter the test input data using the keyboard (not the onscreen numbers) and compare the results to the specified output.

Source: Ron Patton, Software Testing, Second Edition, Sams Publishing

11.1.5 Test Organization and Tracking

An important consideration that needs to be taken into account when creating the test case documentation is the organization and tracking of information. Some questions that a tester or the test team should be able to answer may include:

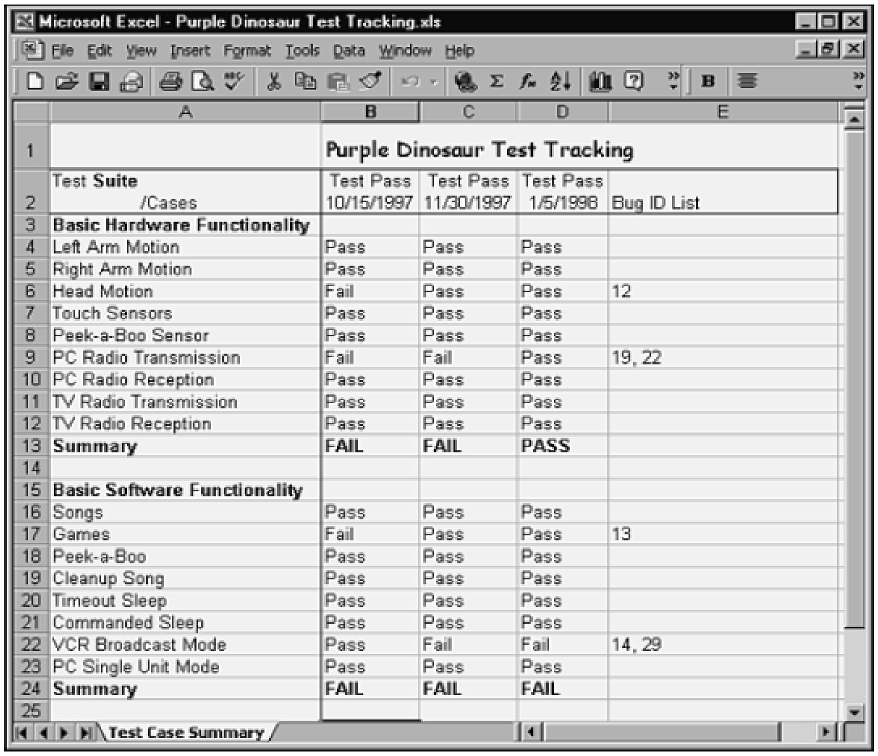
1. Which are the test cases you plan to run?
2. What is the number of test cases you plan to run? What will be the time taken to run them?
3. Can you decide on choosing the test suites (groups of related test cases) to run particular features of the software?
4. When the test cases are run, will you be able to record pass and fail result of each test case?
5. Of the cases that failed, which ones failed the last time too?
6. What is the percentage of the test cases that passed?

These types of questions might be asked over the course of a typical project. However, initially some sort of a process needs to be in place that allows you to manage your test cases and track the results of running them. There are basically four types of systems. They are:

1. ***In Your Head:*** You must not consider this, even for the simplest projects, unless you are testing software for your own personal use and you do not want to track your testing.
2. ***Paper/Documents:*** It is possible to develop test cases on paper for very minor projects. Tables and charts of checklists have proved to be effective. The advantage of maintaining a written document is that if a written checklist includes a tester's initials or signature indicating that the tests were run, it serves as an excellent proof in the court-of-law that testing was performed.
3. ***Spreadsheet:*** A preferred and practical method of tracking test cases is by using a spreadsheet. A spreadsheet can provide a quick view of the status of your testing. This is because all details concerning the test cases are maintained in one place. Spreadsheets are easy to set up and use. They behave as a good proof of testing. A spreadsheet can be used to effectively track and manage test suites and test cases.

Example: Figure 11.4 shows an example of a spreadsheet application used to manage test suites and test cases.

Figure 11.4 A Sample Spreadsheet


Microsoft Excel - Purple Dinosaur Test Tracking.xls
File Edit View Insert Format Tools Data Window Help
Test Case Summary

	A	B	C	D	E
1		**Purple Dinosaur Test Tracking**			
2	Test **Suite** /Cases	Test Pass 10/15/1997	Test Pass 11/30/1997	Test Pass 1/5/1998	Bug ID List
3	**Basic Hardware Functionality**				
4	Left Arm Motion	Pass	Pass	Pass	
5	Right Arm Motion	Pass	Pass	Pass	
6	Head Motion	Fail	Pass	Pass	12
7	Touch Sensors	Pass	Pass	Pass	
8	Peek-a-Boo Sensor	Pass	Pass	Pass	
9	PC Radio Transmission	Fail	Fail	Pass	19, 22
10	PC Radio Reception	Pass	Pass	Pass	
11	TV Radio Transmission	Pass	Pass	Pass	
12	TV Radio Reception	Pass	Pass	Pass	
13	**Summary**	**FAIL**	**FAIL**	**PASS**	
14					
15	**Basic Software Functionality**				
16	Songs	Pass	Pass	Pass	
17	Games	Fail	Pass	Pass	13
18	Peek-a-Boo	Pass	Pass	Pass	
19	Cleanup Song	Pass	Pass	Pass	
20	Timeout Sleep	Pass	Pass	Pass	
21	Commanded Sleep	Pass	Pass	Pass	
22	VCR Broadcast Mode	Pass	Fail	Fail	14, 29
23	PC Single Unit Mode	Pass	Pass	Pass	
24	**Summary**	**FAIL**	**FAIL**	**FAIL**	
25					

Source: Ron Patton, Software Testing, Second Edition, Sams Publishing

4. ***Custom Database:*** The best way to track test cases is to use a Test Case Management Tool. It is a database, programmed specifically to handle test cases. There are many commercially available applications that can be set up to perform just this task. If you want to create your own tracking system, then database software like FileMaker Pro, Microsoft Access, and many others provide almost drag-and-drop database creation facility. These software allow you to build a database that is mapped to the IEEE 829 standard in a very short time.

Few examples of test case management tools are:

Example:

1. *HP Quality Centre:* It is a web based test management software application.
2. *qaManager:* It is a simple web based application for managing QA Projects/Teams effectively and efficiently.
3. *ApTest:* It improves the consistency and control throughout the testing process.
4. *InformUp:* A simple application life cycle management tool.
5. *Testopia:* It is a generic tool for tracking test cases.

An important thing to know is that the number of test cases is usually in thousands and without a means to manage them, you and the other testers could find yourselves lost in large volumes of documentation.

11.2 Bug's Life Cycle

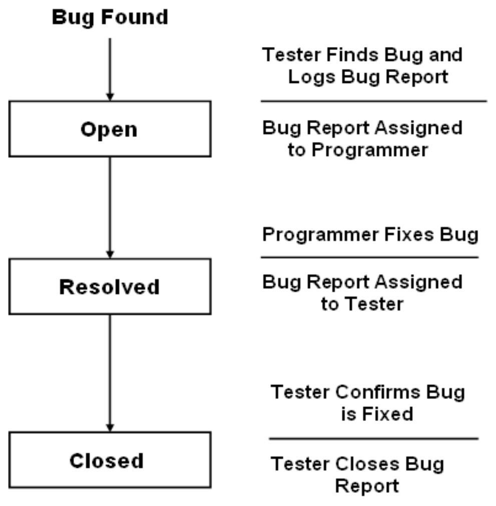
In software, a bug is a coding error or mistake. These bugs must be removed or fixed before the software is put to use. The activity of finding bugs in a program begins after the coding process is completed. It is then integrated with other units of programming to form a software product. The concept behind software bug life stages is similar to a real bug's life stages.

11.2.1 Stages of a Bug

In the software development process, a bug is considered to have a life cycle. In order to be fixed or closed, the bug should complete the life cycle. A specific life cycle guarantees that the process is standardized. The bug reaches different forms during the life cycle.

Example: Figure 11.5 shows an illustration of a simple software bug life cycle.

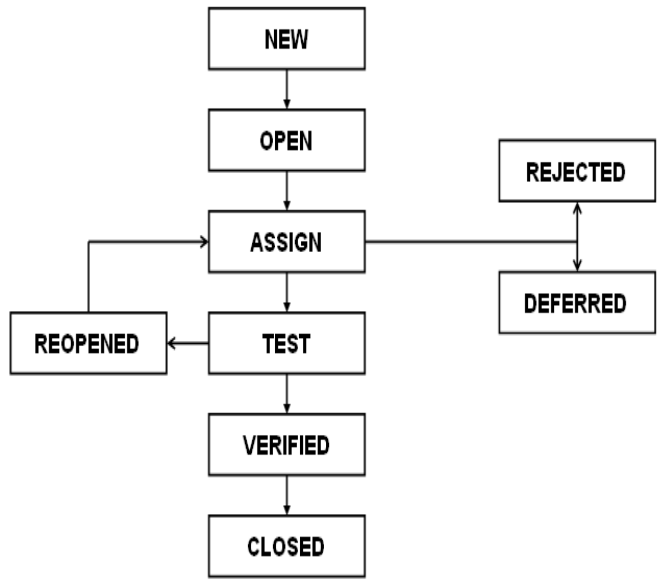
Figure 11.5 A Simple Software Bug Life Cycle


Bug Found
Tester Finds Bug and Logs Bug Report
Open
Bug Report Assigned to Programmer
Programmer Fixes Bug
Resolved
Bug Report Assigned to Tester
Tester Confirms Bug is Fixed
Closed
Tester Closes Bug Report

The example illustrates that when a bug is first found by a software tester, a report is recorded and given to a programmer to be fixed. This state is called the open state. After the programmer fixes the bug, he/she gives the report back to the tester and the bug now enters the resolved state. The tester then carries out a verification test to confirm that the bug is indeed fixed. The bug then enters the closed state which is its final state.

In some cases, the life cycle gets a bit more complicated as illustrated in Figure 11.6.

Figure 11.6 A Complicated Software Bug Life Cycle



Source: http://www.exforsys.com/tutorials/testing/bug-life-cycle-guidelines.html

The different stages of a bug can now be summarized as follows:

1. ***New:*** A bug's state will be "NEW" when it is posted for the first time, that is, the bug is not yet accepted.
2. ***Open:*** After the tester posts the bug, the project manager agrees that the bug is legitimate and changes the state to "OPEN".
3. ***Assign:*** Once the project manager changes the state to "OPEN", the bug is given to the developer. Now the state of the bug is changed to "ASSIGN".
4. ***Test:*** Once the developer fixes the bug, the bug is given back to the testing team for the next round of testing. Before delivering the software with the bug fixed, the developer changes the state of the bug to "TEST". It indicates that the bug has been fixed and is given to the testing team.
5. ***Deferred:*** If the bug state says "DEFERRED", it means the bug is likely to be fixed in later releases. The rationale behind changing the bug to this state is influenced by many factors. Some factors maybe: the priority of the bug may be low, the software release date might be too close, and the bug may not have major effect on the software and so on.
6. ***Rejected:*** If the developer considers that the bug is not legitimate, the bug is rejected. Then the state of the bug is changed to "REJECTED".
7. ***Duplicate:*** When the bug is repeated two times or the two bugs refer to the same feature, then the status of one bug is changed to "DUPLICATE".
8. ***Verified:*** After the bug is fixed the status is changed to "TEST", and then the tester tests the bug. If the bug is not present in the software, then the tester attests that the bug is fixed and changes the status to "VERIFIED".
9. ***Reopened:*** If the bug is still found to exist even after the developer fixes the bug, the tester changes the status to "REOPENED". The bug goes through the complete life cycle once again.

To gain the maximum benefit of quality control system, it is very important that skilled software staff conduct the audit. If the audit report indicates that the software product quality is not up to the mark, then corrective management action is taken accordingly. This helps in maintaining the delivery schedule for the supply of a product to the customer or end-user.

Example: Consider a software project with a requirement of user interface design and a SQL database execution.

The Software Quality Assurance team would develop a quality plan that contains specified standards, methods and procedures that have to be applied to the software project. They would then involve themselves in identifying and preparing the quality plan that needs to be followed.

When these requirements are produced, the Software Quality Control team ensures that the development team has in fact followed the set standards and presents the audit report to the management.

12.2 Testing and Quality Assurance at Workplace

While developing software, the team members must be aware of terms like software testing, software quality assurance, software quality control, software verification and validation, and software integration and testing. Teams based on these projects and the clients should be made aware of the standards being set for quality of the software being developed. The team members should be made aware of common practices that are being followed at the workplace to accomplish software quality assurance. We will start the same by understanding the difference between software testing and quality assurance.

12.2.1 Differences between Software Testing and Quality Assurance

Quality assurances and testing are two overlapping and confusing terminologies. Though they are closely related, yet they are different. Both quality assurance and testing are necessary to effectively manage the risks of creating and maintaining software products.

Software Testing

Software testing is an essential part of software quality assurance, which denotes a review of specification, design and coding related to software products. In simple words, software testing can be described as an assessment, report, and follow-up-task for accomplishing quality goals. Software testing involves the operation of a system under specified conditions and includes continuous evaluation of the output. These specified conditions include both normal and abnormal conditions. Software testing is mainly detection-oriented and is performed by a software tester.

Following are the tasks performed under software testing:

1. Recognizing the most appropriate implementation approach for a given test.
2. Setting up and executing the tests with the intention of finding bugs.
3. Preparing the verification and validation reports of the test plans, test procedures, and test reports.
4. Involving in customer meetings to know the status of projects and design reviews.

The role of a software tester is to assess, report, and follow-up by tracking down the bugs in the software product and also by ensuring that they are fixed. The main aim of the software tester is to find bugs and ensure that they are fixed as soon as possible.

Notes

Role of the software tester

An effective tester needs to take personal responsibility for the bugs he/she finds, track them through bug life cycle, and convince the software development team to fix it at the earliest.

Quality Assurance

Software quality assurance guarantees that the principles, methods, and procedures are in place, and are suitable for the project and can be implemented properly.

Did you know? Bell Labs in the year 1916 introduced formal quality assurance and control function for the first time. This function was successfully implemented by different companies throughout the world.

Software quality assurance involves different means of preventing occurrence of defects in the software being developed. The various methods that are implemented to ensure the quality are ISO 9000 model or CMM model. The aim of software quality assurance is to continuously improve a clearly defined process and to control every step of the process.

The responsibility of the software quality assurance individual is to inspect and calculate the present software development process and to discover ways to improve it with an intention of preventing the occurrence of bugs. The responsibility and scope of a Software Quality Assurance group is larger than the software testing group.

Example: Consider a User interface standard that has no distinction between an "open new window" and a "replace current window", in the “Go To” button.

During usability testing, it is discovered that, lack of distinction between the “open new window" and a "replace current window", caused the end-user’s to search for correct window.

The SQA team sends this information (usability metric) back to the user interface designers and an alteration to the user interface standards are made.

As per the definition of assurance that is “a guarantee or pledge” or “a freedom from doubt,” a quality assurance group’s role is to guarantee that the quality of the product is good. Quality Assurance (QA) group is authorised to inspect the standards and methodologies followed in a software project. It also monitors and evaluates the software development process to provide feedback solutions to the process problems and performs the testing process to decide whether the product is ready for release in the market.

Example: Mean Time Between Failure (MTBF) is a common software measure that tracks down how often the system fails to perform the given task. This measure, in turn, helps to find out the reliability of the product.

Following are the major quality assurance tasks:

1. To develop a standard process for software development and quality assurance and to ensure that that there is no deviation from the standards set.
2. To set guidelines for every step of the process, such as requirement templates, design methodologies, coding standards, and so on.
3. To create checklists for every step of the process and verify the results of each step against the subsequent guidelines and checklists.
4. To develop quality factors, quality criteria, and quality metrics and define a complete set of quality factors.

Example: Software quality assurance includes checklist and other important specifications related to the software program.

Did you know? Most of the firms have adopted total quality management programs since 1980s to retain competitiveness in order to achieve customer satisfaction in the era of globalization.

Caution Several companies are supporting the "process improvement" mantra as though it is the solution for all their software development and quality problems. However, though a standard process is necessary, it is not adequate. There is no hard evidence that conformance to process standards guarantees good products.

12.3 Quality Management in IT

Quality management, which coordinates the various activities to direct and control the organization with regard to quality, is useful for any industry. Most of the principles and theories of quality management that are applied to software development and maintenance activities can also be applied to all other Information Technology (IT) activities.

The principles and theories that are applied emphasize on the following:

1. To identify the important IT processes and their sequence.
2. To plan for defect prevention versus detection by applying IT best practices.
3. To use and implement various standards to achieve appropriate levels of IT governance.
4. To resolve the IT issues equivalent to bugs, defects, and errors.
5. To determine and document the requirements of the customers.
6. To observe and quantify the service performance.
7. To assure the procurement of quality when outsourcing important IT processes.

The term "Information Technology can be defined as any equipment or interconnected system that is used in automatic attainment, storage, control, and transmission of data. Information technology consists of computers, ancillary equipment, and software."

In order to apply the principles of quality management to the "organized activities" executed by an IT company, it is essential to be familiar with the important IT processes. An IT infrastructure depicts all the components that are utilized in the delivery of the IT services to the end-users, including the computing and telecommunication services. These components and their use should be effectively managed. Therefore, a proper IT infrastructure management should be in place.

The management of IT infrastructure and IT services together is called as IT Service Management (ITSM). ITSM establishes the principles and practices of designing, delivering, and maintaining IT services to an agreed-upon level of quality, with respect to the customer requirement. This section explains the important processes found in a typical IT Company. The processes are quite similar to the software engineering processes found in a software life cycle.

ITSM Processes

IT Service Management (ITSM) is based on two major categories of IT service -- one is IT service support and the other is IT service delivery. The ITSM processes assure that within these two categories, the levels of quality that are agreed-upon are achieved.

Service Support

IT service support consists of processes oriented towards the efficient delivery of IT operational services. The processes are as follows:

1. ***Service Desk Function:*** It serves as a main point of contact for customers and supports the management process in providing the resolution.
2. ***Incident Management:*** It is accountable for maintaining regular position of IT service operations as quickly as possible to minimize the adverse impact on business processes.
3. ***Problem Management:*** It reduces the unfavorable impact of incidents and problems on the business operations caused due to errors in the IT infrastructure and effectively manages the problems that occur.
4. ***Configuration Management:*** It is accountable for recognizing, recording, and reporting on configuration items and their relationships to the supporting IT service department.
5. ***Change Management:*** It organizes and controls all changes to the IT services to reduce the adverse impact of those changes on business operations and the end users of IT services.
6. ***Release Management:*** It implements the various changes to IT services by taking various views of customers, employees, technology, and IT governance that supports all aspects of a change which include planning, designing, building and testing activities.

Service Delivery

IT service delivery includes processes that are associated with the long-term planning, control, and managerial aspects of IT services. The processes include:

1. ***Service-level Management:*** It organizes, plans, coordinates, negotiates, reports, and supervises the quality of IT services at the agreed upon cost.
2. ***Availability Management:*** It is accountable for increasing the ability of the IT infrastructure services and supports organizations to deliver a cost-effective and sustained level of service that meets business requirements.
3. ***Capacity Management:*** It ensures that future and current capacity and performance aspects of the IT infrastructure are supplied on time to meet up business requirements at an agreed upon cost.
4. ***Service Continuity Management:*** It is accountable for business stability management functions by ensuring that IT services are stable even in the event of a major business interruption.
5. ***Financial Management for IT Services:*** It provides budgeting, accounting, and charging services in order to manage and control the IT costs and expenditures.
6. ***Security Management:*** It is accountable for reducing security-related incidents by effectively managing privacy, discretion, integrity, and availability of IT services.
7. ***Applications Management:*** It handles various applications from the initial business setup throughout the stages of application life cycle till retirement.
8. ***Software Asset Management:*** It implements good corporate governance system to manage, control, and safeguard the organization's software assets, as well as those risks arising from the use of software assets.

Example: The major standards established by the ITSM processes are ISO 20000 and CobiT.

Recently, the best practices and standards of IT identified quality as a process and recognized the value of a quality management system.

LOVELY PROFESSIONAL UNIVERSITY
Jalandhar-Delhi G.T. Road (NH-1)
Phagwara, Punjab (India)-144411
For Enquiry: +91-1824-521360
Fax.: +91-1824-506111
Email: odl@lpu.co.in

ISBN 978-93-87034-85-3
9 789387 034853