Software Testing

Lecture 11

Fundamentals of Software Testing

Observations about Testing

- "Testing is the process of executing a program with the intention of finding errors." – Myers
- Process used to identify the correctness, completeness and quality of developed computer software.
- **Includes a set of activities conducted with the intent of finding errors in the software**
- An activity to check that the software system is defect free.
- Process of executing a program / application under positive and negative conditions by manual or automated means. It checks for the :-

  - Specification
  - Functionality
  - Performance

What is Software Testing?

The process of finding evidence of defects in software systems.

- Establishing confidence that a program does what it is supposed to do.

1. Software testing is not debugging.
2. Software testing is not quality assurance

Testing Vs Debugging

- **Testing** is focused on identifying the problems in the product
- Done by Tester
- Need not to know the source code

- **Debugging** is to make sure that the bugs are removed or fixed
- Done by Developer
- Need to know the source Code

Testing and Debugging

- **Testing** is the process of determining if a program has any errors. When testing reveals an error, the process used to determine the cause of this error and to remove it, is known as **debugging**.
- Localization and correction of defects are tasks for a software developer and are often called debugging.
- **Debugging is often equated with testing, but they are entirely different activities.** Debugging is the task of localizing and correcting faults.
- The goal of testing is the (more or less systematic) detection of failures (that indicate the presence of defects).

Testing &Debugging Difference

- **Testing:** It involves the identification of bug/error/defect in the software without correcting it. Normally professionals with a Quality Assurance background are involved in the identification of bugs. Testing is performed in the testing phase.
- **Debugging:** It involves identifying, isolating and fixing the problems/bug. Developers who code the software conduct debugging upon encountering an error in the code. Debugging is the part of White box or Unit Testing. Debugging can be performed in the development phase while conducting Unit Testing or in phases while fixing the reported bugs.

Software Testing Vs Quality Assurance

Testing is necessary but not enough for QA process.

- Testing contributes to improve quality by helping to identify problems.

QA sets standards that project team (including testers) should follow in order to build a better software.

Software Testing Purposes

Testing software has **different purposes:**

- Executing a program to find failures
- Executing a program to measure quality
- Executing a program to provide confidence
- Analyzing a program or its documentation to prevent failures

Introduction- Software Testing?

Software testing is defined as "a formal process in which a software unit, several integrated software units or an entire package are examined by running the programs on a computer. All the associated tests are performed according to approved test procedures on approved test cases" (Galin, 2004).

Basic Question's about Testing

What is Software Testing?

Correctness of software with respect to requirements;

Performance of software under various conditions;

Robustness of software, its ability to handle erroneous input and unanticipated conditions;

Installation and other facets of a software release.

What is Software Testing? Defect Reduction

Testing

- What to test
  - **Black-box** (or functional) testing verifies the correct handling of the external functions
  - **White-box** (or structural) testing verifies the correct implementation of internal units, structures, and relations among them
  - **White-box test examines source code with focus on:**
    - Control flow
    - Data flow

Software Testing

- **Computer programs are designed and developed** by humans and hence are prone to errors.
- **Unchecked,** they can lead to a lot of problems.
- **Testing** the software becomes an essential part of the software development lifecycle.
- **Carrying out the testing activities** for projects has to be practiced with proper planning and must be implemented correctly.

Why Testing?

- To find and correct defects.
- To check whether the Client/User needs are satisfied
- To avoid user detecting problems
- Also to provide Quality Product.
- It also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. So it can be either done manually or using automated tools.

Why Testing?

- To find and correct defects.
- To check whether the Client/User needs are satisfied
- To avoid user detecting problems
- Also to provide Quality Product.
- It also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. So it can be either done manually or using automated tools.

Root Causes of Failures

Inaccurate understanding of end user requirements.

- Inability to deal with changing requirements.

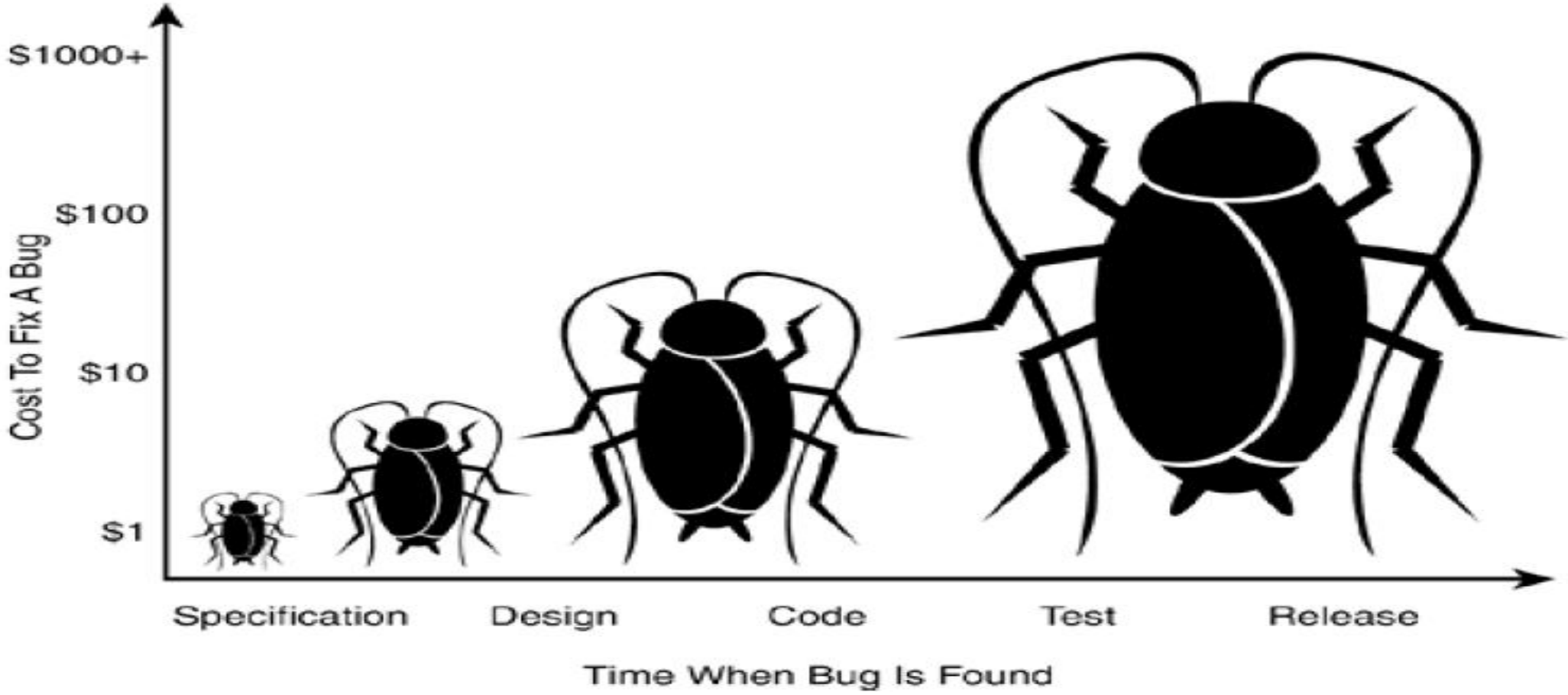
Late discovery of serious project flaws.

- For example, modules that do not fit together.

Untrustworthy build & release process.

- Implementation team's chaos.

Failure Costs



Basic Questions on Testing

What to test?

Any working product which forms part of the software application has to be tested. Both data and programs must be tested.

How often to test?

- When a program (source code) is modified or newly developed, it has to be tested

Who does testing ?

- Software Tester
- Software Developer
- Project Lead/Manager
- End User/Customer/Third Party

When starts testing

Testing is done in different forms at every phase of SDLC:

- **During the requirements gathering phase**, the analysis and verification of requirements are also considered as testing.
- **Reviewing the design in the design phase** with the intent to improve the design is also considered as testing.
- **Testing performed by a developer on completion of the code** is also categorized as testing.

When to Stop Testing?

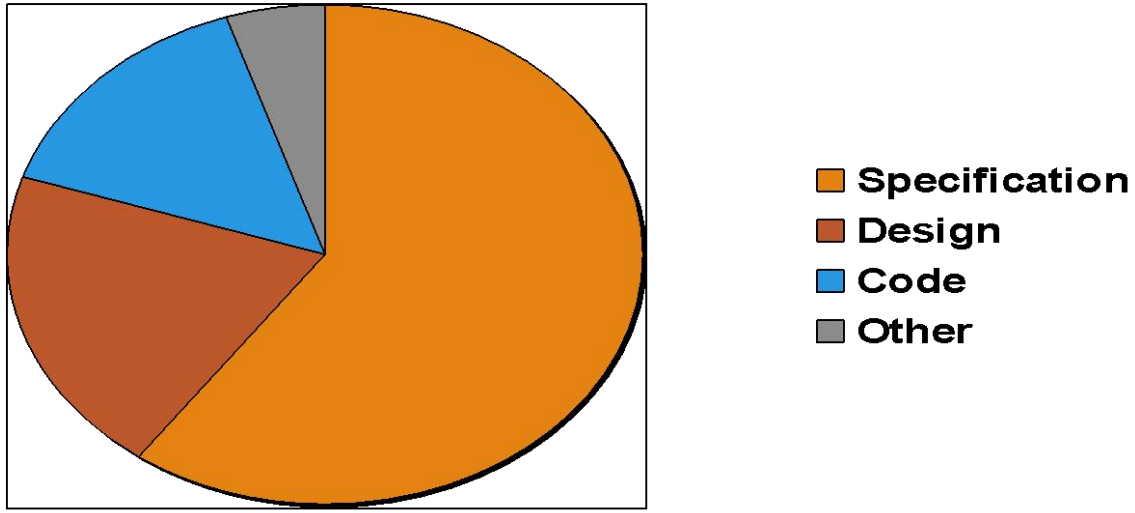
The following **aspects are to be considered for stopping** the testing process:

- Testing Deadlines
- Completion of test case execution
- Completion of functional and code coverage to a certain point
- Bug rate falls below a certain level and no high-priority bugs are identified
- Management decision

Why does S/W have bugs?

- Miscommunication or No Communication (That we are not clear about what an application should do or shouldn't do)
- Time Pressure (Time scheduling)
- Changing Requirements
- Software Complexity (Without experience its bit tough to do the tough application)
- Programming Mistakes

Why do Failures Occur?

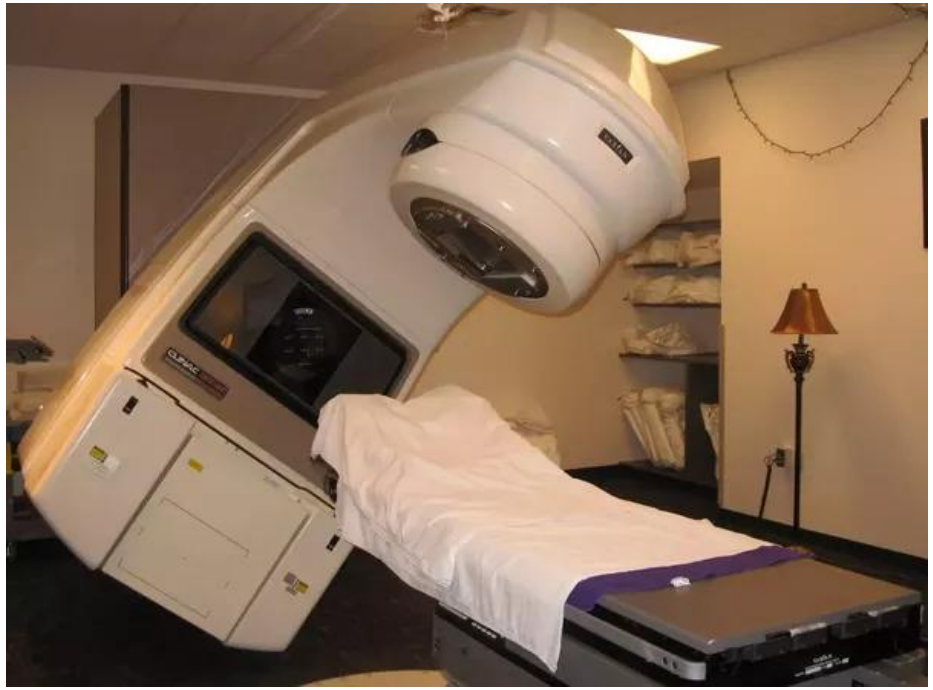


Why testing important Examples?

- China airplane crash on April 26th 1994, due to software bug in controls and caught fire during landing, killing 264 people.

Other Examples……

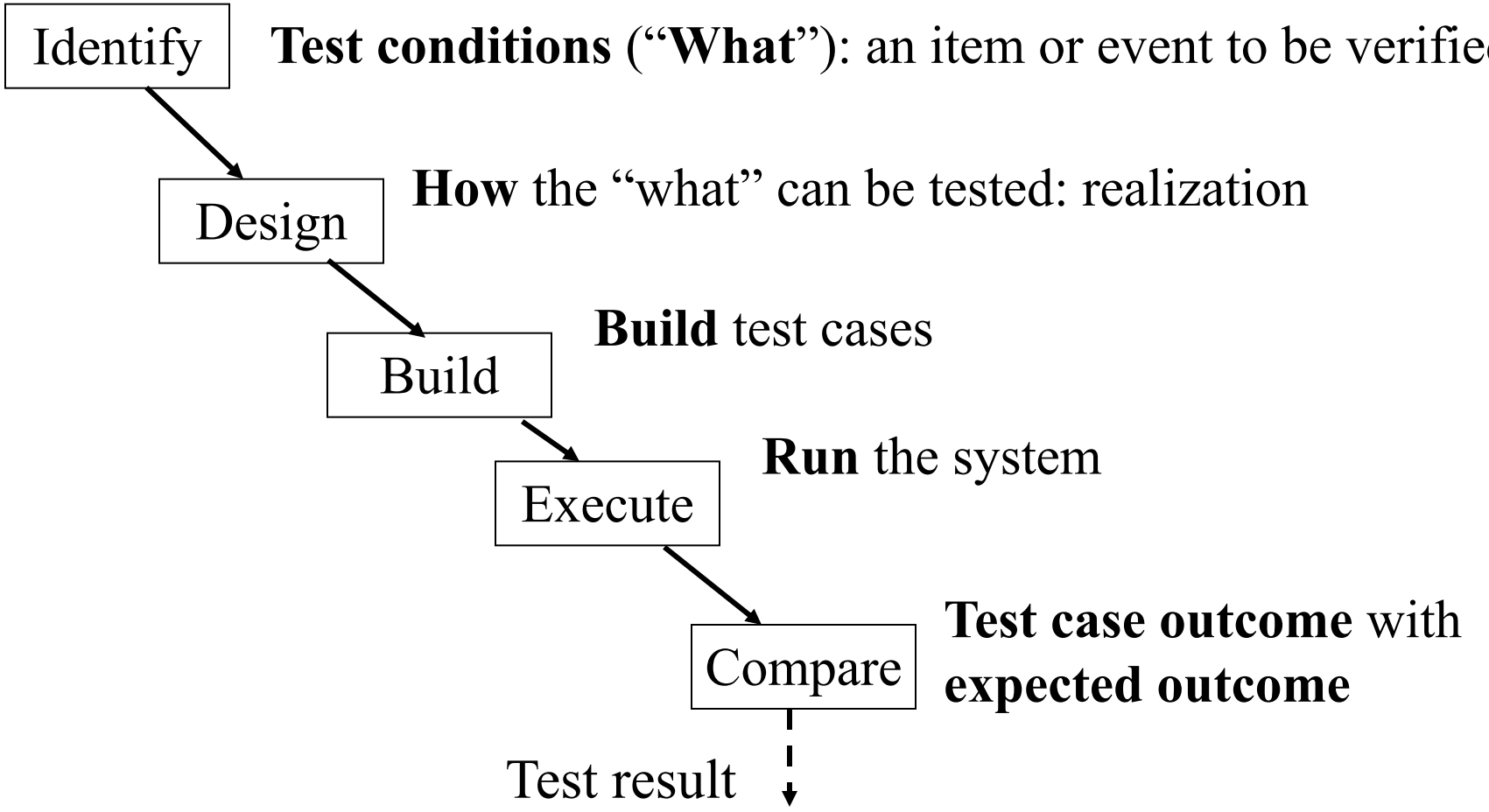
- In 1985, Therac-25 Radiation Therapy Machine kill 3 people and 3 injured due to massive overdoses of radiations.



Other Examples…..

- **In April of 1999**, a software bug caused the failure of a $1.2 billion military satellite launch, the expensive accident in history
- **In 2015 fighter plane F-35** fell victim to a software bug, making it unable to detect targets correctly.
- **Some of the Amazon's third party retailers** saw their product price is reduced to 1p due to a software glitch. They were left with heavy losses.
- **In may of 1996**, a software bug caused the bank accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars.

Testing Activities



Types of Testing

- Sanity Testing
- End-to-end Testing
- Install/Uninstall Testing
- Functional Testing
- Regression
- Integration Testing
- Usability Testing
- Interface Testing
- Performance Testing
- Configuration Testing
- Unit Testing

- Integration Testing
- System Testing
- Usability Testing
- Interface Testing
- Performance Testing
- Stress Testing
- Load Testing
- Configuration Testing
- Compatibility Testing – Focus On Browsers And Windows Etc
- Recovery Testing
- Acceptance Testing
- Beta Testing
- Documentation Testing

Types of Testing

Sanity Testing

- determines whether it is possible and reasonable to proceed with further testing

End-to-end Testing

- End-to-end testing is a methodology used to test whether the flow of an application is performing as designed from start to finish

Install/uninstall Testing

- check that software application is successfully installed/uninstalled & it is working as expected after installation and rest of the system is working fine after uninstallation

Functional Testing

- to ensure that the specified functionality required in the system requirements works.

Regression

- testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results

Unit Testing

- testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

Integration Testing

- testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation

Types of Testing

Usability Testing

- is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design?

Interface Testing

- is the process of testing a product's graphical user interface to ensure it meets its written specifications.

Performance testing

- is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements.

Stress Testing

- is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications

Load Testing

- is the testing to evaluate how system behaves under defined favorable conditions.

Configuration Testing

- is a special type of testing to verify the operation of software for various system configurations.

System Testing

- testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment.

Basic Terms

Error:

- Errors are a part of our daily life.
- Humans make errors in their thoughts, actions, and in the products that might result from their actions.
- Errors occur wherever humans are involved in taking actions and making decisions.
- Software Error: An incorrect internal state that is the manifestation of some fault.

Examples:

Hearing:

Spoken: He has a garage for repairing foreign cars.

Heard : He has a garage for repairing falling cars.

Medicine:

Incorrect Antibiotic prescribed.

Sports:

Incorrect call by a referee in a tennis match.

Basic Definitions

Failure

- There is a deviation of the observed behavior of a program or a system from its specification.

Fault

- An incorrect step, process or data definition.

Error

- Difference between computed, observed or measured value and the true or theoretically correct value or condition.

Basic Definitions: Definitions

Failure: external behavior

- Deviation from expected/specified behavior

Fault: internal characteristics - *cause for failures*

- An incorrect step, process, or data definition in a computer program

Error: refers to a missing or incorrect human action such as human misconceptions, misunderstandings, etc resulting in certain fault(s) being injected into a software

- during design, coding and data entry

Basic Definitions: Definitions

Defect:

- Generally refers to some problem, either with external behavior or with internal characteristics
- error/fault/failure are collectively referred to as defects

Bug/debug:

- not good terms, avoid
- People have raised moral or philosophical objection to the use of bugs as evading responsibility for something people committed
- Instead use defect detection & removal

Basic Terms

Error:

- Errors are a part of our daily life.
- Humans make errors in their thoughts, actions, and in the products that might result from their actions.
- Errors occur wherever humans are involved in taking actions and making decisions.
- Software Error: An incorrect internal state that is the manifestation of some fault.

Examples:

Hearing:

Spoken: He has a garage for repairing foreign cars.

Heard : He has a garage for repairing falling cars.

Medicine:

Incorrect Antibiotic prescribed.

Sports:

Incorrect call by a referee in a tennis match.

Error, Fault & Failure

Error :

- An error is a mistake, misconception, or misunderstanding on the part of a software developer that cause fault or we can say defective programming by the developer.

Fault/Bug/Defect :

- A fault is introduced into the software as the result of an error. OR Discrepancy in code that causes failure.
- An incorrect step, process or data definition in a computer program which causes the program to perform in an unintended or unanticipated manner.
- It is a condition that causes the software to fail to perform its required function
- Example: statements in the program

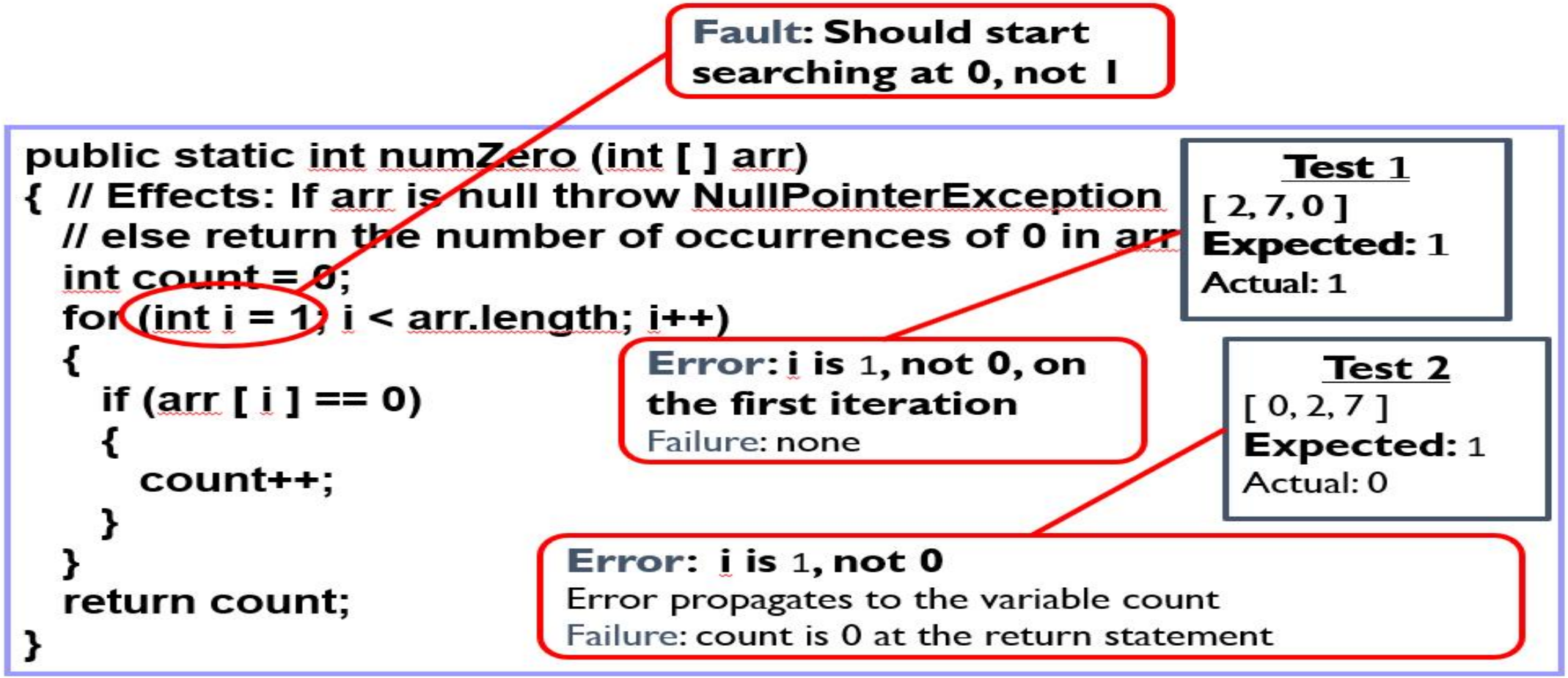
Failure :

- Deviation of the software from its expected result. OR failure means that a given requirement is not fulfilled; it is a discrepancy between the actual result or behavior and the expected result or behavior
- *Software Failure:* External, incorrect behavior with respect to the requirements or another description of the expected behavior.

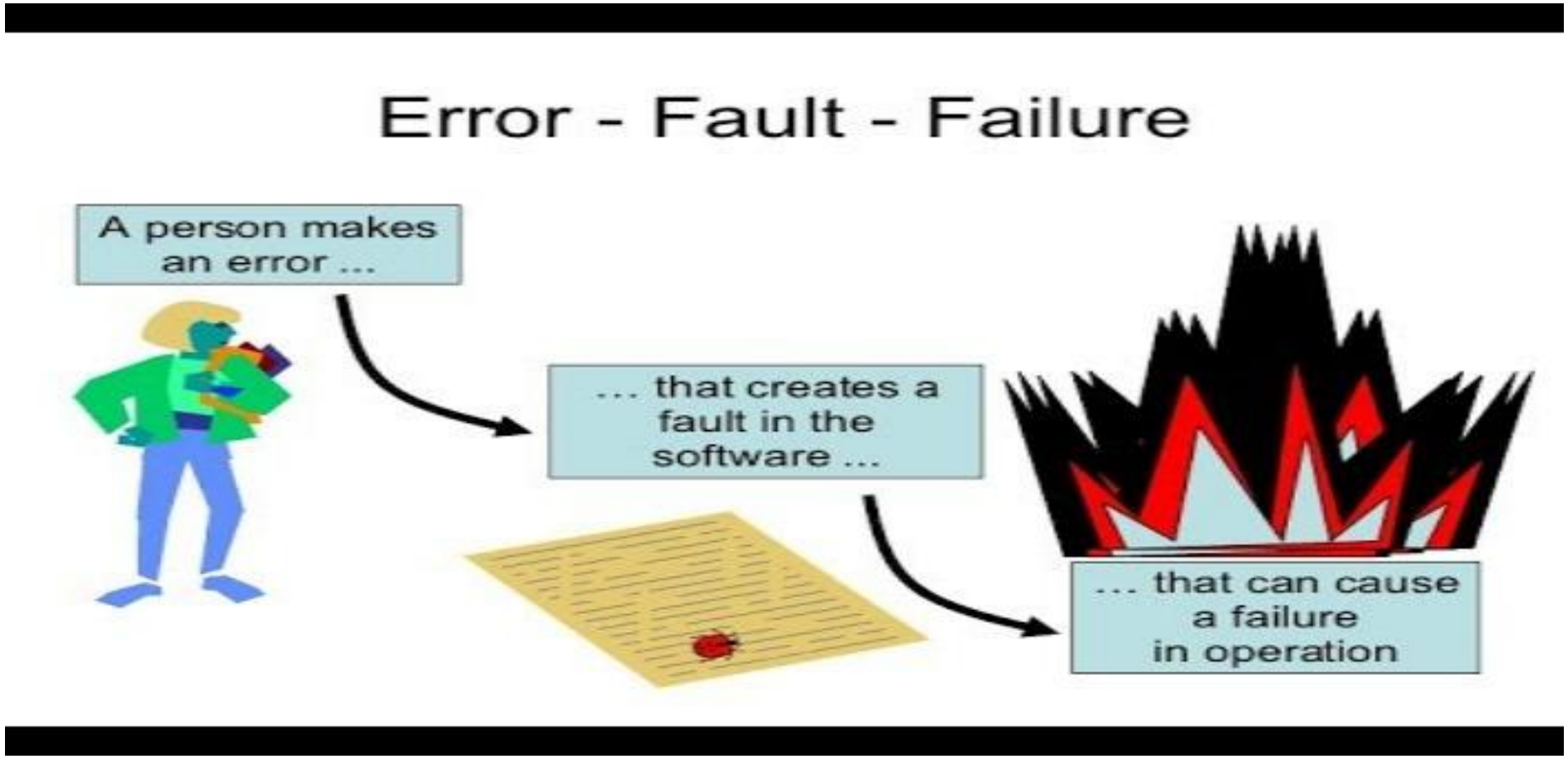
Example:

- There is a product that is too difficult to use.
- Or an output is wrong or the program crashes.

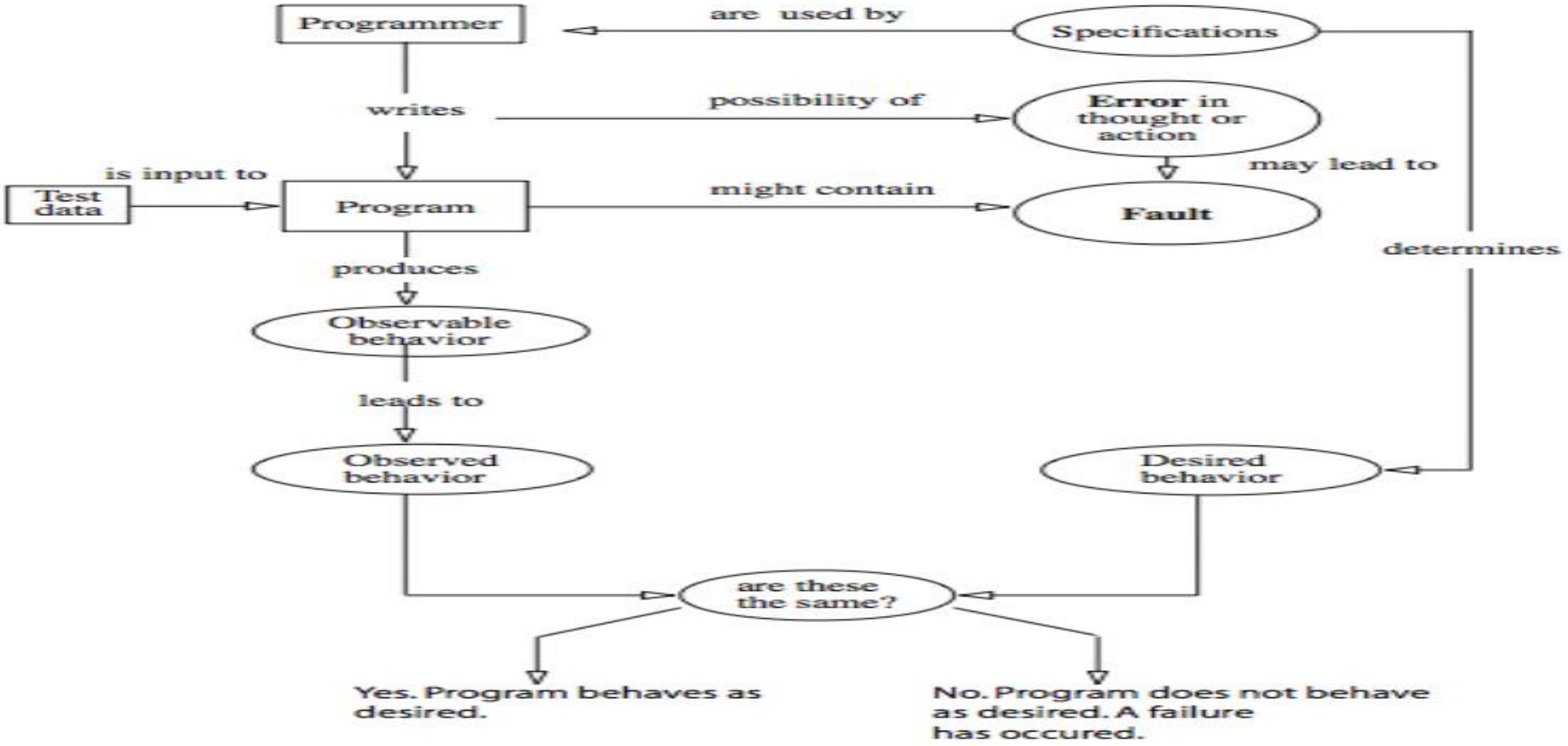
Error, Fault & Failure



Error, Fault and Failure



Error, Fault and Failure



Software Quality attributes

- **Availability:** Is it available when and where I need to use it?
- **Efficiency:** How few system resources does it consume?
- **Flexibility:** How easy is it to add new features?
- **Install ability:** How easy is it to correctly install the product.
- **Interoperability:** How easily does it interconnect with other systems?
- **Maintainability:** How easy it is to correct defects or make changes?
- **Portability:** Can it be made to work on other platforms?

Software Quality Attributes

- **Reliability:** How long does it run before causing a failure?
- **Reusability:** How easily can we use components in other systems?
- **Testability:** Can I verify that it was implemented correctly?
- **Usability:** How easy is it for people to learn or to use?
- **Performance:** the response time, utilization, and throughput behavior of the system.
- **Security:** system's ability to resist unauthorized attempts at usage or behavior modification, while still providing service to authorized users.

Software Quality

Static Quality Attributes:

- Structured, maintainable, testable code as well as the availability of correct and complete documentation.

Dynamic Quality Attributes:

- Software reliability, correctness, completeness, consistency, usability, and performance.

Software Quality (contd.)

Completeness:

- **It refers to the availability** of all features listed in the requirements, or in the user manual.
- An incomplete software is one that does not fully implement all features required.

Consistency:

- **It refers to adherence to a common set of conventions and assumptions**. For example, all buttons in the user interface might follow a common color coding convention.
- An example of inconsistency would be when a database application displays the date of birth of a person in the database.

Software Quality (contd.)

Usability:

- Refers to the ease with which an application can be used.
- This is an area in itself and there exist techniques for usability testing. Psychology plays an important role in the design of techniques for usability testing.

Performance:

- Refers to the time the application takes to perform a requested task. It is considered as a non-functional requirement.
- It is specified in terms such as ``This task must be performed at the rate of X units of activity in one second on a machine running at speed Y, having Z gigabytes of memory."

Audit and Inspection

Audit:

- **An independent examination of a work product** or set of work products to assess compliance with specifications, standards, contractual agreements, or other criteria.
- As per IEEE, it is a review of documented processes that organizations implement and follow. Types of audit include Legal Compliance Audit, Internal Audit, and System Audit.
- At high level audits are a “Check”.

Inspection

Inspection :

- Process of evaluating or examining things
- A formal evaluation technique in which software requirements, design, or code are examined in detail by person or group other than the author to detect faults, violations of development standards, and other problems.
- It improves product quality and documentation
- At high level inspections are a “do”.

Defect Masking

Masked defect

- **Existing error that has not yet caused a failure just because another error has prevented that piece of the code from being performed.**
- Masked defect hides other defects in the system.
- E.g. There is a link to add employee in the system. On clicking this link you can also add a task for the employee. Let's assume, both the functionalities have bugs. However, the first bug (Add an employee) goes unnoticed. Because of this the bug in the add task is masked.

Latent defect

- **Existing error that has not yet caused a failure because the accurate set of conditions was never met.**
- E.g. February has 28 days. The system could have not considered the leap year which results in a latent defect
- These defects do not cause damage to the system immediately but wait for a particular event sometime to cause damage and show their presence.

Test Effort

- Test effort is often shown as the proportion between the number of testers and the number of developers.
- **The proportion varies from 1 tester per 10 developers to up to 3 testers per developer.**
- The conclusion is that test efforts or the budget spent for testing vary enormously.
- **Testing cannot prove the absence of faults.**
- **In order to do this**, a test would need to execute a program in every possible situation with every possible input value and with all possible conditions.
- But in practice complete testing is not feasible.
- **Test effort is difficult as it depends very much on the character of the project.**

Test Effort

- Define test intensity and test extent depending on risk (For every software program it must be decided how intensively and thoroughly it shall be tested. This decision must be made based upon the expected risk of failure of the program)
- Select adequate test techniques(Every technique especially focuses on and checks particular aspects of the test object)
- Test of extra functionality (The product should provide only the required functionality)
- Test case explosion(The testing effort can grow very large)
- Limited resources

Testing and Debugging

- **Testing** is the process of determining if a program has any errors. When testing reveals an error, the process used to determine the cause of this error and to remove it, is known as **debugging**.
- Localization and correction of defects are tasks for a software developer and are often called debugging.
- Debugging is often equated with testing, but they are entirely different activities. Debugging is the task of localizing and correcting faults.
- The goal of testing is the (more or less systematic) detection of failures (that indicate the presence of defects).

Testing &Debugging Difference

- **Testing:** It involves the identification of bug/error/defect in the software without correcting it. Normally professionals with a Quality Assurance background are involved in the identification of bugs. Testing is performed in the testing phase.
- **Debugging:** It involves identifying, isolating and fixing the problems/bug. Developers who code the software conduct debugging upon encountering an error in the code. Debugging is the part of White box or Unit Testing. Debugging can be performed in the development phase while conducting Unit Testing or in phases while fixing the reported bugs.

Automated Vs. Manual Testing

ADVANTAGES	
Automated Testing	**Manual Testing**
• If you have to run a set of tests repeatedly automation is a huge gain	• If Test Cases have to be run a small number of times it's more likely to perform manual testing
• Helps performing "compatibility testing" - testing the software on different configurations	• It allows the tester to perform more ad-hoc (random testing)
• It gives you the ability to run automation scenariosto perform regressions in a shorter time	• Short term costs are reduced
• It gives you the ability to run regressions on a code that is continously changing	• The more time tester spends testing a module the grater the odds to find real user bugs
• Can be run simultaneously on different machines thus decreasing testing time	
• Long term costs are reduced	
DISADVANTAGES	
Automated Testing	**Manual Testing**
• It's more expensive to automate. Initial investments are bigger than manual testing	• Manual tests can be very time consuming
• You cannot automate everything, some tests still have to be done manually	• For every release you must rerun the same set of tests which can be tiresome
OTHER FACTORS	
• The performance of yout test tools	
• The knowledge level of your testing team	
• The continous growth of software to be tested	
• Number of necessary regressions	

Product Vs Process Metrics

Software metrics may be broadly classified as

- Product metrics:
  - The product is measured to improve its quality
- Process metrics:
  - The process is measured to be improved


Thank
you