Fundamentals of Testing

Compiled by - Nazmus Sakib Akash

Understanding Software Testing

The economic importance of software

- The functioning of machines and equipment depend largely on software.
- We cannot imagine large systems in telecommunication, finance or traffic control running without software.

Software quality

- More and more, the quality of software has become the determining factor for the success of technical or commercial systems and products.

Testing for quality improvement

- Testing and reviewing insure the improvement of the quality of software products as well as the quality of software development process itself.

Failure Example 1: Ariane 5 Launch

Flight 501, which took place on June 4, 1996, was the first test flight of the Ariane 5 expendable launch system. It was not successful; the rocket tore itself apart **37 seconds after launch** because of a **malfunction in the in the control software**, making the fault one of the most expensive computer bugs in history.

The Ariane 5 software **reused the specifications*** the Ariane 4, but the Ariane 5's flight path was considerably different and beyond the range for which the reused code had been designed.

Specifically, the Ariane 5's greater acceleration caused the back-up and primary inertial guidance computers to crash, after which the launcher's nozzles were directed spurious data.

Pre-flight tests had never been performed on the re-alignment code under simulated Ariane 5 flight conditions, so the **error was not discovered before launch**.

Failure Example 2: Lethal X-Rays

Because of a software failure, a number of patients received a lethal dose of gamma rays.

- **Therac-25** was a radiation therapy machine produced by atomic Energy of Canada Limited.
- It was involved with at least six known accidents between 1985 and 1987. At least five patients died of the overdoses.
- These accidents highlighted the dangers of software control of safety-critical system.

Causes of Software Failures

Human error

- A defect was introduced into the software code, the data or the configuration parameters.

  Causes of human error:

  - Time pressure,
  - Excessive demands,
  - Complexity distractions.

Environmental conditions

- Changes of environmental conditions.

  Causes of negative environmental conditions:

  - Radiation,
  - Magnetism,
  - Electronic field on pollution sun spots,
  - Hard disk crashes,
  - Power functions.

Definition: Error, Defect & Failure

"Defects cause failure"

Error (IEEE 610):

- A human action that produces an incorrect result, e.g. a programming error.

Defect:

- A flaw in a component or system to fail to perform its required function, e.g. an incorrect statement of data definition.

Failure:

- The physical or functional manifestation of a defect. A defect, if encountered during execution, may cause a failure.
- Deviation of the component or system from its expected delivery, service or result.

Testing during Software Development, Maintenance & Operation

Increasing software quality:

- Testing helps to furnish the software with the desired attributes, i.e. to remove defects leading to failures.

Reduction of the risk of encountering errors:

- Appropriate test activities will reduce the risk of errors being encountered during software operation.

Meeting obligations:

- Tests might be mandatory because of client's or legal regulation as well as to meet industrial standards.

Relative Cost of Bugs

"Bugs found later cost more to fix"

Cost to fix a bug increases exponentially (10x)

i.e., it increases tenfold as time increases.

Example:

A bug found during specification costs $1 to fix.

… if found in *design* cost is $10

… if found in *code* cost is $100

… if found in *released software* cost is $1000

Definition: Quality, Software, Software Quality

Software (as per IEEE 610):

- Computer programs, procedures and possibly associated documentation and data pertaining to the operation a computer system.

Quality (as per IEEE 610):

- The degree to which a component, system or process meets specified requirements and/or user/customer needs and expectations.

Software Quality (as per IEEE 610):

- The totality of functionality and features of a software product that contribute to its ability to satisfy stated or implied needs.

Software Quality

According to **ISO/IEC 9126**, software quality consists of:

- **Functional Quality Attributes,**
- **Non-functional Quality Attributes.**

Functional Quality Attributes:

- Accuracy
- Compliance
- Interoperability
- Stability
- Security

Non-functional Quality Attributes:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Functional Q-attributes

Functionality means:

- *Correctness:* the functionality meets the required attributes / capabilities

- *Completeness:* the functionality meets all(functional) requirements

Functionality includes (as per ISO/IEC 9126):

- Suitability,
- Accuracy,
- Interoperability,
- Security.

Non-functional Q-attributes (I)

Reliability

- Maturity, fault tolerance, recovery after failure.

  Characteristics: Under given conditions, a software/system will keep its capabilities/functionality over a period of time.

- Reliability = quality/time

Usability

- Learn ability, understanding ability, attractiveness.

  Characteristics: Easy to learn, compliance with guidelines, intuitive handling

Non-functional Q-attributes (II)

Efficiency

- System behavior: Functionality and time behavior.
- *Characteristics:* The system requires a minimal use of resources (e.g. CPU-time) for executing the given task

Maintainability

- Verifiability, stability, analyzability, changeability
- *Characteristics:* Amount of effort needed to introduce changes in system components

Portability

- Reparability, compliance, install ability.
- Ability to transfer the software to a new environment (software, hardware, organization)
- *Characteristics:* Easy to install and uninstall, parameters

Types of Quality Assurance (QA):

Constructive activities:

- to prevent defects, e.g. through appropriate methods of software engineering.

Analytical activities:

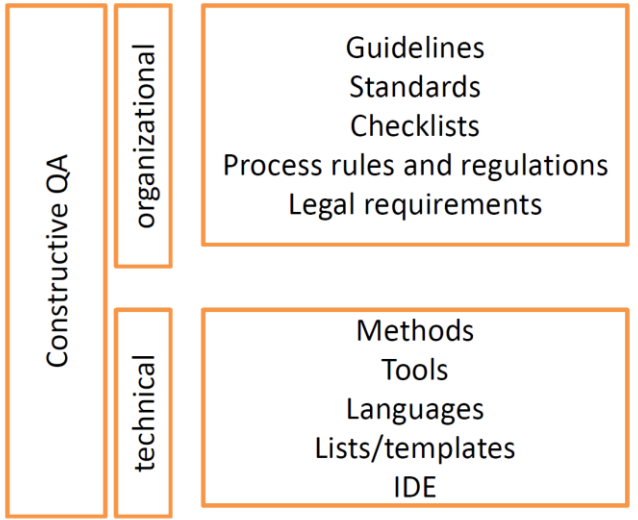
- for finding defects, e.g. through testing leading to correcting defects and preventing failures, hence increasing the software quality.

Constructive Quality Assurance

Quality of process - Quality management

Motto

- Defects not made , need not be fixed
- Defects that were made need not be repeated
- Prevent defects



Analytical Quality Assurance

Quality of product - Verification and test procedure.

Motto

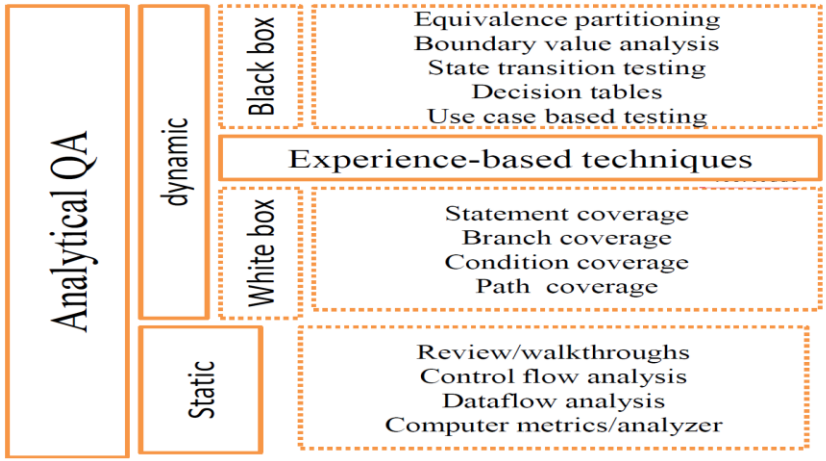
- Defects should be detected early as possible in the process.

Static testing

- Examination without excluding the program.

Dynamic testing

- Includes executing the program.



Quality Attributes

- Some software quality attributes are influenced reciprocally, Because of this, depending on the test object, attributes must be prioritized, e.g. efficiency vs. portability.

- Different kinds of tests will be performed in order to measure the different kinds of attributes .

Test Goals

Gain knowledge about defects in the test objects:

- Defects contained in the test objects must be detected and be described in such a way as to facilitate their correction

Poor functionality:

- System functionality should be implemented as specified

Generating information:

- Before handing over a software system to the users, information about possible risks has to be provided. Gaining such information might be one of the test goals.

Gaining confidence

- Software that has been well tested in trusted to meet the expected functionality and to have a high quality level.

How Much Testing Is Enough?

Exit criteria:

- Not finding (any more) defects is not an appropriate criterion to stop testing activities.
- Other metrics are needed to adequately reflect the quality level reached.

Risk based testing:

- Levels of risk determine the extent of testing carried out, i.e. liability for damages in case of failures occurring, economic and project related aspects.

Time and budget testing:

- The amount of resources available (personal, time and budget) might determine the extent of testing efforts.

Test Case, Test Basis

Test Case: Includes at least the following information

- Pre-condition
- Set of input values
- Set of expected results
- Expected post conditions
- Unique identifier
- Dependence on other test cases
- Reference to the requirement that will be tested
- How to execute the test and check results (optional)
- Priority (optional)

Test Basis: (aka Test Base)

- Set of documents defining the requirements of a component or system. Used as the basis for the development of test cases.

Definitions: Software Development and Reviews

Code (aka Source Code):

- A computer program, written in a programming language, which can be read by human beings.

Debugging:

- Locates and corrects defects in the source code.

Software Development:

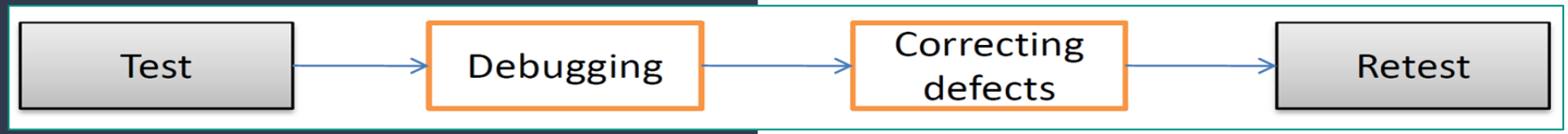
- Is a complex process/sequence of activities aiming at implementing a computer system. It usually follows a software development model.

Requirement:

- A requirement describes a functional attribute that is desired or seen as obligatory.

Review (after IEEE 1028):

- Evaluation of a product or project status to find discrepancies from planned result and to recommend improvements



Testing and Debugging

Test and Retest are test activities:

- Testing shows system failures.
- Re-testing proves, that the defect has been corrected.

Debugging and Correcting defects are developer activities:

- Through debugging, developers can reproduce failures, investigate the state of programs and find the corresponding defect in order to correct it.

Seven Testing Principles (I)

Principle 1: Testing shows the presence of defects.

- Testing can prove the presence of defects.
- Deviations discovered while testing show a failure.
- The cause of the failure might not be obvious.
- Testing reduces the probability of defects remaining undiscovered. The absence of failure does not prove the correctness of the software.
- The test procedure itself might contain error.
- The test conditions might be unsuitable for finding errors.

Seven Testing Principles (II)

Principle 2: Exhaustive testing is not possible.

Exhaustive testing

- A test approach in which the test suite comprises all combinations of input values and preconditions.

Test case explosion

- Defines the exponential increase of efforts and costs when testing exhaustively.

Sample test

- The test includes only a (systematically or randomly derived) subset of all possible input values
- Under real life conditions, sample tests are generally used. Testing all combination of inputs and preconditions is only economically feasible in trivial cases.

Seven Testing Principles (III)

Principle 3: Early Testing.

- The earlier a defect is discovered, the less costly is its correction.
- Highest cost effectiveness when errors are corrected before implementation
- Concepts and specifications may already be tested.
- Defects discovered at the conception phase are corrected with the least effort and costs.
- Preparing a test is time consuming as well.
- Testing involves more than just test execution.
- Test activities can be prepared before software development is completed.
- Testing activities (including reviews) should run in parallel to software specification and design.

Seven Testing Principles (IV)

Principle 4: Defect Clustering.

- Find a defect and you will find more defects nearby.
- Defects often appear clustered like mushrooms or cockroaches.
- It is worth screening the same module where one defect was found.
- Testers must be flexible.
- Once a defect is found it is a good idea to reconsider the direction of further testing.
- The location of a defect might be screened at higher detail level, e.g. starting additional tests or modifying existing tests.

Seven Testing Principles (V)

Principle 5: Pesticide Paradox.

Repeating tests under the same conditions is ineffective.

- Each test case should contain a unique combination of input parameters for a signal test objects, otherwise no additional information can be gained.
- If the same tests are repeated over and over again, no new bugs can be found.

Tests must be revised regularly for different code modules.

- It is necessary to repeat a test after changes have been made in the code (bug fixing, new functionality).
- Automating tests can be an advantage if a group of tests cases is used regularly.

Seven Testing Principles (VI)

Principle 6: Testing is content dependent.

- Testing is done differently in different contexts
- Different test objects are tested differently.
  - The engine controller of a car requires tests different than those of an ecommerce application
- Test environmental (test bed) vs. production environment
  - Test take place on an environment other than the production environment. The test environment should be very similar to the production environment.
- There will always be deviations between test environment and the production environment. These deviations impeach the conclusions drawn after testing.

Seven Testing Principles (VII)

Principle 7: Absence of errors fallacy

- Successful testing finds the most serious failure.
- In most cases, testing will find all defects of the system (see principle 2), but the most serious defects should be found.
- This alone does not prove the quality of the software.
- The functionality of the software may not meet the needs and expectations of the users.
- You can not test quality into the product, it must be built in from the very beginning!