Software Engineering 2UCCE501

Module 5

Module 5 Testing & Maintenance

5.1 Testing Concepts: Purpose of Software Testing, Testing Principles, Goals of Testing, Testing aspects: Requirements, Test Scenarios, Test cases, Test scripts/procedures,

5.2 Strategies for Software Testing, Testing Activities: Planning Verification and Validation, Software Inspections, FTR

5.3 Levels of Testing : unit testing, integration testing, regression testing, product testing, acceptance testing and White-Box Testing

5.4 Black-Box Testing: Test case design criteria, Requirement based Testing, Boundary value analysis, Equivalence Class Partitioning

5.5 Object Oriented Testing: Review of OOA and OOD models, class testing, integration testing, validation testing

5.6 Reverse & Reengineering, types of maintenance

Purpose of Software Testing

- Testing is the **process of analyzing a system** or system component **to detect the differences** between specified (**required**) and observed (**existing**) behavior.

- Testing is a **set of activities** that can be **planned in advance** and conducted systematically

11 Principles of Testing

1. All tests should be traceable to customer requirements

- testing proves the presence of errors
- it does not verify that no more bugs exist
- Testing is not a prove that the system is free of errors.
- most severe defects are those that cause the program to fail to meet its requirements.

2. Exhaustive testing is not possible

- An exhaustive test which considers all possible input parameters, their combinations and different pre-conditions can not be accomplished.
- Test are always spot tests , hence efforts must be managed.

3. Tests should be planned long before testing begin

11 Principles of Testing

4. Test early and regularly

- Testing activities should begin as early as possible within the software life cycle.
- Early testing helps detecting errors at an early stage of the development process which simplifies error correction

5. Accumulation of errors

- There is no equal distribution of errors within one test object.
- The place where one error occurs, it’s likely to find some more. The testing process must be flexible and respond to this behavior.

6. Fading effectiveness

- The effectiveness of software testing fades over time.
- If test-cases are only repeated, they do not expose new errors.
- Errors, remaining within untested functions may not be discovered.
- to prevent this effect, test-cases must be altered and reworked time by time.

11 Principles of Testing

7. Testing depends on context

- No two systems are the same and therefore can not be tested the same way.

8. False conclusion: no errors equals usable system

- Error detection and error fixing does not guarantee a usable system matching the users expectations.
- Early integration of users and rapid prototyping prevents unhappy clients and discussions.

9. The Pareto principle applies to software testing

10.Testing should begin “in the small” and progress toward testing “in the large.”

11. To be most effective, testing should be conducted by an independent third party.

11 Principles of Testing

1. All tests should be traceable to customer requirements
2. Exhaustive testing is not possible
3. Tests should be planned long before testing begin
4. Test early and regularly5. Accumulation of errors
5. Accumulation of errors
6. Fading effectiveness
7. Testing depends on context
8. False conclusion: no errors equals usable system
9. The Pareto principle applies to software testing
10. Testing should begin "in the small" and progress toward testing "in the large."
11. To be most effective, testing should be conducted by an independent third party.

Goals of Testing

- The testing process has two distinct goals:

1. **To demonstrate to the developer and the customer that the software meets its requirements.**

- The first goal leads to **validation testing,** where you expect the system to perform correctly using a given set of test cases that reflect the system's expected use.

Goals of Testing

2. To discover situations in which the behavior of the software is incorrect, undesirable, or does not conform to its specification.**(Verification)**

- The second goal leads to defect testing, where the test cases are designed to expose defects.

- The set of activities that ensure that software correctly implements a specific function or algorithm

Testing Concepts

- A **test component** is a part of the system that can be isolated for testing.
- A **fault**, also called **bug or defect**, is a design or coding mistake that may cause abnormal component behavior.
- An **erroneous state** is an indication of a fault during the execution of the system.
- A **failure** is a deviation between the specification and the actual behavior. A **failure is triggered** by one or more **erroneous states**.

Requirements of Testing

1. Testability

- Software testability is simply how easily [a computer program] can be tested.
- There are certainly metrics that could be used to measure testability.

2. Operability

- "The better it works, the more efficiently it can be tested.“
- The system has few bugs (bugs add analysis and reporting overhead to the test process).

3. Observability

- **"What you see is what you test."**
- Distinct output is generated for each input.
- System states and variables are visible or queriable during execution.
- Past system states and variables are visible or queriable (e.g., transaction logs).
- All factors affecting the output are visible.
- Incorrect output is easily identified.
- Internal errors are automatically detected through self testing mechanisms.
- Internal errors are automatically reported.
- Source code is accessible.

Requirements of Testing

4. Controllability.

- "The better we can control the software, the more the testing can be automated and optimized."
- All possible outputs can be generated through some combination of input.
- All code is executable through some combination of input.
- Software and hardware states and variables can be controlled directly by the test engineer.
- Input and output formats are consistent and structured.
- Tests can be conveniently specified, automated, and reproduced.

Requirements of Testing

5. Decomposability

- "By controlling the scope of testing, we can more quickly isolate problems and perform smarter retesting."
- The software system is built from independent modules.
- Software modules can be tested independently.

6. Simplicity.

- **"The less there is to test, the more quickly we can test it.“**
- **Functional simplicity** (e.g., the feature set is the minimum necessary to meet requirements
- **Structural simplicity** (e.g., architecture is modularized to limit the propagation of faults).
- **Code simplicity** (e.g., a coding standard is adopted for ease of inspection and maintenance).

Requirements of Testing

7. Stability

- **"The fewer the changes, the fewer the disruptions to testing."**
    - Changes to the software are infrequent.
    - Changes to the software are controlled.
    - Changes to the software do not invalidate existing tests.
    - The software recovers well from failures.

8. Understandability.

- **"The more information we have, the smarter we will test."**
    - The design is well understood.
    - Dependencies between internal, external, and shared components are well understood.
    - Changes to the design are communicated.
    - Technical documentation is instantly accessible.
    - Technical documentation is well organized.
    - Technical documentation is specific and detailed.
    - Technical documentation is accurate.

Test Case

- A test case is a set of **conditions or variables under which a tester will determine** whether a **system under test** satisfies requirements or works correctly.

- The process of developing test cases can also help find problems in the requirements or design of an application.

5 attributes of a Test Case

1. The **name** of the test case allows the tester to distinguish between different test cases.
   - **For example:** testing a use case Deposit(), call the test case Test_Deposit*.*

2. The **location** attribute describes where the test case can be found.
   - path name or the URL to the executable of the test program and its inputs.

3. **Input (data)** describes the set of input data or commands to be entered by the actor of the test case.
   - The test data, or links to the test data, that are to be used while conducting the test.

Test Case

4. The expected behavior is described by the **oracle** attribute.

5. The **log** is a set of time-stamped correlations of the observed behavior with the expected behavior for various test runs.