CSE375

Software Testing

Course Outcomes

- CO1 :: understand software testing fundamentals and its importance
- CO2 :: differentiate between various testing techniques used in industries
- CO3 :: apply appropriate test strategy for evaluating a system's performance
- CO4 :: test an application and write test case documents
- CO5 :: apply test effort estimation techniques
- CO6 :: understand different software testing standards

- **Text Books:** 1. THE ART OF SOFTWARE TESTING by GLENFORD J. MYERS, TOM BADGETT, COREY, SANDLER, WILEY
- **References:** 1. FOUNDATIONS OF SOFTWARE TESTING ISTQB CERTIFICATION by REX BLACK, DOROTHY GRAHAM, ERIK VAN VEENENDAAL, CENGAGE LEARNING

Testing

- The aim of testing is to identify all defects in a software product.
- However, in practice even after thorough testing:
  - one cannot guarantee that the software is error-free.

Testing

- The input data domain of most software products is very large:
  - it is not practical to test the software exhaustively with each input data value.

Testing

- Testing does however expose many errors:
  - testing provides a practical way of reducing defects in a system
  - increases the users' confidence in a developed system.

Testing

- Testing is an important development phase:
  - requires the maximum effort among all development phases.
- In a typical development organization:
  - maximum number of software engineers can be found to be engaged in testing activities.

Testing

- Many engineers have the wrong impression:
  - testing is a secondary activity
  - it is intellectually not as stimulating as the other development activities, etc.

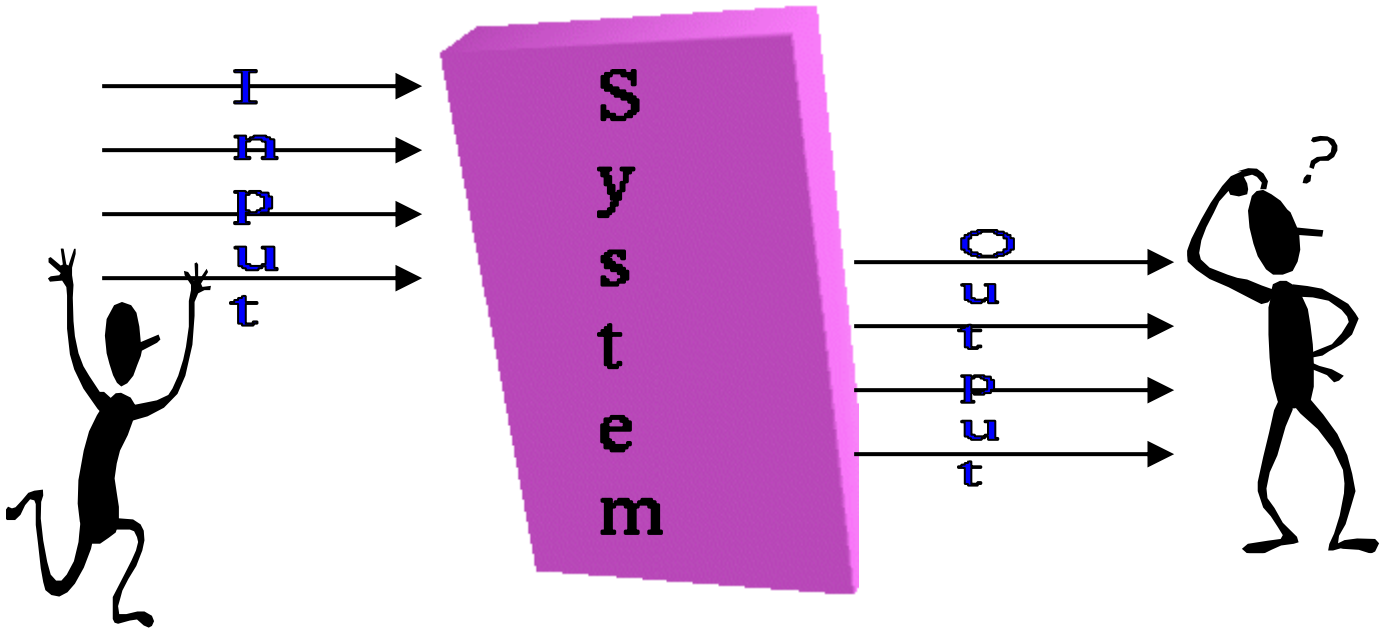
Testing

- Testing a software product is in fact:
  - as much challenging as initial development activities such as specification, design, and coding.
- Also, testing involves a lot of creative thinking.

How do you test a program?

- Input test data to the program.
- Observe the output:
  - Check if the program behaved as expected.

How do you test a system?



How do you test a system?

- If the program does not behave as expected:
  - note the conditions under which it failed.
  - later debug and correct.

Basic Concepts and terminologies

Error, Faults, and Failures

- Error is a mistake committed by the developer team during any of the development phases.
- An error is sometimes referred to as a fault, a bug or a defect.

Basic Concepts and terminologies

Error, Faults, and Failures

- A failure is a manifestation of an error (aka defect or bug).
  - mere presence of an error may not lead to a failure.

Basic Concepts and terminologies

Test cases and Test suites

- A test case is a triplet [I,S,O]
  - I is the data to be input to the system,
  - S is the state of the system at which the data will be input,
  - O is the expected output of the system.

Basic Concepts and terminologies

Test cases and Test suites

- Test a software using a set of carefully designed test cases:
  - the set of all test cases is called the test suite

Verification versus Validation

- Verification is the process of determining:
  - whether output of one phase of development conforms to its previous phase.
- Validation is the process of determining
  - whether a fully developed system conforms to its SRS document.

Verification versus Validation

- Aim of Verification:
  - phase containment of errors
- Aim of validation:
  - final product is error free.

Verification versus Validation

- Verification:
  - are we doing right?
- Validation:
  - have we done right?

Design of Test Cases

- Exhaustive testing of any non-trivial system is impractical:
  - input data domain is extremely large.
- Design an optimal test suite:
  - of reasonable size and
  - uncovers as many errors as possible.

Design of Test Cases

- If test cases are selected randomly:
  - many test cases would not contribute to the significance of the test suite,
  - would not detect errors not already being detected by other test cases in the suite.
- Number of test cases in a randomly selected test suite:
  - not an indication of effectiveness of testing.

Black-box Testing

- Test cases are designed using only functional specification of the software:
  - without any knowledge of the internal structure of the software.
- For this reason, black-box testing is also known as functional testing.

White-box Testing

- Designing white-box test cases:
  - requires knowledge about the internal structure of software.
  - white-box testing is also called structural testing.
  - In this unit we will not study white-box testing.

Levels of Testing

- Software products are tested at three levels:
  - Unit testing
  - Integration testing
  - System testing

Unit testing

- During unit testing, modules are tested in isolation:
  - If all modules were to be tested together:
    - it may not be easy to determine which module has the error.

Unit testing

- Unit testing reduces debugging effort several folds.
  - Programmers carry out unit testing immediately after they complete the coding of a module.

Integration testing

- After different modules of a system have been coded and unit tested:
  - modules are integrated in steps according to an integration plan
  - partially integrated system is tested at each integration step.

System Testing

- System testing involves:
  - validating a fully developed system against its requirements.