Software Testing (Contd.)

Organization of this Lecture:

- **INTEGRATION TSETING**
- **SYSTEM TESTING**
- **Performance testing**
- **Test summary report**
- **Summary**

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.

Testing

- **Unit testing:**
  - test the functionalities of a single module or function.
- **Integration testing:**
  - test the interfaces among the modules.
- **System testing:**
  - test the fully integrated system against its functional and non-functional requirements.

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:
  - validate a fully developed system against its requirements.

Integration Testing

- Develop the integration plan by examining the structure chart :
  - big bang approach
  - top-down approach
  - bottom-up approach
  - mixed approach

Big bang Integration Testing

- Big bang approach is the simplest integration testing approach:
  - all the modules are simply put together and tested.
  - this technique is used only for very small systems.

Big bang Integration Testing

- Main problems with this approach:
  - if an error is found:
    - it is very difficult to localize the error
    - the error may potentially belong to any of the modules being integrated.
  - debugging errors found during big bang integration testing are very expensive to fix.

Bottom-up Integration Testing

- Integrate and test the bottom level modules first.
- A disadvantage of bottom-up testing:
  - when the system is made up of a large number of small subsystems.
    - This extreme case corresponds to the big bang approach.

Top-down integration testing

- Top-down integration testing starts with the main routine:
  - and one or two subordinate routines in the system.
- After the top-level 'skeleton' has been tested:
  - immediate subordinate modules of the 'skeleton' are combined with it and tested.

Mixed integration testing

- Mixed (or sandwiched) integration testing:
  - uses both top-down and bottom-up testing approaches.
  - Most common approach

Integration Testing

- In top-down approach:
  - testing waits till all top-level modules are coded and unit tested.
- In bottom-up approach:
  - testing can start only after bottom level modules are ready.

Phased versus Incremental Integration Testing

- Integration can be incremental or phased.
- In incremental integration testing,
  - only one new module is added to the partial system each time.

Phased versus Incremental Integration Testing

- In phased integration,
  - a group of related modules are added to the partially integrated system each time.
- Big-bang testing:
  - a degenerate case of the phased integration testing.

Phased versus Incremental Integration Testing

- Phased integration requires less number of integration steps:
  - compared to the incremental integration approach.
- However, when failures are detected,
  - it is easier to debug if using incremental testing
    - since errors are very likely to be in the newly integrated module.

System Testing

- System tests are designed to validate a fully developed system:
  - to assure that it meets its requirements.

System Testing

- There are essentially three main kinds of system testing:
  - Alpha Testing
  - Beta Testing
  - Acceptance Testing

Alpha testing

- System testing is carried out
  - by the test team within the developing organization.

Beta Testing

- Beta testing is the system testing:
  - performed by a select group of friendly customers.

Acceptance Testing

- Acceptance testing is the system testing performed by the customer
  - to determine whether he should accept the delivery of the system.

System Testing

- During system testing, in addition to functional tests:
  - performance tests are performed.

Performance Testing

- Addresses non-functional requirements.
  - May sometimes involve testing hardware and software together.
  - There are several categories of performance testing.

Stress testing

- Evaluates system performance
  - when stressed for short periods of time.
- Stress testing
  - also known as endurance testing.

Stress testing

- Stress tests are black box tests:
  - designed to impose a range of abnormal and even illegal input conditions
  - so as to stress the capabilities of the software.
  - Input data volume, input data rate, processing time, utilization of memory, etc. are tested beyond the designed capacity.

Stress Testing

- If the requirements is to handle a specified number of users, or devices:
  - stress testing evaluates system performance when all users or devices are busy simultaneously.

Stress Testing

- If an operating system is supposed to support 15 multiprogrammed jobs,
  - the system is stressed by attempting to run 15 or more jobs simultaneously.
- A real-time system might be tested
  - to determine the effect of simultaneous arrival of several high-priority interrupts.

Stress Testing

- Stress testing usually involves an element of time or size,
  - such as the number of records transferred per unit time,
  - the maximum number of users active at any time, input data size, etc.
- Therefore stress testing may not be applicable to many types of systems.

How many errors are still remaining?

- Seed the code with some known errors:
  - artificial errors are introduced into the program.
  - Check how many of the seeded errors are detected during testing.

Error Seeding

- Let:
  - N be the total number of errors in the system
  - n of these errors be found by testing.
  - S be the total number of seeded errors,
  - s of the seeded errors be found during testing.

Error Seeding

- $n/N = s/S$
- $N = S\ n/s$
- remaining defects:

$$N - n = n\ ((S - s)/ s)$$

Example

- 100 errors were introduced.
- 90 of these errors were found during testing
- 50 other errors were also found.
- Remaining errors= 50 (100-90)/90 = 6

Error Seeding

- The kind of seeded errors should match closely with existing errors:
  - However, it is difficult to predict the types of errors that exist.
- Categories of remaining errors:
  - can be estimated by analyzing historical data from similar projects.

Volume Testing

- Addresses handling large amounts of data in the system:
  - whether data structures (e.g. queues, stacks, arrays, etc.) are large enough to handle all possible situations
  - Fields, records, and files are stressed to check if their size can accommodate all possible data volumes.

Configuration Testing

- Analyze system behavior:
  - in various hardware and software configurations specified in the requirements
  - sometimes systems are built in various configurations for different users
  - for instance, a minimal system may serve a single user,
    - other configurations for additional users.

Compatibility Testing

- These tests are needed when the system interfaces with other systems:
  - check whether the interface functions as required.

Compatibility testing Example

- If a system is to communicate with a large database system to retrieve information:
  - a compatibility test examines speed and accuracy of retrieval.

Recovery Testing

- These tests check response to:
  - presence of faults or to the loss of data, power, devices, or services
  - subject system to loss of resources
    - check if the system recovers properly.

Maintenance Testing

- Diagnostic tools and procedures:
  - help find source of problems.
  - It may be required to supply
    - memory maps
    - diagnostic programs
    - traces of transactions,
    - circuit diagrams, etc.

Maintenance Testing

- Verify that:
  - all required artifacts for maintenance exist
  - they function properly

Documentation tests

- Check that required documents exist and are consistent:
  - user guides,
  - maintenance guides,
  - technical documents

Documentation tests

- Sometimes requirements specify:
  - format and audience of specific documents
  - documents are evaluated for compliance

Usability tests

- All aspects of user interfaces are tested:
  - Display screens
  - messages
  - report formats
  - navigation and selection problems

Environmental test

- These tests check the system's ability to perform at the installation site.
- Requirements might include tolerance for
  - heat
  - humidity
  - chemical presence
  - portability
  - electrical or magnetic fields
  - disruption of power, etc.

Test Summary Report

- Generated towards the end of testing phase.
- Covers each subsystem:
  - a summary of tests which have been applied to the subsystem.

Test Summary Report

- Specifies:
  - how many tests have been applied to a subsystem,
  - how many tests have been successful,
  - how many have been unsuccessful, and the degree to which they have been unsuccessful,
    - e.g. whether a test was an outright failure
    - or whether some expected results of the test were actually observed.

Regression Testing

- Does not belong to either unit test, integration test, or system test.
  - In stead, it is a separate dimension to these three forms of testing.

Regression testing

- Regression testing is the running of test suite:
  - after each change to the system or after each bug fix
  - ensures that no new bug has been introduced due to the change or the bug fix.

Regression testing

- Regression tests assure:
  - the new system's performance is at least as good as the old system
  - always used during phased system development.