What is Software Testing

- Testing is the examination of a software system through sampling
  - A process of proving software quality
    - To demonstrate the software works
  - A process of improving software quality
    - To find errors and faults
  - A development activity to ensure software quality
    - To prevent errors

Static Testing vs. Dynamic Testing

- Static Testing
  - Review and inspect software documents and readable code without execute the program
- Dynamic testing
  - Execute the program and observe its dynamic behavior and input / output
Definitions of Terminology

- Fault:
  - A defect in software system
- Failure:
  - A case when the software’s output is incorrect as the result of an execution
- Error:
  - A mistake that made by the software developer
- Bug:
  - A defect in program code

Current Practice of Software Production

- Software testing is indispensable to all software development
- Software quality can be achieved to certain extent through systematic application of testing methods
- Software testing takes about 50% of development effort and resources
- Systematic application of testing methods through effective uses of software tools becomes the trend in software production

Software Testing Technique

- **White Box Testing**, or **Structure Testing** is derived directly from the implementation of a module and able to test all the implemented code
- **Black Box Testing**, or **Functional Testing** is able to test any functionality is missing from the implementation.

White Box Testing Technique

- White Box Testing of software is predicated on close examination of procedural detail.
- Logical paths through the software are tested by providing test cases that exercise specific sets of conditions and / or loops.
- The status of the program may be examined at various points to determine if the expected or asserted status corresponds to the actual status.
**Process of White Box Testing**

- Tests are derived from an examination of the source code for the modules of the program.
- These are fed as input to the implementation, and the execution traces are used to determine if there is sufficient coverage of the program source code.

**Black Box Testing**

- Black box testing attempts to find errors in the following categories:
  - Incorrect or missing functions
  - Interface errors
  - Errors in data structures or external databases access
  - Performance errors
  - Initialization and termination errors.

**Process of Black Box Testing**

**Software Quality Attributes**

- Correctness
  - The consistency between program code and specification
- Reliability
  - The probability that a software system provides required functions in a given period of time operation in a specific environment
- Safety
  - The property that the software will not cause any loss of human life and substantial environmental damage
- Maintainability
  - The easiness that a software system to be maintained, to enhance functions and to adapt to changed environment.
Validation and Verification

Validation
Whether the developed system is what the user wanted.
- To prove that a system satisfies users’ requirements
- Users’ requirements may or may not be elicited
- Users’ requirements may or may not be documented accurately and completely
- Whether or not users’ requirements are satisfied cannot be formally proved
- Users’ requirements change frequently

Verification
Whether the developed system is what have been specified
- To prove that a program is consistent with respect to the specification.
- A program formally proved to be consistent with respect to a specification can still fail to satisfy users’ requirements
- Consistency between software artefacts can be formally defined and proved
- Program can be derived from a specification

The Context of Software Testing

The V-Model

Requirements analysis
- Requirement definition
- Functional specification
- Architectural design
- Architectural design spec
- Detailed design
- Module design spec
- Debugged modules
- Linked between phases
- Consistency check

Operation and maintenance
- Validated system
- Acceptance test and release
- System software integration
- Integrated software
- Debugged modules
- Implementation

Testing at Requirements Analysis Stage

- The verification of the internal consistency of the requirements definition
- The validation of the specified requirements with respect to what the customer actually expects
- The analysis of the feasibility of the requirements
- Preparation for dynamic testing
  - The derivation of verification requirements, which are converted into system tests and acceptance tests at later stages of the development
  - The development of a test plan
Static Testing Methods: Formal Review

- **Applicability:**
  - Designs, program codes, and various documents produced at various stages of development

- **The purpose:**
  - To analyse internal consistency, satisfaction of requirements, and suitability for implementation

- **Technique: Walk-through**
  - Test data are selected and the software is simulated manually.
  - The test data are "walked through" the system, with intermediate results kept on a blackboard or a sheet of paper

Key Issues in Formal Review

- Keep the test data simple
- Encourage discussion, not just to complete the simulation
- Most errors are discovered by questioning the developer's decisions, rather than by examining the test data

Example: Walk-Through

```
assignManager() /assignStaff()

/assignManager()

/commissioned

Authorized(authorizationCode)
[contract signed]
/setCampaignActive()

Active

campaignCompleted()
/prepareFinalStatement()

Completed

paymentReceived()
[paymentDue <= zero]

 Paid

archiveCampaign() /unassignStaff() /unassignManager()
```

Static Testing Methods: Inspections

- A step-by-step reading of the software engineering product, with each step checked against a predetermined list of criteria, called check list.
- These criteria usually include checks for historically common errors, adherence to programming standards, and consistency with program specifications.
- Inspection requires a team of testers including the software developer.
- The developer narrates the reading of the product and finds many errors just by the simple review act of reading aloud.
- Other errors are determined as a result of discussion with team members and by applying the checklist.
**Test Plan**

- A test plan should address the following issues:
  - The organizational responsibilities for the various tasks in the testing programme;
  - The methods that are to be used to review documents associated with the test process;
  - The general testing strategy, including selection of test methods and test quality evaluation criteria;
  - The test schedule, i.e. a list of tests that are to be carried out together with the expected times that they will be executed.

**Testing at Functional Specification Stage**

- Validation and Verification activities:
  - The verification of the functional specification against the requirements definition
  - The verification of the internal consistency of the functional specification and its implementability
  - The preparation for dynamic testing
    - The generation of functional test data which usually will form the core of the final test set;
    - Further development of test plan if it has not been completed during the requirements analysis phase.

**Modelling Techniques**

- Animation of formal specifications
  - To derive the required output by mechanically evaluation of the formal specification.
  - Applicable only if there is a formal specification
  - Need tool support when the specification is complicated

- Prototyping software systems
  - The primary aim is to communicate the specifier's interpretation of the requirements to the customer in order to locate misunderstandings.
  - A prototype can also be used to produce test cases for use at later phases in development process.

**Symbolic Evaluation of Program**

- A program is "executed" over symbols rather than actual values.
  - Expression
    - Substitute the symbolic value of each variable into the expression
    - Simplify the result expression by application of algebraic laws
  - Assignment
    - The resulting symbolic value of the right-hand side expression becomes the new symbolic value of the variable on the left-hand side
  - Conditional branching
    - The predicate becomes a constraint on the symbolic value
  - Output
    - The symbolic value of the variables is the result
Testing at Top Level Design Stage

- Verification activity:
  - The verification of the architectural design against functional specification
  - The verification of the internal consistency of the design
  - Preparation for dynamic testing:
    - The generation of test cases:
      - to exercise the functions introduced during the design
      - based upon the structure of the system
  - Acquire/develop validation support tools
  - The construction of test procedures
  - The development of a test coverage matrix

Testing at Detailed Design Stage

- Verification activity:
  - The verification of the detailed design and module specification against the top level design
  - The verification of the internal consistency of each module specification and the consistency between modules
  - Preparation of dynamic testing
    - The expansion of the test set for checking module interfaces and for checking design functions of each module, etc.;
    - The construction of the test procedure for unit testing.

Testing at Implementation Stage

- Unit Testing:
  - To verify the implementation of the module against the module specification and design
  - The actual execution of the code produced during this phase
  - It should follow the unit test procedure specified in the test plan
  - Need additional test cases

Basic Elements of Dynamic Testing

- The program under test
  - Must be executable
  - May need additional code to make it executable
- The test case
  - The input data to run the program
  - The expected output / dynamic behaviour
- The observation
  - The aspects of behaviour to be observed
  - Means of observation
- The analysis of test results
  - The correctness of behaviour
  - The adequacy
The Process of Dynamic Testing

- Preparing testing environment
- How to select test cases?
- How to observe & record program behaviour?
- Select a test case
- Execute program on one test data
- When to stop?
- Check if testing is adequate
- Is the program correct?
- Analyse test results

May need to write a piece of code to make a fragment of program code executable

Some Notions of Software Testing

- Test data selection criteria:
  - Rules that determine how to select test cases
- Test observation scheme:
  - Rules that determine how to observe and record system’s behaviour
  - Software instrumentation – the code inserted into program to implement an observation scheme
- Test data adequacy criteria:
  - Rules that determine when testing can stop
  - Rules that measure how well the testing has been done
- Test oracle
  - It determines whether program’s behaviour on an input is correct
  - It can be another program, a formal specification, or a human being, such as a tester or domain expert.

Testing at Software Integration Stage

- Integration Testing
  - The primary aim is to verify the design
  - It should follow the test plan
  - The program modules should be integrated progressively according to a specified strategy
  - The focus is the correctness of information passing through the interface between modules

Top-Down Integration Strategy
**Bottom-Up Integration Strategy**

![Diagram showing the Bottom-Up Integration Strategy]

**Test Driver, Stub and Test Script**

- **Test Driver**
  - A piece of code written to call the piece of code, such as a procedure or a module of program, so that it can be executed.

- **Stub**
  - A piece of program code written to replace the modules or procedures that the program under test depends on and calls so that it can be executed.

- **Test harness**
  - Both stub and test driver are code specially written for testing purposes. They are called the test harness.

- **Test script**
  - Some test tools can support the generation of such code, but the tester may need to describe the environment in a particular language. Such description is usually called test script.

**System Test Stage**

- Executing the test procedures associated with the verification tests
  - Domain analysis
  - System functional testing
  - Random testing

- Measuring the test adequacy

**Random Testing**

- Random testing use test data selected at random according to certain probability distribution over the input space

- Representative random testing
  - The probability distribution use to sample the input data represents the operation of the software, e.g. data obtained in the operation of the old system or similar systems

- Non-representative random testing
  - The probability distribution has no-relationship with the operation of the system
Advantages & Disadvantages

Advantages
- Reliability can be estimated especially when representative random testing is used
- Low cost in the selection of test cases, which can be automated to a great extent
- Can achieve a high fault detection ability

Disadvantages
- Less confidence can be obtained from the testing
- Still need to validate the correctness of output, which may be more difficult than deliberately selected test cases.

Acceptance Test and Release

Objective:
- To deliver the validated system to customer

Main activity:
- To demonstrate that the system is acceptable to the customer

It is usually a process of executing the test procedures associated to a subset of the verification requirements agreed by both the customer and the developer as being an adequate representation of user requirements.

Testing During Operation & Maintenance

Main activities
- Operation of the system
- Maintenance
  - Modifications to correct errors
  - Modifications to enhance capability and functionality
  - Modifications to fit into new environment

Verification activities:
- The testing after each modification
  - To check that the errors are corrected
  - To ensure that there is no bad effects of the modification

Regression Testing

Regression testing is the process of re-executing a subset of the test cases that are related to the modified modules of the program code and the effected functions of the system.

The test coverage matrix will be found useful to identify which subset of the test cases should be re-executed.

The output recorded from previous test execution will be useful to check the regression test output.

Interaction between the tester and the software system can also be recorded and replayed to automate regression testing process.