Basic Concepts and Preliminaries

Course Code: CSC4133 Course Title: Software Quality and Testing

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**2**	**Week No:**	**1**	**Semester:**	
Lecturer:	*Prof. Dr. kamruddin Nur,* *kamruddin@aiub.edu*				

Lecture Outline

- Meeting People's Quality Expectations
- Verification & Validation
- SQE (Software Quality Engineering) process
- Major SQE activities
- Software Quality
- Quality Models
- Role of Testing
- Error, Fault, Failure, and Defect
- The Notion of Software Reliability
- The Objectives of Testing
- The Concept of Complete Testing
- Testing Levels
- Testing Techniques
- Manual Testing vs. Automated Testing

Meeting People's Quality Expectations

General Expectations:

➔ "**good**" software quality

Meeting People's Quality Expectations

- **People: Consumers vs. Producers**
  - ➔Quality expectations by consumers
  - ➔To be satisfied by producers through software quality engineering (SQE)
- **Deliver software system that…**
  - ➔Does what it is *supposed to do*
    - –needs to be "validated"
  - ➔Does the things *correctly*
    - –needs to be "verified"

Validation & Verification (V&V)

- **Validation**
  - Validation is a process that ensures the software product meets the customer requirements
  - Software systems must do what they are supposed to do; they must *do the right things*
  - Building the correct product
- **Verification**
  - Verification is a process that ensures the software product works properly
  - Software systems must perform the specific tasks correctly; they must *do the things right*
  - Building the product correctly

Meeting Quality Expectations

- Difficulties in achieving good quality:
  - Size: MLOC products common
  - Complexity
  - Environmental stress/constraints
  - flexibility/adaptability expected
- Other difficulties/factors:
  - product type
  - cost and market conditions
- No “silver bullet”, but…
  - **SQE** (Software Quality Engineering) helps

Main Tasks for SQE

- The tasks for software QA and quality engineering are to ensure software quality through the related validation and verification activities. These activities need to be carried out by the people and organizations responsible for developing and supporting these software systems in an overall quality engineering process:
  - Quality planning
  - Execution of selected QA or software validation & verification activities
  - Measurement & Analysis to provide convincing evidence to demonstrate software quality to all parties involved
- Customers and users need to have the assurance that their quality expectations are satisfied by the delivered software systems.

Major SQE Activities

- Testing: remove defect & ensure quality
- Other QA *alternatives* to testing
  - –Inspection/Technical Review/Review/Walkthrough etc.
  - – Defect Prevention
  - – Formal Verification
  - – Fault Tolerance

Software Quality

- Five Views of Software Quality (by Kitchenham & Pfleeger)
  - Transcendental view
  - User's view
  - Manufacturing view
  - Product view
  - Value-based view
- Software Quality in terms of **quality factors** and **quality criteria** (by McCall, Richards, & Walters)
  - A ***quality factor*** represents behavioral characteristic of a system.
    - Examples: correctness, reliability, efficiency, testability, maintainability, reusability
  - A ***quality criterion*** is an attribute of a quality factor that is related to software development. Example: modularity is an attribute of software architecture

Quality Models

- **ISO 9126** (International Organization for Standardization)
- **CMM** (Capability Maturity Model)/
  CMMI (Capability Maturity Model Integration)
- **TPI** (Test Process Improvement)
- **TMM** (Test Maturity Model)

Role of Testing

- Software testing is one of the most important activities of SQA.
- **Software quality assessment divide into two categories**:
  - **Static analysis**
    - It examines the code/document and reasons over all behaviors that might arise during run time
      - Examples: Code review, inspection, and algorithm analysis
  - **Dynamic analysis**
    - *Actual program execution* to expose possible program failure
    - One observe some representative program behavior, and reach conclusion about the quality of the system
- Static and Dynamic Analysis are ***complementary*** in nature.
- Focus is to combine the strengths of both approaches.

Error, Fault, Failure, and Defect

- **Error:**
  - A human action that produces an incorrect result
  - Missing or incorrect human action resulting in certain fault(s) being injected into a software
- **Fault:**
  - An incorrect step, process, or data definition in a computer program
  - An underlying condition within a software that causes certain failure(s) to occur

Error, Fault, Failure, and Defect

- **Failure:**
  - The inability of a system or component to perform its required functions within specified performance requirements
  - A behavioural deviation from the user requirement or the product specification
- **Defect:**
  - Failures, faults, and errors are collectively referred to as defects.
- Note: Software problems or defects, are also commonly referred to as **"bugs"**

The Notion of Software Reliability

- It is defined as the *probability of failure-free operation of a software system for a specified time in a specified environment.*
- It can be estimated via *random testing.*
- Test data must be drawn from the input distribution to closely resemble the future usage of the system.
  - Future usage pattern of a system is described in a form called *operational profile(OP).*

The Objectives of Testing

- Main objective: detecting defects/bugs
- Reduce the risk of failures
- Reduce the cost of testing

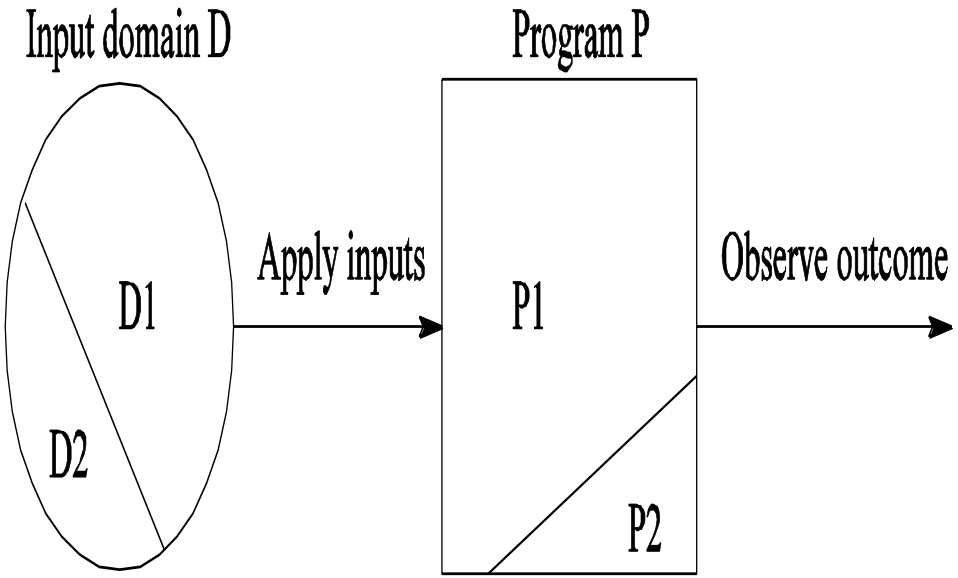
The Concept of Complete Testing

- **Complete/Exhaustive testing means –**
  - – Testing a software with all sorts of inputs (valid and invalid) under all execution environments
  - – “There are no undisclosed faults at the end of test phase”
  - – Complete testing is impossible/impractical

The Concept of Complete Testing

- Complete testing is nearly impossible for most of the systems, Because..
  - The domain of possible inputs of a program is too large (valid inputs and invalid inputs)
  - The design issues may be too complex to completely test
  - It may not be possible to create all possible execution environments of the system

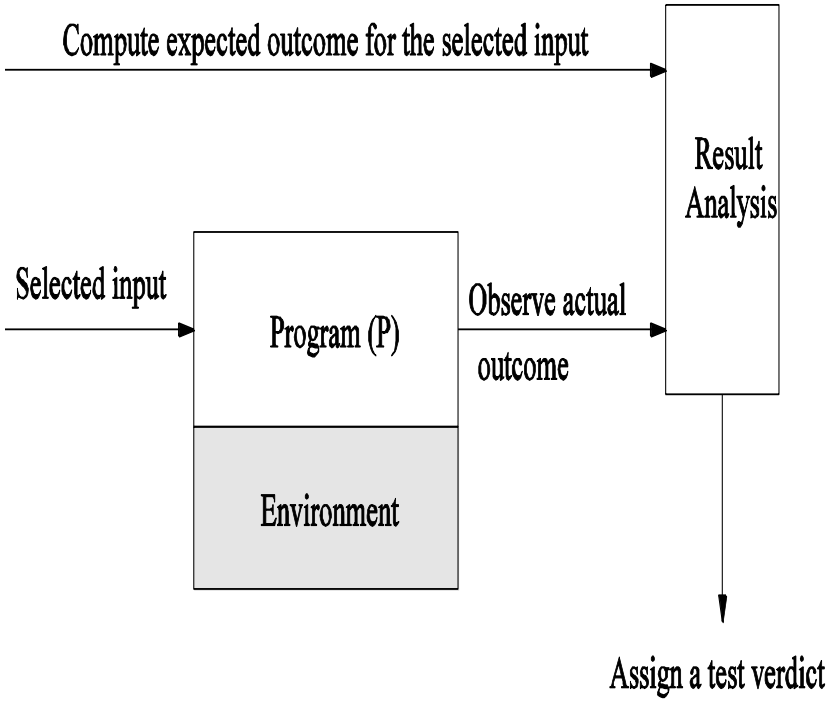
The Central Issue in Testing



- Divide the input domain D into D1 and D2
- Select a subset D1 of D to test program P
- It is possible that D1 exercise only a part P1 of P

Fig. A subset of the input domain exercising a subset of the program behavior

Testing Activities



- Identify the objective to be tested
- Select inputs
- Compute the expected outcome
- Set up the execution environment of the program
- Execute the program
- Analyze the test results

Testing Levels

1. **Unit testing**
   - Testing of one individual unit
   - Individual program units, such as procedure, methods in isolation
2. **Integration testing**
   - Testing of two or more units to identify interface defects
   - Modules are assembled to construct larger subsystem and tested
3. **System testing**
   - Testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements.
   - Includes wide spectrum of testing such as functionality, performance
4. **Acceptance testing**
   - Formal testing conducted to determine whether a system satisfies its acceptance criteria. Performed by customers/end users (preferably).

Testing Levels

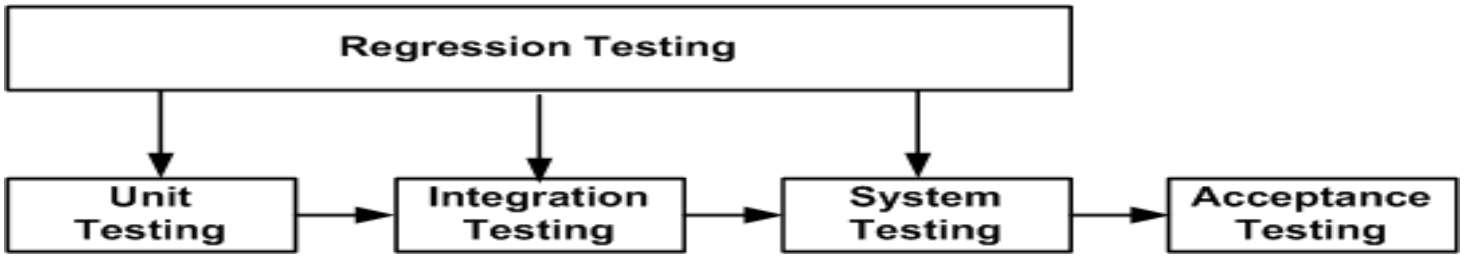


Figure : Regression testing at different software testing levels

- **Regression Testing:**
  - Regression testing is the re-testing of a system or component of a system to verify that modifications have not introduced any new defects and that the system/component still complies with its specified requirements.
  - New test cases are not designed
  - Tests are selected, prioritized and executed
  - To ensure that nothing is broken in the new version of the software

Testing Techniques

- Basically two categories –
  1. White-Box Testing
  2. Black-Box Testing

White-Box vs. Black-Box Testing

White-box testing:

- Internal implementations are tested
- View components as transparent
- Based on knowledge of the internal logic
- Done by programmers (usually)

Black-box testing:

- External functionalities and behavior are tested
- View components as opaque
- Based on requirements and functionality
- Without any knowledge of internal design, code or language
- Done by professional testers (usually)

White-Box Testing

- **White-box testing** *a.k.a.* **structural testing, clear-box testing, glass-box testing, transparent-box testing**
- Examines source code with focus on:
- Control flow
- Data flow
- Control flow refers to flow of control from one instruction to another
- Data flow refers to propagation of values from one variable or constant to another variable
- It is applied to individual units of a program
- Software **developers (programmers)** perform structural testing on the individual program units they write

Black-Box Testing

- Black-box testing a.k.a. **functional testing, behavioral testing**
- Examines the program that is accessible from outside
- Applies the input to a program and observe the externally visible outcome
- It is applied to both an entire program as well as to individual program units
- It is performed at the external interface level of a system
- It is conducted by a separate software quality assurance group(preferred)

Manual Testing vs. Automated Testing

- Manual Testing:
- Oldest and most rigorous type of software testing
- Requires a tester to perform manual test operations
  - Hard to repeat
  - Not always reliable
  - Costly
    - time consuming
    - labor intensive

Automated Testing

- Automated Testing:
- Testing employing software tools
- Execute tests without manual intervention
  - Fast
  - Repeatable
  - Reliable
  - Reusable
  - Programmable
  - Saves time

Books

1. Software Quality Engineering: Testing, Quality Assurance and Quantifiable Improvement - Jeff Tian

2. Software Testing And Quality Assurance – Theory and Practice - Kshirasagar Naik & Priyadarshi Tripathy

References

- *Software Quality Assurance: From Theory to Implementation*, by Daniel Galin
- *The Art of Software Testing*, by Glenford J. Myers, Corey Sandler and Tom Badgett
- *Software Testing Fundamentals: Methods and Metrics* by Marnie L. Hutcheson