Software Quality Assurance
Lecture 2

Basic Concepts and Preliminaries

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"

Verification and Validation

- **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.

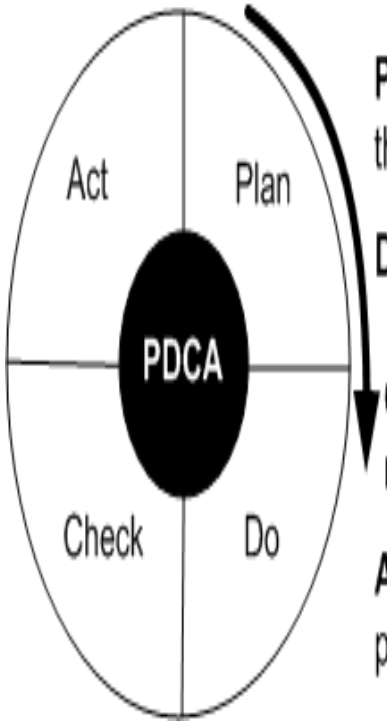
SQE as an Answer

- Major SQE activities:
  - Testing: remove defect & ensure quality
  - Other QA *alternatives* to testing
    - Inspection/Review/Walkthrough
    - Defect Prevention
    - Formal Verification
    - Fault Tolerance

The Quality Revolution

- Started in Japan by Deming, Juran, and Ishikawa during 1940s
- In 1950s, Deming introduced statistical quality control to Japanese engineers
- Statistical quality control (SQC) is a discipline based on measurement and statistics
  - SQC methods use seven basic quality management tool
    - Pareto analysis, Trend Chart, Flow chart, Histogram, Scatter diagram, Control chart, Cause and effect diagram
- "**Lean principle**" was developed by Taiichi Ohno of Toyota
  - "A systematic approach to identifying and eliminating waste through continuous improvement, flowing the product at the pull of the customer in pursuit of perfection."

The Quality Revolution



Plan – Establish the objective and process to deliver the results.

Do – Implement the plan and measure its performance.

Check – Assess the measurements and report the results to decision makers.

Act – Decide on changes needed to improve the process.

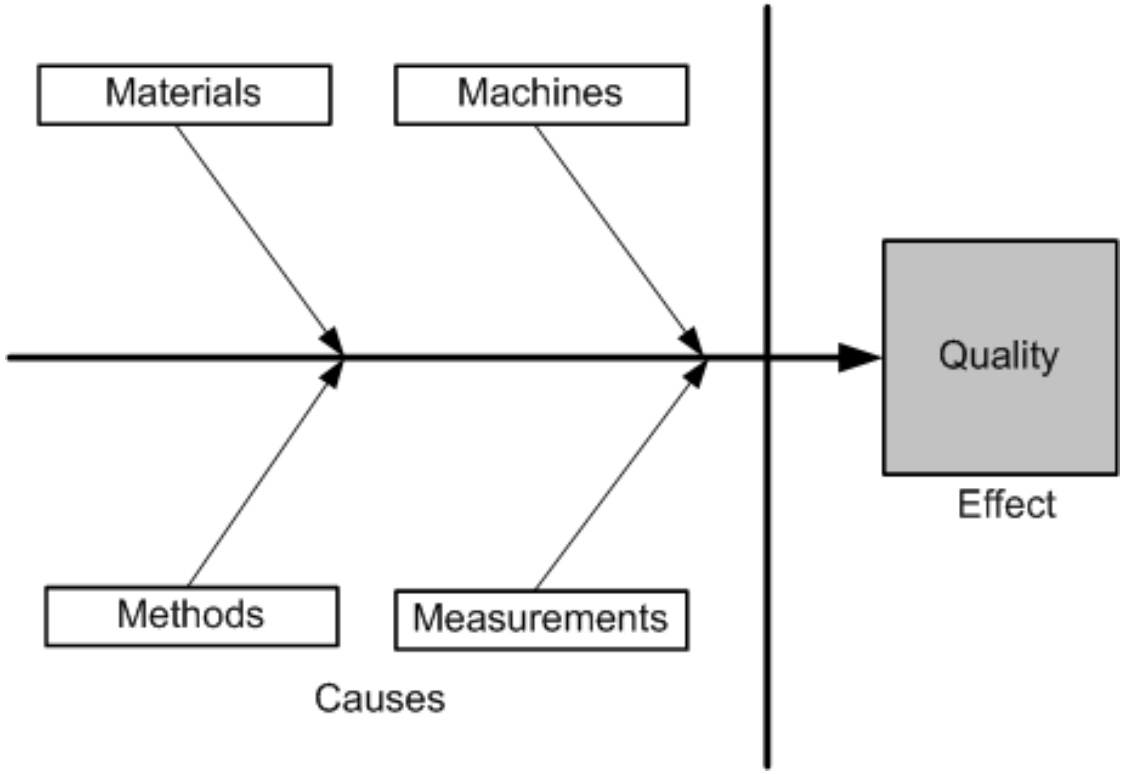
The Shewhart cycle

- **Deming** introduced Shewhart's PDCA cycle to Japanese researchers
- It illustrate the activity sequence:
  - Setting goals
  - Assigning them to measurable milestones
  - Assessing the progress against the milestones
  - Take action to improve the process in the next cycle

The Quality Revolution

- In 1954, Juran spurred the move from SQC to TQC (Total Quality Control).
- Key Elements of TQC:
  - Quality comes first, not short-term profits
  - The customer comes first, not the producer
  - Decisions are based on facts and data
  - Management is participatory and respectful of all employees
  - Management is driven by cross-functional committees
- An innovative methodology developed by Ishikawa called *cause-and-effect diagram*.

Ishikawa diagram



The Quality Revolution

- National Broadcasting Corporation (NBC) of United States broadcast a documentary:
  - "If Japan Can ... Why Can't We?" on June 24th, 1980
- Leaders in United States started emphasizing on quality
- In 1987 Malcolm Baldrige National Quality Award was introduced in U.S.A
  - Similar to the Deming prize in Japan
- In Baldrige National Award the quality is viewed as:
  - Something defined by the customer
- In Deming prize, the quality is viewed as:
  - Something defined by the producer by conformance to specifications

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**
- **TPI** (Test Process Improvement)
- **TMM** (Test Maturity Model)

Role of Testing

- Software testing is one of the most important activities of QA
- 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
- **Failure**
  - The inability of a system or component to perform its required functions within specified performance requirements
  - A behavioral deviation from the user requirement or the product specification
- **Defect**
  - *Failures*, *faults*, and *errors* are collectively referred to as ***defects*** in literature.
- **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

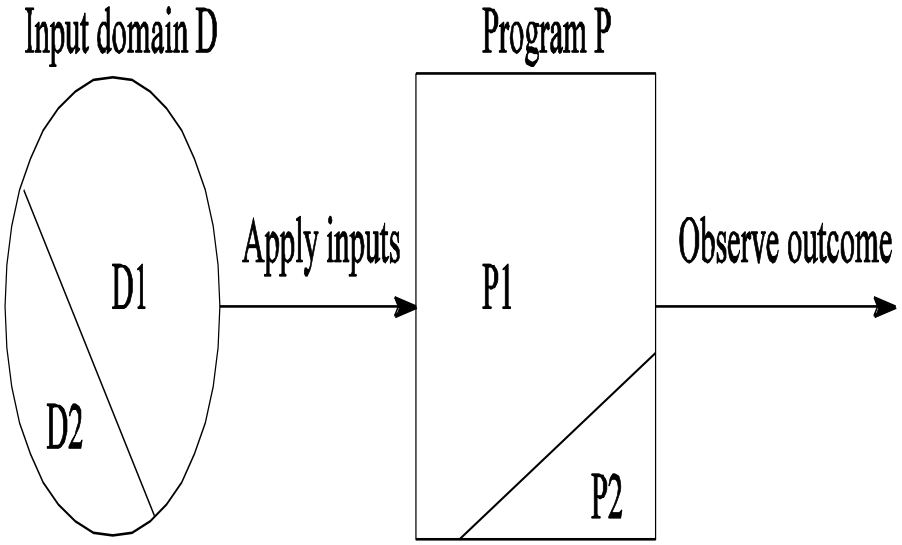
- Main objective: detecting bugs
- It does work
- It does not work
- Reduce the risk of failures
- Reduce the cost of testing

The Concept of Complete Testing

- Complete/Exhaustive testing means –

  "There are no undisclosed faults at the end of test phase"
- Complete testing is nearly impossible for most of the systems, Because..
  - The domain of possible inputs of a program is too large
    - Valid inputs
    - 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

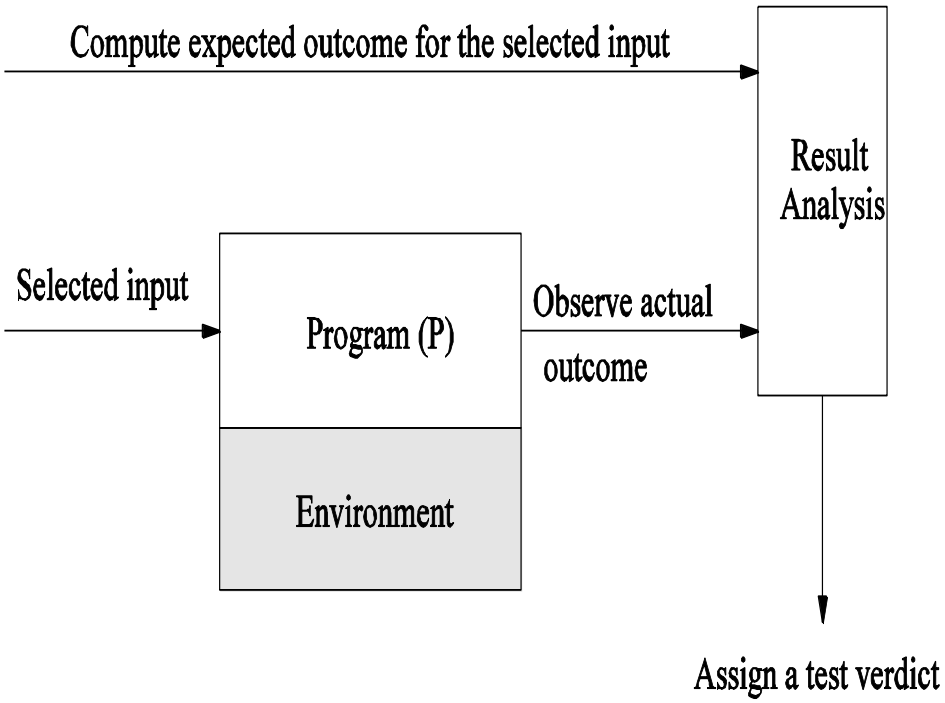
The Central Issue in Testing



- 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
   - Individual program units, such as procedure, methods in isolation
2. Integration testing
   - Modules are assembled to construct larger subsystem and tested
3. System testing
   - Includes wide spectrum of testing such as functionality, performance
4. Acceptance testing
   - Customer's expectations from the system

Testing Levels

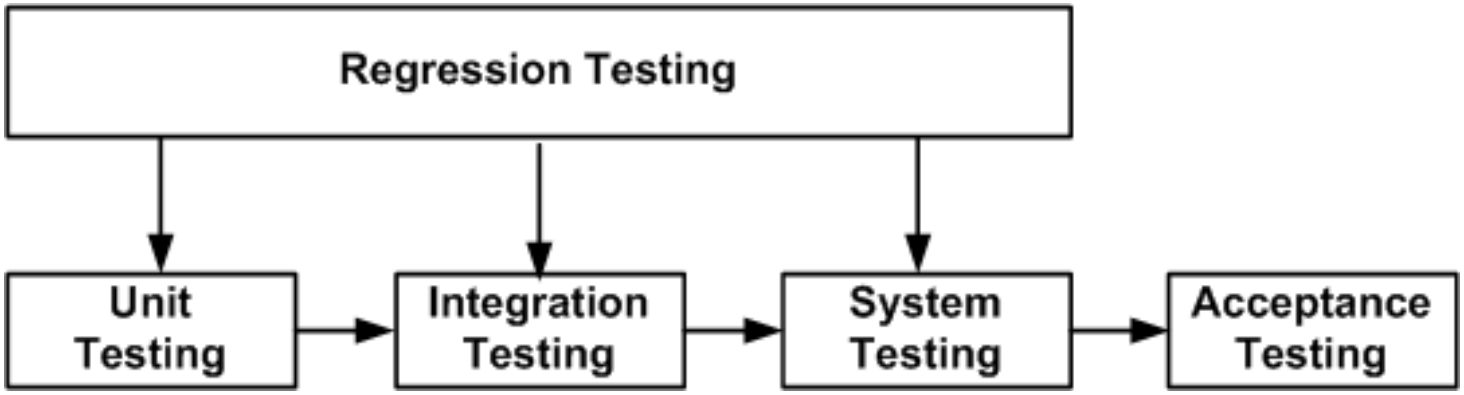


Figure : Regression testing at different software testing levels

- Regression Testing:
  - 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 and Black-Box Testing

White-box testing:

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

Black-box testing:

- View components as opaque
- Based on requirements and functionality
- Without any knowledge of internal design, code or language.

White-Box Testing

- White-box testing a.k.a. **structural 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 perform structural testing on the individual program units they write

Black-Box Testing

- Black-box testing a.k.a. **functional 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

- Software Testing –
  - Manual Testing
  - Automated Testing

Manual 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