CHAPTER – 1
BASIC CONCEPTS AND PRELIMINARIES

OUTLINE OF CHAPTER

- The Quality Revolution
- Software Quality
- Role of Testing
- Verification and Validation
- Failure, Error, Fault and Defect
- The Notion of Software Reliability
- The Objectives of Testing
- Testing Activities
- Testing Level
- White-box and Black-box Testing
- Test Planning and Design
- Monitoring and Measuring Test Execution
- Test Tools and Automation
- Test Team Organization and Management

HOW TO MEASURE QUALITY?

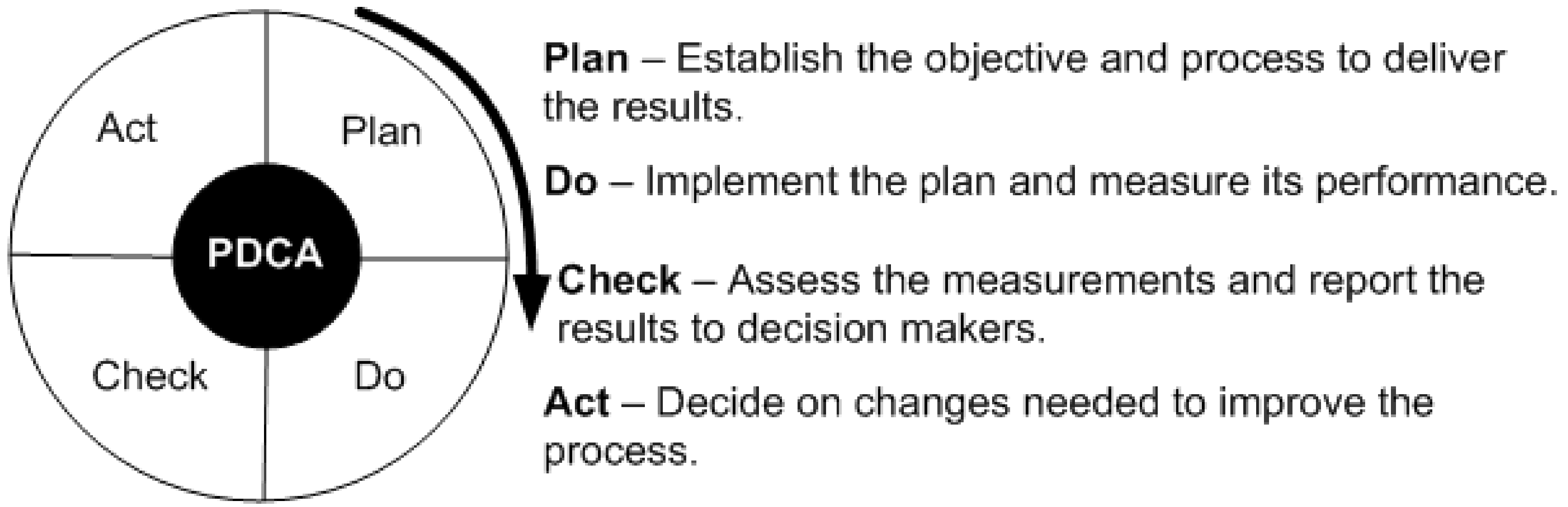
FOCUS OF QUALITY PROCESS

- Paying much attention to customer’s requirements
- Making efforts to continuously improve quality
- Integrating measurement processes with product design and development
- Pushing the quality concept down to the lowest level of the organization
- Developing a system-level perspective with an emphasis on methodology and process
- Eliminating waste through continuous improvement
- Adhering to deadlines

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.

Figure 1: The Shewhart cycle

- Deming introduced Shewhart's PDCA cycle to Japanese researchers
- It illustrates 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

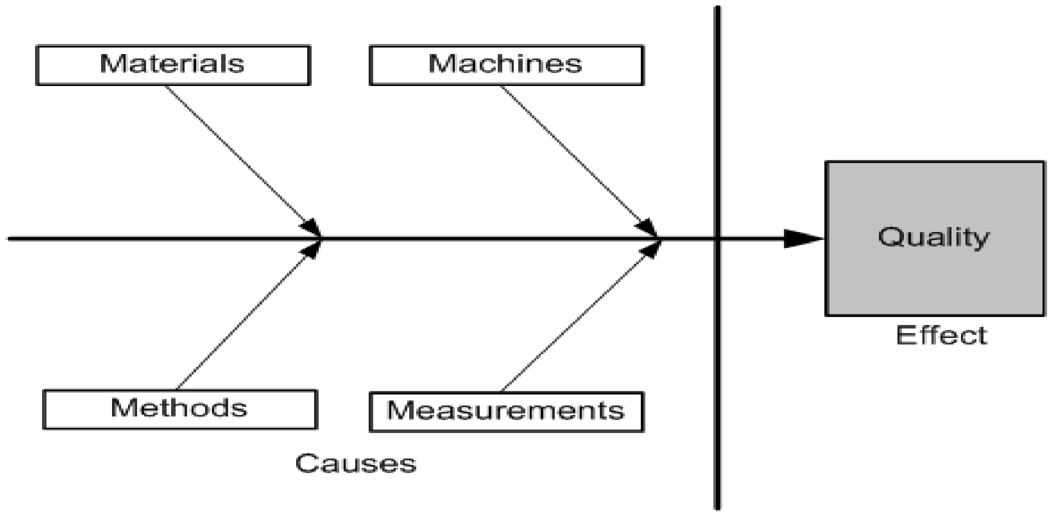


Figure 2: Ishikawa diagram

ISHIKAWA DIAGRAM

- Kaoru Ishikawa found from statistical data that dispersion in product quality came from four common causes, namely *materials*, *machines*, *methods*, and *measurements*, known as the 4 Ms.
- Materials often differ when sources of supply or size requirements vary.
- Machines, or equipment, also function differently depending on variations in their parts, and they operate optimally for only part of the time.
- Methods, or processes, cause even greater variations due to lack of training and poor handwritten instructions.
- Finally, measurements also vary due to outdated equipment and improper calibration.
- Variations in the 4 Ms parameters have an effect on the quality of a product.

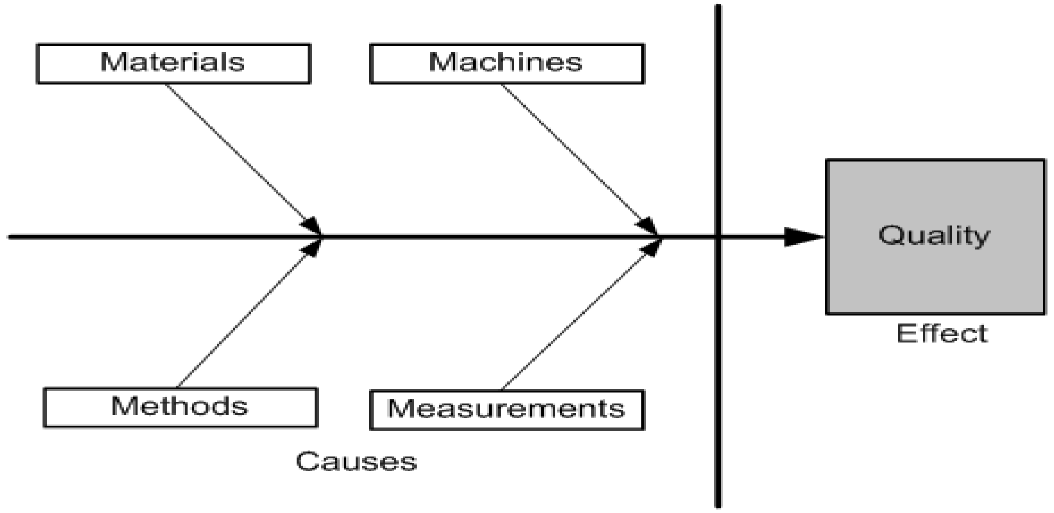


Figure 2: Ishikawa diagram

SOFTWARE QUALITY

- Five Views of Software Quality:
  - Transcendental view
    - It envisages quality as something that can be recognized but is difficult to define.
    - *I know it when I see it.*
  - User's view
    - Does the product satisfy user needs and expectations?
  - Manufacturing view
    - Quality is understood as conformance to the specification.
    - The quality level of a product is determined by the extent to which the product meets its specifications.
  - Product view
    - Quality is viewed as tied to the inherent characteristics of the product. A product's inherent characteristics, that is, internal qualities, determine its external qualities.
  - Value-based view
    - Depends on the amount a customer is willing to pay for it.

SOFTWARE QUALITY

- Software quality in terms of quality factors and criteria
  - A quality factor represents a behavioural characteristic of a system
    - Examples: correctness, reliability, efficiency, and testability
  - 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
  - Examples: ISO 9126, CMM, TPI, and TMM

ROLE OF TESTING

- Software quality assessment divide into two categories:
  - Static analysis
    - It examines the code and reasons overall behavior that might arise during run time
    - Identify issues within the logic and techniques
      - Examples: Code review, inspection, and algorithm analysis
  - Dynamic analysis
    - Actual program execution to expose possible program failure
    - Involves running code and examining the outcome, which also entails testing possible execution paths of the code.
    - Static and Dynamic Analysis are complementary in nature
- Focus is to combine the strengths of both approaches

VERIFICATION AND VALIDATION

- Verification
  - Evaluation of software system that helps in determining whether the product of a given development phase satisfies the requirements established before the start of that phase
    - Building the product correctly
    - Are we building the product right?
    - It is static testing
    - It includes checking documents, design, codes and programs.
- Validation
  - Evaluation of software system that helps in determining whether the product meets its intended use
    - Building the correct product
    - Are we building the right product?
    - It is dynamic testing
    - It includes testing and validating the actual product i.e. it includes execution of the code.

FAILURE, ERROR, FAULT AND DEFECT

- Failure
  - A *failure* is said to occur whenever the external behavior of a system does not conform to that prescribed in the system specification
- Error
  - An *error* is a state of the system.
  - An *error* state could lead to a *failure* in the absence of any corrective action by the system
- Fault
  - A *fault* is the adjudged cause of an *error*
- Defect
  - It is synonymous of *fault*
  - It a.k.a. *bug*
- The process of failure manifestation can therefore be succinctly represented as a behavior chain as follows: fault → error → failure.

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
- The level of reliability of a system depends on those inputs that cause failures to be observed by the end users.
- 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*
-

THE OBJECTIVES OF TESTING

- It does work
  - To find what is working [can be a single unit, integration of units or entire system]
- It does not work
  - To find what is not working
- Reduce the risk of failures
  - Complex system fail from time to time and gives rise to a notion of *failure rate*
- Reduce the cost of testing
  - the cost of designing, maintaining, and executing test cases,
  - the cost of analyzing the result of executing each test case,
  - the cost of documenting the test cases, and
  - the cost of actually executing the system and documenting it.

WHAT IS A TEST CASE?

- Test Case is a simple pair of
  - **<input, expected outcome>**
- State-less systems: A compiler is a stateless system
  - Test cases are very simple
  - A compiler is a stateless system because to compile a program it does not need to know about the programs it compiled previously
  - Outcome depends solely on the current input
- State-oriented: ATM is a state-oriented system
  - Test cases are not that simple. A test case may consist of a sequences of <**input, expected outcome**>
    - The outcome depends both on the current state of the system and the current input
    - ATM example:
      - **< check balance, $500.00 >,**
      - **< withdraw, “amount?” >,**
      - **< $200.00, “$200.00” >,**
      - **< check balance, $300.00 >**

EXPECTED OUTCOME

- An outcome of program execution may include
  - Value produced by the program
  - State Change
  - A sequence of values which must be interpreted together for the outcome to be valid

- A *test oracle* is a mechanism that verifies the correctness of program outputs
  - Generate expected results for the test inputs
  - Compare the expected results with the actual results of execution of the IUT

THE CONCEPT OF COMPLETE TESTING

- Complete or exhaustive testing means
  - “There are no undisclosed faults at the end of the test phase”
- Complete testing is nearly impossible for most of the system
  - 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

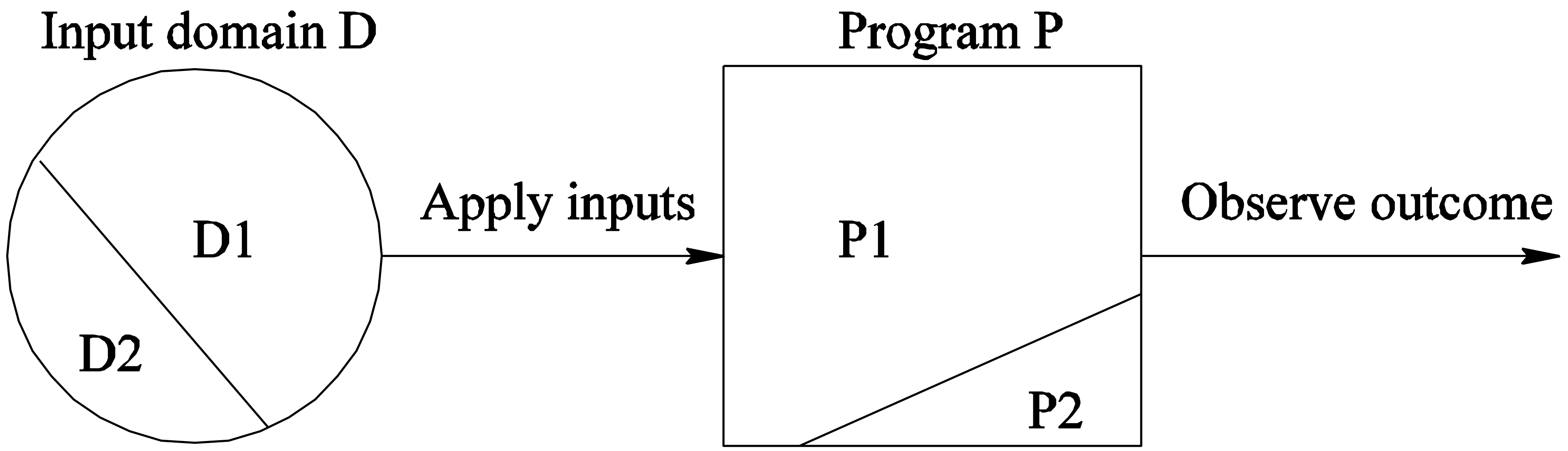


Figure 3: A subset of the input domain exercising a subset of the program behavior

- 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

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

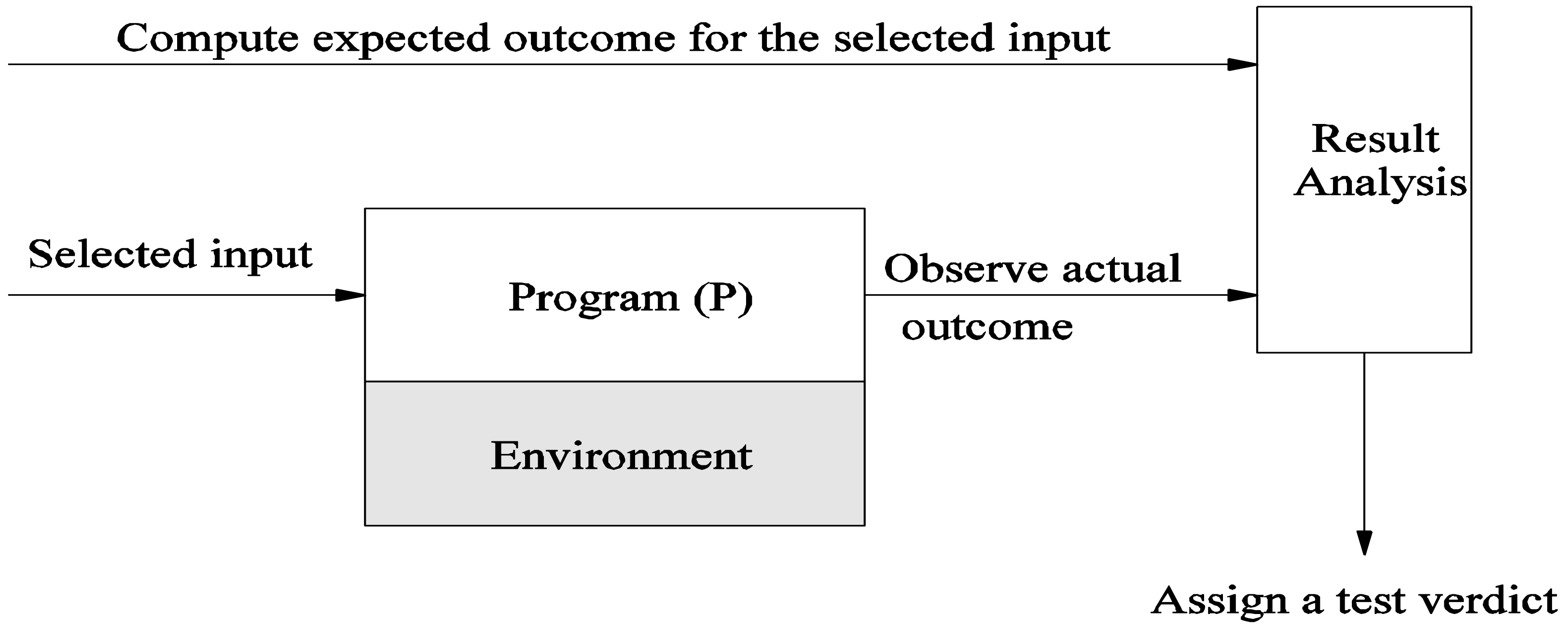


Figure 4: Different activities in process testing

TESTING LEVEL

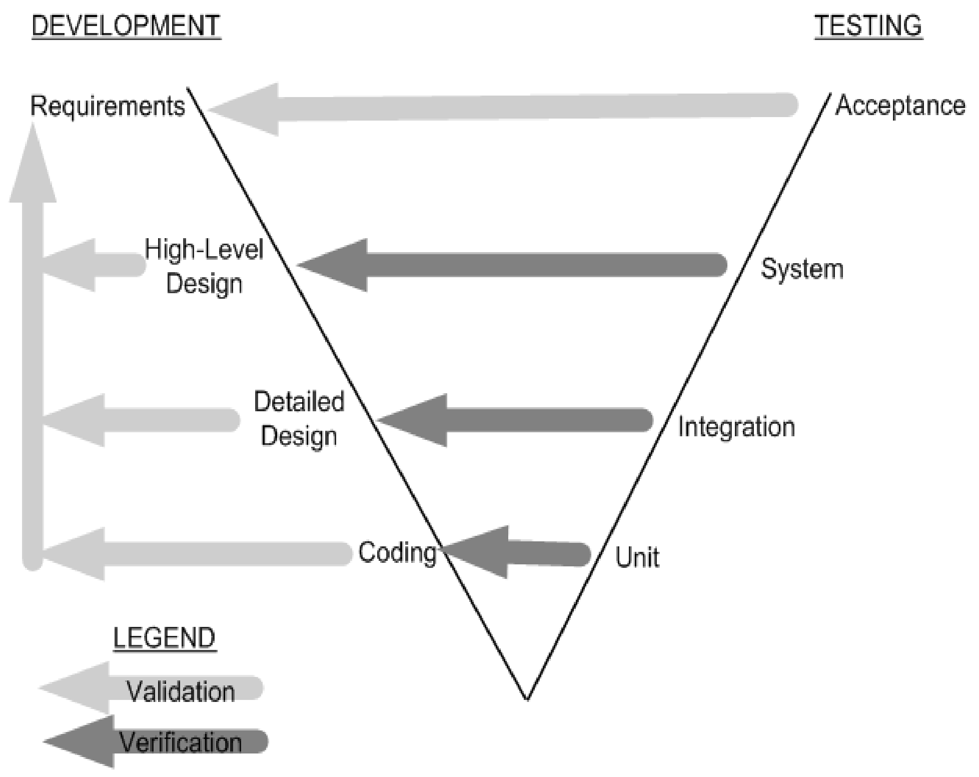


Figure 5: Development and testing phases in the V model

Unit testing

- Individual program units, such as procedure, methods in isolation

Integration testing

- Modules are assembled to construct larger subsystem and tested

System testing

- Includes wide spectrum of testing such as functionality, and load

Acceptance testing

- Customer’s expectations from the system
- Two types of acceptance testing
  - User Acceptance Test (UAT)
  - Business Acceptance Test (BAT)
- UAT: System satisfies the contractual acceptance criteria
- BAT: System will eventually pass the user acceptance test

TESTING LEVEL

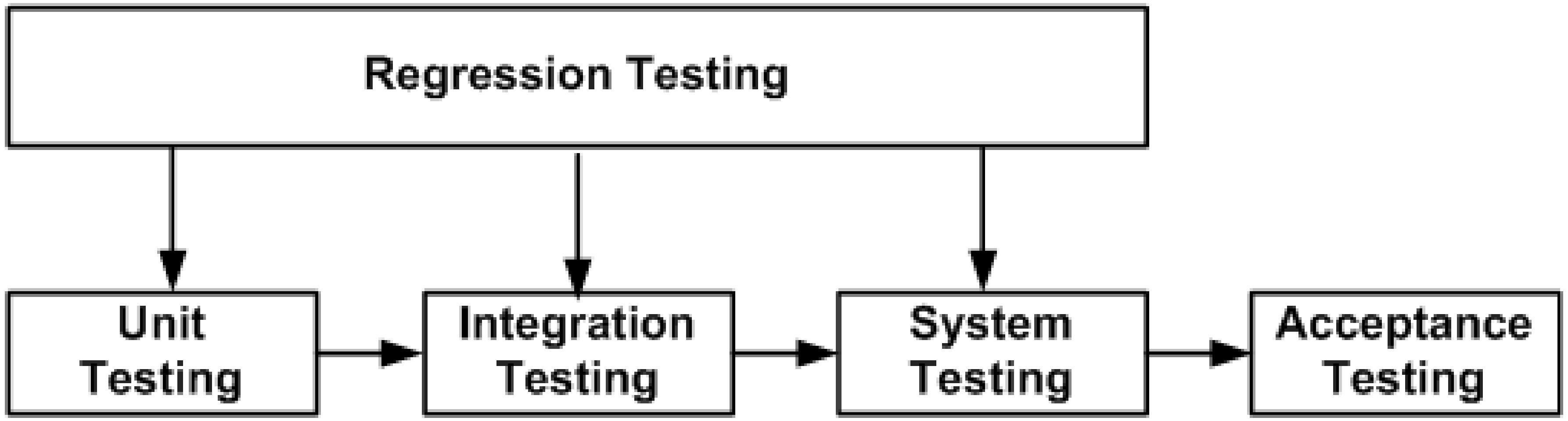


Figure 6: Regression testing at different software testing levels

- New test cases are not designed
- Test are selected, prioritized and executed
- To ensure that nothing is broken in the new version of the software

SOURCE OF INFORMATION FOR TEST SELECTION

- Requirement and Functional Specifications
- Source Code
- Input and output Domain
- Operational Profile
- Fault Model
  - Error Guessing
  - Fault Seeding
  - Mutation Analysis

WHITE-BOX AND BLACK-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 a.k.a. **functional testing**

Examines the program that is accessible from outside

Applies the input to a program and observes 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

TEST PLANNING AND DESIGN

- The purpose is to get ready and organized for test execution
- A test plan provides a:
  - Framework
    - A set of ideas, facts or circumstances within which the tests will be conducted
  - Scope
    - The domain or extent of the test activities
  - Details of resources needed
  - Effort required
  - Schedule of activities
  - Budget
- Test objectives are identified from different sources
- Each test case is designed as a combination of modular test components called test steps
- Test steps are combined together to create more complex tests

MONITORING AND MEASURING TEST EXECUTION

- Metrics for monitoring test execution
- Metrics for monitoring defects
- Test case effectiveness metrics
  - Measure the “defect revealing ability” of the test suite
  - Use the metric to improve the test design process
- Test-effort effectiveness metrics
  - Number of defects found by the customers that were not found by the test engineers

TEST TOOLS AND AUTOMATION

Increased productivity of the testers

Better coverage of regression testing

Reduced durations of the testing phases

Reduced cost of software maintenance

Increased effectiveness of test cases

The test cases to be automated are well defined

Test tools and infrastructure are in place

The test automation professionals have prior successful experience in automation

Adequate budget have been allocation for the procurement of software tools

TEST TEAM ORGANIZATION AND MANAGEMENT

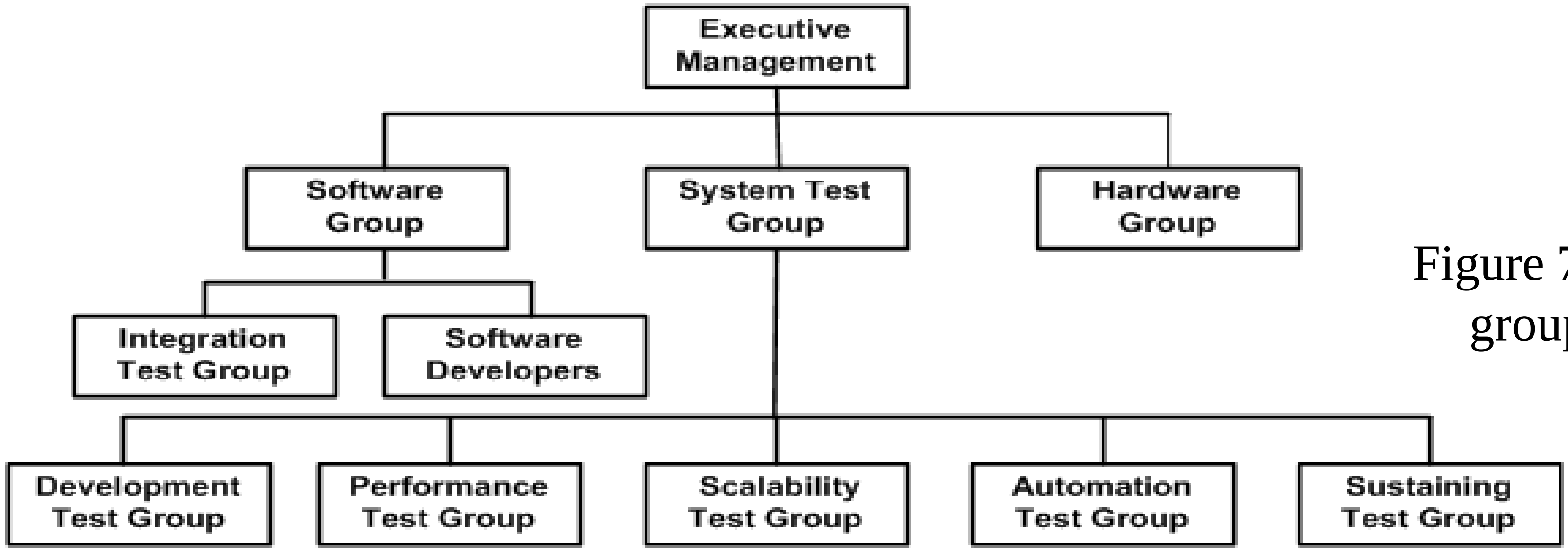


Figure 7: Structure of test groups

- Hiring and retaining test engineers is a challenging task
- Interview is the primary mechanism for evaluating applicants
- Interviewing is a skills that improves with practice
- To retain test engineers management must recognize the importance of testing efforts at par with development effort

THANK YOU!!