

Software Testing and Quality Assurance

Theory and Practice

Chapter 1

Basic Concepts and Preliminaries

- 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
- What is a Test Case?
- Expected Outcome
- The Concept of Complete Testing
- The Central Issue in Testing
- Testing Activities
- Testing Level
- Source of Information for Test Selection
- White-box and Black-box Testing
- Test Planning and Design
- Monitoring and Measuring Test Execution
- Test Tools and Automation
- Test Team Organization and Management

- 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.”

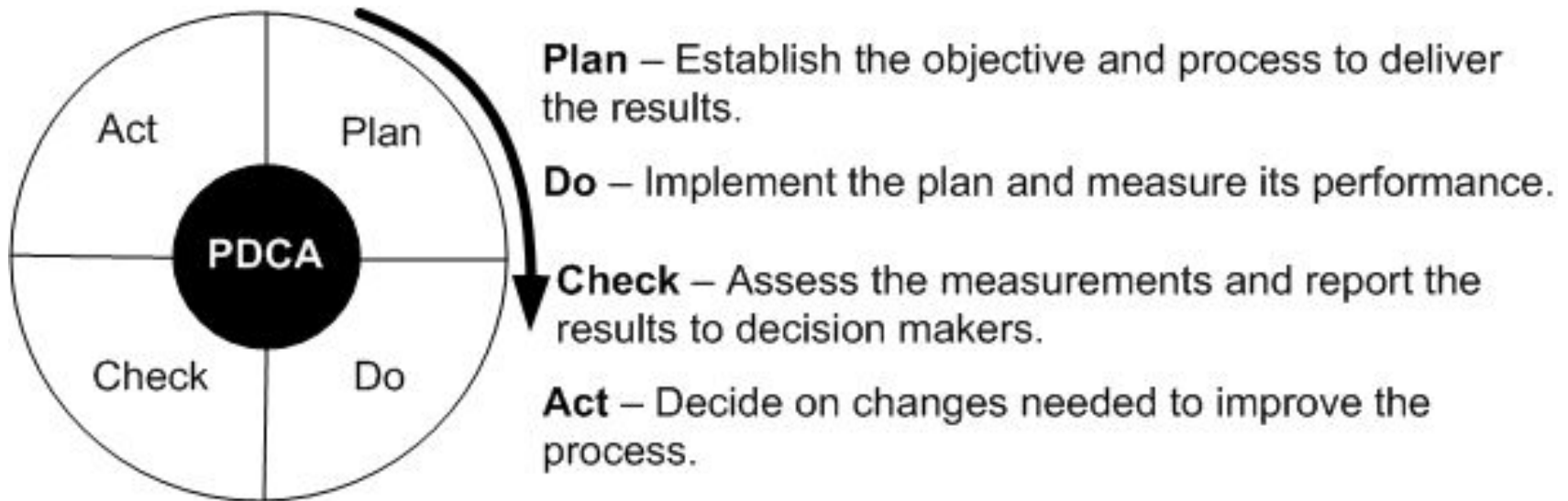
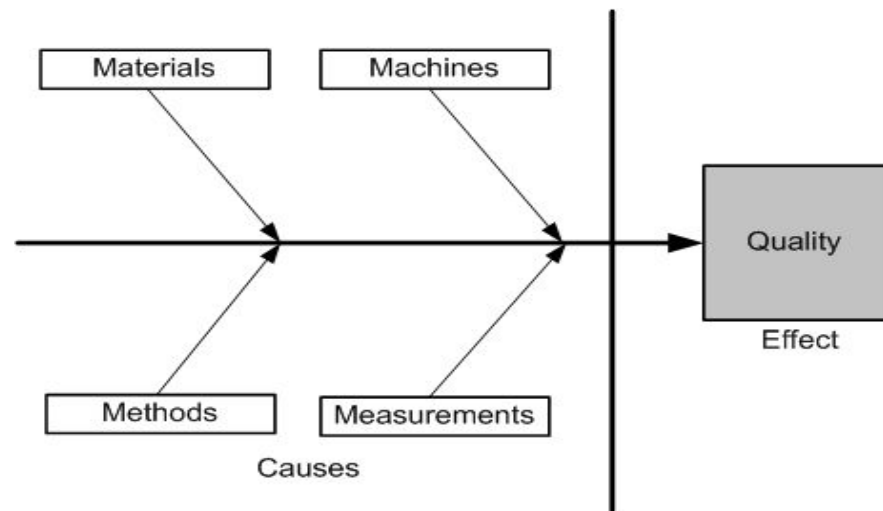


Figure 1.1: The Shewhart cycle

- 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

- 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 1.2: Ishikawa diagram



- 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

- Five Views of Software Quality:
 - Transcendental view
 - User's view
 - Manufacturing view
 - Product view
 - Value-based view
- Software Quality in terms of quality factors and criteria
 - A quality factor represents behavioral 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

- Software quality assessment divide into two categories:
 - Static analysis
 - It examines the code 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 combines the strengths of both approaches

- Verification
 - Evaluation of software system that help in determining whether the product of a given development phase satisfy the requirements established before the start of that phase
 - Building the product correctly

- Validation
 - Evaluation of software system that help in determining whether the product meets its intended use
 - Building the correct product

- 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*

- 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*

- It does work
- It does not work
- Reduce the risk of failures
- Reduce the cost of testing

- Test Case is a simple pair of
<input, expected outcome>
- State-less systems: A compiler is a stateless system
 - Test cases are very simple
 - 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 >**

- 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

- Complete or exhaustive testing means
“There are no undisclosed faults at the end of test phase”
- Complete testing is near 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

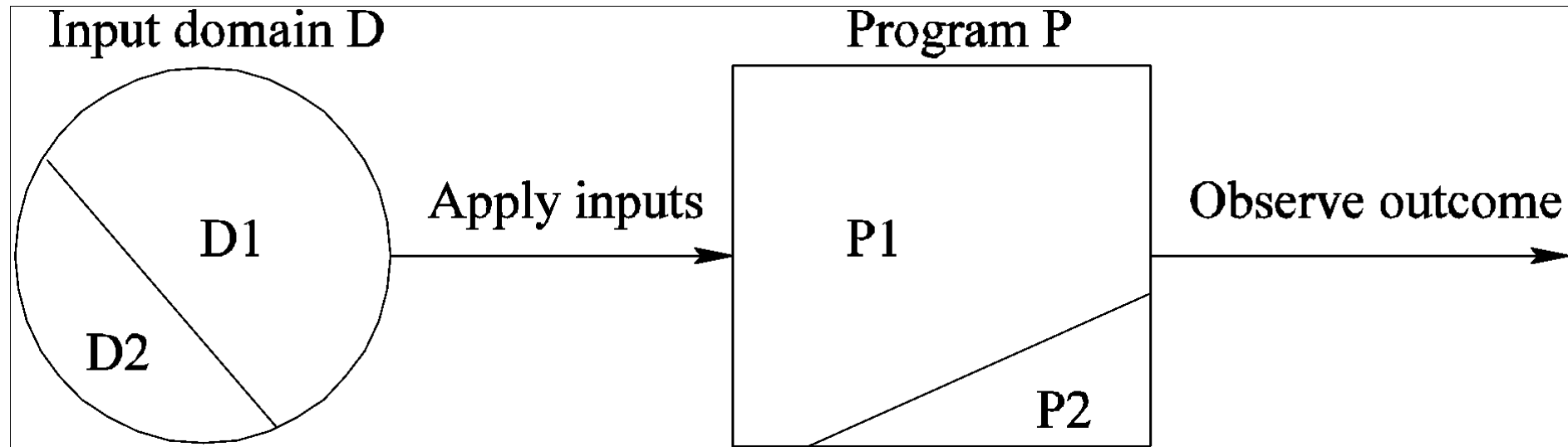


Figure 1.5: A subset of the input domain exercising a subset of the program behavior

- 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

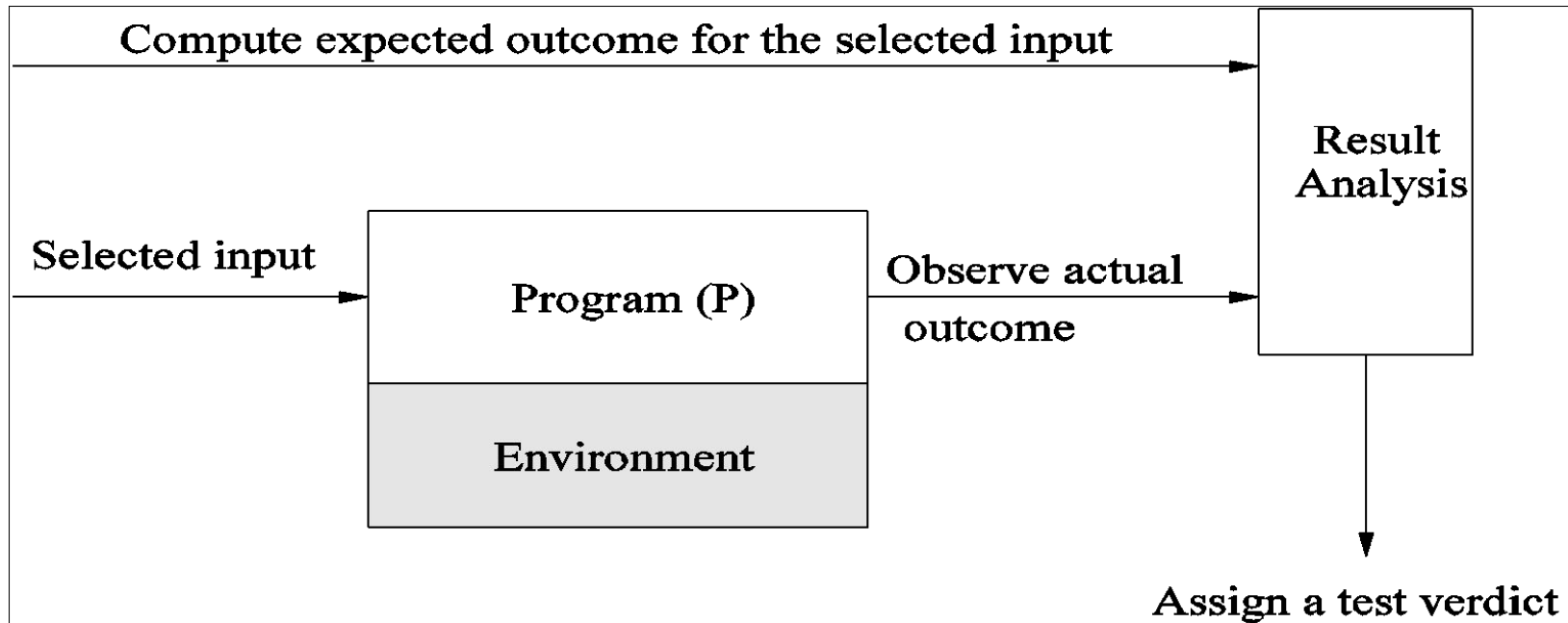


Figure 1.6: Different activities in process testing

- 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

- 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
 - UAT
 - BAT
 - UAT: System satisfies the contractual acceptance criteria
 - BAT: System will eventually pass the user acceptance test

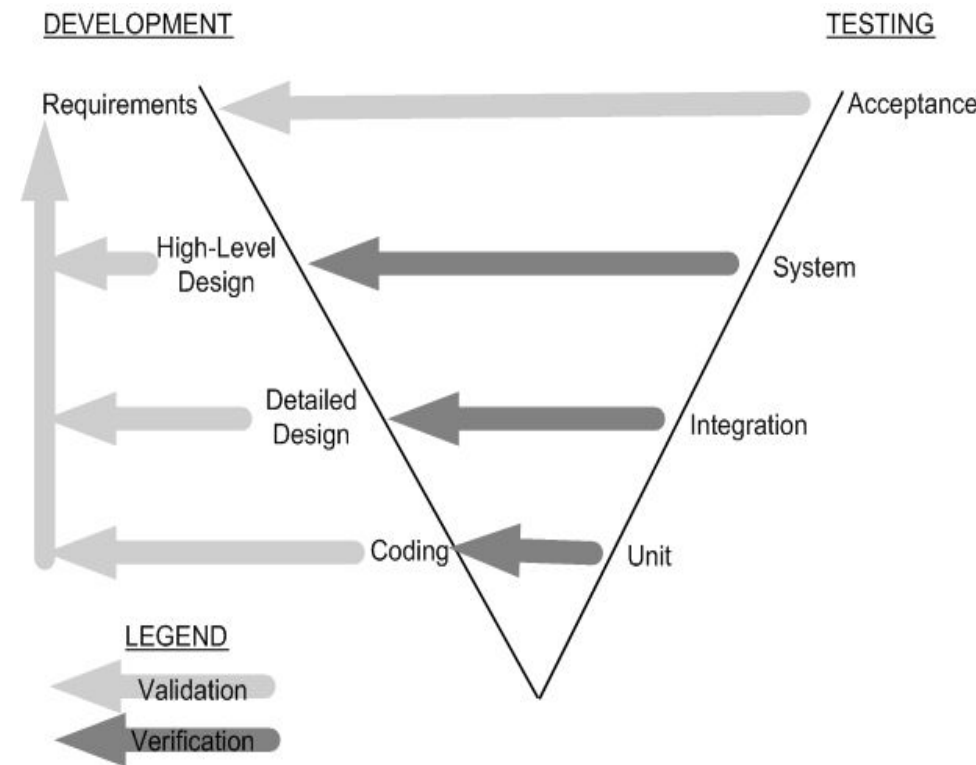


Figure 1.7: Development and testing phases in the V model

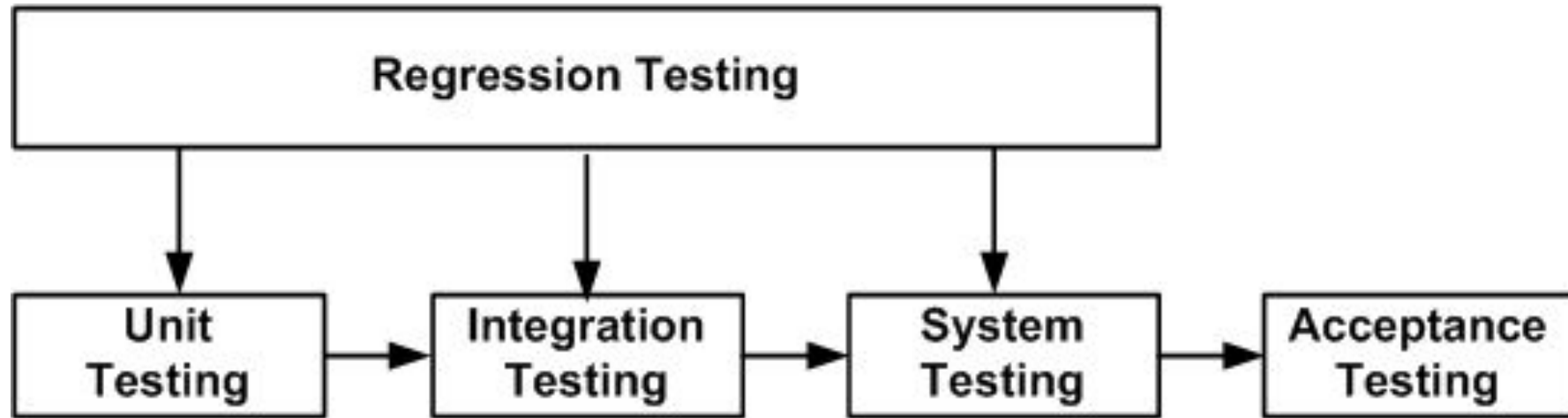


Figure 1.8: Regression testing at different software testing levels

- 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

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

- 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 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

- 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 resource 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

- 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

- 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 an 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

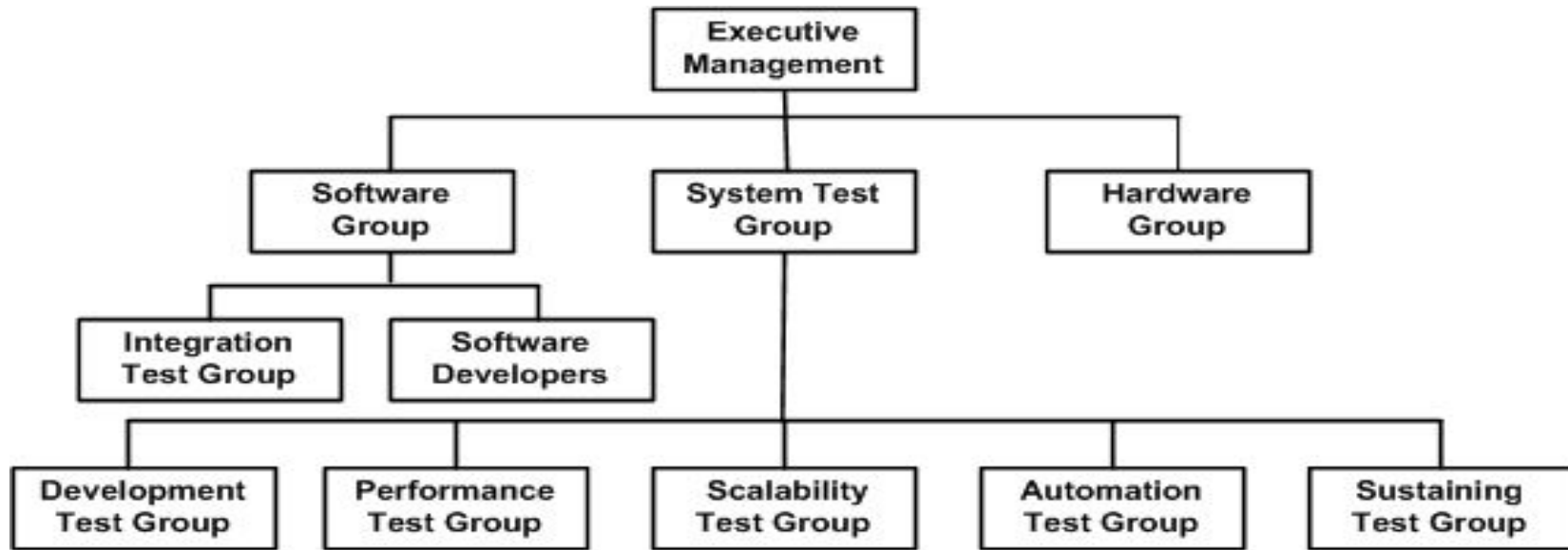


Figure 16.1: Structure of test groups

- 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