

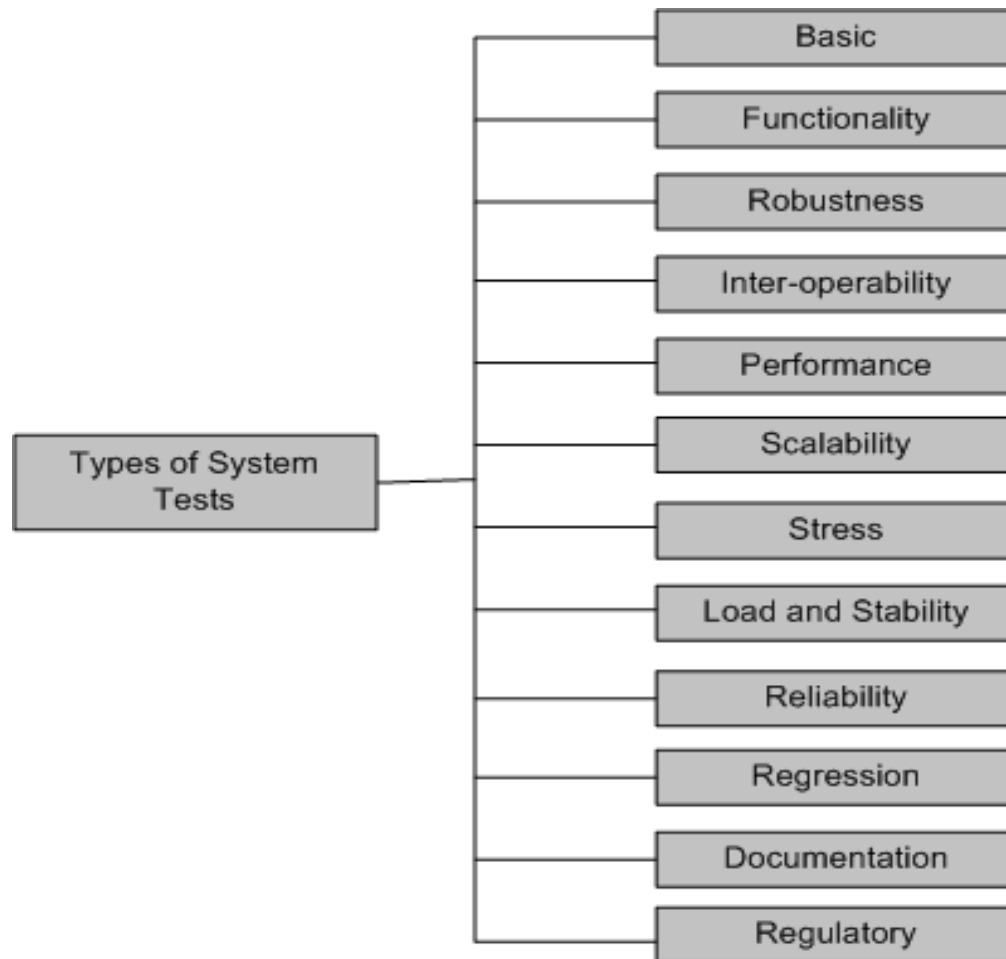
# Chapter Seven: System Testing

Sem. II - 2017

Department of Software Engineering  
ITSC-AAIT

Natnael A.

# Types of System Tests



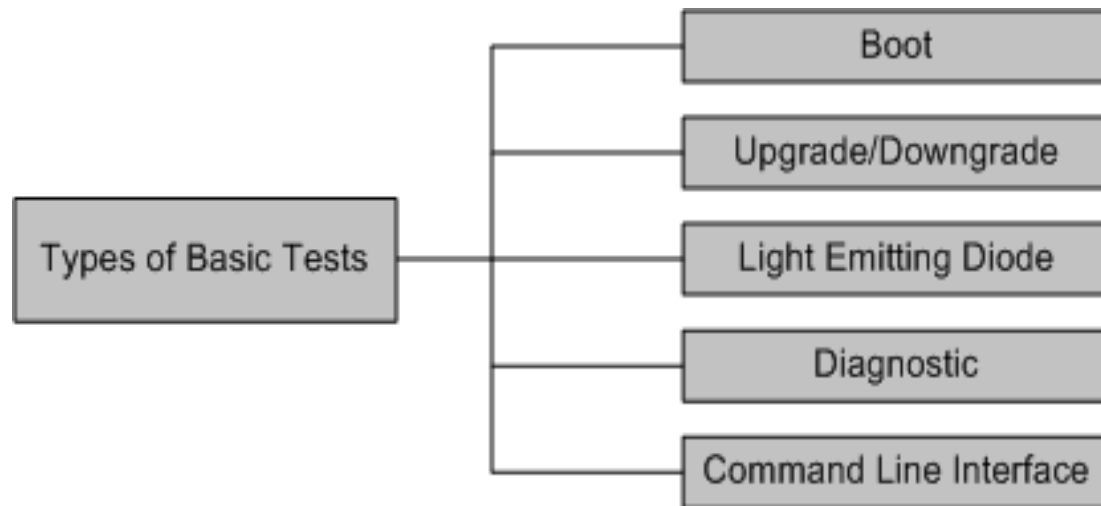
# Types of System Tests

- Basic tests provide an evidence that the system can be installed, configured and be brought to an operational state.
- Functionality tests provide comprehensive testing over the full range of the requirements, within the capabilities of the system.
- Robustness tests determine how well the system recovers from various input errors and other failure situations.
- Inter-operability tests determine whether the system can inter-operate with other third party products.
- Performance tests measure the performance characteristics of the system, e.g., throughput and response time, under various conditions.

# Types of System Tests

- Scalability tests determine the scaling limits of the system, in terms of user scaling, geographic scaling, and resource scaling
- Stress tests put a system under stress in order to determine the limitations of a system and, when it fails, to determine the manner in which the failure occurs
- Load and Stability tests provide evidence that the system remains stable for a long period of time under full load
- Reliability tests measure the ability of the system to keep operating for a long time without developing failures.
- Regression tests determine that the system remains stable as it cycles through the integration of other subsystems and through maintenance tasks.
- Documentation tests ensure that the system's user guides are

# Basic Tests

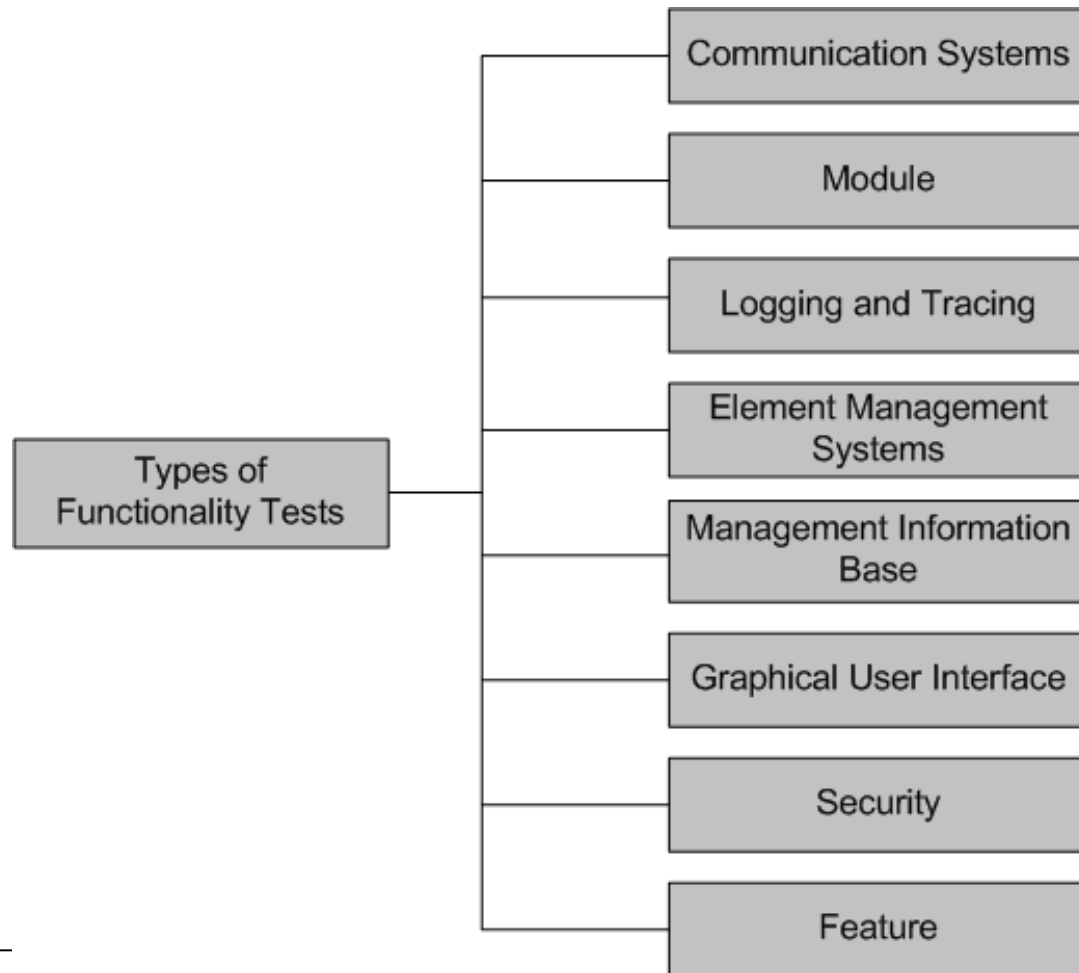


- Boot: Boot tests are designed to verify that the system can boot up its software image (or, build) from the supported boot options.
- Upgrade/Downgrade: Upgrade/downgrade tests are designed to verify that the system software can be upgraded or downgraded (rollback) in a graceful manner.

# Basic Tests

- Light Emitting Diode: The LED (Light Emitting Diode) tests are designed to verify that the system LED status indicators functioning as desired.
- Diagnostic: Diagnostic tests are designed to verify that the hardware components (or, modules) of the system are functioning as desired
  - Power-On Self Test
  - Ethernet Loop Back Test
  - Bit Error Rate Test
- Command line Interface: Command Line Interface (CLI) tests are designed to verify that the system can be configured.

# Functionality Tests



# Functionality Tests

- Communication Systems Tests
  - These tests are designed to verify the implementation of the communication systems as specified in the customer requirements specification
  - Four types of communication systems tests are recommended
    - Basic interconnection tests
    - Capability tests
    - Behavior tests
    - System resolution tests
- Module Tests
  - Module Tests are designed to verify that all the modules function individually as desired within the systems
  - The idea here is to ensure that individual modules function correctly within the whole system.
    - For example, an Internet router contains modules such as line cards, system controller, power supply, and fan tray. Tests are designed to verify each of the functionalities



# Functionality Tests

- Logging and Tracing Tests
  - Logging and Tracing Tests are designed to verify the configurations and operations of logging and tracing
  - This also includes verification of “flight data recorder: non-volatile Flash memory” logs when the system crashes.
- Element Management Systems (EMS) Tests
  - EMS tests verifies the main functionalities, which are to manage, monitor and upgrade the communication systems network elements(NEs)
  - It includes both EMS client and EMS servers functionalities
- Management Information Base (MIB) Tests
  - MIB tests are designed to verify
    - Standard MIBs including MIB II
    - Enterprise MIBs specific to the system

# Functionality Tests

- Graphical User Interface Tests
  - Tests are designed to look-and-feel the interface to the users of an application system
  - Tests are designed to verify different components such as icons, menu bars, dialog boxes, scroll bars, list boxes, and radio buttons.
  - The GUI can be utilized to test the functionality behind the interface, such as accurate response to database queries
  - Tests the usefulness of the on-line help, error messages, tutorials, and user manuals
  - The **usability** characteristics of the GUI is tested, which includes the following
    - **Accessibility**: Can users enter, navigate, and exit with relative ease?
    - **Responsiveness**: Can users do what they want and when they want in a way that is clear?
    - **Efficiency**: Can users do what they want to with minimum number of steps and time?
    - **Comprehensibility**: Do users understand the product structure

# Functionality Tests

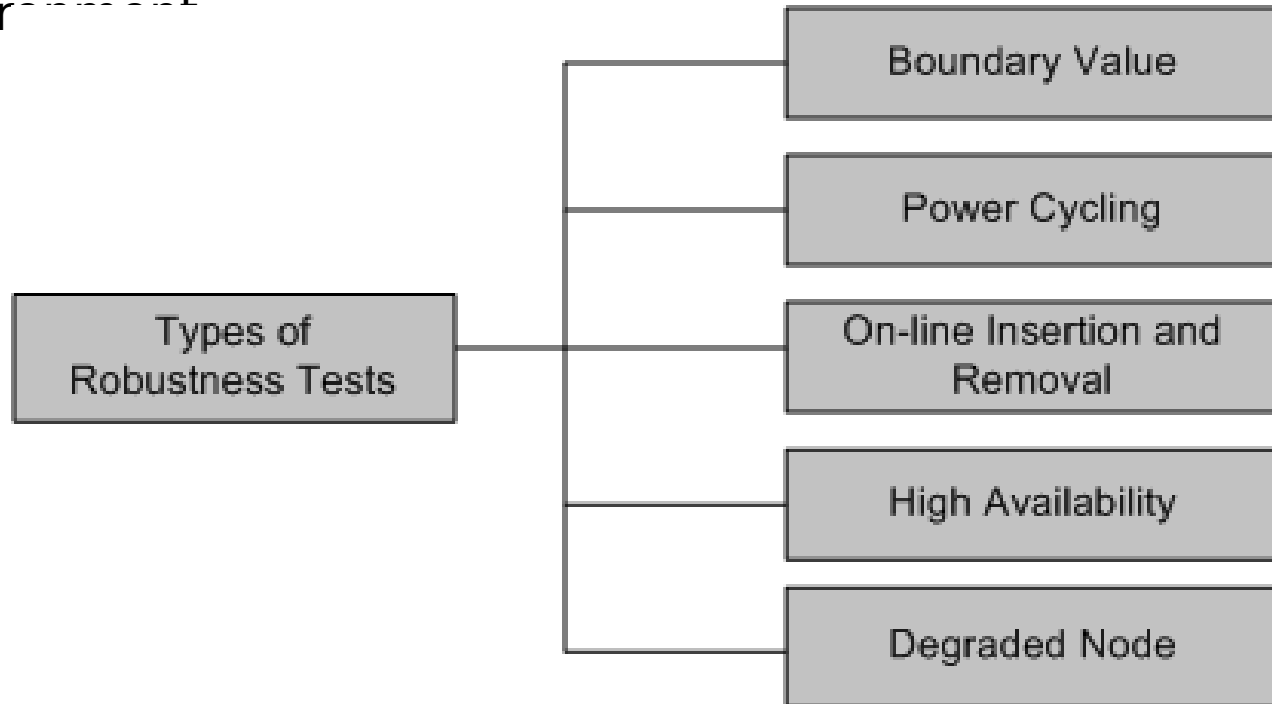
- Security Tests
  - Security tests are designed to verify that the system meets the security requirements
    - Confidentiality
      - It is the requirement that data and the processes be protected from unauthorized disclosure
    - Integrity
      - It is the requirement that data and process be protected from unauthorized modification
    - Availability
      - It is the requirement that data and processes be protected from the denial of service to authorized users
  - Security test scenarios should include negative scenarios such as misuse and abuse of the software system

# Functionality Tests

- Security Test: Useful types of security tests includes the following:
  - Verify that only authorized accesses to the system are permitted
  - Verify the correctness of both encryption and decryption algorithms for systems where data/messages are encoded.
  - Verify that illegal reading of files, to which the perpetrator is not authorized, is not allowed
  - Ensure that virus checkers prevent or curtail entry of viruses into the system
  - Ensure that the system is available to authorized users when a zero-day attack occurs
  - Try to identify any “backdoors” in the system usually left open by the software developers
- Feature Tests
  - These tests are designed to verify any additional functionalities which are defined in requirement specification but not covered in the other functional Testing categories. Examples: Data conversion testing and Cross-functionality testing

# Robustness Tests

- Robustness means how much sensitive a system is to erroneous input and changes its operational environment
- Tests in this category are designed to verify how gracefully the system behaves in error situations and in a changed operational environment



# Robustness Tests

- Boundary value
  - Boundary value tests are designed to cover boundary conditions, special values, and system defaults
  - The tests include providing invalid input data to the system and observing how the system reacts to the invalid input.
- Power cycling
  - Power cycling tests are executed to ensure that, when there is a power glitch in a deployment environment, the system can recover from the glitch to be back in normal operation after power is restored
- On-line insertion and removal
  - On-line Insertion and Removal (OIR) tests are designed to ensure that on-line insertion and removal of modules, incurred during both idle and heavy load operations, are gracefully handled and recovered.

# Robustness Tests

- High Availability
  - The concept of high availability is also known as fault tolerance
  - High availability tests are designed to verify the redundancy of individual modules, including the software that controls these modules.
  - The goal is to verify that the system gracefully and quickly recovers from hardware and software failures without adversely impacting the operation of the system
  - High availability is realized by means of proactive methods to maximize service up-time, and to minimize the downtime
- Degraded Node
  - Degraded node (also known as failure containment) tests verify the operation of a system after a portion of the system becomes non-operational
  - It is a useful test for all mission-critical applications.

# Interoperability Tests

- Tests are designed to verify the ability of the system to inter-operate with third party products
- The re-configuration activities during interoperability tests is known as configuration testing
- Another kind of inter-operability tests is called (backward) compatibility tests
  - Compatibility tests verify that the system works the same way across different platforms, operating systems, data base management systems
  - Backward compatibility tests verify that the current software build flawlessly works with older version of platforms.



# Performance Tests

- Tests are designed to determine the performance of the actual system compared to the expected one.
- Tests are designed to verify response time, execution time, throughput, resource utilization and traffic rate.
- One needs to be clear about the specific data to be captured in order to evaluate performance metrics.
- For example, if the objective is to evaluate the response time, then one needs to capture.
  - End-to-end response time (as seen by external user)
  - CPU time
  - Network connection time
  - Database access time
  - Network connection time
  - Waiting time

# Scalability Tests

- Tests are designed to verify that the system can scale up to its engineering limits
- Scaling tests are conducted to ensure that the system response time remains the same, or increases by a small amount, as the number of users are increased.
- There are three major causes of these limitations:
  - data storage limitations
  - network bandwidth limitations
  - speed limit
- Extrapolation is often used to predict the limit of scalability

# Stress Tests

- The goal of stress testing is to evaluate and determine the behavior of a software component while the offered load is in excess of its designed capacity
- The system is deliberately stressed by pushing it to and beyond its specified limits
- It ensures that the system can perform acceptably under worst-case conditions, under an expected peak load. If the limit is exceeded and the system does fail, then the recovery mechanism should be invoked
- Stress tests are targeted to bring out the problems associated with one or more of the following:
  - Memory leak
  - Buffer Allocation and Memory Carving

# Load and Stability Tests

- Tests are designed to ensure that the system remains stable for a long period of time under full load
- When a large number of users are introduced and applications that run for months without restarting, a number of problems are likely to occur:
  - the system slows down
  - the system encounters functionality problems
  - the system crashes altogether
- Load and stability testing typically involves exercising the system with virtual users and measuring the performance to verify whether the system can support the anticipated load.
- This kind of testing help one to understand the ways the system will work in real-life situations.

# Reliability Tests

- Reliability tests are designed to measure the ability of the system to remain operational for long periods of time.
- The reliability of a system is typically expressed in terms of mean time to failure (MTTF)
- The average of all the time intervals between successive failures is called the MTTF.
- After a failure is observed, the developers analyze and fix the defects, which consumes some time – let us call this interval the repair time.
- The average of all the repair times is known as the mean time to repair (MTTR)
- Now we can calculate a value called mean time between failure (MTBF) as  $MTBF = MTTF + MTTR$

# Regression Tests

- In this category, new tests are not designed, instead, test cases are selected from the existing pool and executed
- The main idea in regression testing is to verify that no defect has been introduced into the unchanged portion of a system due to changes made elsewhere in the system
- During system testing, many defects are revealed and the code is modified to fix those defects
- One of four different scenarios can occur for each fix:
  - The reported defect is fixed
  - The reported defect could not be fixed inspite of making an effort
  - The reported defect has been fixed, but something that used to work before has been failing
  - The reported defect could not be fixed inspite of an effort, and something that used to work before has been failing

# Regression Tests

- One possibility is to re-execute every test case from version  $n - 1$  to version  $n$  before testing anything new
- A full test of a system may be prohibitively expensive.
- A subset of the test cases is carefully selected from the existing test suite to
  - maximize the likelihood of uncovering new defects
  - reduce the cost of testing

# Documentation Tests

- Documentation testing means verifying the technical accuracy and readability of the user manuals, tutorials and the on-line help
- Documentation testing is performed at three levels:
  - Read test: In this test a documentation is reviewed for clarity, organization, flow, and accuracy without executing the documented instructions on the system
  - Hands-on test: Exercise the on-line help and verify the error messages to evaluate their accuracy and usefulness.
  - Functional test: Follow the instructions embodied in the documentation to verify that the system works as it has been documented.



# Regulatory Tests

- In this category, the final system is shipped to the regulatory bodies in those countries where the product is expected to be marketed
- The idea is to obtain compliance marks on the product from various countries
- Most of these regulatory bodies issue safety and EMC (electromagnetic compatibility)/ EMI (electromagnetic interference) compliance certificates (emission and immunity)
- The regulatory agencies are interested in identifying flaws in software that have potential safety consequences
- The safety requirements are primarily based on their own published standards

# Software Safety

- A hazard is a state of a system or a physical situation which when combined with certain environmental conditions, could lead to an accident or mishap
- An accident or mishap is an unintended event or series of events that results in death, injury, illness, damage or loss of property, or harm to the environment
- Software safety is defined in terms of hazards
- A software in isolation cannot do physical damage. However, a software in the context of a system and an embedding environment could be vulnerable
- Examples:
  - A software module in a database application is not hazardous by itself, but when it is embedded in a missile navigation system, it

# Safety Assurance

- There are two basic tasks performed by a safety assurance engineering team:
  - Provide methods for identifying, tracking, evaluating, and eliminating hazards associated with a system
  - Ensure that safety is embedded into the design and implementation in a timely and cost effective manner, such that the risk created by the user/operator error is minimized

# Reading

Software Testing and Quality Assurance: Theory and Practice

Page [224-253]

And read other online references

Assessment  
Software Test Planning