ELYSIUM SPARK NOTES

ELYSIUM ACADEMY
Milestone of Cognizance

MANUAL TESTER ISTQB

Elysium Academy Spark Notes

VERSION 2.7

01. Introduction to Software Testing

What is Software Testing?

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The main aim of testing is to ensure that the software meets user requirements and is free of defects.

- Validation: Are we building the right product?
- Verification: Are we building the product right?

- **Objectives of Software Testing:**
  - Detect defects in the software.
  - Prevent defects from entering the system by early identification.
  - Ensure the product meets its requirements.
  - Increase confidence that the product works as expected.
  - Reduce the risk of failures after the product is released.
- **Importance of Testing:**
  - **Quality Assurance:** Testing ensures that a product is of high quality.
  - **Cost Saving:** Finding and fixing defects earlier in the lifecycle is cheaper than later stages.
  - **User Satisfaction:** Testing ensures that the product meets the needs of its users.
- **Principles of Testing:**
  - **Testing shows the presence of defects:** Testing can reveal defects, but cannot prove the absence of them.
  - **Exhaustive testing is impossible:** Testing all possible inputs and scenarios is not feasible; therefore, risk-based testing is employed.
  - **Early testing:** The earlier a defect is detected, the cheaper it is to fix.
  - **Defect clustering:** A small number of modules usually contains most defects.
  - **Pesticide paradox:** Re-running the same test cases repeatedly will stop finding new defects.
  - **Testing is context dependent:** Testing strategies and approaches vary based on the application.
  - **Absence of errors fallacy:** A defect-free system may still fail to meet user expectations.

- **The Testing Process and Life Cycle:**

The software testing process typically involves the following phases:

- **Test Planning:** Defining the objectives, strategy, scope, and resources for testing.
- **Test Design:** Identifying test conditions and creating test cases.
- **Test Execution:** Running test cases, comparing actual results with expected outcomes.
- **Defect Reporting:** Logging defects found during testing.
- **Test Closure:** Evaluating the test cycle, reporting test metrics, and archiving test cases.

02. Fundamentals of Testing

- **Verification vs Validation:**
  - **Verification:** Ensures that the product is built according to the specifications and requirements (are we building it right?).
  - **Validation:** Ensures the product fulfills its intended use when deployed in its environment (are we building the right thing?).
- **Testing Terminology:**
  - **Defect/Bug:** A flaw in a software product that causes it to behave incorrectly.
  - **Test Case:** A set of conditions and variables to determine whether a system meets its requirements.
  - **Test Suite:** A collection of test cases.
  - **Test Data:** Input given to the software during testing to ensure it behaves as expected.
- **Debugging vs Testing:**
  - **Testing:** Identifying defects.
  - **Debugging:** Fixing the identified defects.
- **Static vs Dynamic Testing:**
  - **Static Testing:** Examining the code and documentation without executing the program (e.g., reviews, walkthroughs).
  - **Dynamic Testing:** Involves executing the code and checking the output against expected results.

03. The Software Development Life Cycle (SDLC) and Testing Models

- **Waterfall Model:**

A sequential development model where each phase (requirements, design, implementation, testing, deployment) must be completed before the next one begins. Testing is done afte r development is complete.

- **V-Model (Verification and Validation):**

An extension of the Waterfall model where testing activities are planned in parallel with corresponding development stages. Each phase has its validation activity (e.g., design has test design, coding has unit testing).

- **Agile Model:**

Agile emphasizes collaboration, flexibility, and incremental development. Testing is continuous and performed throughout development. Agile uses iterative cycles called sprints, where testing is integrated with development.

- **Testing in DevOps:**

In DevOps, testing is integrated into continuous integration/continuous deployment (CI/CD) pipelines. Testing becomes automated and part of the ongoing development process, allowing for frequent releases with minimal risk.

- **STLC (Software Testing Life Cycle):**

The Software Testing Life Cycle consists of several phases that ensure systematic testing:

- **Requirement Analysis:** Understand what is being tested.
- **Test Planning:** Determine testing scope, strategy, and resources.
- **Test Case Development:** Create detailed test cases based on requirements.
- **Test Environment Setup:** Prepare the testing environment.
- **Test Execution:** Run test cases and log defects.
- **Test Cycle Closure:** Evaluate testing success and wrap up the cycle

04. Test Levels

- **Unit Testing:**
  - Tests individual units or components of the software.
  - Usually performed by developers.
  - Helps ensure that individual functions or modules work correctly in isolation.
- **Integration Testing:**
  - Verifies the interactions between integrated units or components.
  - Can be done using different strategies, such as Big Bang or Incremental (Top-Down or Bottom-Up).
- **System Testing:**
  - Tests the entire integrated system to validate that it meets specified requirements.
  - Ensures that the system works end-to-end as a whole.
- **Acceptance Testing:**
  - Performed by the end users to determine if the system meets business needs.
  - User Acceptance Testing (UAT) and Operational Acceptance Testing (OAT) are common types.

05. Test Types

- **Functional Testing:**
  - Verifies that the software functions according to requirements.
  - Includes:
    1. **Smoke Testing:** Preliminary testing to check if the basic functionality works.
    2. **Sanity Testing:** Focused testing on specific functionalities after a change.
    3. **Regression Testing:** Ensures that new changes do not introduce new defects.

- **Non-Functional Testing:**
  - Tests aspects of the software not related to specific behaviors, such as:
    1. **Performance Testing:** Checks the system's performance under various conditions.
    2. **Usability Testing:** Ensures the system is easy to use.
    3. **Security Testing:** Ensures the system is protected against unauthorized access and attacks.
- **Regression Testing:**
  - Ensures that changes to the code do not introduce new defects in previously working functionality.
  - Usually performed after bug fixes, updates, or code modifications.
- **Maintenance Testing:**
  - Performed to ensure that updates, upgrades, or fixes to the system do not negatively impact the overall functionality.

06. Test Design Techniques

- **Black Box Techniques:**
  - Focuses on testing the system's external behavior without knowledge of the internal code structure.
    1. **Equivalence Partitioning:** Dividing input data into valid and invalid partitions.
    2. **Boundary Value Analysis (BVA):** Testing at the boundaries between partitions.
    3. **Decision Table Testing:** Used when the system's output depends on different combinations of inputs.
- **White Box Techniques:**
  - Testing based on the internal structure and logic of the system.
    1. **Statement Coverage:** Ensures every statement in the code is executed at least once.
    2. **Branch Coverage:** Ensures every decision branch (true/false) is tested.

- **Experience-Based Techniques:**
  - Relies on the tester's experience, intuition, and knowledge of past bugs.
    1. **Exploratory Testing:** Involves simultaneous learning, test design, and test execution.
    2. **Error Guessing:** The tester guesses where defects may be based on experience.

07. Test Management

- **Test Planning:**
  - Defining the test strategy, objectives, resources, schedule, and deliverables.
  - The test plan document outlines the scope, approach, resources, and schedule of testing activities.
- **Test Control:**
  - Ongoing activities to track the test progress and adjust the plan if necessary.
- **Test Estimation:**
  - Estimating the time and effort required to perform the testing activities. Common techniques include:
    1. **Work Breakdown Structure (WBS)**
    2. **Expert-based Estimation**
    3. **Historical Data-based Estimation**
- **Test Reporting and Metrics:**
  - Reporting test results, defects, and overall product quality.
  - Test Metrics: Quantitative data, such as test coverage, defect density, test execution rate.
- **Entry and Exit Criteria:**
  - **Entry Criteria:** Defines the conditions that must be met before testing can begin.
  - **Exit Criteria:** Defines when testing can be concluded, such as after a certain percentage of test cases are executed and passed.

08. Defect Management

- **Defect Lifecycle:**
  - **New:** A defect is identified and reported.
  - **Assigned:** The defect is assigned to a developer to fix.
  - **Open:** The developer starts working on the defect.
  - **Fixed:** The defect is fixed and ready for retesting.
  - **Closed:** The defect is verified as fixed by the tester.
  - **Reopened:** If the defect reoccurs, it is reopened.
- **Severity vs Priority:**
  - **Severity:** Refers to the impact of the defect on the system (e.g., critical, major, minor).
  - **Priority:** Refers to the urgency of fixing the defect (e.g., high, medium, low).
- **Defect Reporting:**
  - Reporting defects involves logging them into a defect tracking system with all relevant details (e.g., steps to reproduce, expected vs actual results, severity, priority).

09. Testing Tools

- **Test Management Tools:**
  - Help in planning, executing, and managing the overall testing process.

    Examples: JIRA, TestRail, HP ALM
- **Defect Tracking Tools:**
  - Tools for reporting, tracking, and managing defects.

    Examples: Bugzilla, Mantis, JIRA
- **Automation Tools:**
  - Automation tools help reduce manual testing effort, especially for regression tests.

    Examples: Selenium, QTP, JUnit

10. Testing in Agile

- **Agile Testing Principles:**
  - Testing is integrated into the development process and occurs continuously.
  - Collaboration between testers, developers, and business stakeholders is key
- **Continuous Testing:**
  - In Agile, testing is continuous and happens throughout the development cycle.
  - Automated testing tools are widely used to perform regression and functional tests after each code change.
- **Exploratory Testing:**
  - Testers explore the system to identify defects that may not be caught by formal test cases.
  - It involves simultaneous learning, test design, and execution.

11. Conclusion

This ISTQB Manual Tester covers all fundamental topics essential for manual testing according to the ISTQB syllabus. From understanding the basics of testing and the software development life cycle, to mastering test management, defect management, and test design techniques. This guide serves as a quick reference for key testing concepts, methodologies, and best practices that every manual tester should know.

ELYSIUM SPARK NOTES

ELYSIUM ACADEMY
Milestone of Cognizance

Thank You

For Your Learning Today

elysiumacademy.org | info@elysiumacademy.org

Scan Here for More
Spark Notes