Testing Concepts

Lesson 1: Fundamentals of Testing

Capgemini

Lesson Objectives

To understand the following topics:

- Some Facts about Software Systems
- What is Testing?
  - Typical Objectives of Testing
  - Testing and Debugging
- Why is Testing Necessary?
  - Testing's Contributions to Success
  - Quality Assurance and Testing
  - Errors, Defects, and Failures - Reasons behind Errors
  - Defects, Root Causes and Effects
  - Cost of Software Defects
  - Importance of Testing Early in SDLC phases

Lesson Objectives

- Seven Testing Principles
  - Economic of Testing
  - Scope of Software Testing
  - Factors influencing Software Testing
- Test Process
  - Test Process in Context
  - Test Activities and Tasks
  - Test Work Products
  - Traceability between the Test Basis and Test Work Products
- The Psychology of Testing
  - Human Psychology and Testing
    - Attributes of a good Tester
    - Code of Ethics for Tester
  - Tester’s and Developer’s Mindsets
- Limitations of Software Testing

Some Facts about Software Systems !!

- Software systems are an integral part of life, from business applications (e.g., banking) to consumer products (e.g., cars).
- Most people have had an experience with software that did not work as expected.
- Software that does not work correctly can lead to many problems, including loss of money, time, or business reputation, and even injury or death.

Examples :

1. Excel gives 77.1 x 850 = 100000 instead of 65535
2. Y2K problem in Payroll systems designed in 1974
3. Disney's Lion King – Simba

1.1 What is Testing ?

Software testing is a way to assess the quality of the software and to reduce the risk of software failure in operation.

- Misperceptions of Testing :
  - Testing only consists of running tests i.e. executing the software and checking the results. But, software testing is a process which includes many different activities and test execution is just one of these activities. The test process includes activities such as test planning, analyzing, designing, implementing and executing the tests, reporting test progress and results, and evaluating the quality of a test object.
  - Testing focuses entirely on verification of requirements, user stories, or other specifications. But, testing also involves checking whether the system meets specified requirements, it also involves validation, which is checking whether the system will meet user and other stakeholder needs in its operational environment(s).

1.1 What is Testing ? (Cont.)

Testing involves both Dynamic testing and Static testing.

- Testing that involves the execution of the component or system being tested; such testing is called **dynamic testing.**
- Testing that involves the reviewing of work products such as requirements, user stories, and source code; such testing is called **static testing.**

Test activities are organized and carried out differently in different lifecycles, explained in lesson 2.

1.1.1 Objectives of Software Testing

- To evaluate work products such as requirements, user stories, design, and code
- To verify whether all specified requirements have been fulfilled
- To validate whether the test object is complete and works as the users and other stakeholders expect
- To build confidence in the level of quality of the test object
- To prevent defects
- To find failures and defects in the Software before User finds it
- To provide sufficient information to stakeholders to allow them to make informed decisions, especially regarding the level of quality of the test object.

1.1.1 Objectives of Software Testing (Cont.)

- To reduce the level of risk of inadequate software quality (e.g., previously undetected failures occurring in operation) and contribute to the delivery of higher quality software product

- To comply with contractual, legal, or regulatory requirements or standards, and/or to verify the test object's compliance with such requirements or standards

Objectives of testing can vary, depending upon the context of the component or system being tested, the test level, and the software development lifecycle model.

Software Testing - Definitions

The process of executing a program (or part of a program) with the intention of finding errors - G. J. Myers

Software testing is the process of testing the functionality and correctness of software by running it

Testing is the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies specified requirements - IEEE 83a

The process of analyzing a system to detect the difference between existing and required conditions and to evaluate the feature of the system - IEEE/ANSI, 1983 [Std 829-1983]

1.1.2 Testing and Debugging

Testing and debugging are different.

- Executing tests can show failures that are caused by defects in the software.
- Debugging is the development activity that finds, analyzes, and fixes such defects.

Subsequent confirmation testing checks whether the fixes have resolved the defects.

In some cases, testers are responsible for the initial test and the final confirmation test, while developers do the debugging and associated component testing.

However, in Agile development and in some other lifecycles, testers may be involved in debugging and component testing.

1.2 Why is Software Testing necessary ?

- Rigorous testing of components and systems, and their associated documentation, can help reduce the risk of failures occurring during operation.
- When defects are detected, and subsequently fixed, this contributes to the quality of the components or systems.
- Software testing is also required to meet contractual or legal requirements or industry-specific standards.
- SDLC consists of many stages and if bugs are caught in the earlier stages it costs much less to fix them. This saves money and time.
- Software testing provides product security – the user gets a trustworthy product.
- Customer satisfaction - Software Testing helps in bringing out the best user experience possible.

1.2.1 Testing's Contributions to Success

Use of appropriate test techniques applied with the appropriate level of test expertise, in the appropriate test levels, and at the appropriate points in the SDLC can reduce the frequency of problematic deliveries.

Examples :

- Testers involved in requirements reviews reduces the risk of incorrect functionality being developed.
- Testers working closely with system designers reduces the risk of fundamental design defects.
- Testers working closely with developers reduces the risk of defects within the code and the tests.
- Testers verifying and validating the software prior to release can detect failures that increases the likelihood that the software meets stakeholder needs and satisfies requirements.
- Achievement of defined test objectives contributes to the success of overall software development and maintenance.

1.2.2 Quality Assurance and Testing

- Quality Assurance (QA) and Testing are not the same, but they are related - Quality management, ties them together.

- Quality management includes both **quality assurance** and **quality control.**

1.2.2 Quality Assurance and Testing (Cont.)

Quality Assurance	Quality Control
It is an preventive approach – prevents the faults from occurring by providing rules and methods.	It is a corrective approach – corrects the faults when they occur.
It is a task conducted in the process.	It is a task conducted on the product.
Gives confidence to customer.	Gives confidence to producer.
It is a set of planned and systematic set of activities that provides adequate confidence and assures that the product conforms to specified requirements.	It is the process by which product quality is compared with applicable standards and appropriate action is taken when non-conformance is detected.
Entire team (designers, coders, testers, etc.) is responsible for QA.	Only tester is responsible for QC.
Examples : use of CASE (engineering) and CAST (testing) tools, training and	Examples: Walkthrough, inspections, etc.

1.2.3 Errors, Defects and Failures

- A person can make an error (mistake), which can lead to the introduction of a defect (fault or bug) in the software code or in some other related work product.
- An error that leads to the introduction of a defect in one work product can trigger an error that leads to the introduction of a defect in a related work product.

Example :

- A requirements elicitation error can lead to a requirements defect, which then results in a programming error that leads to a defect in the code. If a defect in the code is executed, this may cause a failure, but not necessarily in all circumstances. For example, some defects require very specific inputs or preconditions to trigger a failure, which may occur rarely or never.

1.2.3 Errors, Defects and Failures (Cont.)

- Error(Mistake): A human action that produces an incorrect result
- Fault: A stage caused by an error which leads to unintended functionality of the program
- Bug: It is an evidence of the fault. It causes the program to perform in unintended manner. It is found before application goes into beta version
- Failure: Inability of the system to perform functionality according to its requirement
- Defect: It is a mismatch of the actual and expected result identified while testing the software in the beta version

Reasons behind Errors

- Time pressure
- Human fallibility
- Inexperienced or insufficiently skilled project participants
- Miscommunication between project participants, including miscommunication about requirements and design
- Complexity of the code, design, architecture, the underlying problem to be solved, and/or the technologies used
- Misunderstandings about intra-system and inter-system interfaces, especially when such intra-system and inter-system interactions are large in number
- New, unfamiliar technologies
- Environmental conditions - for example, radiation, electromagnetic fields, and pollution can cause defects in firmware or influence the execution of software by changing hardware conditions.

1.2.4 Defects, Root Causes and Effects

- The root causes of defects are the earliest actions or conditions that contributed to creating the defects.
- Defects can be analyzed to identify their root causes.
- By focusing on most significant root causes, root cause analysis can lead to process improvements that prevent future defects from being introduced.

Example :

- Suppose incorrect interest payments, due to a single line of incorrect code, results in customer complaints. The defective code was written for a user story which was ambiguous, due to the product owner's misunderstanding of how to calculate interest. Therefore, it means that :
  - customer complaints – **effects**
  - incorrect interest payments – **failures**
  - improper calculation in the code – **defect**
  - lack of knowledge on the part of the product owner - **root cause** of this defect
  - Due to lack of knowledge, Product owner makes a **mistake** while writing user story.

Cost of Software Defects

It is Easy to find and fix defect in early stages rather than in the later phases of software.

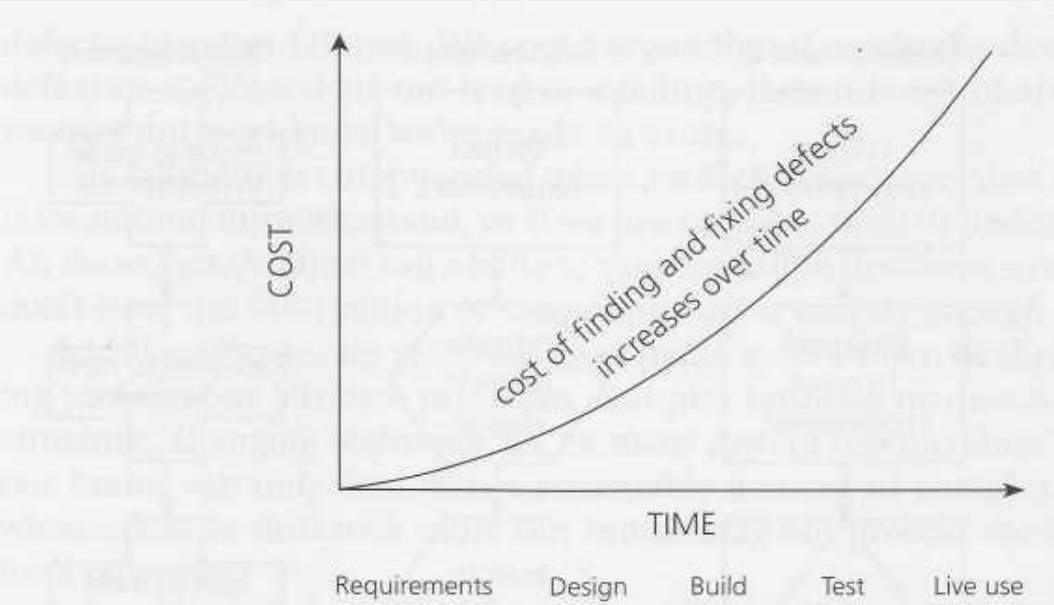


FIGURE 1.2 Cost of defects

Importance of Testing Early in SDLC Phases

- Prevents future Problems thus lowering the cost
- Testing will not be a bottleneck anymore
- Testers become more familiar with the software, as they are more involved with the evolution of the product.
- Reduces the chances of failure
- The test environment can be prepared in advance
- The risk of having a short time for testing is greatly reduced
- Maintains “quality culture” in the organization

1.3 Seven Testing Principles

Principle 1 - Testing shows presence of defects, not their absence

Principle 2 - Exhaustive testing is impossible

Principle 3 - Early testing saves time and money

Principle 4 - Defects Cluster together

Principle 5 – Beware of the Pesticide Paradox

Principle 6 - Testing is context dependent

Principle 7 - Absence of Errors is fallacy

Economics of Testing

Economics of Testing

- It is both the driving force and the limiting factor

Driving - Earlier the errors are discovered and removed in the lifecycle, lowers the cost of their removal.

Limiting - Testing must end when the economic returns cease to make it worth while i.e. the costs of testing process significantly outweigh the returns

Scope of Software Testing

Bad news : You can't test everything

Good news : There is such a thing as "good enough"

Bad news : Good enough may cost too much

What do we do : Increase focus via systematic process of elimination

What you might test?

- Those areas which are within the scope of your project

What you should test?

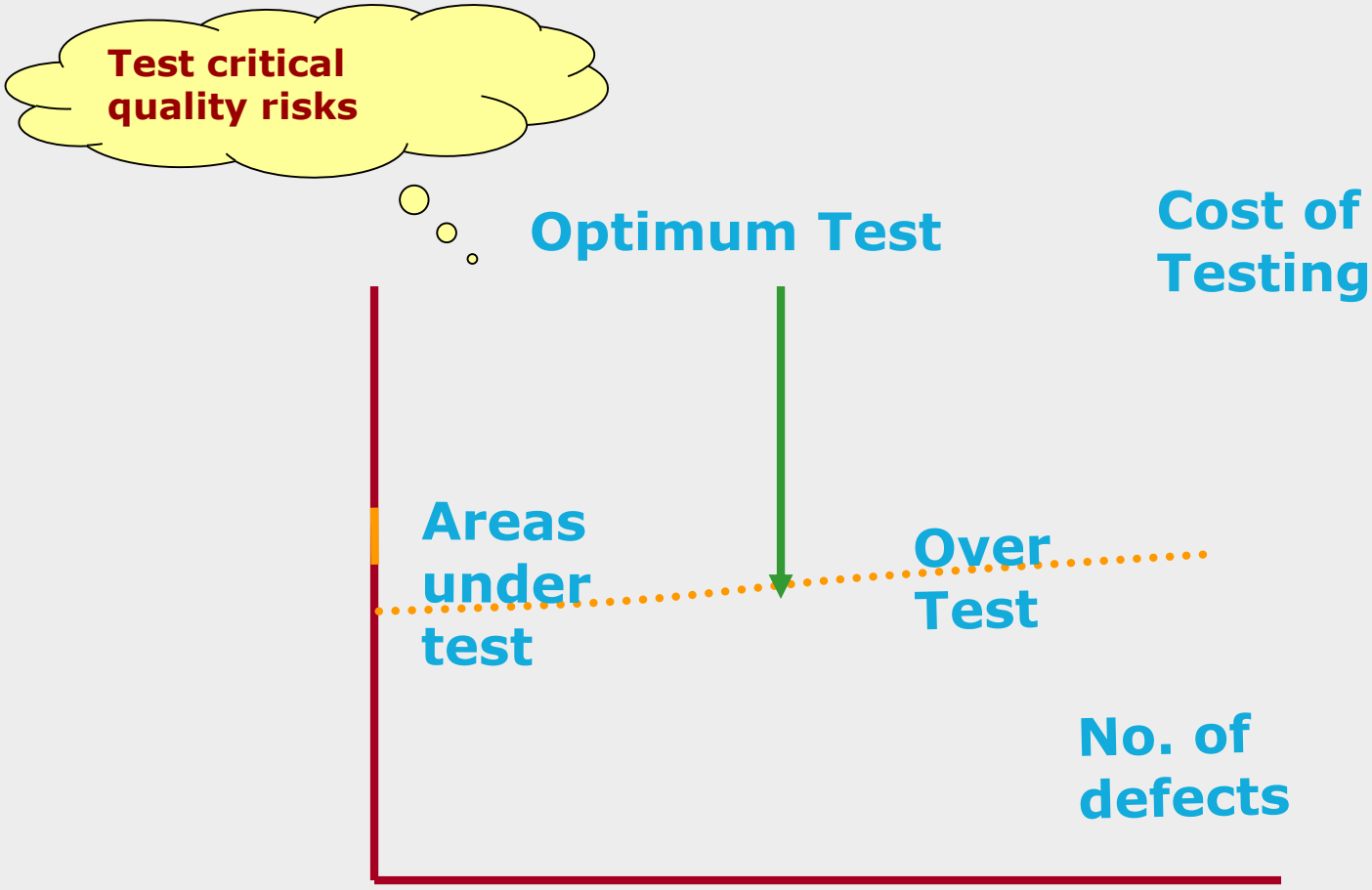
- The critical system functionality which effects the customers & users experience of quality

What you can test?

- Estimate time & resource for risk driven effort

Scope of Software Testing

"Understanding risk is the key to Optimum testing"



Factors influencing the Scope of Testing

Contractual requirements

Legal requirements

- Privacy related laws
- Non-disclosure of identity

Industry-specific requirements

- Aircraft safety equipment

Scope of Testing is about identifying the correct test cases for automation

The steps involved are:

- Identify various factors that form the basis of identifying the candidate test cases
- Apply 'Divide & rule' strategy : Break the application into smaller modules
- Analyze each module to identify the candidate test cases
- Calculate ROI

Factors influencing the scope of testing :

- In small projects

  Test case writing
  Test case execution
  Regression testing

- In Large projects

  Setting up test bed
  Generating test data, test scripts, etc.

1.4 Test Process

Testing is a process rather than a single activity.

The quality and effectiveness of software testing is primarily determined by the quality of the test processes used.

The activities of testing can be divided into the following basic steps:

- Planning and Control
- Analysis and Design
- Implementation and Execution - Evaluating and Reporting
- Test Completion/Closure activities

1.4.1 Test Process in Context

The proper, specific software test process in any given situation depends on many factors.

Few of the Contextual factors that influence the test process are :

- SDLC model and project methodologies being used
- Test levels and test types being considered
- Product and project risks
- Business domain
- Operational constraints that include :
  - Budgets and resources
  - Timescales
  - Complexity
  - Contractual and regulatory requirements
- Organizational policies and practices
- Required internal and external standards

1.4.1 Test Process in Context (Cont.)

The following sections describe general aspects of organizational test processes in terms of the following:

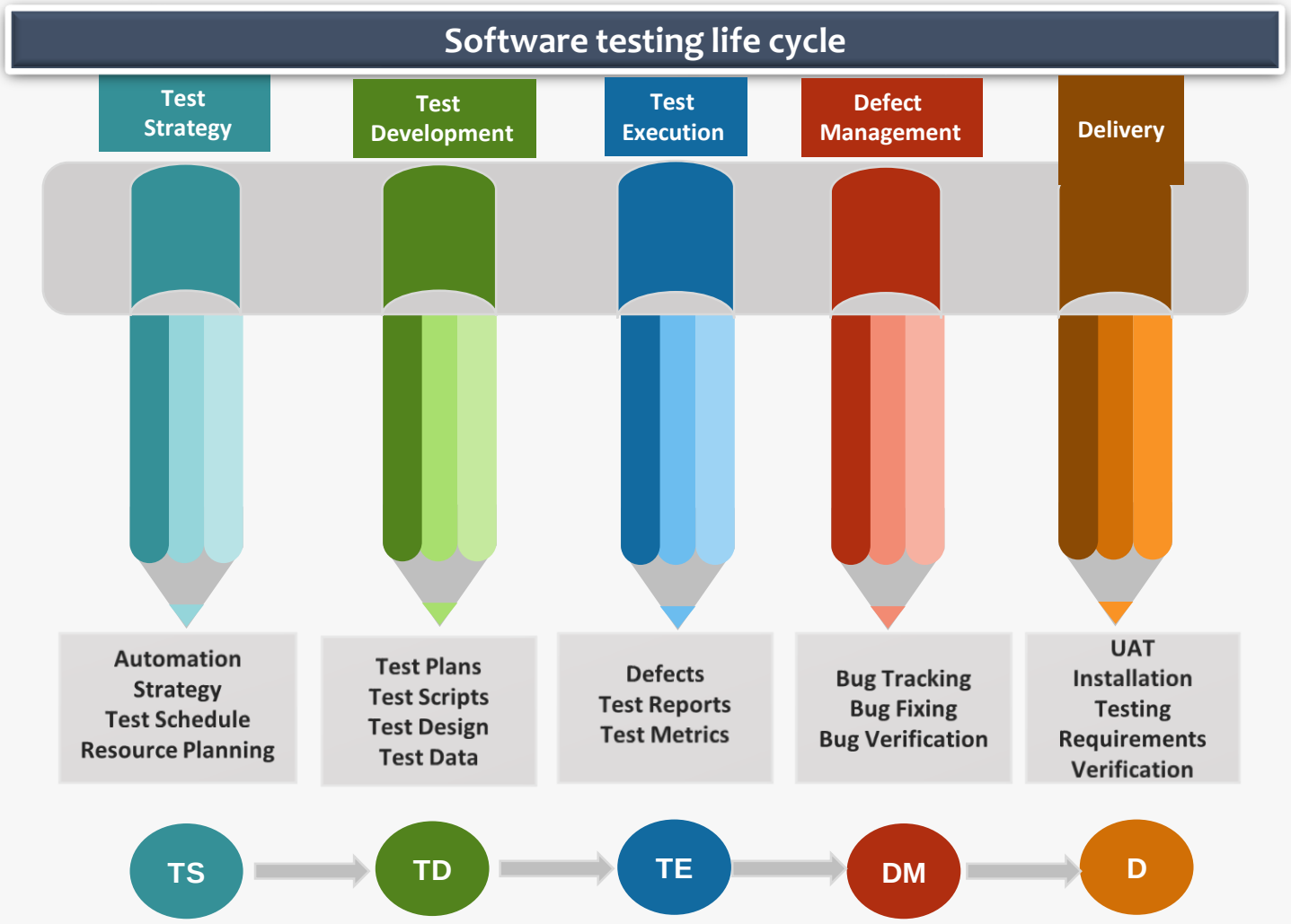
- Test activities and tasks
- Test work products
- Traceability between the test basis and test work products

It is very useful if the test basis (for any level or type of testing that is being considered) has measurable coverage criteria defined.

The coverage criteria can act effectively as key performance indicators (KPIs) to drive the activities that demonstrate achievement of software test objectives (defined in section 1.1.1).

1.4.2 Test Activities and Tasks

- Test planning
- Test monitoring and control
- Test analysis
- Test design
- Test implementation
- Test execution
- Test completion


Software testing life cycle
Test Strategy
Test Development
Test Execution
Defect Management
Delivery
Automation Strategy Test Schedule Resource Planning
Test Plans Test Scripts Test Design Test Data
Defects Test Reports Test Metrics
Bug Tracking Bug Fixing Bug Verification
UAT Installation Testing Requirements Verification
TS
TD
TE
DM
D

Test Planning

- Test planning involves activities that define the testing objectives (refer section 1.1.1) and the approach for meeting test objectives within constraints imposed by the context.

  For example, specifying suitable test techniques and tasks, and formulating a test schedule for meeting a deadline.

- Test plans may be revisited based on feedback obtained from monitoring and control activities.

Test Monitoring and Control

- Test monitoring involves an on-going comparison of actual progress against the test plan using any test monitoring metrics defined in the test plan.
- Test control involves taking actions necessary to meet the objectives of the test plan.
- Test monitoring and control are supported by the evaluation of exit criteria.
- For example, the evaluation of exit criteria for test execution as part of a given test level may include:
  - Checking test results and logs against specified coverage criteria
  - Assessing the level of component or system quality based on test results and logs
  - Determining if more tests are needed (e.g., if tests originally intended to achieve a certain level of product risk coverage failed to do so, requiring additional tests to be written and executed)

Test Analysis

Test analysis determines “what to test” in terms of measurable coverage criteria.

Test analysis includes the following major activities:

1. Analyzing the test basis appropriate to the test level being considered.
   - Requirement specifications, such as business requirements, functional requirements, system requirements, user stories, epics, use cases, or similar work products that specify desired functional and non-functional component or system behavior
   - Design and implementation information, such as system or software architecture diagrams or documents, design specifications, call flows, modelling diagrams (e.g., UML or ER diagrams), interface specifications, or similar work products that specify component or system structure
   - The implementation of the component or system itself, including code, database metadata and queries, and interfaces
   - Risk analysis reports, which may consider functional, non-functional, and structural aspects of the component or system

Test Analysis (Cont.)

2. Evaluating the test basis and test items to identify defects of various types, such as:
   - Ambiguities
   - Omissions
   - Inconsistencies
   - Inaccuracies
   - Contradictions
   - Superfluous statements

Test Analysis (Cont.)

3. Identifying features and sets of features to be tested
4. Defining and prioritizing test conditions for each feature based on analysis of the test basis, and considering functional, non-functional, and structural characteristics, other business and technical factors, and levels of risks
5. Capturing bi-directional traceability between each element of the test basis and the associated test conditions (see sections 1.4.3 and 1.4.4)

Test Design

- During test design, the test conditions are elaborated into high-level test cases, sets of high-level test cases, and other testware. So, test analysis answers the question “what to test?” while test design answers the question “how to test?”
- Test design includes the following major activities:
- Designing and prioritizing test cases and sets of test cases
- Identifying necessary test data to support test conditions and test cases
- Designing the test environment and identifying any required infrastructure and tools
- Capturing bi-directional traceability between the test basis, test conditions, test cases, and test procedures (see section 1.4.4)

The elaboration of test conditions into test cases and sets of test cases during test design often involves using test techniques (see chapter 4).

Test Implementation

- During test implementation, the testware necessary for test execution is created and/or completed, including sequencing the test cases into test procedures.
- Test design answers the question “how to test?” while test implementation answers the question “do we now have everything in place to run the tests?”

Test Implementation (Cont.)

Test implementation includes the following major activities:

- Developing and prioritizing test procedures, and, potentially, creating automated test scripts
- Creating test suites from the test procedures and automated test scripts
- Arranging the test suites within a test execution schedule in a way that results in efficient test execution (see section 5.2.4)
- Building the test environment (including, potentially, test harnesses, service virtualization, simulators, and other infrastructure items) and verifying that everything needed has been set up correctly
- Preparing test data and ensuring it is properly loaded in the test environment
- Verifying and updating bi-directional traceability between the test basis, test conditions, test cases, test procedures, and test suites (see section 1.4.4)

Test Execution

During test execution, test suites are run in accordance with the test execution schedule.

Test execution includes the following major activities:

- Recording the IDs and versions of the test item(s) or test object, test tool(s), and testware
- Executing tests either manually or by using test execution tools
- Comparing actual results with expected results
- Analyzing anomalies to establish their likely causes (e.g., failures may occur due to defects in the code, but false positives also may occur (see section 1.2.3))
- Reporting defects based on the failures observed (see section 5.6)
- Logging the outcome of test execution (e.g., pass, fail, blocked)
- Repeating test activities either as a result of action taken for an anomaly, or as part of the planned testing (e.g., confirmation testing, or regression testing)
- Verifying and updating bi-directional traceability between the test basis, test conditions, test cases, test procedures, and test results.

Test Completion

Test completion includes the following major activities:

- Checking whether all defect reports are closed, entering change requests or product backlog items for any defects that remain unresolved at the end of test execution
- Creating a test summary report to be communicated to stakeholders
- Finalizing and archiving the test environment, the test data, the test infrastructure, and other testware for later reuse
- Handing over the testware to the maintenance teams, other project teams, and/or other stakeholders who could benefit from its use
- Analyzing lessons learned from the completed test activities to determine changes needed for future iterations, releases, and projects
- Using the information gathered to improve test process maturity.

1.4.3 Test Work Products

Test work products are created as part of the test process. Many of the test work products can be captured and managed using test management tools and defect management tools (see chapter 6).

- Test planning work products
- Test monitoring and control work products
- Test analysis work products
- Test design work products
- Test implementation work products
- Test execution work products
- Test completion work products

1.4.4 Traceability between Test Basis and Test Work Products

It is important to establish and maintain traceability throughout the test process between each element of the test basis and the various test work products associated with it.

In addition to the evaluation of test coverage, good traceability supports:

- Analyzing the impact of changes
- Making testing auditable
- Meeting IT governance criteria
- Improving the understandability of test progress reports and test summary reports to include the status of elements of the test basis (e.g., requirements that passed their tests, requirements that failed their tests, and requirements that have pending tests)
- Relating the technical aspects of testing to stakeholders in terms that they can understand
- Providing information to assess product quality, process capability, and project progress against business goals.

Attributes of a good Tester

- A good test engineer has a 'test to break' attitude
- Need to take a different view, a different mindset (“What if it isn’t?”, “What could go wrong?”)
- An ability to understand the point of view of the customer
- A passion for quality and attention to detail
- Notice little things that others miss/ignore (See symptom not bug)
- Ability to communicate fault information to both technical (developers) and non-technical (managers and customers)
- Tact and diplomacy for maintaining a cooperative relationship with developers
- Work under worst time pressure (at the end)
- “Patience”

1.5 Psychology of Testing

- Identifying defects during a static test or identifying failures during dynamic test execution, was perceived as criticism of the product and of its author.
- Since developers expect their code to be correct, they have a confirmation bias that makes it difficult to accept that the code is incorrect.
- As a result of these psychological factors, some people may perceive testing as a destructive activity, even though it contributes greatly to project progress and product quality
- To try to reduce these perceptions and tensions between the testers and the analysts, product owners, designers, and developers, the information about defects and failures should be communicated in a constructive way.

1.5.1 Human Psychology and Testing

Testers and test managers need to have good interpersonal skills to be able to communicate effectively about defects, failures, test results, test progress, and risks, and to build positive relationships with colleagues.

Ways to communicate well include the following examples:

- Start with collaboration rather than battles. Remind everyone of the common goal of better quality systems.
- Emphasize on the benefits of testing. For example, for the authors, defect information can help them improve their work products and their skills. For the organization, defects found and fixed during testing will save time and money and reduce overall risk to product quality.
- Communicate test results and other findings in a neutral, fact-focused way without criticizing the author.
- Confirm that the other person has understood what has been said and vice versa.
- Clearly defining the right set of test objectives.

Code of Ethics for Tester

Involvement in software testing enables individuals to learn confidential and privileged information. A code of ethics is therefore necessary, among other reasons to ensure that the information is not put to inappropriate use.

PUBLIC - Tester shall act consistently with public interest

CLIENT AND EMPLOYER - Tester shall act in best interests of their client and employer

PRODUCT - Tester shall ensure that the deliverables they provide meet highest professional standards possible

JUDGMENT - Tester shall maintain integrity and independence in their professional judgment

MANAGEMENT - Tester managers and leaders shall subscribe to and promote an ethical approach to manage software testing

PROFESSION - Tester shall advance the integrity and reputation of the profession consistent with the public interest

COLLEAGUES - Tester shall be fair to and supportive of their colleagues, and promote cooperation with software developers

SELF - Tester shall participate in lifelong learning and shall promote an ethical approach to the practice of the profession

1.5.2 Tester's and Developer's Mindsets

- Developers and testers often think differently. Bringing these mindsets together helps to achieve a higher level of product quality.
- A mindset reflects an individual's assumptions and preferred methods for decision making and problem solving.
- A tester's mindset should include curiosity, professional pessimism, a critical eye, attention to detail, and a motivation for good and positive communications and relationships.
- A tester's mindset tends to grow and mature as the tester gains experience.
- A developer's mindset may include designing and building solutions than in contemplating what might be wrong with those solutions.
- In addition, confirmation bias makes it difficult to find mistakes in their own work. With the right mindset, developers are able to test their own code.
- Having some of the test activities done by independent testers increases defect detection effectiveness, which is particularly important for large, complex, or safety-critical systems.

Limitations of Software Testing

Even if we could generate the input, run the tests, and evaluate the output, we would not detect all faults

Correctness is not checked

- The programmer may have misinterpreted the specs, the specs may have misinterpreted the requirements

There is no way to find missing paths due to coding errors

Summary

In this lesson, you have learnt:

- Testing is an extremely creative & intellectually challenging task
- No software exists without bug
- Testing is conducted with the help of users requirements, design documents, functionality, internal structures & design, by executing code
- Scope of testing
- The cost of not testing is potentially much higher
- Testing is in a way a destructive process
- A successful test case is one that brings out an error in program
- Various principles of testing

Review Question

Question 1: What is visible to end-users is a deviation from the specific or expected behavior is called as

- Defect
- Bug
- Failure
- Fault

Question 2: _____ is a planned sequence of actions.

Question 3: Pick the best definition of Quality :

- Quality is job done
- Zero defects
- Conformance to requirements
- Work as designed

Question 4: One cannot test a program completely to guarantee that it is error free (T/F)

Question 5: One can find missing paths due to coding errors (T/F)