SESSION- 7

Software testing

Dr. Bharati. V W

WHY IS TESTING NECESSARY?

- the way in which a defect in software can cause harm to a person, to the environment or to a company.
- Distinguish between the root cause of a defect and its effects.
- testing is part of quality assurance and how testing contributes to higher quality.
- Recall the terms 'mistake', 'defect', 'fault', 'failure‘ and 'bug'.

Causes of software defects

- Why is it that software systems sometimes don't work correctly? We know that people make mistakes - we are fallible [imperfect/human errors]
- **Error:**
- Human action that produces an incorrect result.
- **If someone makes an error or mistake in using the software, this may lead directly to a problem - the software is used incorrectly and so does not behave as we expected.**

- **defects or bugs or faults**

- **However, people also design and build the software and they can make mistakes during the design and build. These mistakes mean that there are flaws in the software itself. These are called defects or sometimes bugs or faults.**
- **.**

What is ISTQB

"International Software Testing Qualifications Board

The **International Software Testing Qualifications Board** (ISTQB) is a software testing certification board that operates internationally. Founded in Edinburgh in November 2002, the ISTQB is a non-profit association legally registered in Belgium.

What is ISTQB? Full form of ISTQB is **"International Software Testing Qualifications Board."** It offers internationally recognized certifications called "ISTQB Certified Tester." Why should I take the certification?

Terms in software testing

- Error [USER/ CODER]
- An error is a mistake made by human that leads to discrepancy between the actual and the expected result.

- Defect [TESTER]
- A defect is a problem in the functioning of a software system during testing. ISTQB defines a defect as "A flaw in a component or system that can cause the component or system to fail to perform its required function, e.g., an incorrect statement or data definition."

- Fault [SYSTEM]
- A fault is an incorrect step, process or data definition in a software product.

- Bug [DEVELOPER CLEARS]
- A bug is a flaw in a software system that causes the system to behave in an unintended manner.

- Failure [SYSTEM FAILURE]
- A failure is the inability of a software system to perform its operations within the specified performance benchmark. As per ISTQB, "a defect, if encountered during execution, may cause a failure of the component or system".

Terms in software testing

- So, we can say that a mistake made by humans during coding is called error, an error found during the testing phase is called a defect, a defect to be resolved by the development team is called a bug and when a build does not meet its specifications then it is termed as failure.

- **Failure**

- **When the software code has been built , it is executed and then any defects may cause the system to fail to do what it should do causing a failure.**
- **Not all the defects result in failures, and some are in dormant [inactive] in the code and we may never notice them.**
- **Deviation of the component or system from its expected delivery, service or result.**

Does our mistakes matter? yes

- Let's think about the consequences of mistakes. We agree that any human being, programmers and testers included, can make an error. These errors may produce defects in the software code or system, or in a document.
- If a defect in code is executed, the system may experience a failure. So the mistakes we make matter partly because they have consequences for the products for which we are responsible.

Functional and non-functional testing

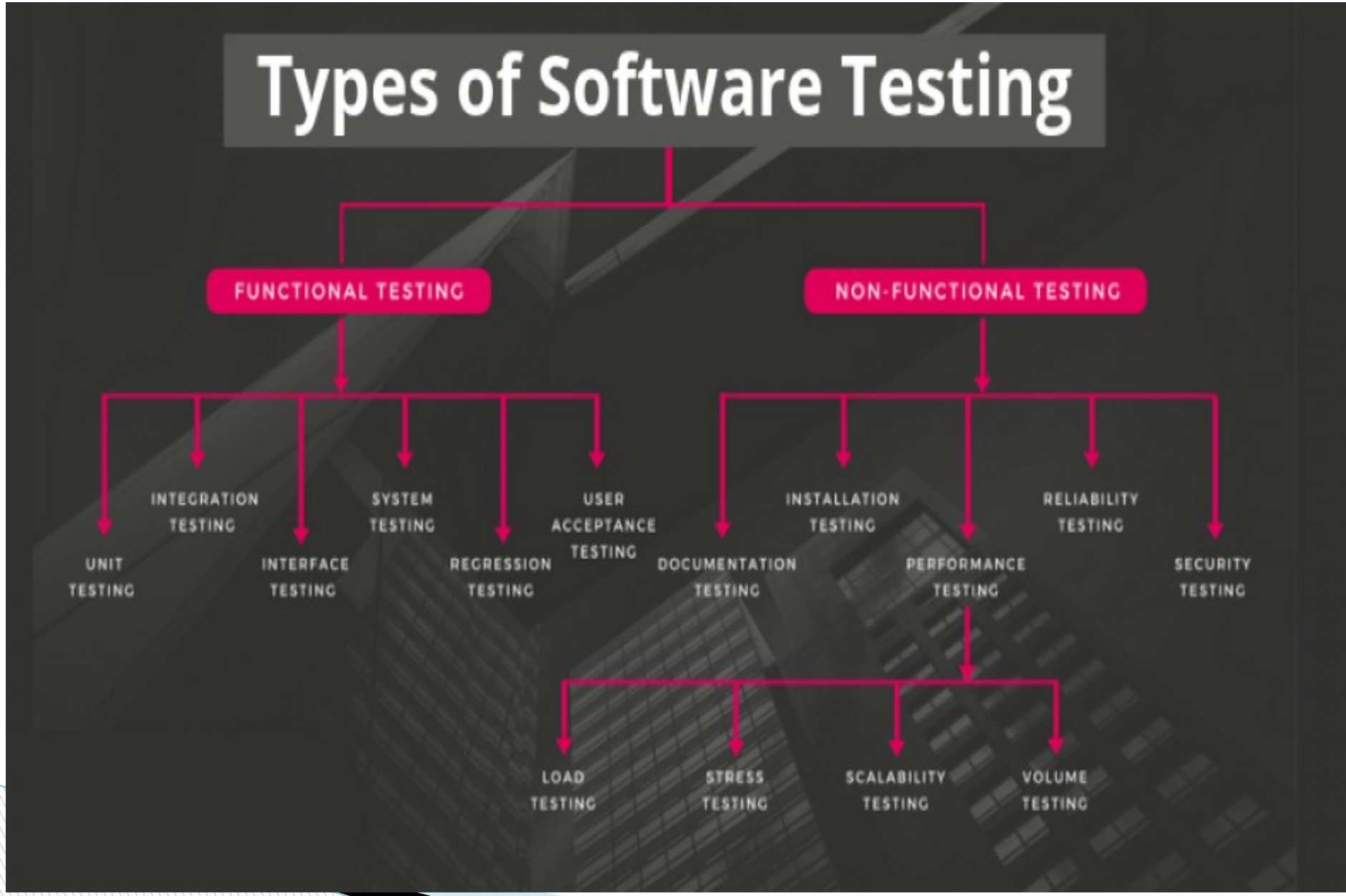
- Functional testing is a type of testing which verifies that each **function** of the software application operates in conformance with the requirement specification. This testing mainly involves black box testing, and it is not concerned about the source code of the application.
- Non-functional testing is a type of testing to check non-functional aspects (performance, usability, reliability, etc.) of a software application. It is explicitly designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

Examples of Functional Testing Types

- Unit testing
- Smoke testing
- User Acceptance
- Integration Testing
- Regression testing
- Localization
- Globalization
- Interoperability

Examples of Non-functional Testing Types

- Performance Testing
- Volume Testing
- Scalability
- Usability Testing
- Load Testing
- Stress Testing
- Compliance Testing
- Portability Testing
- Disaster Recover Testing


Types of Software Testing
FUNCTIONAL TESTING
NON-FUNCTIONAL TESTING
UNIT TESTING
INTEGRATION TESTING
INTERFACE TESTING
SYSTEM TESTING
REGRESSION TESTING
USER ACCEPTANCE TESTING
DOCUMENTATION TESTING
INSTALLATION TESTING
PERFORMANCE TESTING
RELIABILITY TESTING
SECURITY TESTING
LOAD TESTING
STRESS TESTING
SCALABILITY TESTING
VOLUME TESTING

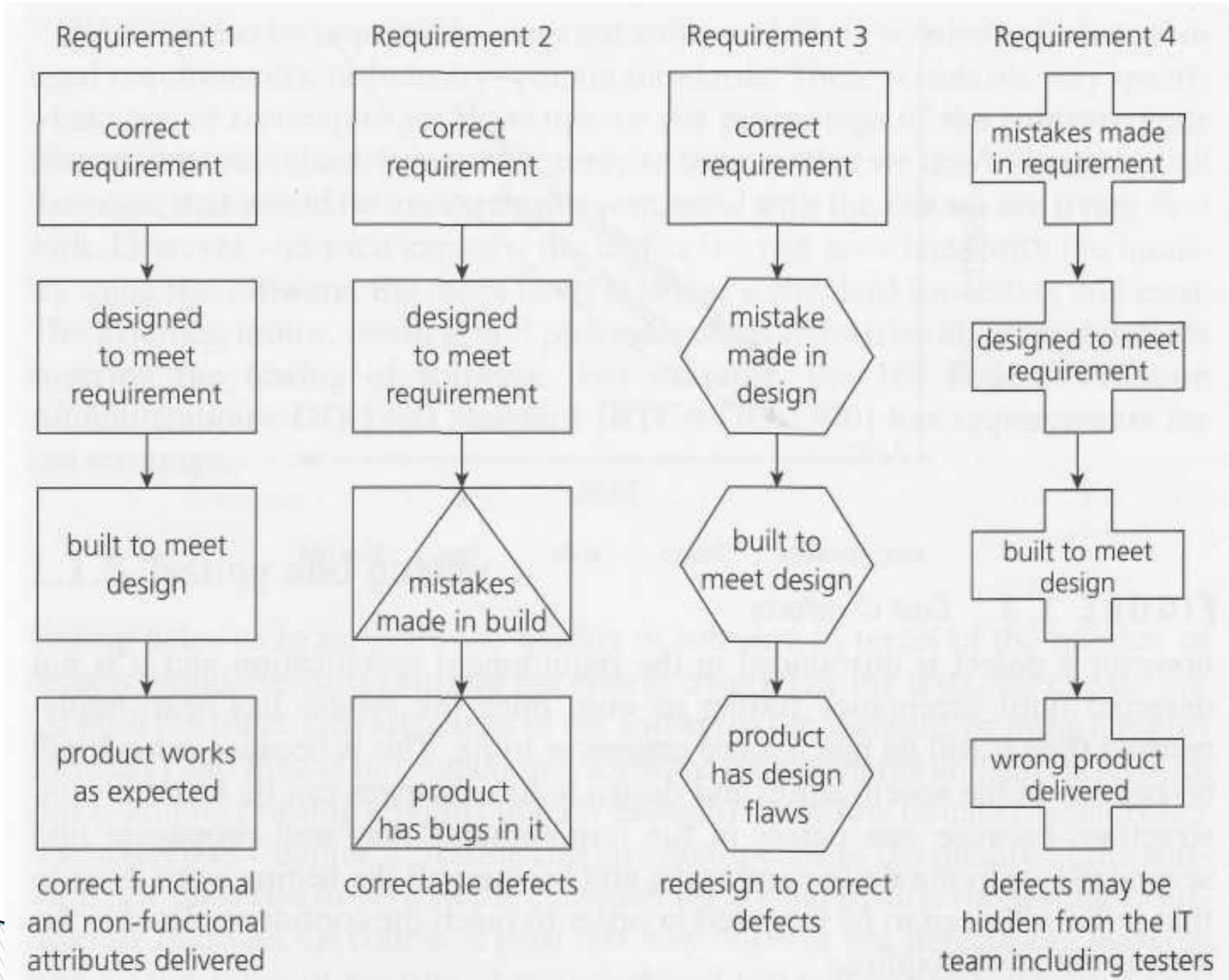


FIGURE 1.1 Types of error and defect

When do defects arise?

- In Figure 1.1 we can see how defects may arise in four requirements for a product

What is the cost of defects?

- As well as considering the impact of failures arising from defects we have not found, we need to consider the impact of when we find those defects.
- The cost of finding and fixing defects rises considerably across the life cycle; think of the old English proverb 'a stitch in time saves nine'.

- "A stitch in time saves nine" is an old English proverb which means that you should fix something as soon as it is damaged, do not wait to deal with a specific problem. If not, things will get worse and the problems will take longer to deal with.
- In other words, this proverb means that a timely effort that will prevent more work later. It certainly means prevention is better than cure.

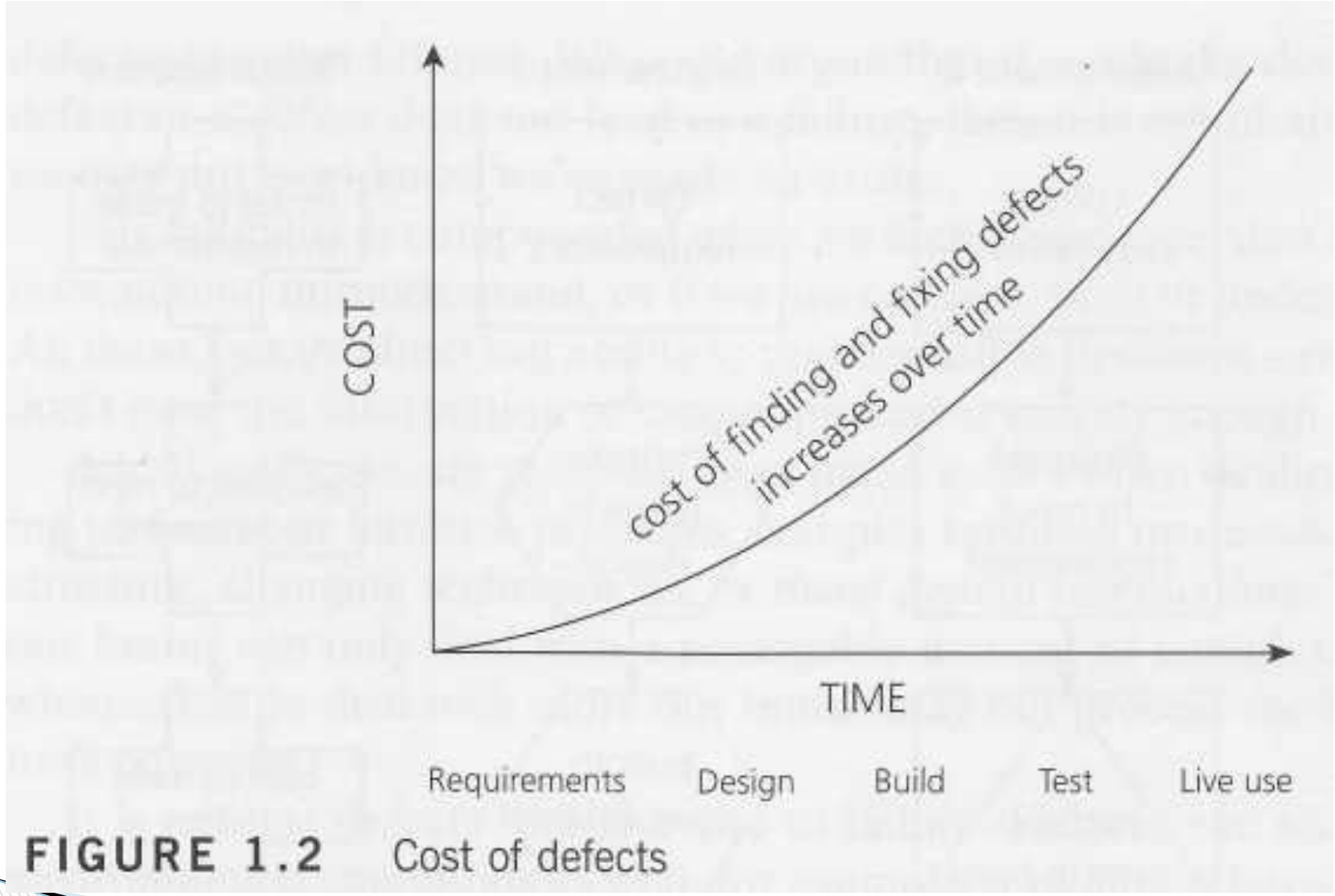


FIGURE 1.2 Cost of defects

WHAT IS TESTING?

- **ISTQB® (International Software Testing Qualifications Board)** is a not-for-profit association legally registered in Belgium.

- Let's break the definition down into parts; the definition has some key phrases to remember. The definition starts with a description of testing as a process and then lists some objectives of the test process.

testing as a process:

- *Process.*
- *All life cycle activities*
- *Both static and dynamic*
- *Planning*
- *Preparation*
- *Software products and related work products*

the some of the objectives for testing -the reasons why we do it:

- Determine that (software products) satisfy specified requirements
- Demonstrate that (software products) are fit for purpose
- Detect defects

What is Software Testing

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the **developed software met** the specified requirements or not and to identify the defects to ensure that the product is defect-free in order to produce a quality product.

What is ST?

- "Software testing is the process to prove that the software works correctly“
- “Testing is the process to prove that the software doesn't work“
- “Testing is the process to detect the defects and minimize the risks associated with the residual[available/ remaing] defects"

When can we meet our test objectives?

- **Testing Principle - Early testing**
  - Testing activities should start as early as possible in the software or system development life cycle and should be focused on defined objectives.

- We can use both dynamic testing and static testing. which often include:

1. finding defects;
2. gaining confidence in and providing information about the level of quality;
3. preventing defects.

7 TESTING PRINCIPLES

Principle 1:	**Testing shows presence of defects**	**Testing can show that defects are present, but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness**

Principle 2:	**Exhaustive testing is impossible**	**Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases. Instead of exhaustive testing, we use risks and priorities to focus testing efforts.**
Principle 3:	**Early testing**	Testing activities should start as early as possible in the software or system development life cycle and should be focused on defined objectives.

Principle 4:	**Defect clustering**	**Most defects are found in just a few parts of the test object. Defects are not evently distributed, but clustered together. Thus if many defects are detected in one place, ther are normally more defects nearby.**
Principle 5:	**Pesticide paradox**	If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new bugs. To overcome this 'pesticide paradox', the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.

Principle 6:	**Testing is context dependent**	**Testing must be adapted to the risks inherent in the use and environment of the application. Every s/w system the test exit criteria, should be decided upon individually . Testing is done differently in different contexts. For example, safety-critical software is tested differently from an e-commerce site.**
Principle 7:	**Absence-of-errors fallacy [the fallacy of assuming that no failures means a useful system]**	Finding failures and repairing defects does not guarantee that the system as a whole meets user expectations and needs. Early involvement of the users in the development process and the use of prototypes are preventive measures.

- **Fault** : It is a condition that causes the software to fail to perform its required function.
  Error : Refers to difference between Actual Output and Expected output.
  Failure : It is the inability of a system or component to perform required function according to its specification.
  IEEE Definitions
  Failure: External behavior is incorrect
- **Fault:** Discrepancy in code that causes a failure.
- **Error:** Human mistake that caused fault
- **Note:**
  Error is terminology of Developer.
- **Bug** is terminology of Tester

THE PSYCHOLOGY OF TESTING

- Independent testing
- Why do we sometimes not get on with the rest of the team
- *the objective of testing is to uncover as many bugs as possible. The testing has to be done without any emotional attachment to the software. If someone points out the bugs, it is only to improve the quality of the software rather than to find fault "*

-developer test
-Blindness to one's own errors
-independent testing team
-reporting of failures
-mutual compreshension

Fundamental Test Process-

- ✔ test planning and control,
- ✔ test analysis and design,
- ✔ test implementation and execution,
- ✔ evaluation of the test exist criteria,
- ✔ test closure activities ,
- ✔ psychology of testing and
- ✔ 7 principles of testing.

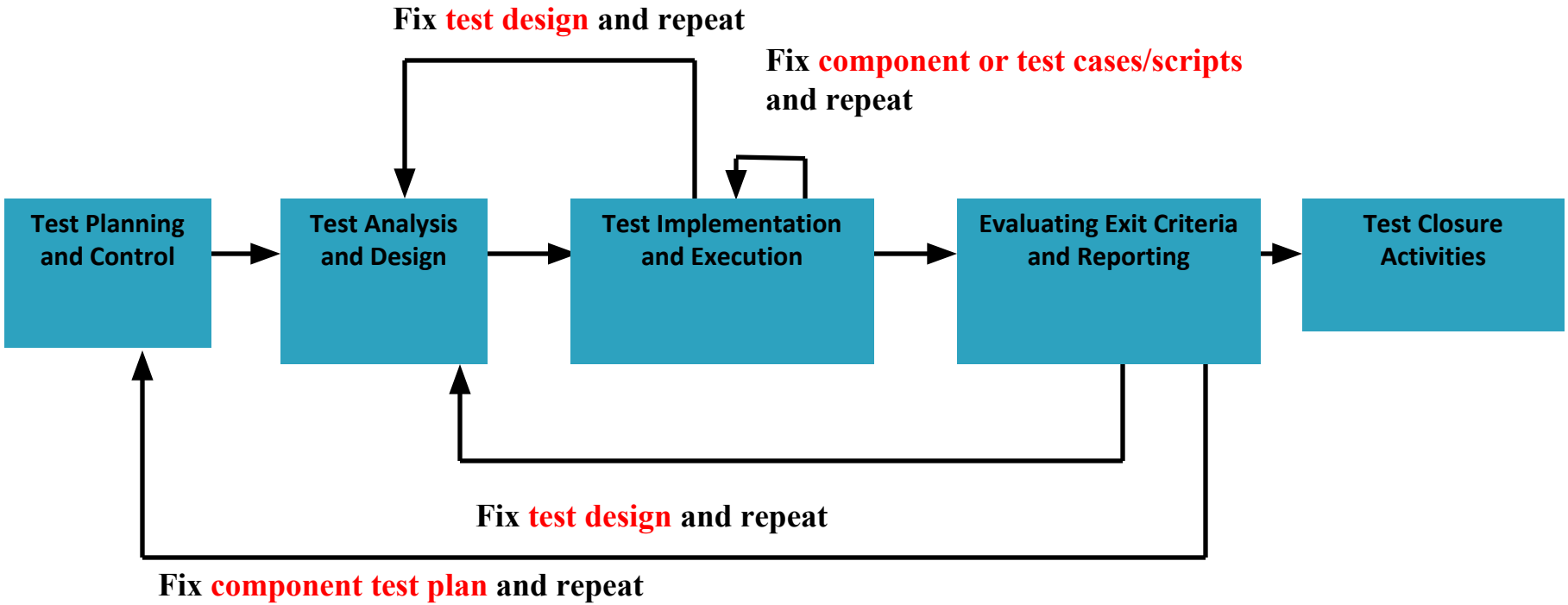
Fundamental Test Process

The five stages of the fundamental test process

- **Test Planning and Control**
- **Test Analysis and Design**
- **Test Implementation and Execution**
- **Evaluating Exit Criteria and Reporting**
- **Test Closure Activities**

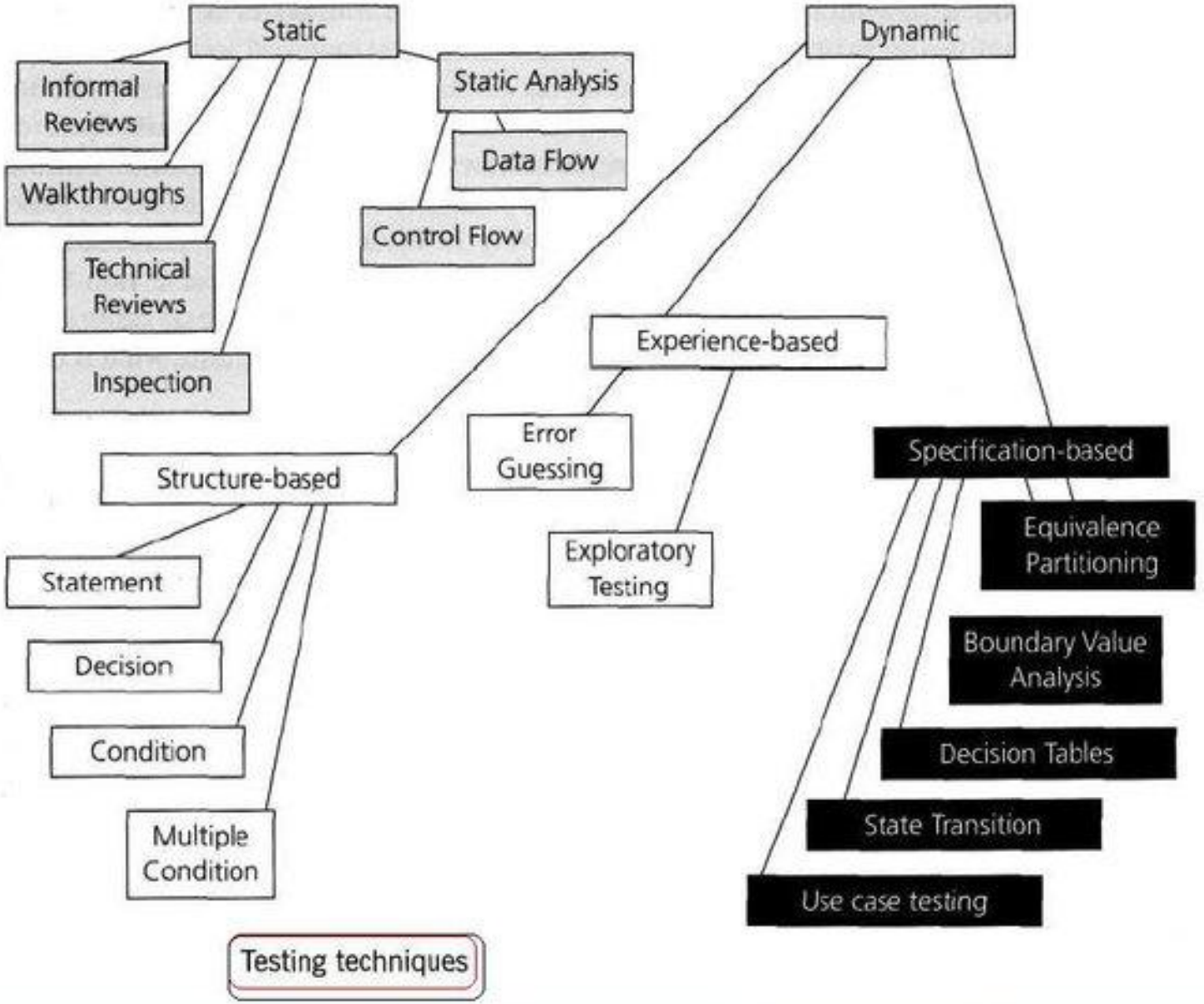
Fundamental Test Process

5 Phases of the Fundamental Test Process



FUNDAMENTAL TEST PROCESS

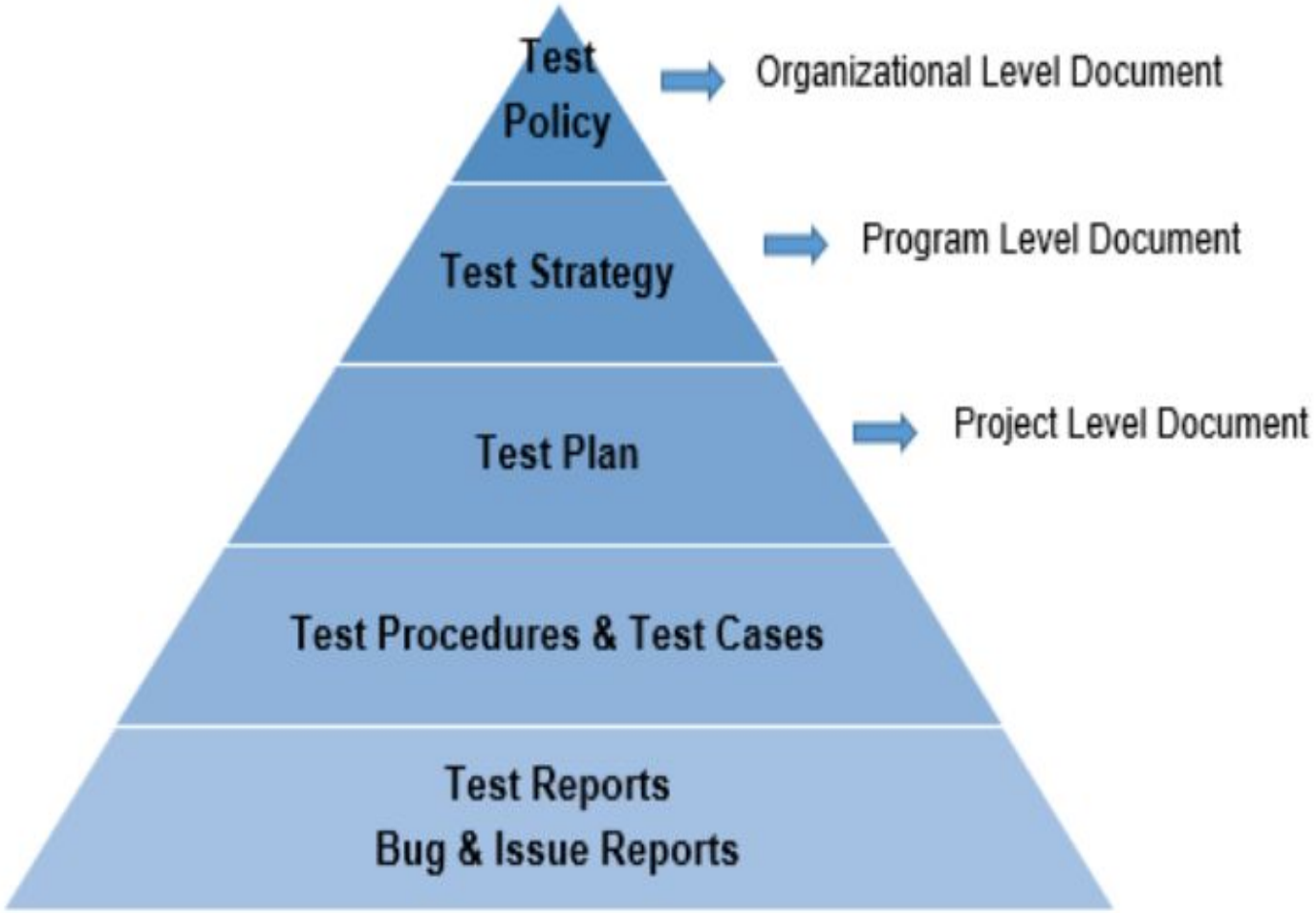
- The fundamental test process and activities. Will start with test planning and continue through to test closure.
- These activities are logically sequential, but, in a particular project, may overlap, take place concurrently and even be repeated.
- This process is particularly used for dynamic testing, but the main headings of the process can be applied to reviews as well.
- For example, we need to plan and prepare for reviews, carry out the reviews, and evaluate the outcomes of the reviews. For some reviews, such as inspections, we will have exit criteria and will go through closure activities.


Static
Informal Reviews
Walkthroughs
Technical Reviews
Inspection
Static Analysis
Data Flow
Control Flow
Dynamic
Experience-based
Error Guessing
Exploratory Testing
Specification-based
Equivalence Partitioning
Boundary Value Analysis
Decision Tables
State Transition
Use case testing
Structure-based
Statement
Decision
Condition
Multiple Condition
Testing techniques

planning and control;

Test planning and control

- During **test planning, understand the goals and objectives of the customers, stakeholders, and the project, and the risks which testing is intended to address. This will give us what is sometimes called the mission of testing or the test assignment.**
- There are three main high level test documentation, which I came across all through software testing. These documents are:
  - Test Policy
  - Test Strategy
  - Test Plan


Test Policy
Organizational Level Document
Test Strategy
Program Level Document
Test Plan
Project Level Document
Test Procedures & Test Cases
Test Reports
Bug & Issue Reports

- **TEST POLICY:**
- Test policy is a document described at the organization level and gives the organizational insight for the test activities.
- It is determined by the senior management of the organization and defines the test principles that the organization has adopt. The test policy is very high-level document and at the top of the test documentation structure.
- Organizations may prefer to publish their test policy in a sentence, as well as a separate document. Also they may use this policy in both development and maintenance projects.
- The test policy shall describe the followings:
  - Clear answer to the question of "What does testing means for the organization"
  - Test objectives that the organization have
  - The definition of the testing process used by the organization to increase the quality of the software developed
  - How the organization will measure the effectiveness and efficiency of the test while achieving goals
  - How the organization will improve its test processes

TEST STRATEGY:

- Test strategy document is prepared at the program level and includes general test strategy, management principles, processes and approaches for the tests to be performed for a software in detail.
- The test strategy document is also a high level document and is usually written by the test manager and the project manager in the top level organization. It is generally prepared in large scale projects and does not need much updating. In small scale projects, test strategies and test approach may be included in the test plan, and also the test strategy document may not be written separately.
- Test approach and test activities included in the test strategy document must be consistent with the test policies of the organization.
- The test strategy document may applicable for a program / system that contains multiple projects and describes;

- Objective / scope of testing
- In-scope / out of scope items for testing
- Test levels (Unit, System, Integration, System Integration)
- Test types (Functional / Non-Functional)
- Entry / Exit / Stop / Resumption Criteria for testing (for different levels / phases)
- Risks to be addressed
- Test environment
- Test case design methodology
- Test methodology (Top-down / bottom-up / risk based)
- Test control and reporting
- Test automation approach
- Test tools to be used
- Defect management approach
- Defect classification
- Retesting & regression approach

- **TEST PLAN:**
- Test plan is a document prepared at the project level. In general it defines work products to be tested, how they will be tested (test cases) and test type distribution among testers. Test plan also includes test environment and test tools to be used during the project, the persons responsible for the tests and their responsibilities, test levels and test types, test schedule planned for test runs, and the principles of management and reporting of errors / bugs.
- Test plan is usually prepared by the test manager or test leader in the test organization and shared with the entire team in the project. It is a living document throughout the project and should be kept under revision control as it's updated.
- The information in the test plan document must be consistent with the organization's test policy and test strategy.
- The test plan may describe the followings:
  - All test strategies specific to the project
  - Test estimations & test schedule
  - Test organization / roles / responsibilities
  - Test deliverables
  - Test reporting principles
- IEEE Std 829 (IEEE Standard for Software Test Documentation) gives a "Test Plan Template" that would be useful for those who will prepare a test plan.

1 Determine the scope and risks and identify the objectives of testing:

- consider what software, components, systems or other products are in scope for testing; the business, product, project and technical risks which need to be addressed;
  - ✔ whether we are testing primarily
- ❖ to uncover defects,
- ❖ to show that the software meets requirements,
- ❖ to demonstrate that the system is fit for purpose or to measure the qualities and attributes of the software.

2 Determine the **test approach** (techniques, test items, coverage, identifying and interfacing with the teams involved in testing, test ware):

- **consider how we will carry out the testing, the techniques to use, what needs testing and how extensively (i.e. what extent of coverage).**
- **We'll look at who needs to get involved and when (this could include developers, users, IT infrastructure teams);**
- **decide what we are going to produce as part of the testing (e.g. testware / test tools such as test procedures and test data). This will be related to the requirements of the test strategy.**

Testware includes documentation, scripts, inputs, expected results, set-up and clear-up procedures, files, databases, environment, and any additional**software** or utilities used in **testing**. Generally, **Testware** is also called as**Testing** Tools.

3. Implement the test policy and/or the test strategy:

- there may be an organization or program policy and strategy for testing. If this is the case, during our planning we must ensure that what we plan to do adhere[stick] to the policy and strategy.
- Test policy: A high level document describing the principles, approach and major objectives of the organization regarding testing
- Test strategy: a high level description of the test levels to be performed and testing within those levels for an organization or prg [one or more projects]

4 Determine the required test resources (e.g. people, test environment, PCs): from the planning we have already done we can now go into detail; we decide on our team make-up and we also set up all the supporting hardware and software we require for the test environment.

5 Schedule test analysis and design tasks, test implementation, execution and evaluation: we will need a schedule of all the tasks and activities, so that we can track them and make sure we can complete the testing on time.

6 Determine the **exit criteria**:

- **we need to set criteria such as coverage criteria (for example, the percentage of statements in the software that must be executed during testing) that will help us track whether we are completing the test activities correctly.**
- **They will show us which tasks and checks we must complete for a particular level of testing before we can say that testing is finished.**

- Management of any activity does not stop with planning it. We need to control and measure progress against the plan. So, **test control** is an ongoing activity. We need to compare actual progress against the planned progress
- And report to the project manager and customer on the current status of testing, including any changes or deviations from the plan .

- Test planning takes into account the feedback from **monitoring and control activities** which take place through out the project. Test control has the following major tasks:

- Test control has the following major tasks:
- Measure and analyze the results of reviews and testing:
- Monitor and document progress, test coverage and exit criteria:
- Provide information on testing:
- Initiate corrective actions:
- Make decisions:

Test analysis and design

Test analysis and design

- To review the test basis. The test basis is the information on which test cases are based, such as requirements, design specifications, product risk analysis, architecture and interfaces.
- To identify test conditions
- To design the tests
- To design the test environment set-up and identify the required infrastructure and tools

Test analysis and design

- In this phase, specifications are analysed to understand what the application's functions are –
  1. the screens,
  2. the actions,
  3. database updates,
  4. error messages,
  5. flow of data/info etc.
- The specifications are analysed to check how clear and complete the information is, for any confusion, ambiguities(unclear), for any discrepancies(conflicting actions), and more importantly how to test and verify the function in the system. The reviews and clarifications done early in this stage helps build a greater understanding of the product and hence an improvement to cost, quality and time in later stages.

- The ISTQB processes advocates a 3-step process to the test design process – Test Conditions, Test Cases, Test Procedures.
- **Test Conditions:** a test condition verifies a small section of the Functional Specification. eg. Registration, Login etc
- **Test Cases:** pre-conditions, a set of inputs, expected outcome and the post-conditions to verify the Test Condition
- **Test Procedure** : actual sequence of actions or steps to execute the test
- **Test Execution Log** – the order in which tests are run logically one after another

Test implementation and execution

Test implementation and execution

- **Test implementation and execution** is the activity where test procedures or scripts are specified by combining the test cases in a particular order and including any other information needed for test execution, the environment is set up and the tests are run.

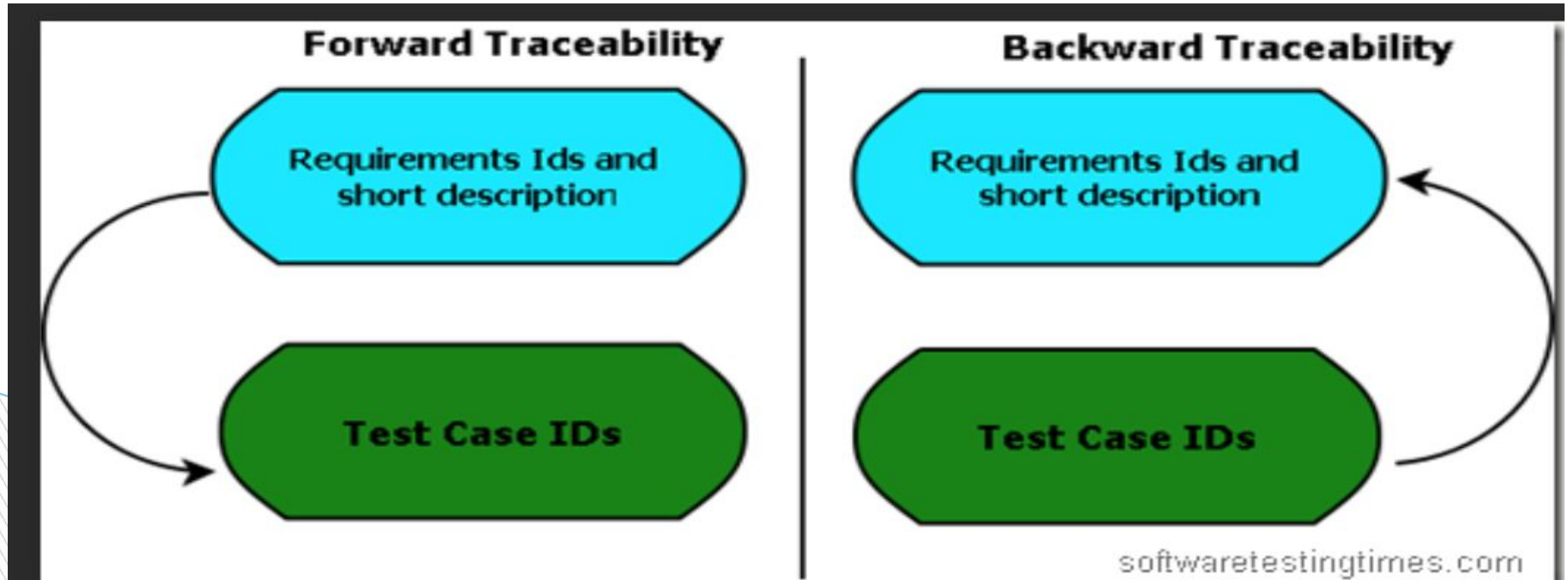
- Test implementation and execution has the following major tasks:

Test execution

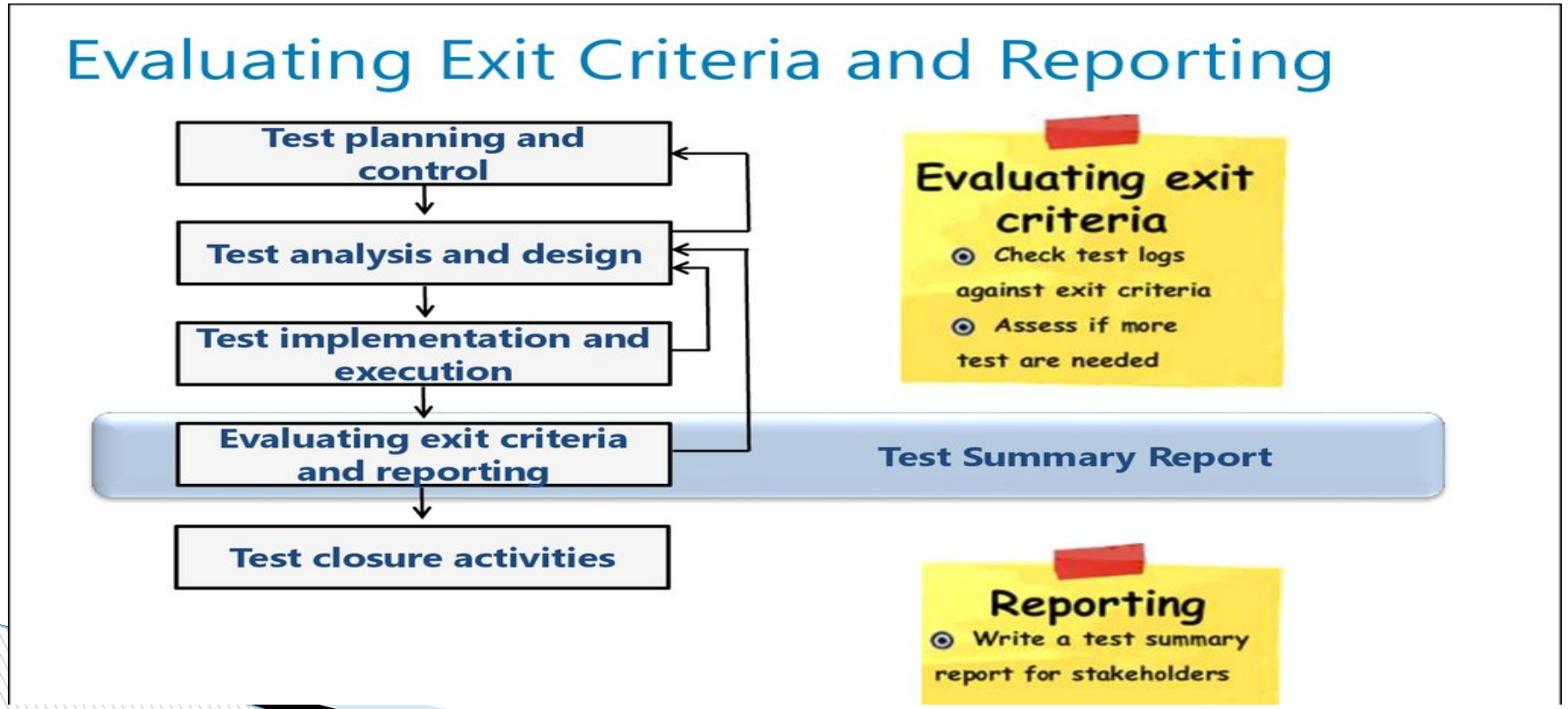
During test execution, test suites are run in accordance with the test execution schedule. Test execution includes the following major activities:

- 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
- Reporting defects based on the failures observed
- 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., execution of a corrected test, confirmation testing, and/or regression testing)
- Verifying and updating bi-directional traceability between the test basis, test conditions, test cases, test procedures, and test results.

Testware includes documentation, scripts, inputs, expected results, set-up and clear-up procedures, files, databases, environment, and any additional**software** or utilities used in **testing**. Generally, **Testware** is also called as**Testing** Tools.



- **Evaluating Exit Criteria and Reporting Exit criteria** i s set of agreed conditions with stakeholders based on which you can officially mark the testing process to be completed for a particular test level. Exit criteria should be set for each test level



Evaluating exit criteria and reporting

- Evaluating exit criteria is the activity where test execution is assessed against the defined objectives. This should be done for each test level, as for each we need to know whether we have done enough testing. Based on our risk assessment, we'll have set criteria against which we'll measure 'enough' or not.
- Exit criteria should be set and evaluated for each test level. Evaluating exit criteria has the following major tasks:

- Check test logs against the exit criteria specified in test planning:
- Assess if more tests are needed or if the exit criteria specified should be changed:
- Write a test summary report for stakeholders:

How to measure exit criteria ?

1. **All the planned requirements must be met**

2. **All the high Priority bugs should be closed**

3. **All the test cases should be executed**

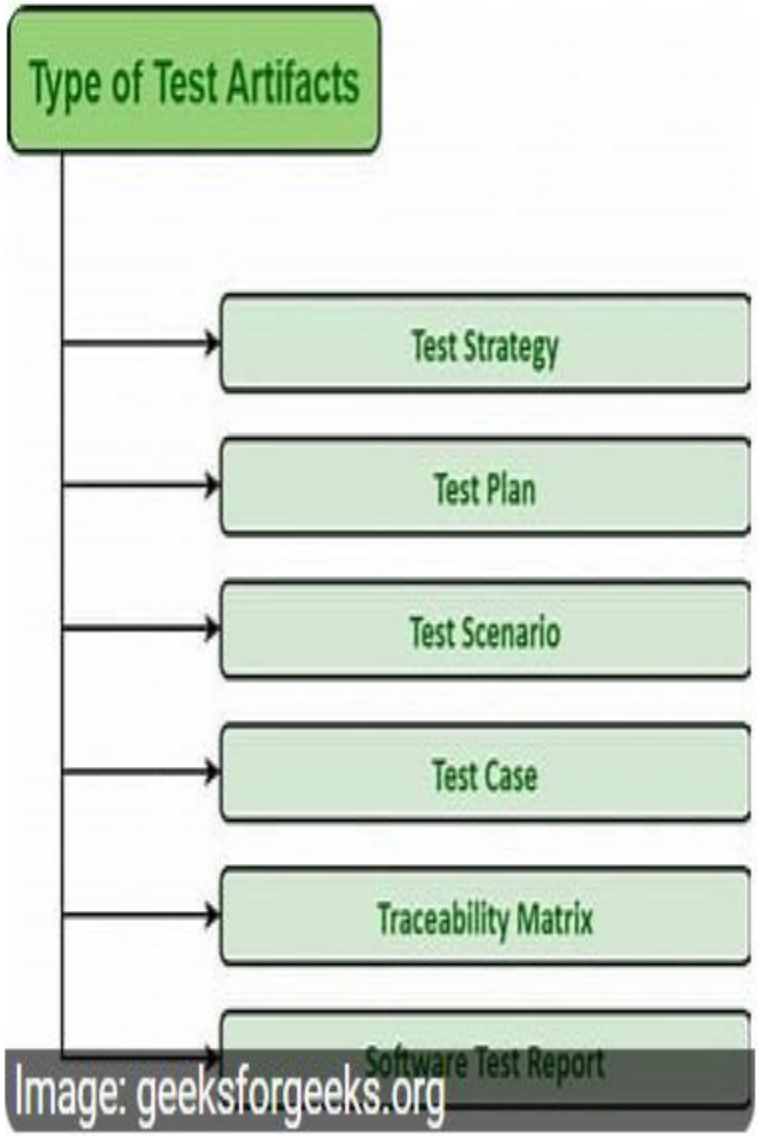
4. **If the scheduled time out is arrived**

5. **Test manager must sign off the release**

Note: All these parameters can be met by percentages(not 100 %)

Test closure activities

- Ensure Test Completion. Making sure that all the testing work has actually been completed and concluded. In complex...
- Handover Test Artifacts. Deliver the test artifacts to people who need it in future. After the release of the software,...
- Project Retrospectives. This is very important activity of test closure, project retrospectives are done to document the...
- Archive Test Work Products. Finally, all the test work products like test results, test logs, test.



- A retrospective is **a structured way to gather knowledge, insights, metrics, and artifacts from** a completed project, phase, or development iteration. Even in daily life, taking the time to reflect on why something unpleasant happened helps you to avoid a recurrence. A formal retrospective provides closure

- **Retrospectives are carried out after every sprint. Basic purpose of this retrospective meeting is –**
  - What went well – [Continue doing]
  - What didn't went well – [Stop doing]
  - Improvement Areas – [Start doing]

- **Some common problems are seen across all teams –**
  - Team members are not giving enough inputs in retrospective. How to get desired inputs from team members?
  - Improvement Areas are identified during retrospective, but How much improvement is made by team after each sprint? Answer is – Team does not know.
  - Retrospective meeting ended up in a blame game – Testers vs Devs. How to stop this?

Example for test closure

Test Closure - Activities and Deliverables

- Following are the list of activities and their respective deliverables:

Activities	Deliverables
Prepare Test Summary Report	Test Summary Report
Prepare Application/Product Stability Report	AQI / PQI Report
Retrospection	Critical Issues List Lessons Learnt Document

- **AQI (Application Quality Index)** and **PQI (Product Quality Index)** are metrics that help measure and communicate the **quality status of an application or product** during or after testing phases. These indices offer a quantitative view of the testing outcome, helping stakeholders make informed decisions.

1. AQI (Application Quality Index):

- AQI measures the **quality of an application under test (AUT)** at a given point in time.
- It considers factors such as:
  - Number of open defects (by severity)
  - Test case pass/fail rates
  - Code coverage
  - Defect density
  - Performance metrics (e.g., response time, throughput)

Typical AQI Formula (example):

```
text                                                    Copy    Edit

AQI = 100 - (Weighted defect score + Test failures + Performance penalties)
```

2. PQI (Product Quality Index):

- PQI is a broader metric that reflects the **overall product quality over time** (often across versions/releases).
- While AQI may be specific to a release cycle or sprint, **PQI captures cumulative quality performance**.

Typical PQI Approach:

```
text                                                    Copy    Edit

PQI = f(Defect leakage + Post-production defects + Customer feedback + Release stability)
```

PQI May Include:

- Historical defect trend analysis
- Post-release defect reports
- Customer-reported defects
- Escaped defects (bugs found in production)
- Mean time to detect and resolve defects (MTTD, MTTR)
- Test automation maturity
- Customer satisfaction scores (CSAT, NPS)

Test closure activities

Testware includes documentation, scripts, inputs, expected results, set-up and clear-up procedures, files, databases, environment, and any additional**software** or utilities used in **testing**. Generally, **Testware** is also called as**Testing** Tools.

- During test closure activities, we collect data from completed test activities to consolidate experience, including checking and filing testware, and analyzing facts and numbers. We may need to do this when software is delivered.
- We also might close testing for other reasons, such as
  - when we have gathered the information needed from testing,
  - when the project is cancelled,
  - when a particular milestone is achieved, or
  - when a maintenance release or update is done.
- Test closure activities include the following major tasks:

- Check which planned deliverables: we actually delivered and ensure all incident reports have been resolved through defect repair or deferral. For deferred defects, in other words those that remain open, we may request a change in a future release. We document the-acceptance or rejection of the software system.

- Finalize and archive testware, such as scripts, the test environment, and any other test infrastructure, for later reuse. It is important to reuse whatever we can of testware; we will inevitable carry out maintenance testing, and it saves time and effort if our testware can be pulled out from a library of existing tests. It also allows us to compare the results of testing between software versions.

- Hand over testware to the maintenance organization who will support the software and make any bug fixes or maintenance changes, for use in confirmation testing and regression testing.

- Evaluate how the testing went and analyze lessons learned for future releases and projects. This might include process improvements for the soft ware development life cycle as a whole and also improvement of the test processes.

THE PSYCHOLOGY OF TESTING

Psychology of Testing

The Testing approach

- Historically testing was viewed as showing the system meets its requirements
- This has evolved to a stage where testing is performed with the primary aim of finding faults rather than proving correctness. It is perceived as a destructive (negative) process
- Seeking to find failures (the right approach) can be viewed as criticism of the product and/or its author
- But looking for failures
  - Time can be saved
  - Risks reduced
  - Costs reduced
  - Skills improved

Psychology of Testing

Traits of Good Testers

- A Tester needs:
  - good communication skills
  - good observation skills
  - people handling skills
  - Curiosity (testing activities interest)
  - patience
  - Reliability (consistency)
  - Thoroughness (carefulness)
  - an inquisitive nature (interfering nature)
  - attention to detail
  - creativity in terms of identifying likely faults
  - Experience
- However as with most other disciplines an effective test team will need a mix of skills so it is difficult to generalise

Psychology of Testing

Developer vs Tester Relationship

- The relationship between a Developer and a Tester is not normally an easy one because:-
  - testers point out problems with software
  - developers like to think their software is perfect
  - testers are perceived as delaying the project by finding faults in the system
  - when the development slips, testers normally have to work long hours to test the product, which in turn can cause anger
- It is important that they work together
- It is also important that they have mutual respect for each other.
- Collaboration is the right approach – we work to a common goal!
- Communicate findings objectively, not subjectively

Psychology of Testing

Independent testing

- There are several levels of Independence (from Low to High):
  - Tests designed by the person(s) who wrote the software under test
  - Tests designed by another person(s) (e.g. from the development team).
  - Tests designed by a person(s) from a different organizational group (e.g. an independent test team).
  - Tests designed by a person(s) from a different organization or company (e.g. outsourcing to an in-house or external test specialist organisation)

questions

- What is software testing?
- explain briefly the fundamental test process?
- Explain briefly the general principles of testing

questions

- What is software testing? And how the test principles apply across the test life cycle?
- Explain the terms error, bug,
- What is software quality and software qualilty assurance
- Define test effort.