SOFTWARE TESTING AND MAINTENANCE

CHAPTER 6

- ANIKET MISHRA

- **Software Testing**
- Testing is the process of executing a program with the aim of finding errors. To make our software perform well it should be error-free. If testing is done successfully it will remove all the errors from the software.

Principles of Testing: -

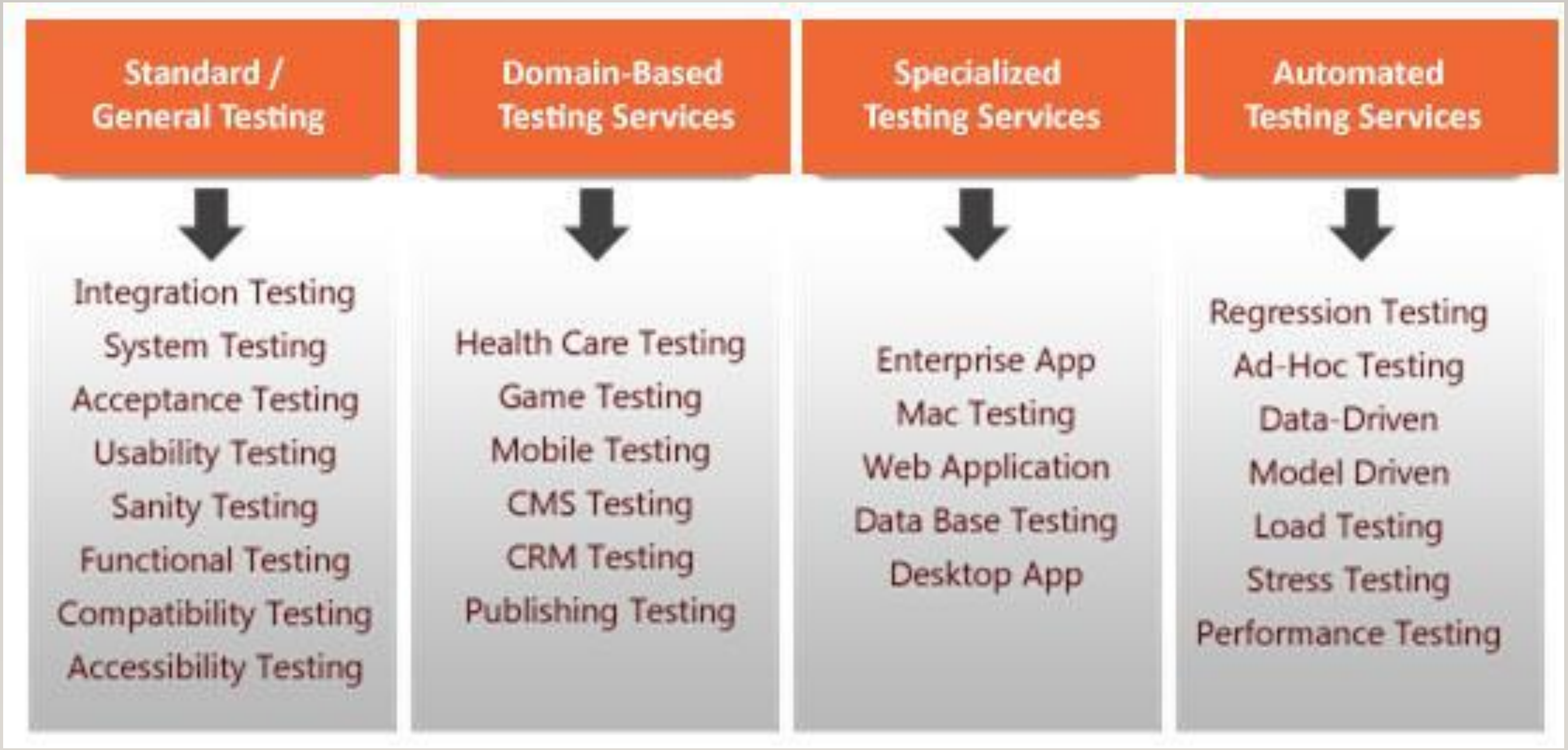
1. All the test should meet the customer requirements
2. To make our software testing should be performed by a third party
3. Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
4. All the test to be conducted should be planned before implementing it
5. It follows the Pareto rule (80/20 rule) which states that 80% of errors come from 20% of program components.
6. Start testing with small parts and extend it to large parts.

- **Why to Learn Software Testing?**

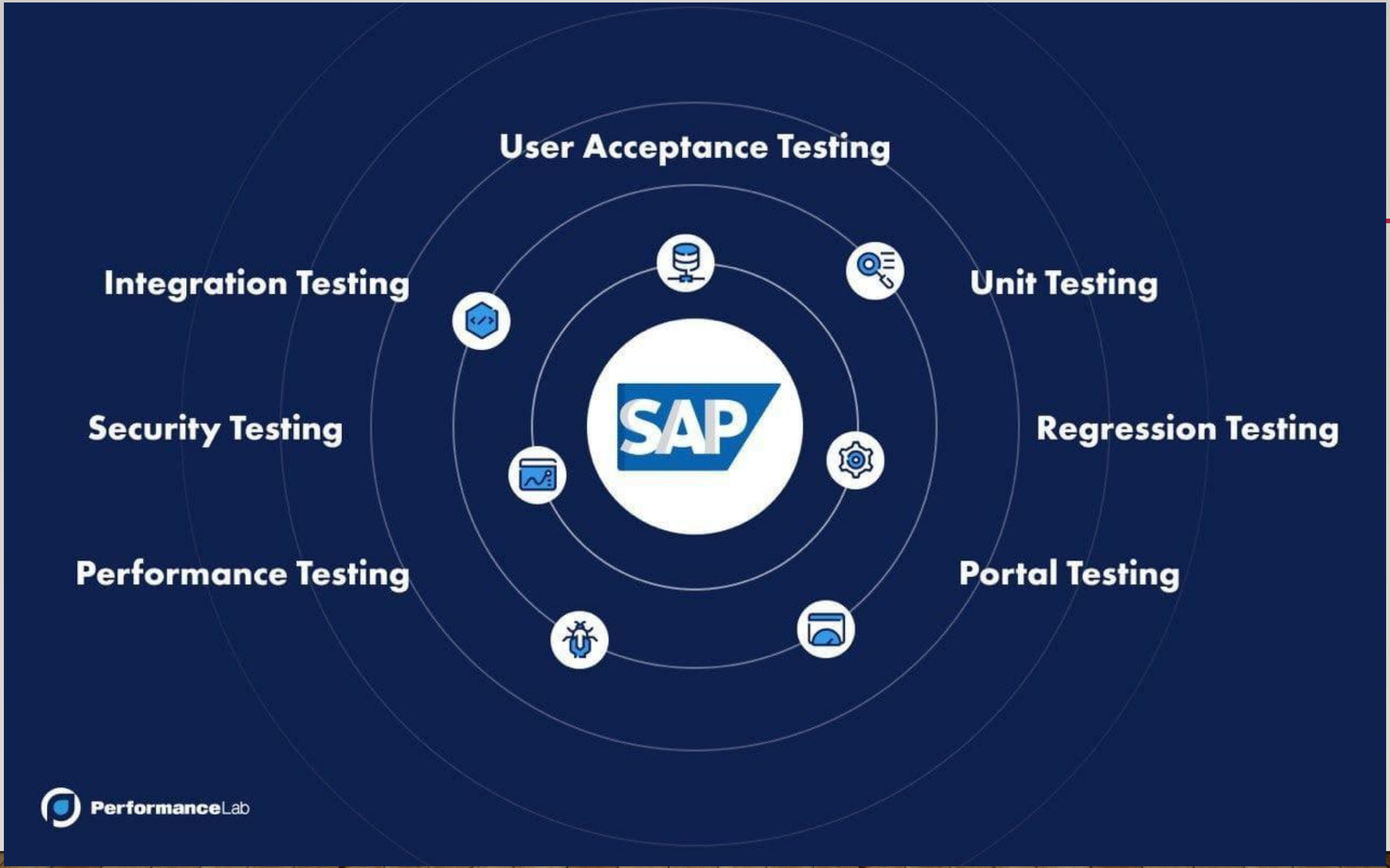
- In the IT industry, large companies have a team with responsibilities to evaluate the developed software in context of the given requirements. Moreover, developers also conduct testing which is called **Unit Testing**. In most cases, the following professionals are involved in testing a system within their respective capacities –

  - Software Tester
  - Software Developer
  - Project Lead/Manager
  - End User

- SOFTWARE APPLICATION TESTING SERVICES
- The Software Testing Team at Watzmann provide the below mentioned Application Testing Services:
- **Enterprise Application Testing** - Enterprise applications are critical to many businesses across various industries. At Watzmann Consulting we have a proficient QA team that can support infrastructure and processes to optimize testing of business applications.
- **Web Application Testing** - Watzmann's dedicated team of testing engineers have good experience and knowledge in web application testing technologies. We are proficient in different technologies has generated in delivering multiple projects at different parts of the globe.
- **Desktop Application Testing** – We provide a wide range of testing services on desktop applications. With the support of our detailed desktop application testing expertise and QA testing experience we have gained substantial experience in this.
- **Mobile App Testing** - Our mobile app testing services team has knowledge of handling complex apps which are intended to function across multiple devices with varying screen sizes, internal hardware, resolutions that can operate under various operating systems.



Standard / General Testing	Domain-Based Testing Services	Specialized Testing Services	Automated Testing Services
Integration Testing	Health Care Testing	Enterprise App	Regression Testing
System Testing	Game Testing	Mac Testing	Ad-Hoc Testing
Acceptance Testing	Mobile Testing	Web Application	Data-Driven
Usability Testing	CMS Testing	Data Base Testing	Model Driven
Sanity Testing	CRM Testing	Desktop App	Load Testing
Functional Testing	Publishing Testing		Stress Testing
Compatibility Testing			Performance Testing
Accessibility Testing			


User Acceptance Testing
Integration Testing
Unit Testing
Security Testing
SAP
Regression Testing
Performance Testing
Portal Testing
PerformanceLab

UNIT TESTING :

UNIT TESTING is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

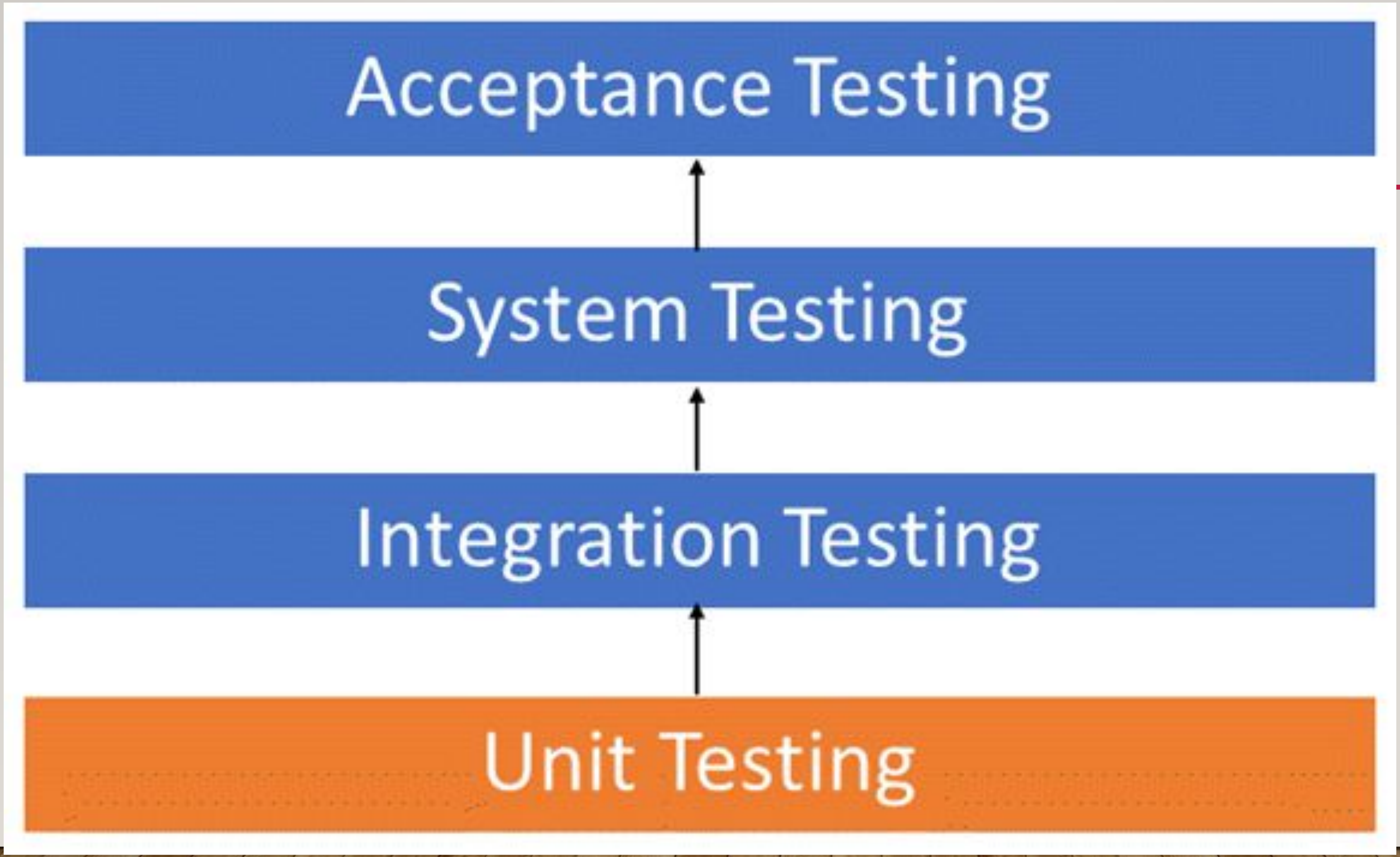
Why Unit Testing?

Unit Testing is important because software developers sometimes try saving time doing minimal unit testing and this is myth because inappropriate unit testing leads to high cost Defect fixing during System Testing, Integration Testing and even Beta Testing after application is built. If proper unit testing is done in early development, then it saves time and money in the end.

Unit Testing is of two types:

- Manual
- Automated

Unit testing is commonly automated but may still be performed manually. Software Engineering does not favor one over the other but automation is preferred. A manual approach to unit testing may employ a step-by-step instructional document.


Acceptance Testing
System Testing
Integration Testing
Unit Testing

Unit Testing Techniques

The **Unit Testing Techniques** are mainly categorized into three parts which are Black box testing that involves testing of user interface along with input and output, White box testing that involves testing the functional behaviour of the software application and Gray box testing that is used to execute test suites, test methods, test cases and performing risk analysis.
Code coverage techniques used in Unit Testing are listed below:

- Statement Coverage
- Decision Coverage
- Branch Coverage
- Condition Coverage
- Finite State Machine Coverage

Unit tests are run automatically on the software engineers' machines during the development process. By using intelligent tools, only the tests affected by the changes made to the relevant software module are run, rather than the whole test suite.

Unit Test Example: Mock Objects

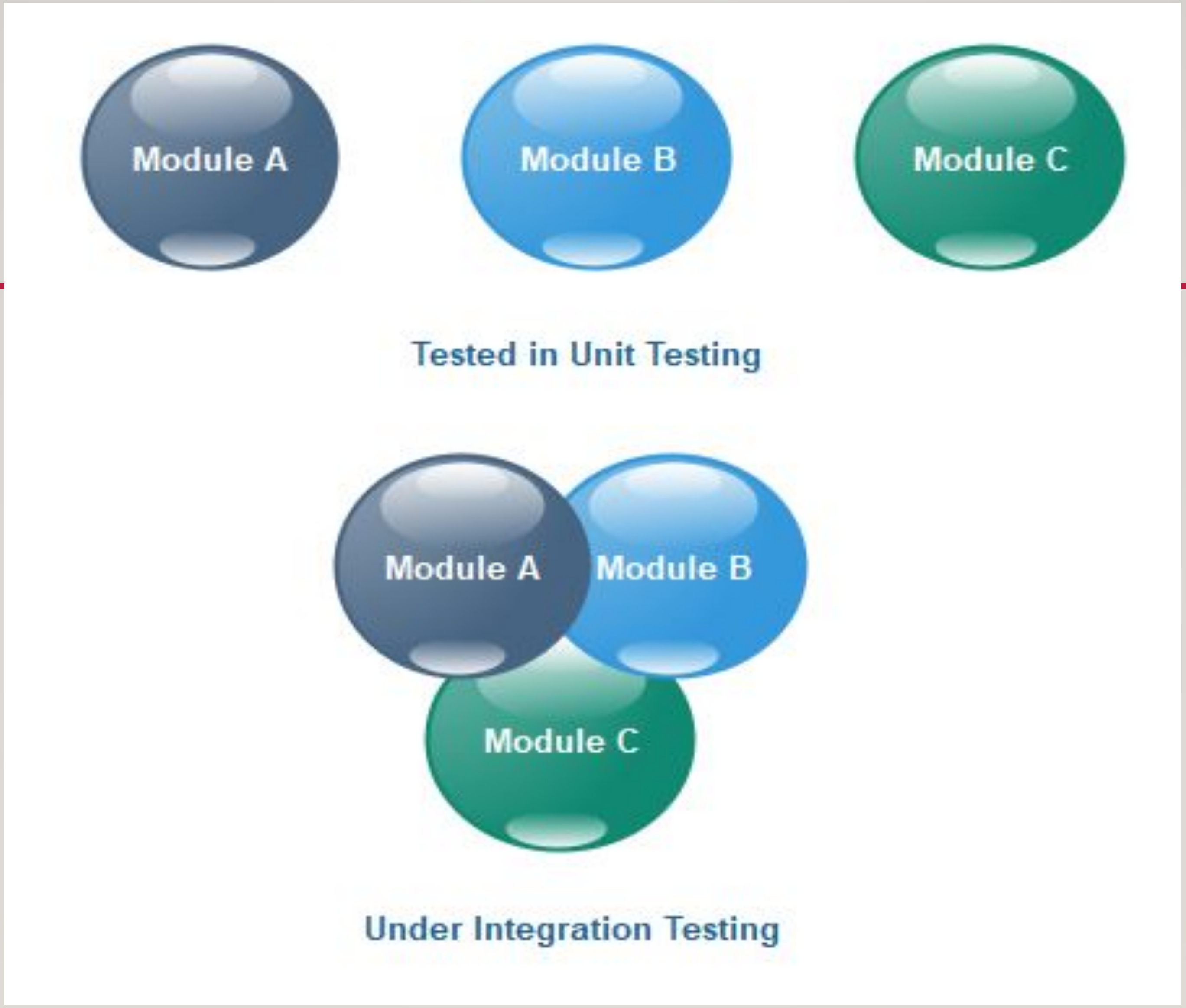
Unit testing relies on mock objects being created to test sections of code that are not yet part of a complete application. Mock objects fill in for the missing parts of the program.
For example, you might have a function that needs variables or objects that are not created yet. In unit testing, those will be accounted for in the form of mock objects created solely for the purpose of the unit testing done on that section of code.

INTEGRATION TESTING :

- **What Is Integration Testing?**
- Let’s start with a formal definition. Integration testing is a type of testing meant to check the combinations of different units, their interactions, the way subsystems unite into one common system, and code compliance with the requirements.
- For example, when we check login and sign up features in an e-commerce app, we view them as separate units. If we check the ability to log in or sign up after a user adds items to their basket and wants to proceed to the checkout, we check the integration between these two functionalities.
- **Why Do We Need Integration Testing?**
- IT specialists are well aware of how unpredictable code can be, even if it seems perfectly written. Besides, there are dynamic projects, where pre-approved requirements can lose their relevance and change – that’s especially true for Agile projects.
- Simultaneous testing of different modules is very convenient, practical, and cost-efficient.
- Integration checks can take place at any stage of the SDLC.
- It is a lifesaver for a team involved in projects with constantly changing requirements or logic that is under review after development has started.
- Unlike other types of tests, integration can cover any amount of program code in one sprint.
- https://www.softwaretestinghelp.com/what-is-integration-testing/

Similarly Mail Box: Check its integration to the Delete Mails Module.

Test Case ID	Test Case Objective	Test Case Description	Expected Result
1	Check the interface link between the Login and Mailbox module	Enter login credentials and click on the Login button	To be directed to the Mail Box
2	Check the interface link between the Mailbox and Delete Mails Module	From Mailbox select the email and click a delete button	Selected email should appear in the Deleted/Trash folder


Module A
Module B
Module C
Tested in Unit Testing
Module A
Module B
Module C
Under Integration Testing

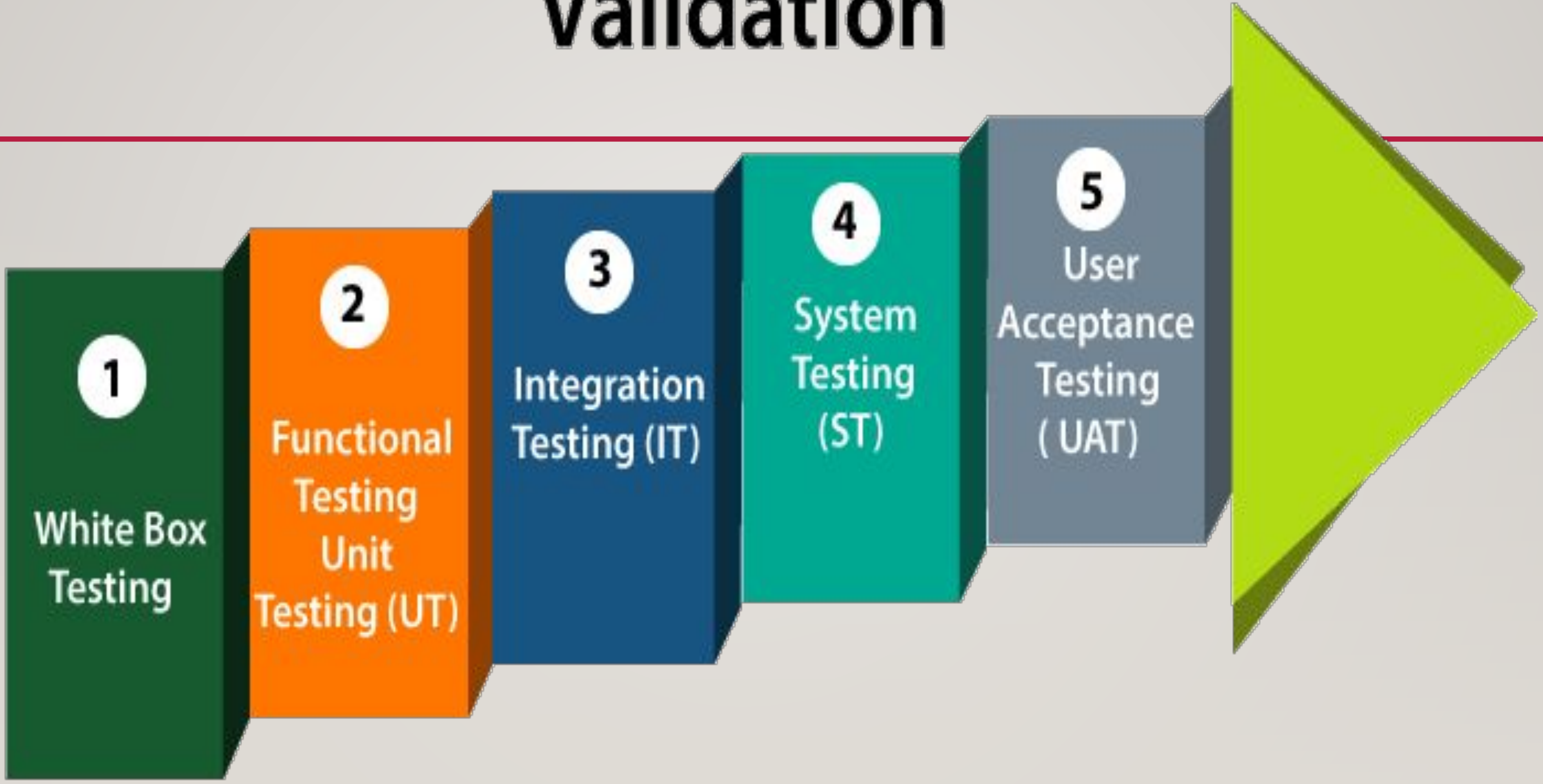
VERIFICATION TESTING :

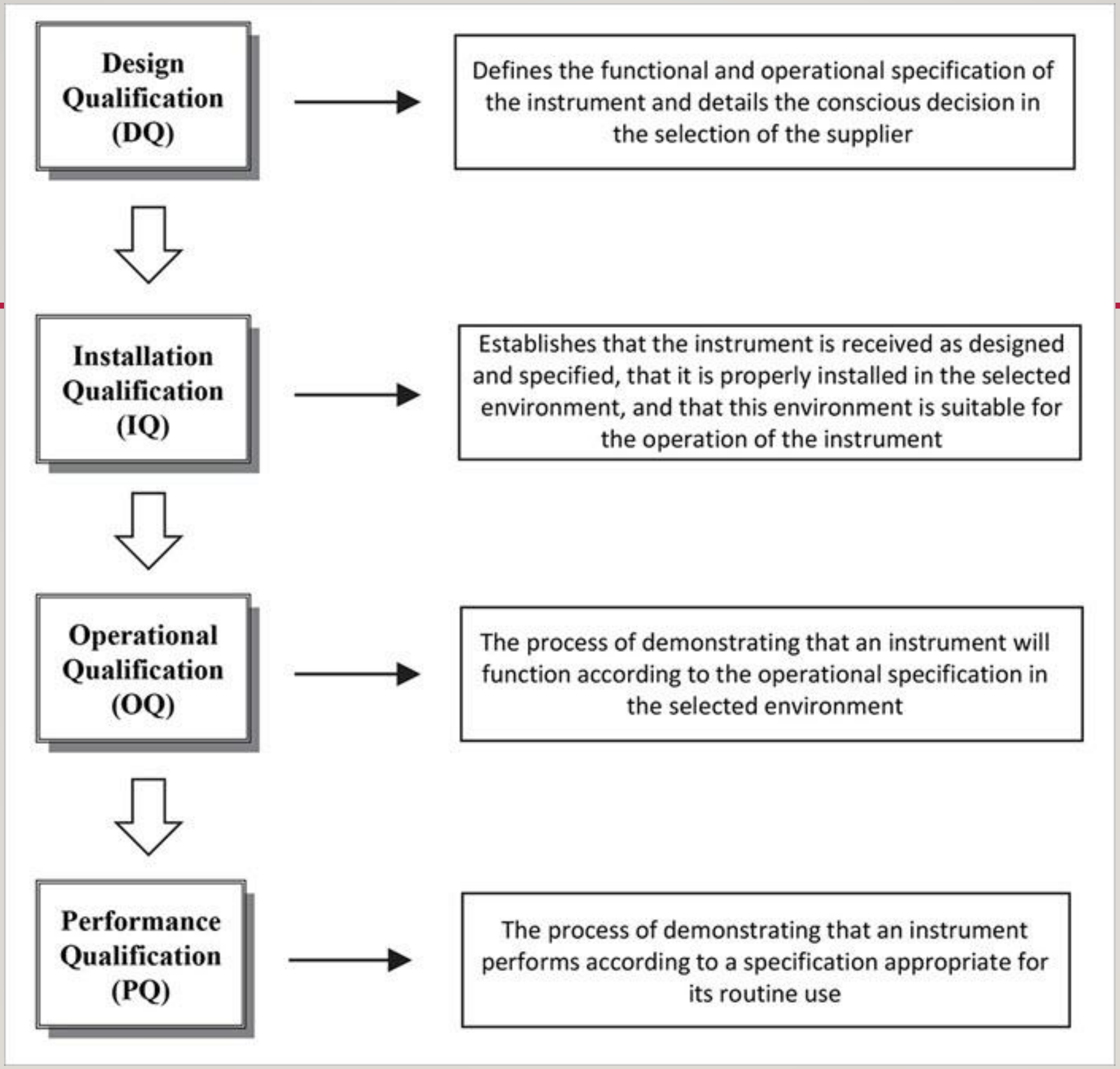
- Verification Testing : is used to confirm that a product meets specifications or requirements as defined in Phase Zero of the product development process. Verification testing should be conducted iteratively throughout a product design process, ensuring that the designs perform as required by the product specifications.
- Product developers achieve verification using an array of methods that can include inspection, demonstration, physical testing, and simulation.
- Verification testing includes different activities such as business requirements, system requirements, design review, and code walkthrough while developing a product.
- It is also known as static testing, where we are ensuring that "**we are developing the right product or not**". And it also checks that the developed application fulfilling all the requirements given by the client.
- Verification testing – Did I make the product correctly?
- Validation testing – Did I make the correct product?
-

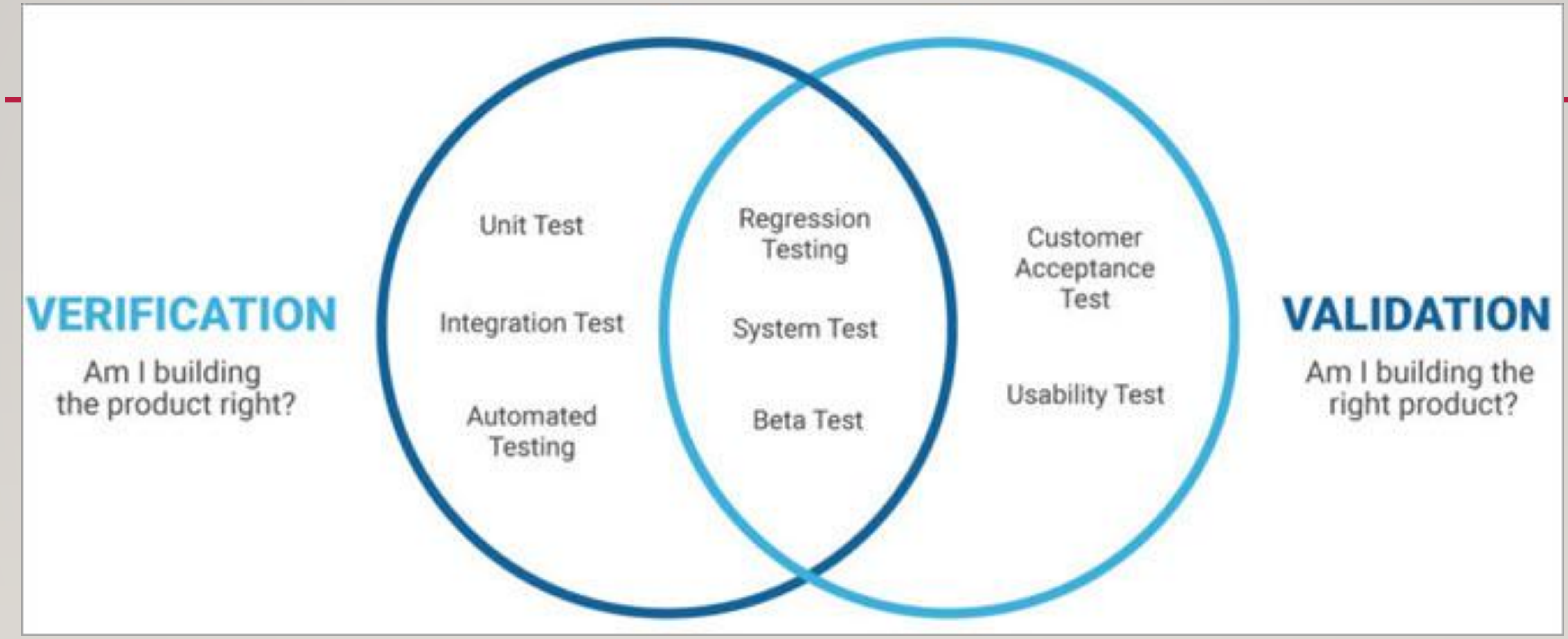
- **What is Validation Testing?**
- Validation testing is the process of ensuring if the tested and developed software satisfies the client /user needs. The business requirement logic or scenarios have to be tested in detail. All the critical functionalities of an application must be tested here.
- Validation testing can be best demonstrated using V-Model. The Software/product under test is evaluated during this type of testing.
- Verification and validation testing play an integral part in the product development process. Creating technology and systems is an iterative cycle, in which development teams are constantly learning and feeding data back into the process.
- Verification and validation activity serve as a system of checks and balances, ensuring that developers regularly evaluate if the product is functioning as intended and meeting the needs of its users. Testing also offers an important opportunity to document the tangible results of your design efforts.

Verification	Validation
We check whether we are developing the right product or not.	We check whether the developed product is right.
Verification is also known as **static testing**.	Validation is also known as **dynamic testing**.
Verification includes different methods like Inspections, Reviews, and Walkthroughs.	Validation includes testing like functional testing , system testing, integration , and User acceptance testing.
It is a process of checking the work-products (not the final product) of a development cycle to decide whether the product meets the specified requirements.	It is a process of checking the software during or at the end of the development cycle to decide whether the software follow the specified business requirements.
Quality assurance comes under verification testing.	**Quality control** comes under validation testing.
The execution of code does not happen in the verification testing.	In validation testing, the execution of code happens.
In verification testing, we can find the bugs early in the development phase of the product.	In the validation testing, we can find those bugs, which are not caught in the verification process.
Verification testing is executed by the Quality assurance team to make sure that the product is developed according to customers' requirements.	Validation testing is executed by the testing team to test the application.
Verification is done before the validation testing.	After verification testing, validation testing takes place.
In this type of testing, we can verify that the inputs follow the outputs or not.	In this type of testing, we can validate that the user accepts the product or not.

Validation


1
White Box Testing
2
Functional Testing Unit Testing (UT)
3
Integration Testing (IT)
4
System Testing (ST)
5
User Acceptance Testing (UAT)


Design Qualification (DQ)
Defines the functional and operational specification of the instrument and details the conscious decision in the selection of the supplier
Installation Qualification (IQ)
Establishes that the instrument is received as designed and specified, that it is properly installed in the selected environment, and that this environment is suitable for the operation of the instrument
Operational Qualification (OQ)
The process of demonstrating that an instrument will function according to the operational specification in the selected environment
Performance Qualification (PQ)
The process of demonstrating that an instrument performs according to a specification appropriate for its routine use


VERIFICATION
Am I building the product right?
Unit Test
Integration Test
Automated Testing
Regression Testing
System Test
Beta Test
Customer Acceptance Test
Usability Test
VALIDATION
Am I building the right product?

- **What is System Testing?**
- System testing is performed in the context of a System Requirement Specification (SRS) and/or a Functional Requirement Specifications (FRS). It is the final test to verify that the product to be delivered meets the specifications mentioned in the requirement document. It should investigate both functional and non-functional requirements.
- **System Testing** is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system.
- Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario.
- Verify thorough testing of every input in the application to check for desired outputs.
- Testing of the user’s experience with the application.

- **System Testing is Blackbox**
- Two Category of Software Testing
- Black Box Testing
- White Box Testing

- 1. Usability Testing - To test if an application or product has good user experience or not.

  2. Regression Testing - To confirm that a code change or addition has not adversely affected existing features.

  3. Load Testing - It is a type of non-functional testing which helps understand the behaviour of the application under a specific expected load.

  4. Functional Testing - It is a type of testing to verify that a product performs and functions correctly according to user specifications.

  5. Migration Testing - Testing of programs used to migrate /convert data from one application to another replacement application.

  6. Compatibility Testing - It performed to validate that software performs same behaviour with different environment.

  7. Boundary Value Testing - It is designed to include representatives of boundary values.

  8. Fuzz Testing - It is used to provide invalid, unexpected, or random data to the inputs of a program.

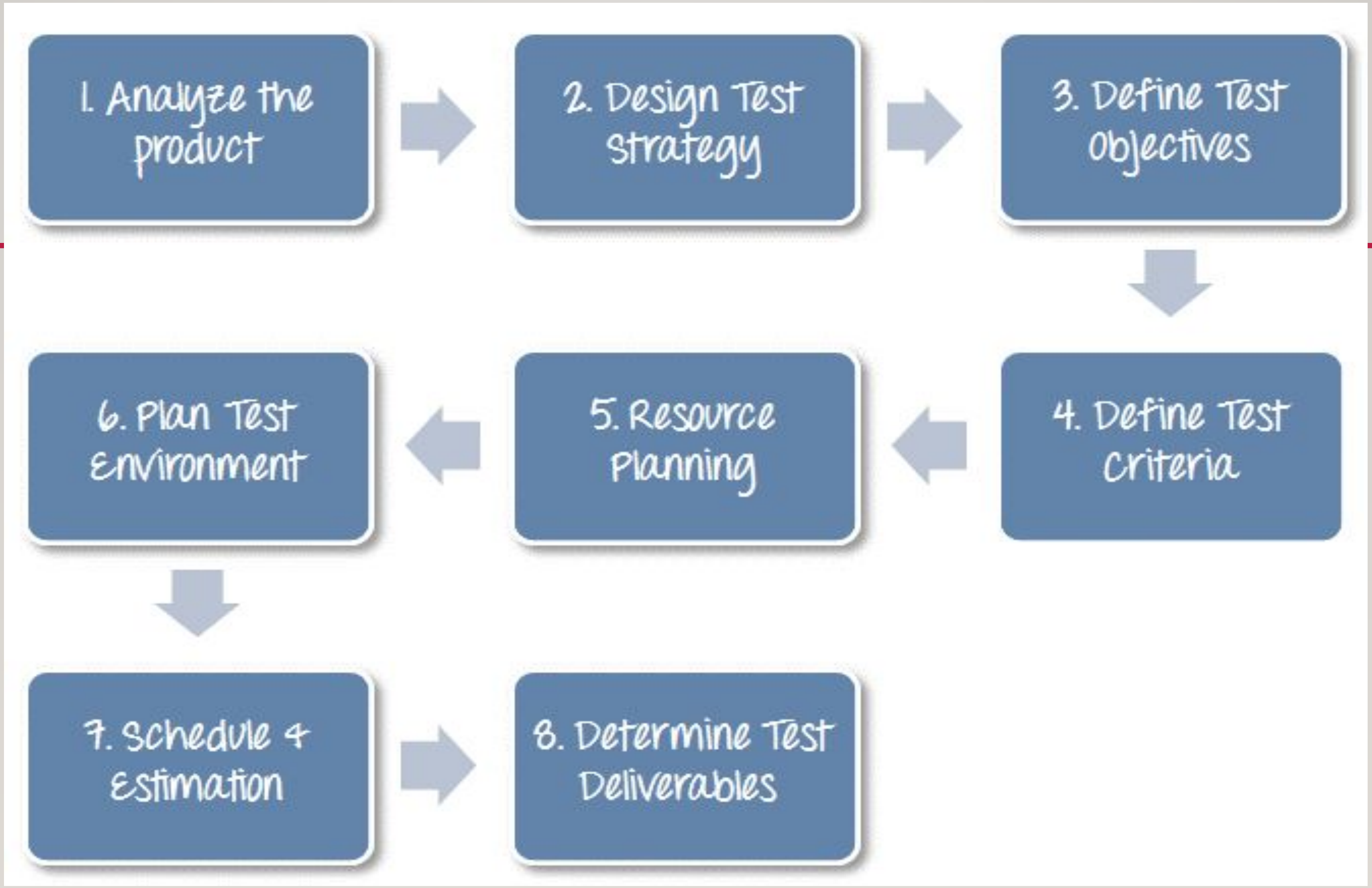
Test Plan :

A **Test Plan** is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

Test Plan is a dynamic document. The success of a testing project depends upon a well-written Test Plan document that is current at all times. Test Plan is more or less like **a blueprint of how the testing activity is going** to take place in a project.

Given below are a few pointers on a Test Plan:

- Test Plan is a document that acts as a point of reference and only based on that testing is carried out within the QA team.
- It is also a document that we share with the Business Analysts, Project Managers, Dev team and the other teams. This helps to enhance the level of transparency of the QA team’s work to the external teams.
- It is documented by the QA manager/QA lead based on the inputs from the QA team members.
- Test Planning is typically allocated with 1/3rd of the time that takes for the entire QA engagement. The other 1/3rd is for Test Designing and the rest is for Test Execution.
- This plan is not static and is updated on an on-demand basis.
- The more detailed and comprehensive the plan is, the more successful will be the testing activity.


1. Analyze the product
2. Design Test Strategy
3. Define Test Objectives
4. Define Test Criteria
5. Resource Planning
6. Plan Test Environment
7. Schedule & Estimation
8. Determine Test Deliverables

1. Analyze the product
2. Design the Test Strategy
3. Define the Test Objectives
4. Define Test Criteria
5. Resource Planning
6. Plan Test Environment
7. Schedule & Estimation
8. Determine Test Deliverables

https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html

WHITE BOX TESTING :

- “White box testing” (also known as clear, glass box or structural testing) is a testing technique which evaluates the code and the internal structure of a program.
- White box testing involves looking at the structure of the code. When you know the internal structure of a product, tests can be conducted to ensure that the internal operations performed according to the specification.
- White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.
- **Blackbox testing, involves testing from an external or end-user type perspective.**

- Why we perform WBT?
- That all independent paths within a module have been exercised at least once.
- All logical decisions verified on their true and false values.
- All loops executed at their boundaries and within their operational bounds internal data structures validity.
- The term “Whitebox” was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software’s outer shell (or “box”) into its inner workings. Likewise, the “black box” in “Black Box Testing” symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.