CSE- 321
Software Engineering

Software Verification & Validation , Testing, Quality

Lecture Outlines

- ✧ Verification and Validation
- ✧ Cleanroom software development
- ✧ Software testing
- ✧ Software testing types
- ✧ Black-box Testing ,White-box Testing, Grey-Box Testing

Verification and Validation

Verification and Validation

Verification:

"**Are we building the product right**".

The software should conform to its specification.

Validation:

"**Are we building the right product**".

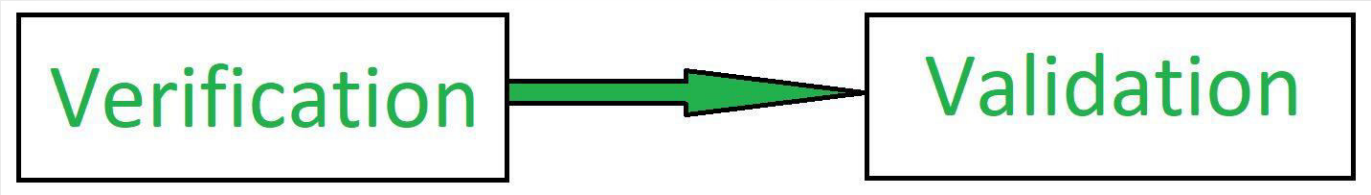
The software should do what the user really requires.

Verification and Validation (V & V)

Verification:

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is **developed is right or not**. It verifies whether the developed product fulfills the requirements that we have.

Activities involved in verification: Inspections Reviews, Walkthroughs, Desk-checking



Validation:

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product , it checks what we are developing is **the right product**. it is validation of actual and expected product.

Includes program reviews, system testing, customer acceptance testing.

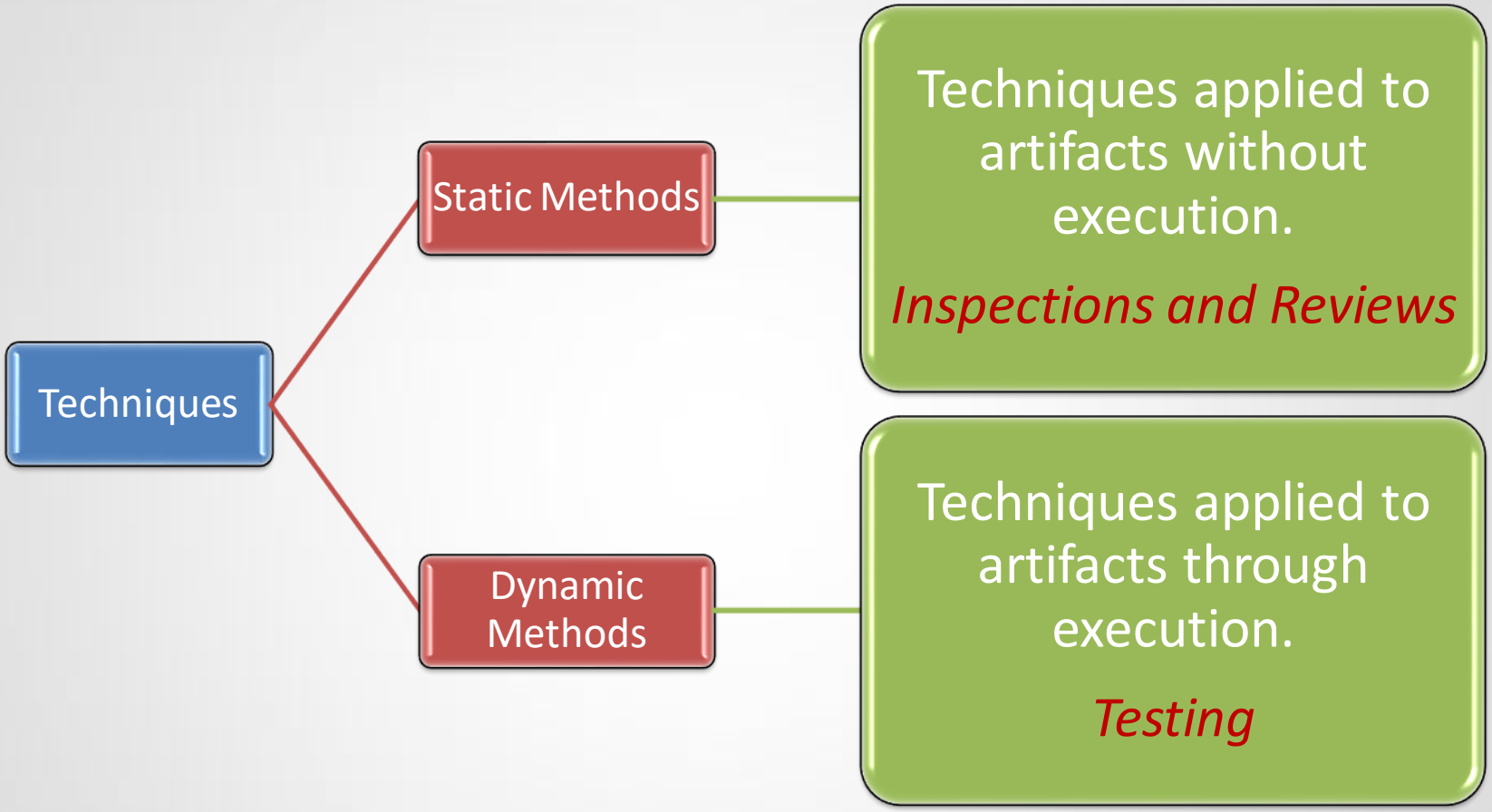
The V & V process

- Is a whole life-cycle process - V & V must be applied at each stage in the software process.
- Has two principal objectives
  - The discovery of defects in a system;
  - The assessment of whether or not the system is useful and useable in an operational situation.

V & V goals

- Verification and validation should establish confidence that the software is fit for purpose.
- This does NOT mean completely free of defects.
- Rather, it must be good enough for its intended use and the type of use will determine the degree of confidence that is needed.

V and V Techniques



Static and dynamic verification

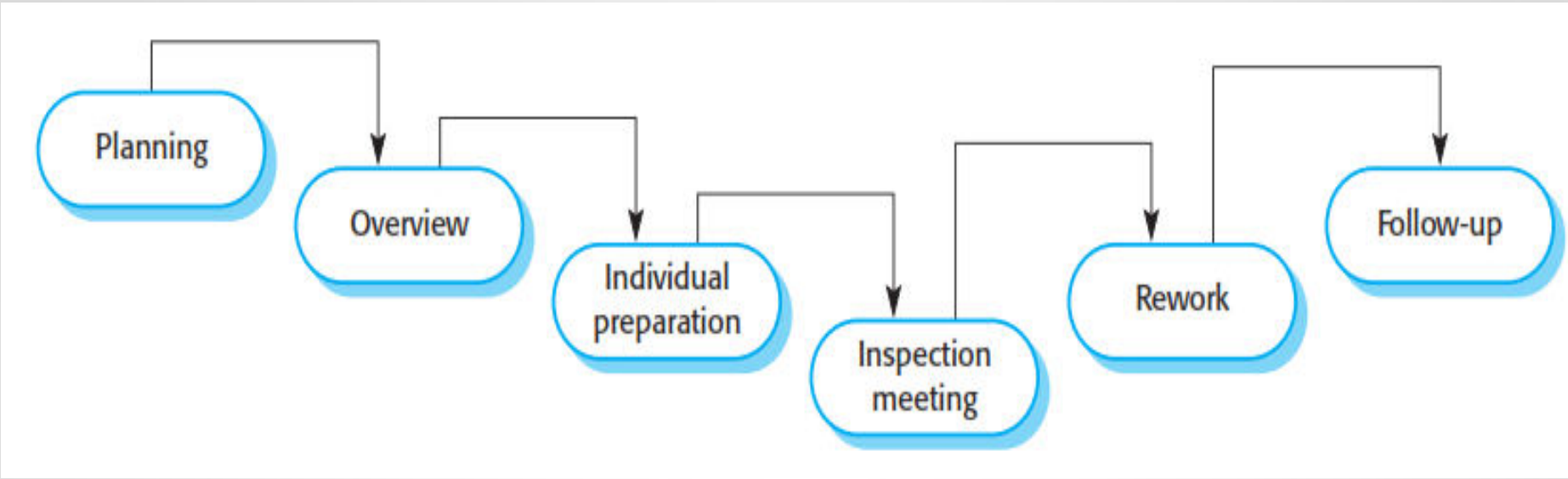
- Software inspections is a static V & V process in which a software system is reviewed to find errors, omissions and anomalies. Generally, inspections focus on source code, but any readable representation of the software such as its requirements or a design model can be inspected.

  - May be supplement by tool-based document and code analysis
  - Inspections do not require execution of a system so may be used before implementation.
  - They may be applied to any representation of the system such as the requirements or design.
  - They have been shown to be an effective technique for discovering program errors.

Static and dynamic verification

- Software inspections
  - Inspections are an old idea. There have been several studies and experiments that have demonstrated that inspections are more effective for defect discovery than program testing.
  - Fagan (Fagan, 1986) reported that more than 60% of the errors in a program can be detected using informal program inspections.
  - Mills et al. (Mills, et al., 1987) suggest that a more formal approach to inspection based on correctness arguments can detect more than 90% of the errors in a program.

The inspection process



- System overview presented to inspection team.
- Code and associated documents are distributed to inspection team in advance for individual preparation.
- Inspection meeting takes place and discovered errors are noted.
- Modifications are made to repair discovered errors (by owner).
- Re-inspection may or may not be required.

Inspection Checks

Possible checks that might be made during the inspection process

Fault class	Inspection check
Data faults	Are all program variables initialised before their values are used? Have all constants been named? Should the upper bound of arrays be equal to the size of the array or Size -1? If character strings are used, is a delimiter explicitly assigned? Is there any possibility of buffer overflow?
Control faults	For each conditional statement, is the condition correct? Is each loop certain to terminate? Are compound statements correctly bracketed? In case statements, are all possible cases accounted for? If a break is required after each case in case statements, has it been included?
Input/output faults	Are all input variables used? Are all output variables assigned a value before they are output? Can unexpected inputs cause corruption?

Inspection Checks

Possible checks that might be made during the inspection process

Fault class	Inspection check
Interface faults	Do all function and method calls have the correct number of parameters? Do formal and actual parameter types match? Are the parameters in the right order? If components access shared memory, do they have the same model of the shared memory structure?
Storage management faults	If a linked structure is modified, have all links been correctly reassigned? If dynamic storage is used, has space been allocated correctly? Is space explicitly de-allocated after it is no longer required?
Exception management faults	Have all possible error conditions been taken into account?

Static and dynamic verification

- Software testing Concerned with exercising and observing product behaviour (dynamic verification)

  - Inspections and testing are complementary and not opposing verification techniques. Both should be used during the V & V process.
  - Inspections can check conformance with a specification but not conformance with the customer's real requirements. Inspections cannot check **non-functional characteristics** such as performance, usability, etc.

  - Software testing is a process of identifying the **correctness of software by considering its all attributes** (Reliability, Scalability, Portability, Re-usability, Usability) and evaluating the execution of software components to find the software bugs or errors or defects.

Program testing

- Can reveal the presence of errors NOT their absence.
- The only validation technique for non-functional requirements as the software has to be executed to see how it behaves.
- Should be used in conjunction with static verification to provide full V&V coverage
- Typically, a commercial software system has to go through three stages of testing:
  - **Development testing:** the system is tested during development to discover bugs and defects.
  - **Release testing**: a separate testing team test a complete version of the system before it is released to users.
  - **User testing:** users or potential users of a system test the system in their own environment.

Verification vs Validation

Verification	Validation
Are we implementing the system right?	Are we implementing the right system?
Evaluating products of a development phase	Evaluating products at the closing of the development process
The objective is making sure the product is as per the requirements and design specifications	The objective is making sure that the product meets user's requirements
Activities included: reviews, meetings, and inspections	Activities included: black box testing, white box testing, and grey box testing
Verifies that outputs are according to inputs or not	Validates that the users accept the software or not
Items evaluated: plans, requirement specifications, design specifications, code, and test cases	Items evaluated: actual product or software under test
Manual checking of the documents and files	Checking the developed products using the documents and files

Verification vs Validation

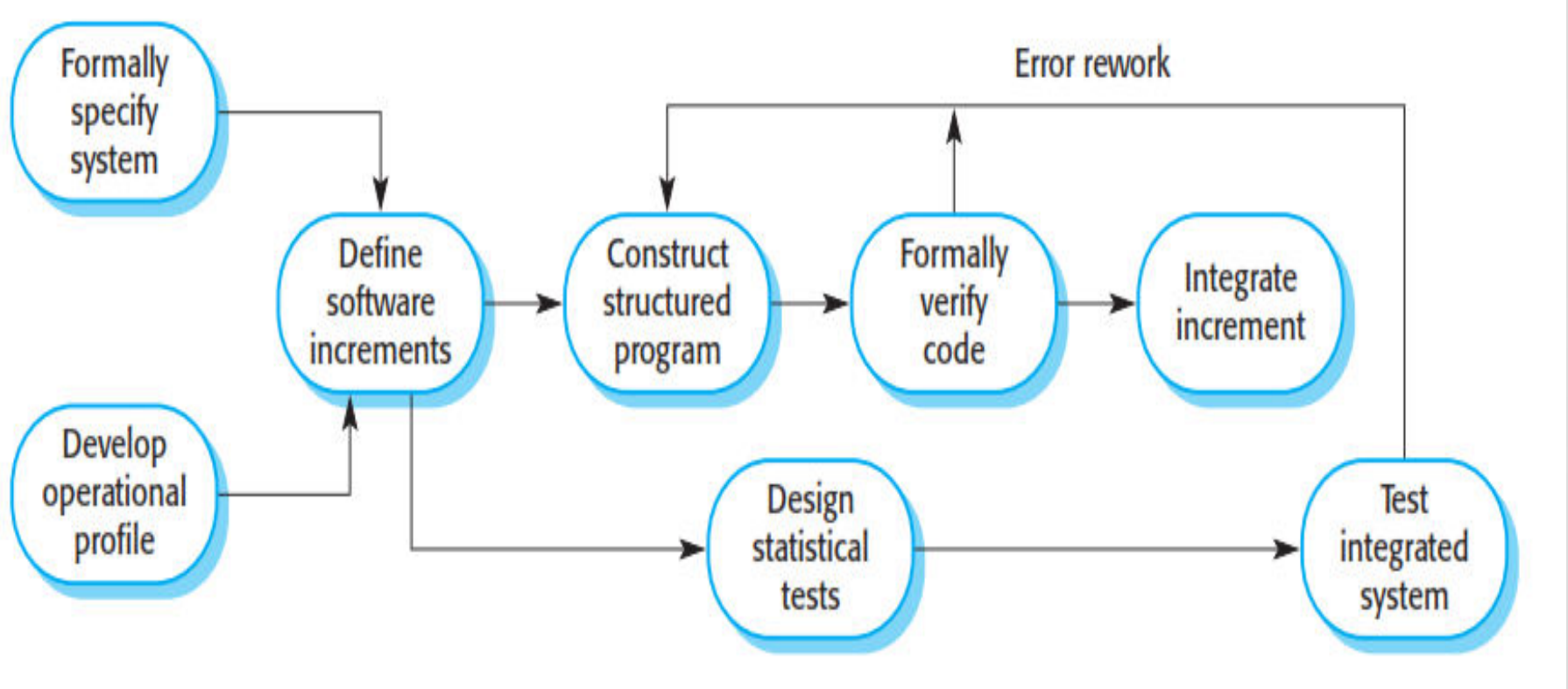
VERIFICATION	VALIDATION
Verification is the static testing.	Validation is the dynamic testing.
It does not include the execution of the code.	It includes the execution of the code.
Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.
It checks whether the software conforms to specifications or not.	It checks whether the software meets the requirements and expectations of a customer or not.
It can find the bugs in the early stage of the development.	It can only find the bugs that could not be found by the verification process.
The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.
Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
It comes before validation.	It comes after verification.
It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer.

Cleanroom software development

- The name is derived from the “Cleanroom” process in semiconductor fabrication.
- **The philosophy is defect avoidance rather than defect removal.**
- Cleanroom software development (Mills, et al., 1987; Cobb and Mills, 1990; Linger, 1994; Prowell, et al., 1999) is a software development philosophy that uses formal methods to support rigorous software inspection.
- A software development process based on:
  - Incremental development (if appropriate)
  - Formal specification
  - Static verification using correctness arguments
  - Statistical testing to certify program reliability

The Cleanroom process

The objective of this approach to software development is zero-defect software.



The Cleanroom process

The Cleanroom approach to software development is based on five key strategies:

1. **Formal specification** The software to be developed is formally specified. A state transition model that shows system responses to stimuli is used to express the specification.

2. **Incremental development** The software is partitioned into increments that are developed and validated separately using the Cleanroom process. These increments are specified, with customer input, at an early stage in the process.

3. **Structured programming** Only a limited number of control and data abstraction constructs are used. The program development process is a process of stepwise refinement of the specification. A limited number of constructs are used and the aim is to systematically transform the specification to create the program code.

The Cleanroom process

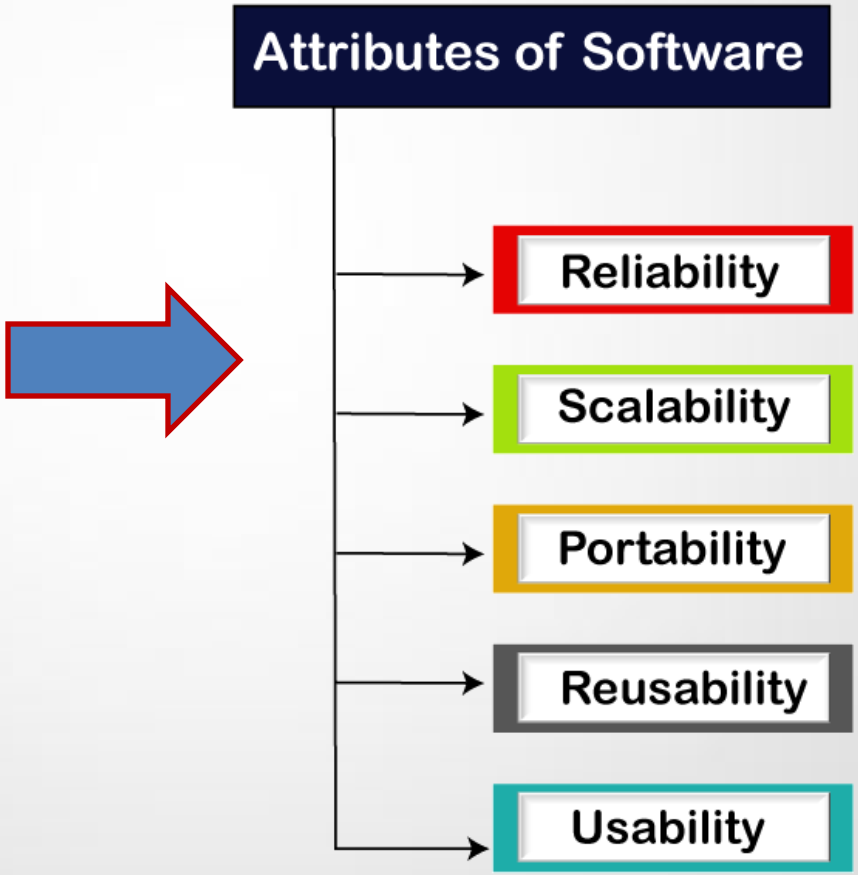
4. **Static verification** The developed software is statically verified using rigorous software inspections. There is no unit or module testing process for code components.

5. **Statistical testing of the system** The integrated software increment is tested statistically, to determine its reliability. These statistical tests are based on an operational profile, which is developed in parallel With the system specification.

Software testing

Software testing

Software testing is a process of identifying the **correctness** of software by considering its all attributes (**Reliability, Scalability, Portability, Re-usability, Usability**) and evaluating the execution of software components to find the software bugs or errors or defects.



Software testing

Software testing is a method to check whether the actual software product **matches expected requirements** and to ensure that software product is **defect free**.

Software Testing Definition according to **ANSI/IEEE 1059** standard – A process of analyzing a software item to detect the differences between existing and required conditions (i.e., defects) and to evaluate the features of the software item.

Testing can only show the **presence of errors** in a program. It cannot demonstrate that **there are no remaining faults**.

Software testing terms

Software testing terms

Error:

- Error is deviation from actual and expected value.
- It represents mistake made by people.

Fault

- Fault is incorrect step, process or data definition in a computer program which causes the program to behave in an unintended or unanticipated manner.
- It is the result of the error.

Bug

- Bug is a fault in the program which causes the program to behave in an unintended or unanticipated manner.
- It is an evidence of fault in the program.

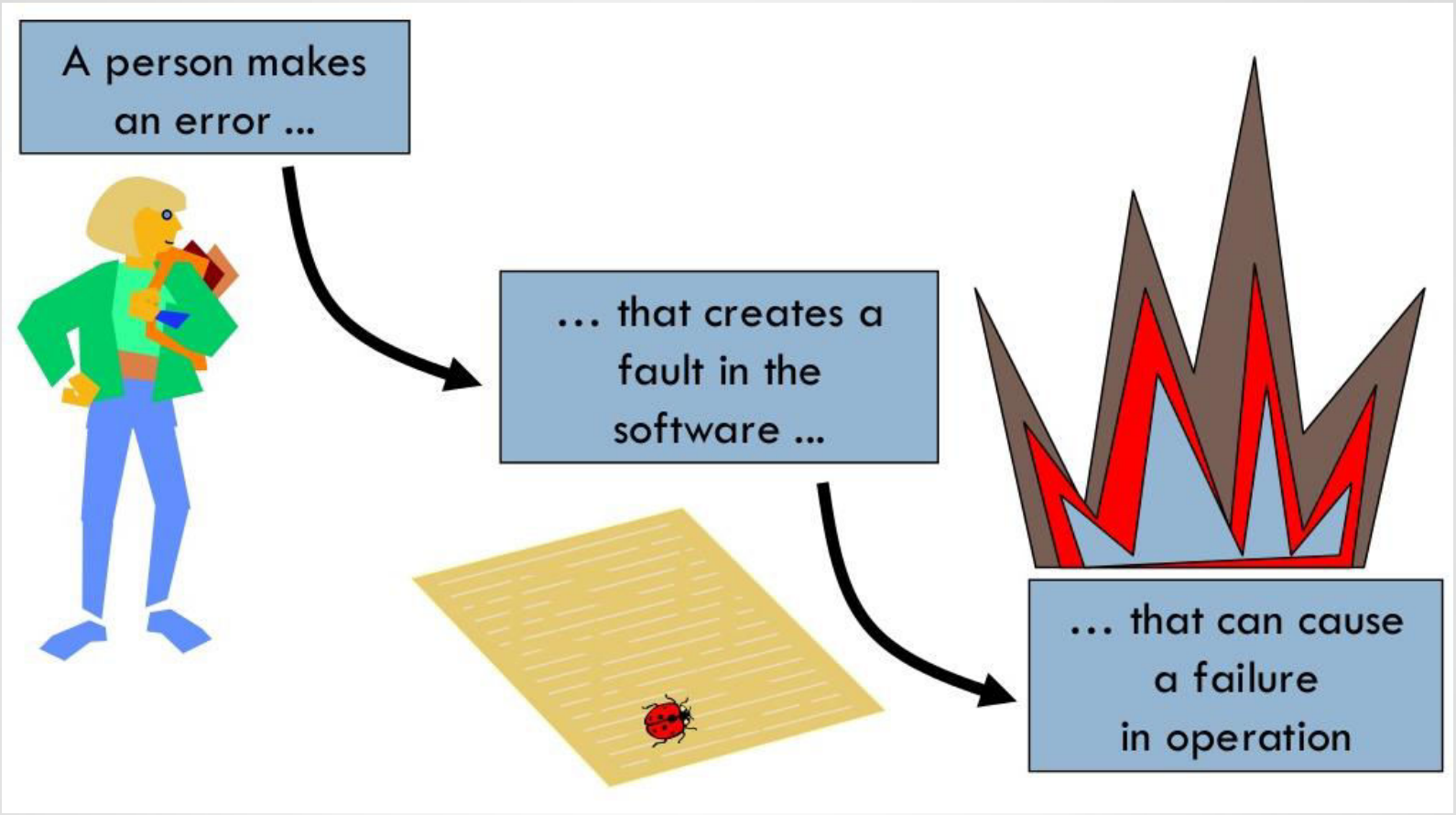
Failure

- Failure is the inability of a system or a component to perform its required functions within specified performance requirements.
- Failure occurs when fault executes.

Defect

- A defect is an error in coding or logic that causes a program to malfunction or to produce incorrect/unexpected results.
- A defect is said to be detected when a failure is observed.

Software testing terms



Error / Mistake	Defect / Bug/ Fault	Failure
Found by ↑	Found by ↑	Found by ↑
Developer	Tester	Customer

Benefits of Software Testing

What are the benefits of Software Testing?

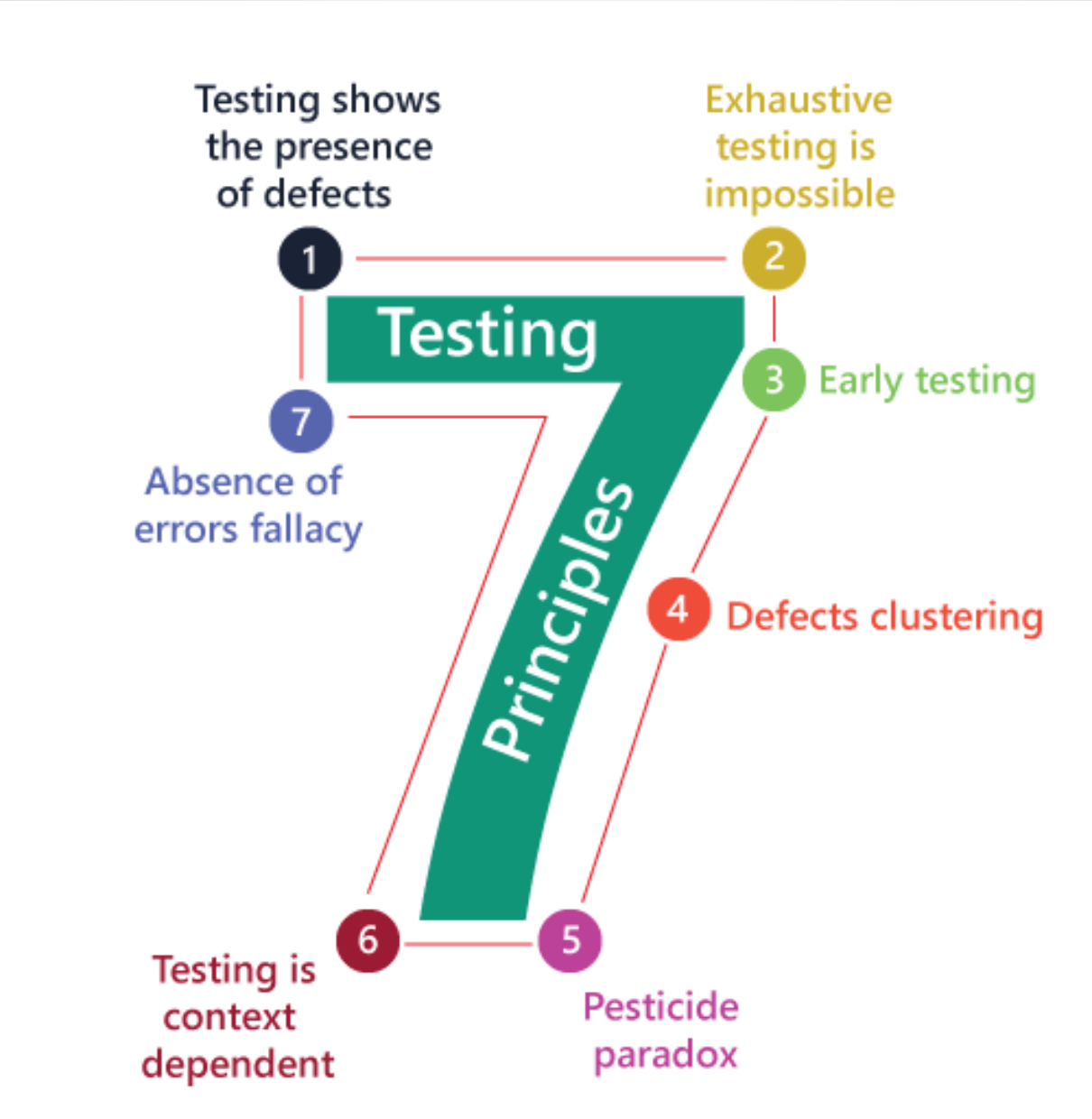
Cost-Effective: It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.

Security: It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

Product quality: It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.

Customer Satisfaction: The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

Software Testing Principles



Software Testing Principles

Testing shows presence of defects:

The goal of software testing is to make the software fail. Software testing reduces the presence of defects. Software testing **talks about the presence of defects and doesn't talk about the absence of defects.** Software testing can ensure that defects are present but it can not prove that software is defects free. Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not removes all defects.

Exhaustive testing is not possible:

It is the process of testing the functionality of a software in all possible inputs (valid or invalid) and pre-conditions is known as exhaustive testing. Exhaustive testing is impossible means the **software can never test at every test cases.** It can test only some test cases and assume that software is correct and it will produce the correct output in every test cases. If the software will test every test cases then it will take more cost, effort, etc. and which is impractical.

Early Testing:

To find the defect in the software, early test activity shall be started. The defect detected in early phases of SDLC will very less expensive. For better performance of software, software testing will start at initial phase i.e. testing will perform at the requirement analysis phase.

Software Testing Principles

Defect clustering:
In a project, a small number of the module can contain most of the defects**. Pareto Principle** to software testing state that **80% of software defect comes from 20% of modules**.

Pesticide paradox:
Repeating the same test cases again and again will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.

Testing is context dependent:
Testing approach depends on context of software developed. **Different types of software need to perform different types of testing**. For example, The testing of the e-commerce site is different from the testing of the Android application.

Absence of errors fallacy:
If a built software is 99% bug-free but it does not follow the user requirement then it is unusable. **It is not only necessary that software is 99% bug-free but it also mandatory to fulfill all the customer requirements.**

Testing Guidelines

- Development team should avoid testing the software
- Testing must be done with unexpected and negative inputs
- Software can never be 100% bug-free after making a number of test cases
- Start as early as possible
- The time available is limited
- Inspecting test results properly
- Validating assumptions

Software Testing Life Cycle

STLC stands for **Software Testing Life Cycle**. STLC is a sequence of different activities performed by the testing team to **ensure the quality of the software or the pro**duct.

- STLC is an integral part of Software Development Life Cycle (SDLC). But, STLC deals only with the testing phases.
- STLC starts as soon as requirements are defined or SRD (Software Requirement Document) is shared by stakeholders.
- STLC provides a step-by-step process to ensure quality software.

Software Testing Life Cycle

There are 6 major phases of STLC –

1. **Requirement Analysis** – When the SRD is ready and shared with the stakeholders, the testing team starts high level analysis concerning the AUT (Application under Test).
2. **Test Planning** – Test Team plans the strategy and approach.
3. **Test Case Designing/Developing** – Develop the test cases based on scope and criteria's.
4. **Test Environment Setup** – When integrated environment is ready to validate the product.
5. **Test Execution** – Real-time validation of product and finding bugs.
6. **Test Closure** – Once testing is completed, matrix, reports, results are documented.

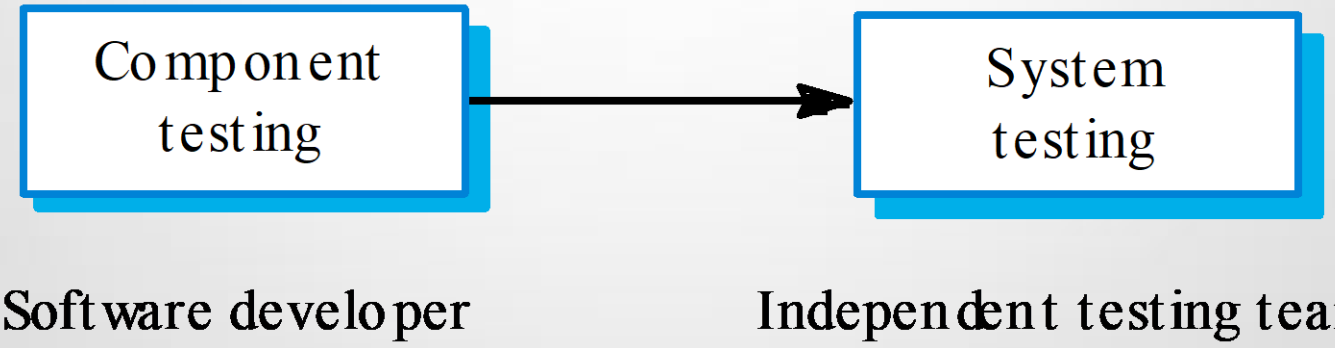
Testing process

Component testing

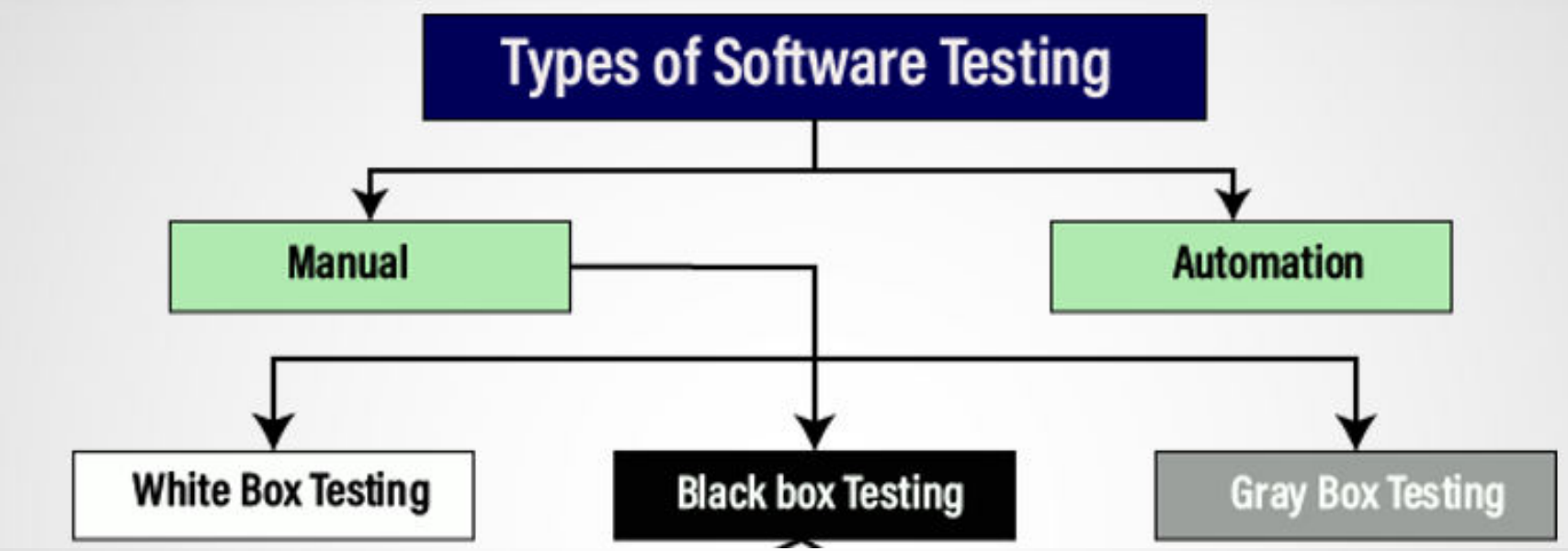
- Testing of individual program components;
- Usually the responsibility of the component developer (except sometimes for critical systems);
- Tests are derived from the developer's experience.

System testing

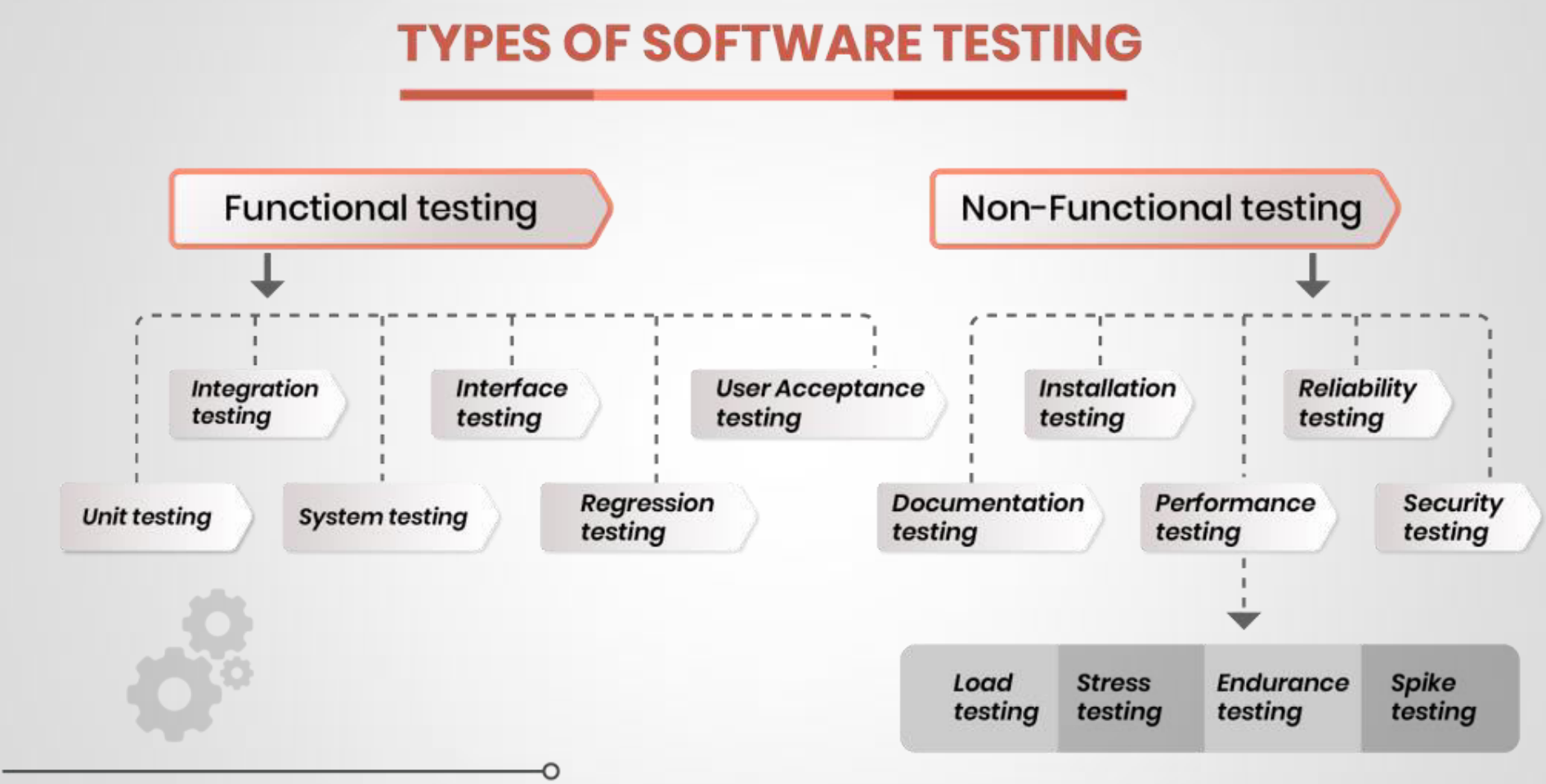
- Testing of groups of components integrated to create a system or sub-system;
- The responsibility of an independent testing team;
- Tests are based on a system specification



Software Testing



Software Testing



Manual testing

The process of checking the functionality of an application as per the customer needs **without taking any help of automation** tools is known as manual testing.

While performing the manual testing on any application, we do not need any specific knowledge of any testing tool, rather than have a proper understanding of the product so we can easily prepare the test document.

Manual testing can be further divided into three types of testing, which are as follows:

- **White box testing**
- **Black box testing**
- **Gray box testing**

Manual testing

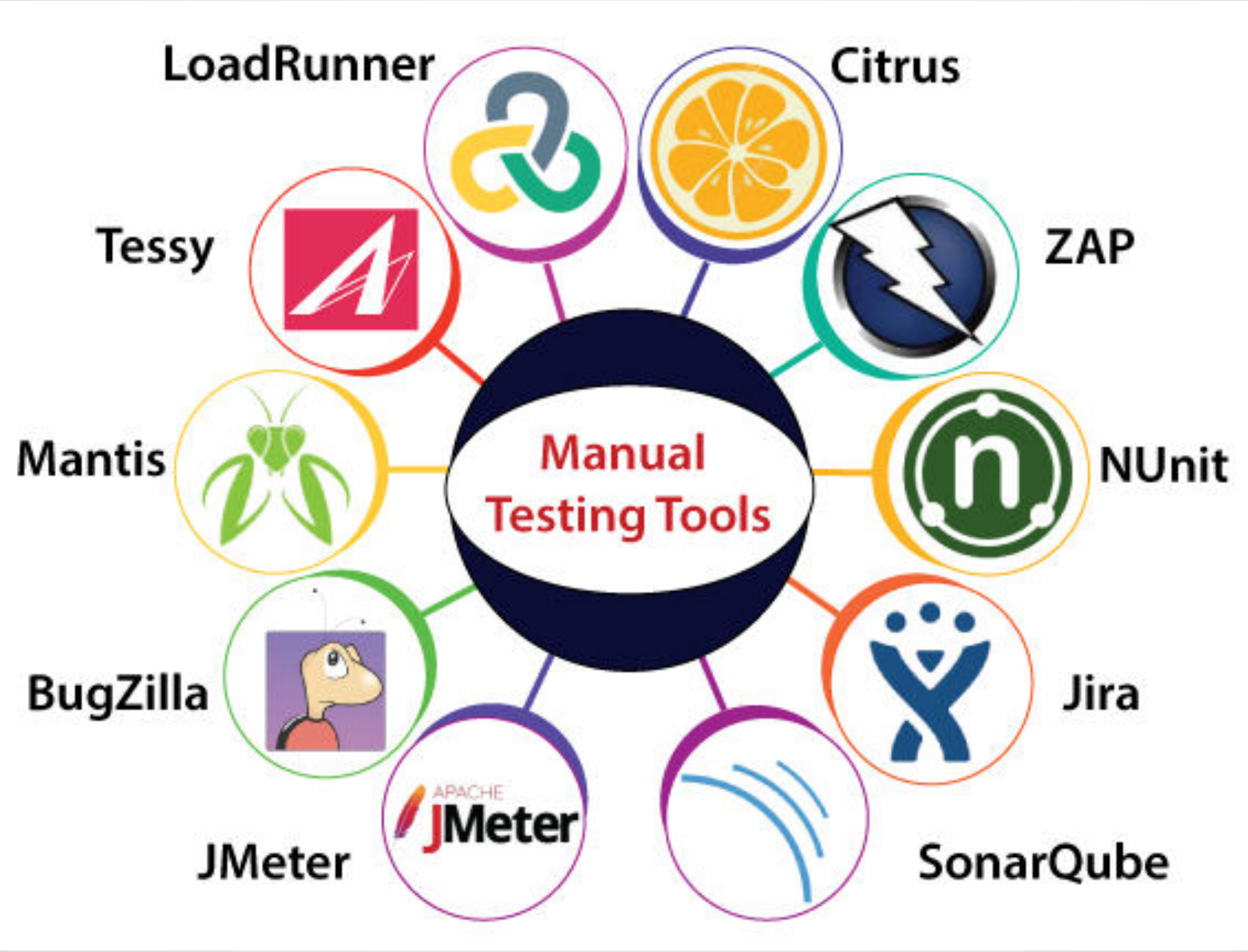
Advantages of Manual Testing

- It does not require programming knowledge while using the Black box method.
- It is used to test dynamically changing GUI designs.
- Tester **interacts with software as a real user** so that they are able to discover usability and user interface issues.
- It is **cost-effective**.
- Easy to learn for new testers.

Disadvantages of Manual Testing

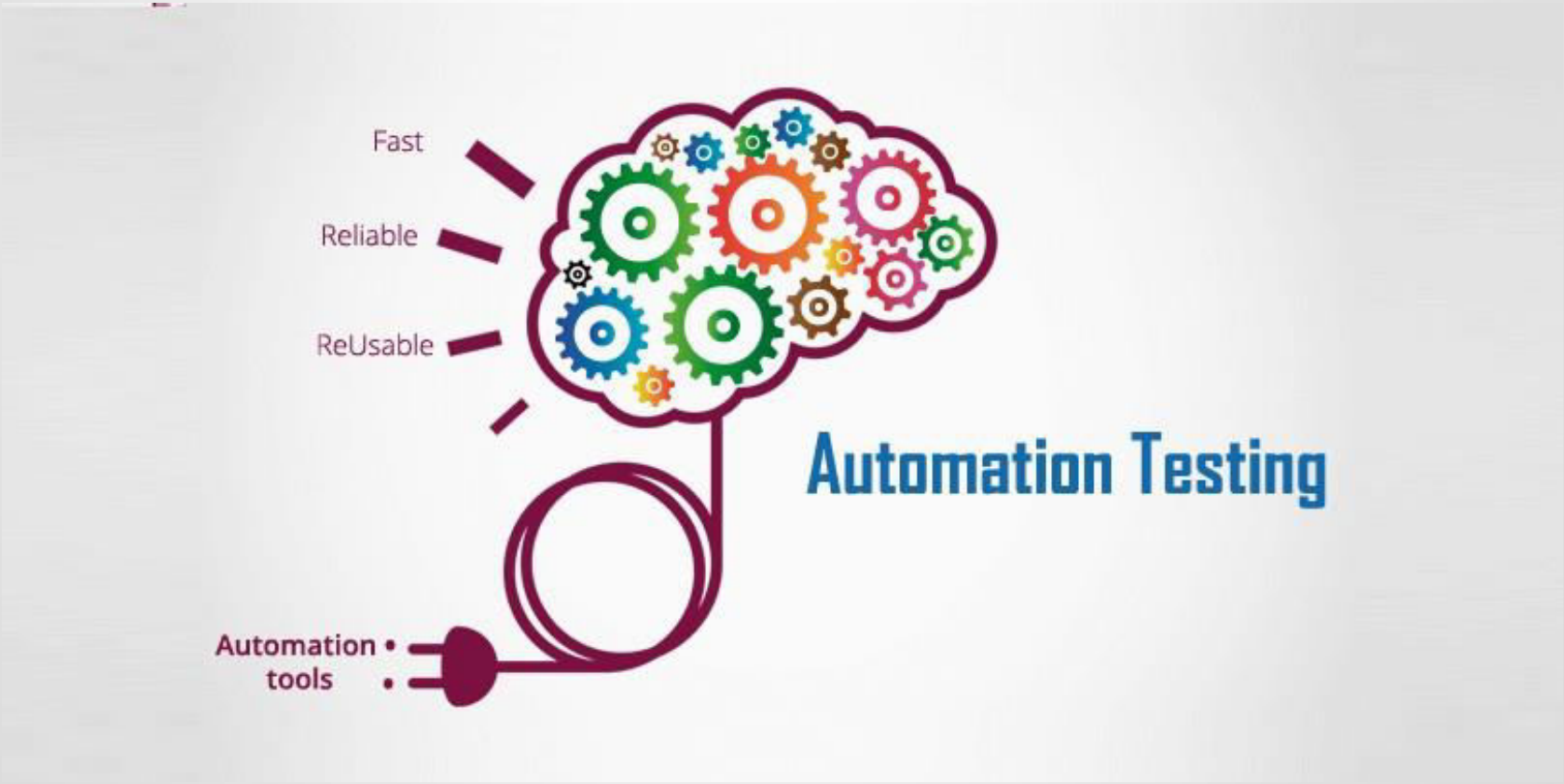
- It requires a large number of human resources.
- It is **very time-consuming**.
- Tester develops test cases based on their skills and experience. There **is no evidence that they have covered all functions or not**.
- Test cases **cannot be used again**. Need to develop separate test cases for each new software.
- It does not provide testing on all aspects of testing.
- Since two teams work together, sometimes it is difficult to understand each other's motives, **it can mislead the process**.

Manual testing tools

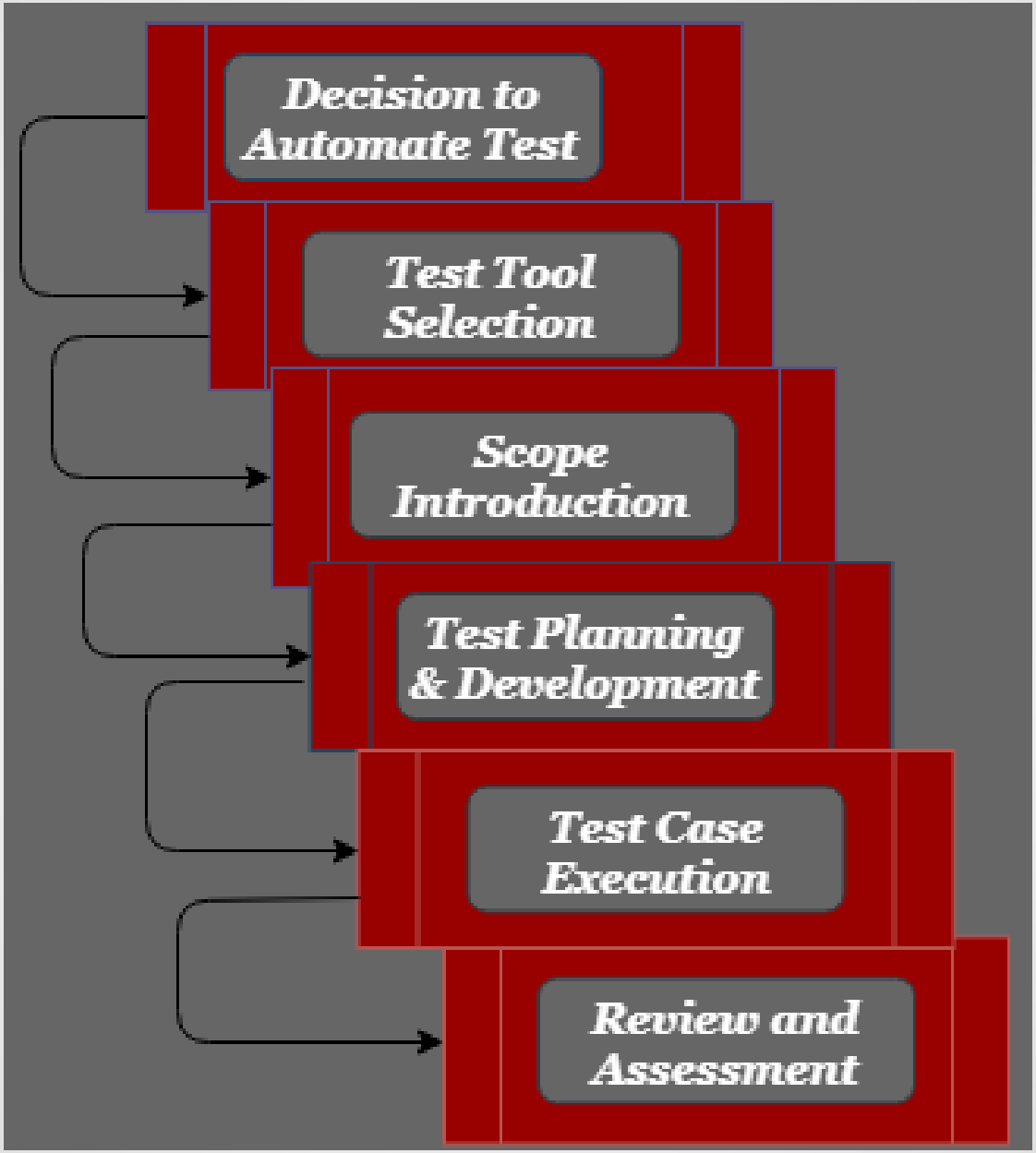


Automation testing

Automation testing is a process of converting any manual test cases into the test scripts with the help of automation tools, or any programming language is known as automation testing.



Life cycle of Automation Testing



Automation Testing

Advantages of Automation Testing

- Automation testing takes **less time** than manual testing.
- Automation Testing provides **re-usability of test cases** on testing of different versions of the same software.
- Automation testing is **reliable as it eliminates hidden errors** by executing test cases again in the same way.
- It does not require many human resources, instead of writing test cases and testing them manually, they need an automation testing engineer to run them.

Disadvantages of Automation Testing

- Automation Testing **requires high-level skilled testers**.
- It requires **high-quality testing tools**.
- When it encounters an unsuccessful test case, the analysis of the whole event is complicated.
- Test **maintenance is expensive** because high fee license testing equipment is necessary.
- Debugging is mandatory if a less effective error has not been solved, it can lead to fatal results.

Black box Testing

- **Black box testing** is a technique of software testing which examines the functionality of software **without peering into its internal structure or coding**.
- Black box testing is a method of software testing that **examines the functionality of an application**
- This method of test can be applied to virtually every level of software
- The tester is oblivious to the system architecture and does not have access to the source code

Black box testing

Generic steps of black box testing

- The black box test is based on the specification of requirements, so it is examined in the beginning.
- In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
- In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
- The fourth phase includes the execution of all test cases.
- In the fifth step, the tester compares the expected output against the actual output.
- In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

Black box testing

Black-box testing techniques:

Decision Table Technique	Decision Table Technique is a systematic approach where various input combinations and their respective system behavior are **captured in a tabular form.** It is appropriate for the functions that have a logical relationship between two and more than two inputs.
Boundary Value Technique	Boundary Value Technique is **used to test boundary values**, boundary values are those that contain the upper and lower limit of a variable. It tests, while entering boundary value whether the software is producing correct output or not.
State Transition Technique	State Transition Technique is used to capture the b**ehavior of the software application when different input values are given to the same function**. This applies to those types of applications that provide the specific number of attempts to access the application.
All-pair Testing Technique	All-pair testing Technique is used **to test all the possible discrete combinations of values**. This combinational method is used for testing the application that uses checkbox input, radio button input, list box, text box, etc.

Black box testing

Black-box testing techniques:

Cause-Effect Technique	Cause-Effect Technique underlines the **relationship between a given result and all the factors affecting the result**.It is based on a collection of requirements.
Equivalence Partitioning Technique	Equivalence partitioning is a technique of software testing in **which input data divided into partitions of valid and invalid values**, and it is mandatory that all partitions must exhibit the same behavior.
Error Guessing Technique	Error guessing is a technique in which there is no specific method for identifying the error. It is based on the experience of the test analyst, where the tester uses **the experience to guess the problematic areas of the software**.
Use Case Technique	Use case Technique used to identify the test cases from the beginning to the end of the system as per the usage of the system. By using this technique, the test team creates a test scenario that can exercise the entire software based on the functionality of each function from start to end.

Black box testing

Advantages of Black Box Testing

– Tester can be non-technical.

– Used to verify contradictions in actual system and the specifications.

– Test cases can be designed as soon as the functional specifications are complete

Disadvantages of Black Box Testing

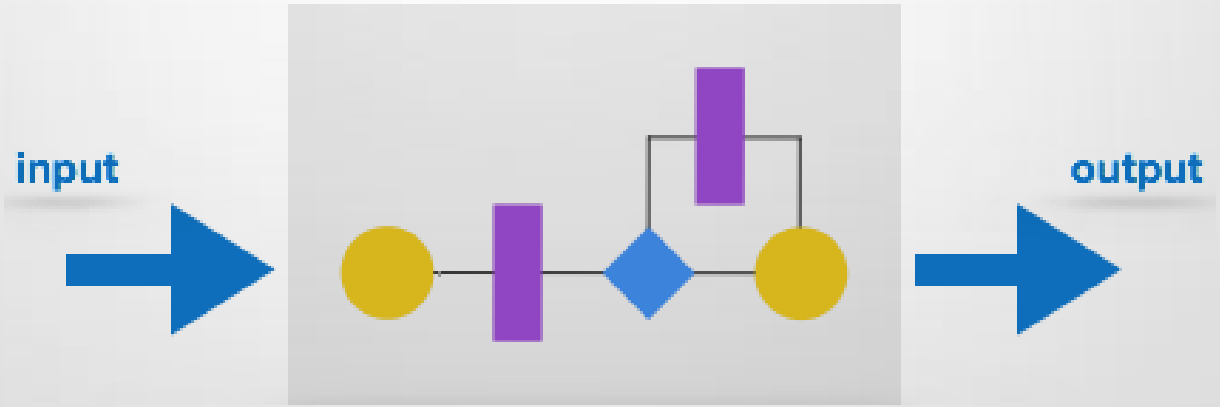
– The test inputs needs to be from large sample space.

– It is difficult to identify all possible inputs in limited testing time. So writing test cases is slow and difficult

– Chances of having unidentified paths during this testing

White Box Testing

- **White box testing** is a method of testing software that **tests internal structures or working of an application**
- In white-box testing an internal perspective of the system , as well as programming skills, are used to design test cases
- It is also known as **clear box testing, glass box testing, transparent box testing, and structural testing**
- White box testing is the detailed investigation of internal logic and structure of the code
- In order to perform white box testing of an application , the tester needs to possess knowledge of the **internal working of the code**

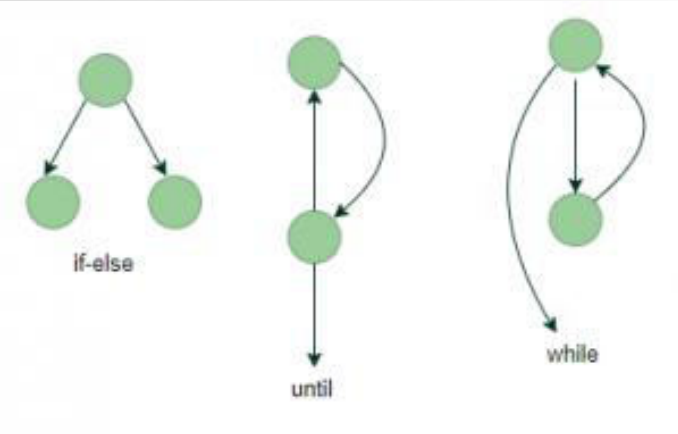


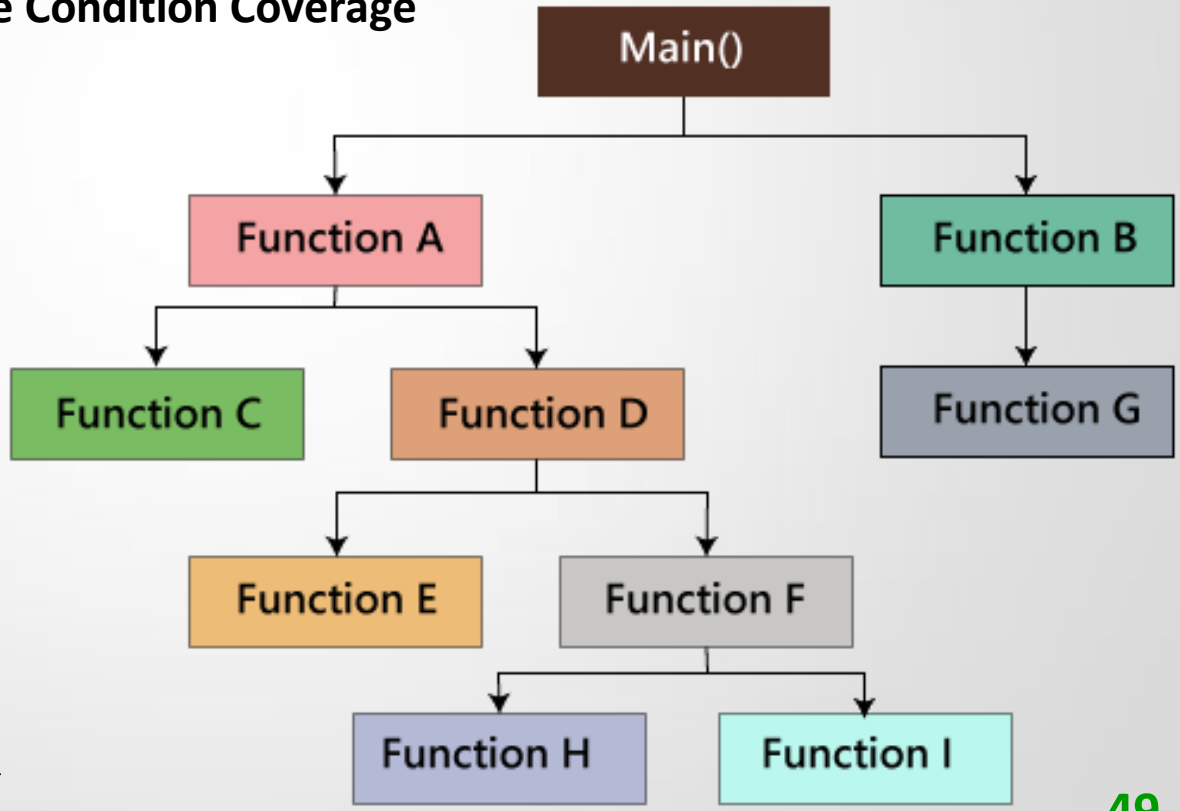
White Box Testing

White Box Testing techniques:

The white box testing contains various parts, which are as follows:

- Statement coverage
- Testing based on the memory (size) perspective
- Condition testing : **Multiple Condition Coverage**
- Basis Path test
- Flow graph notation
- Loop Testing





https://www.geeksforgeeks.org/software-engineering-white-box-testing/?ref=lbp

White Box Testing

Advantages:

- White box testing is very thorough as **the entire code and structures are tested**.
- It results in the optimization of code removing error and helps in removing extra lines of code.
- It can start at an earlier stage as it doesn't require any interface as in case of black box testing.
- Easy to **automate**.

Disadvantages:

- Main disadvantage is that it is **very expensive**.
- Redesign of code and rewriting code needs test cases to be written again.
- Testers are required to **have in-depth knowledge of the code and programming** language as opposed to black box testing.
- Missing functionalities cannot be detected as the code that exists is tested.
- Very complex and at times not realistic.

Black Box Testing vs White Box Testing

BLACK BOX TESTING	WHITE BOX TESTING
It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure r the code or the program of the software.
It is mostly done by **software testers**.	It is mostly done by **software developers**.
No knowledge of implementation is needed.	Knowledge of implementation is required.
It can be referred as outer or external software testing.	It is the inner or the internal software testing.
It is **functional test** of the software.	It is **structural test** of the software.
This testing can be initiated on the basis of requirement specifications document.	This type of testing of software is started after detail design document.

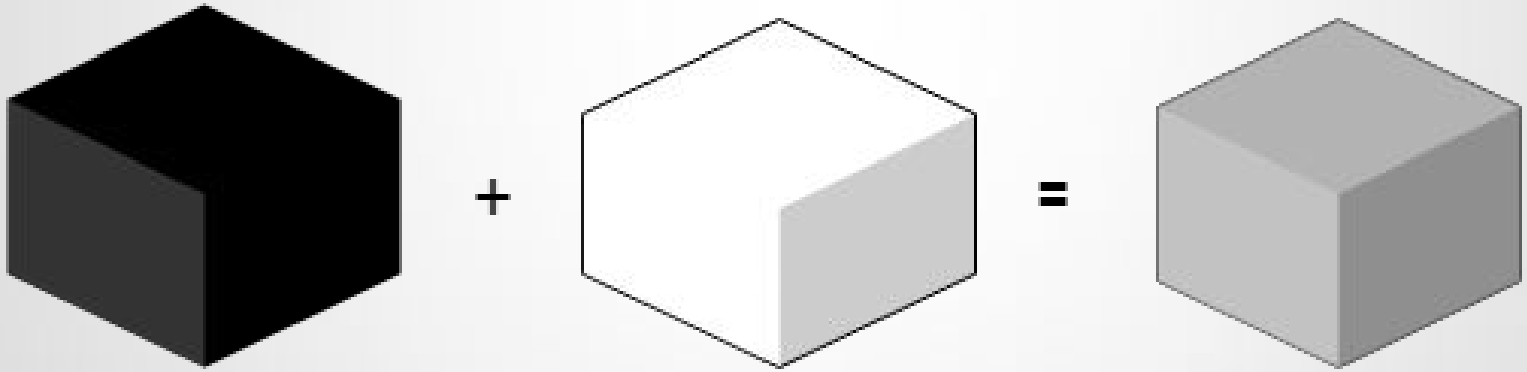
Black Box Testing vs White Box Testing (cont.)

BLACK BOX TESTING	WHITE BOX TESTING
No knowledge of programming is required.	It is mandatory to have knowledge of programming.
It is the **behavior testing** of the software.	It is the **logic** testing of the software.
It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
It is also called **closed testing**.	It is also called as **clear box testing**.
It is least time consuming.	It is most time consuming.
It is not suitable or preferred for algorithm testing.	It is suitable for algorithm testing.
Can be done by trial and error ways and methods.	Data domains along with inner or internal boundaries can be better tested.

GreyBox Testing

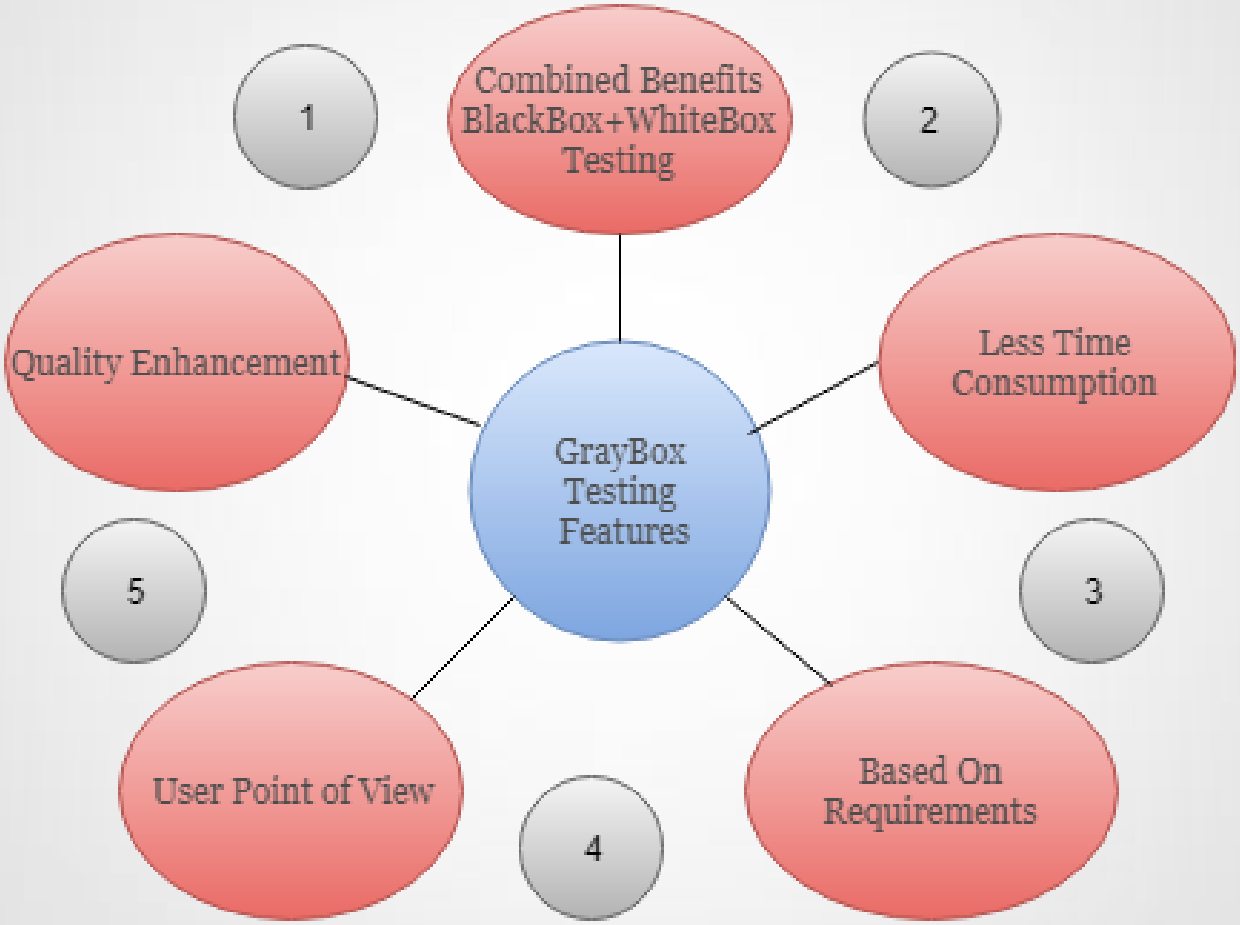
Grey-box testing is a software testing method to test the software application with partial knowledge of the internal working structure.

It is a **combination of black box and white box testing** because it involves access to internal coding to design test cases as white box testing and testing practices are done at functionality level as black box testing.



GreyBox Testing

Why GreyBox testing?



GreyBox Testing

Why GreyBox testing?

Reasons for GreyBox testing are as follows

- It provides combined benefits of both Blackbox testing and WhiteBox testing.
- It includes the input values of **both developers and testers** at the same time to **improve the overall quality** of the product.
- It **reduces time consumption** of long process of functional and non-functional testing.
- It **gives sufficient time** to the developer to fix the product defects.
- It includes user point of view rather than designer or tester point of view.
- It involves examination of requirements and determination of specifications by user point of view deeply.

GreyBox Testing

Gray Box Testing Techniques:

- Matrix Testing
- Pattern Testing
- Orthogonal Array Testing
- Regression Testing

https://www.geeksforgeeks.org/gray-box-testing-software-testing/?ref=rp

GreyBox Testing

Advantages of Gray Box Testing:

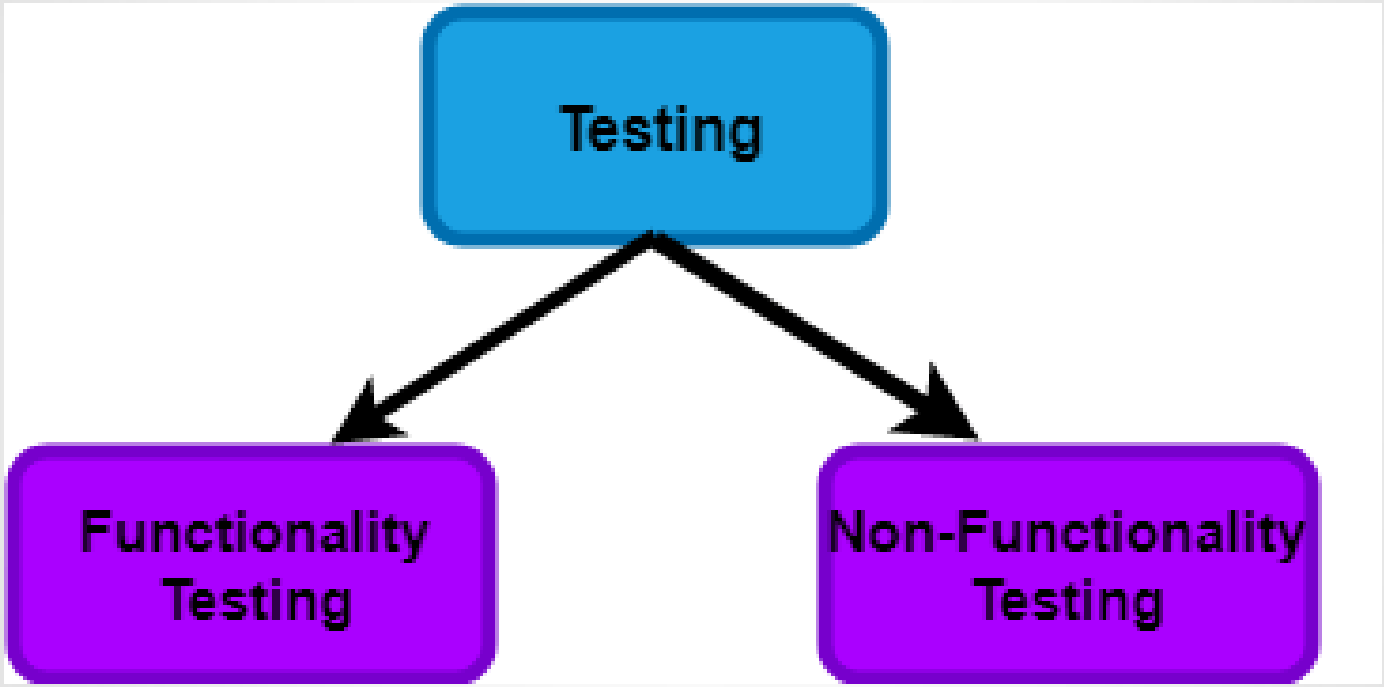
- Gray box testing is mostly done by the **user perspective**.
- Testers are **not required to have high programming skills** for this testing.
- In gray box testing, developers have more **time for defect fixing**.
- By doing gray box testing, benefits of both black box and white box testing is obtained.
- Gray box testing is **unbiased**. It avoids conflicts between a tester and a developer.

Disadvantages of gray box testing:

- Defect association is difficult when gray testing is performed for **distributed systems.**
- Limited access to internal structure leads to limited access for code **path traversal**.
- Because source code cannot be accessed, doing complete white box testing is not possible.
- Gray box testing is not suitable for **algorithm testing**.

https://www.geeksforgeeks.org/gray-box-testing-software-testing/?ref=rp

Testing Approaches



Testing Approaches

Functional Testing:

It is a type of software testing which is used to **verify the functionality of the software application,** whether the function is **working according to the requirement specification.**

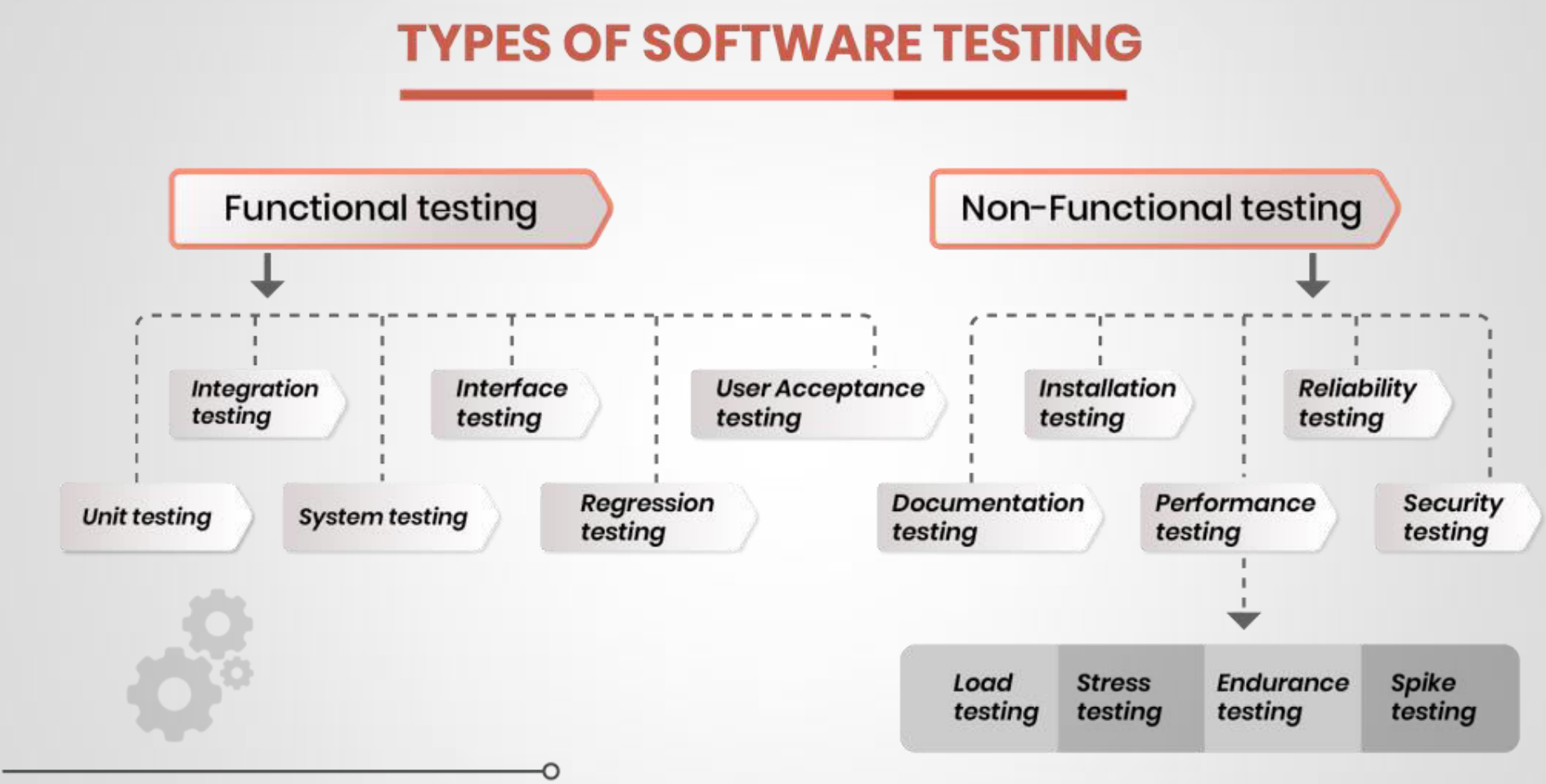
- Tester does verification of the requirement specification in the software application.
- After analysis, the requirement specification tester will make a plan.
- After planning the tests, the tester will design the test case.
- After designing the test, case tester will make a document of the traceability matrix.
- The tester will execute the test case design.
- Analysis of the coverage to examine the covered testing area of the application.
- Defect management should do to manage defect resolving.

Testing Approaches

Non-Functional Testing

Non-functional testing is a type of software testing to test **non-functional parameters** such as reliability, load test, performance and accountability of the software. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters.

Software Testing



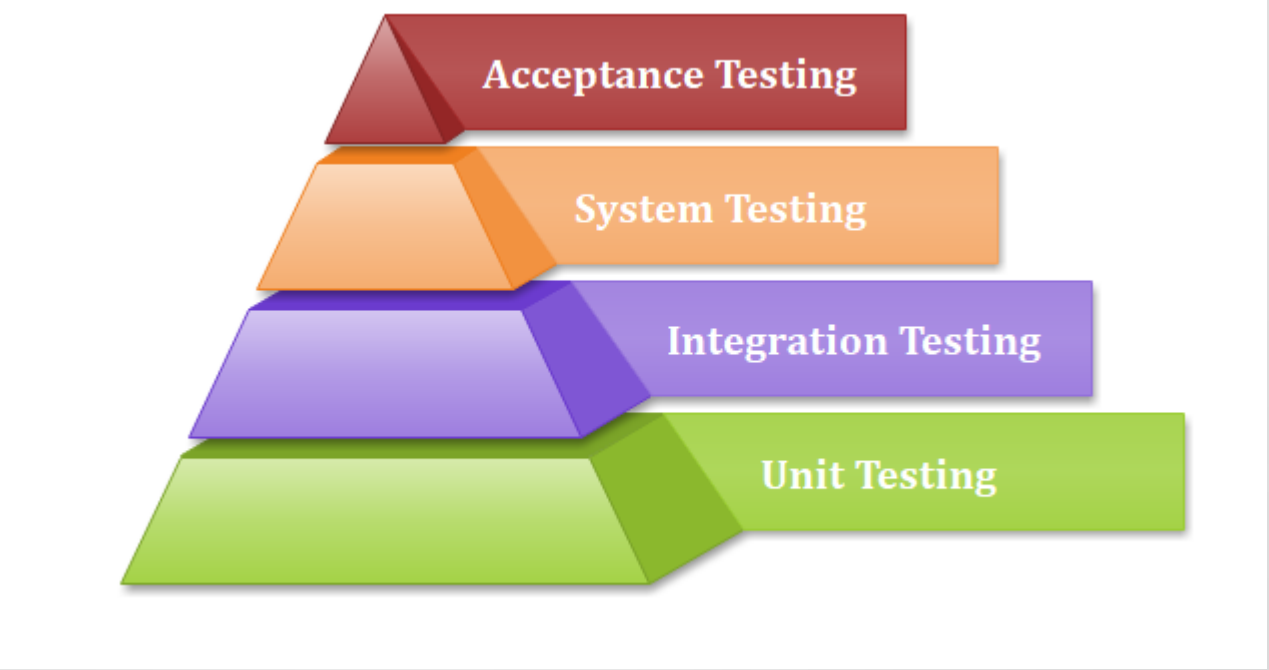
Unit Testing

Unit testing is a type of software testing, where the **individual unit or component of the software tested**. Unit testing, **examine the different part of the application**, by unit testing functional testing also done, because unit testing ensures each module is working correctly.

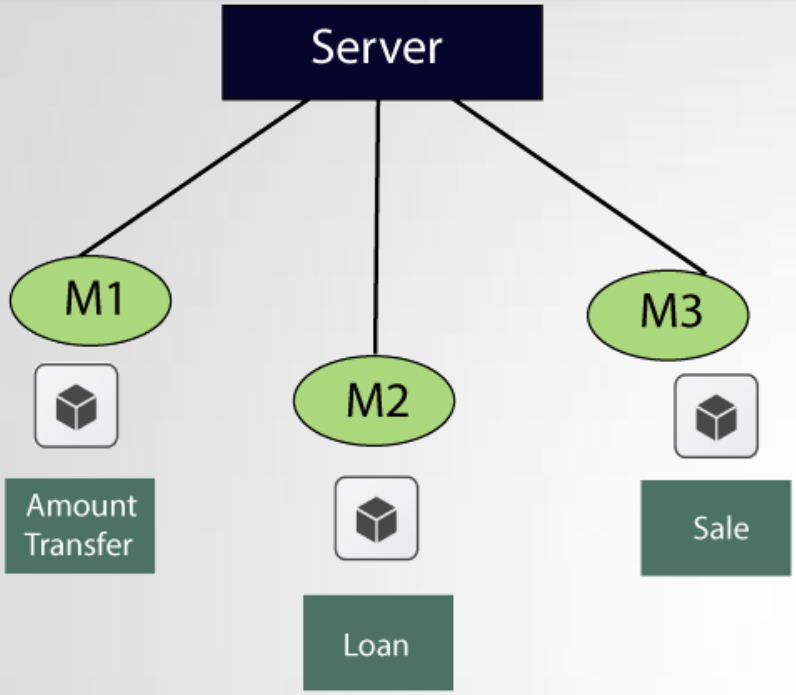
The developer does unit testing. Unit testing is done in the development phase of the application.

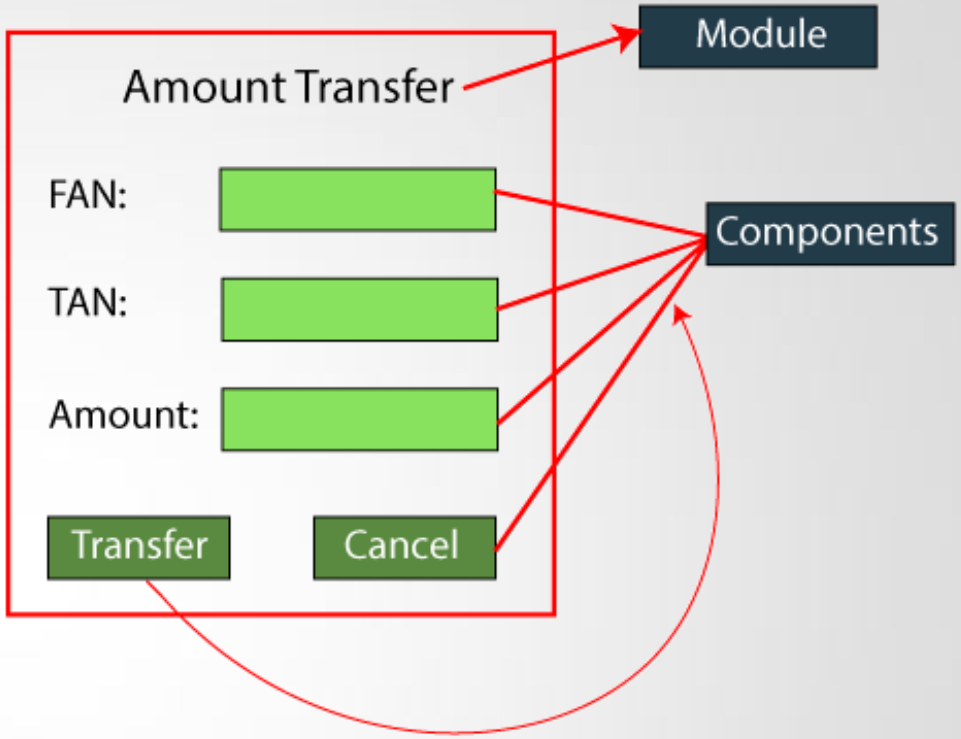
Unit Testing Tools

- NUnit
- JUnit
- PHPunit
- Parasoft Jtest
- EMMA



Unit Testing





Unit Testing Techniques:

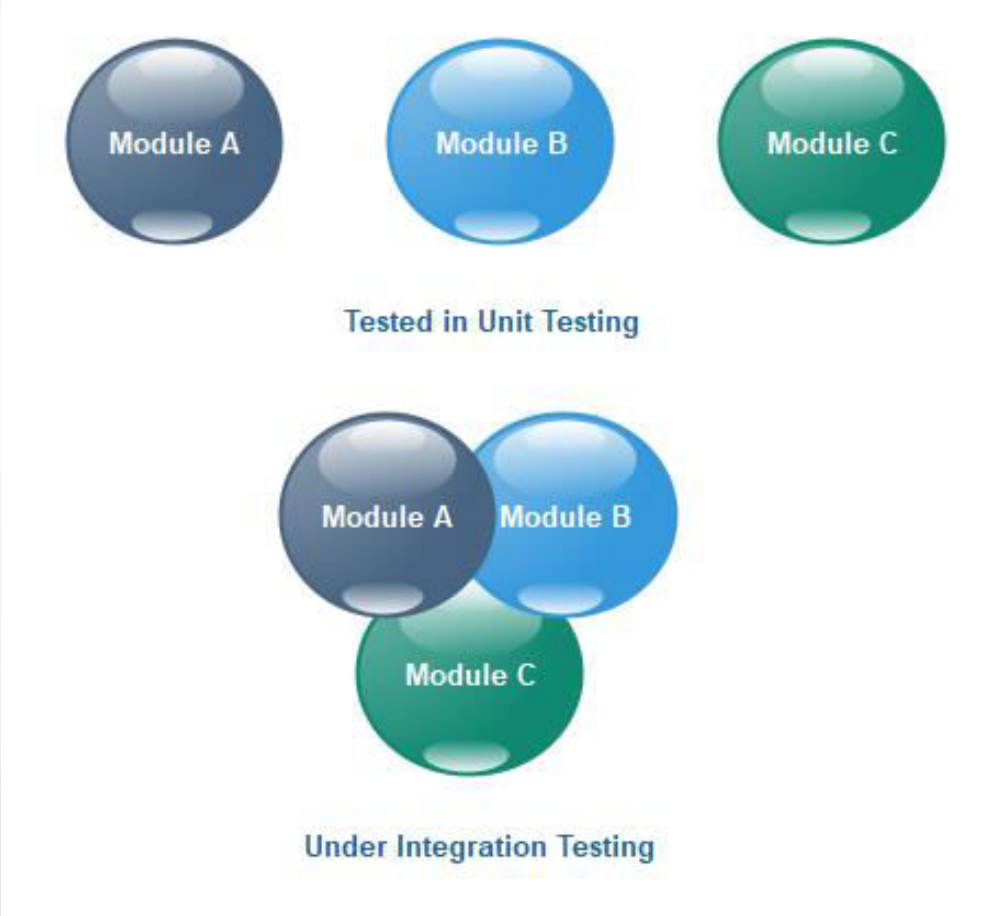
Unit testing uses all white box testing techniques as it uses the code of software application:

- **Data flow Testing**
- **Control Flow Testing**
- **Branch Coverage Testing**
- **Statement Coverage Testing**
- **Decision Coverage Testing**

Integration testing

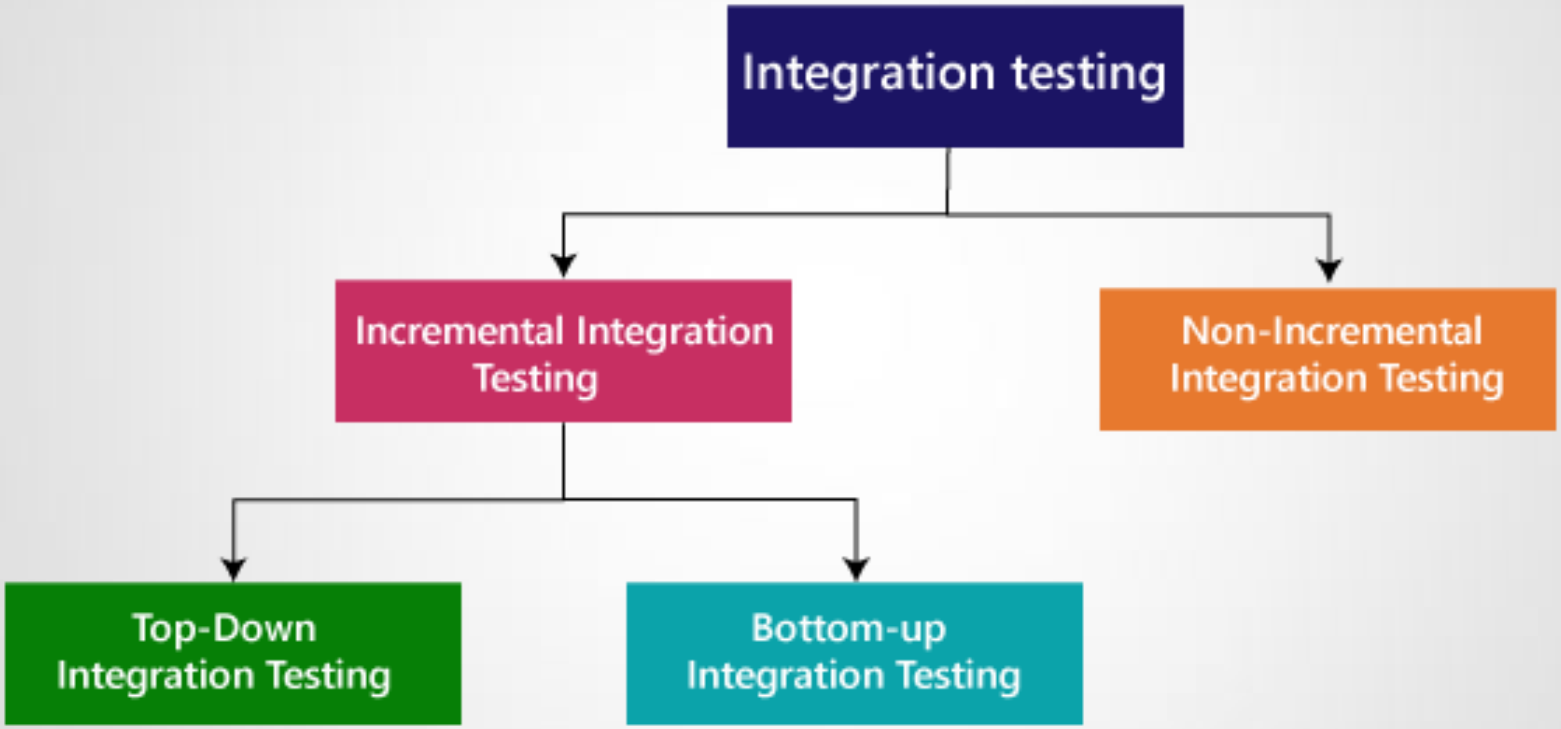
Integration testing

Integration testing is the second level of the software testing process **comes after unit testing**. In this testing, units or individual components of the software are **tested in a grou**p. The focus of the integration testing level is to **expose defects at the time of interaction between integrated components or units**.



Integration testing

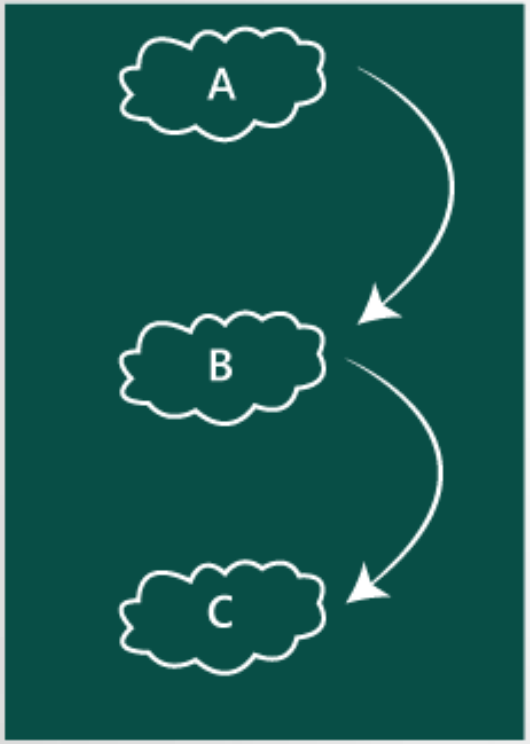
Types of Integration Testing



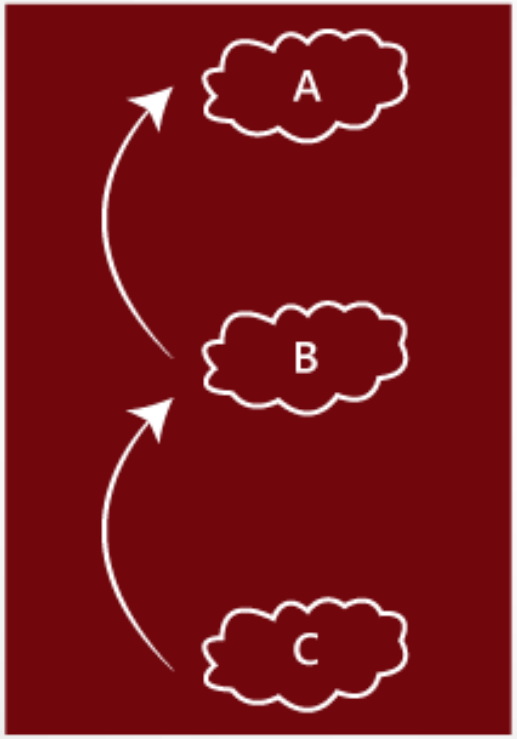
https://www.javatpoint.com/integration-testing

Integration testing

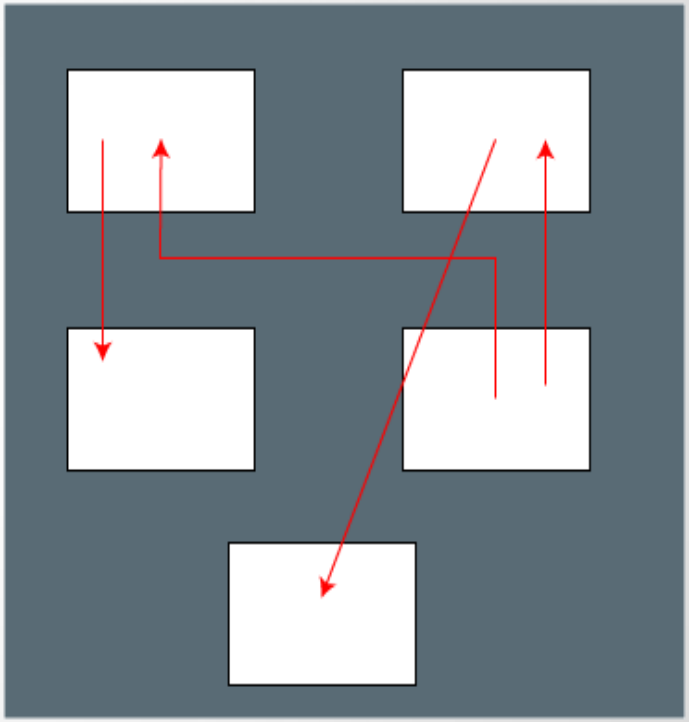
Top-Down Approach



Bottom-up Appoarch



Non- incremental



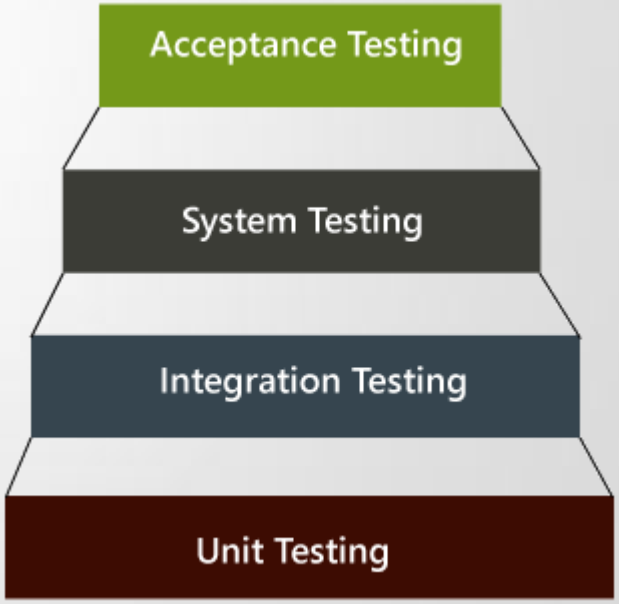
System Testing

System Testing

The software is compiled as product and then it is tested as a whole. This can be accomplished using one or more of the following tests:

- **Functionality testing** - Tests all functionalities of the software against the requirement.
- **Performance testing** - This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.

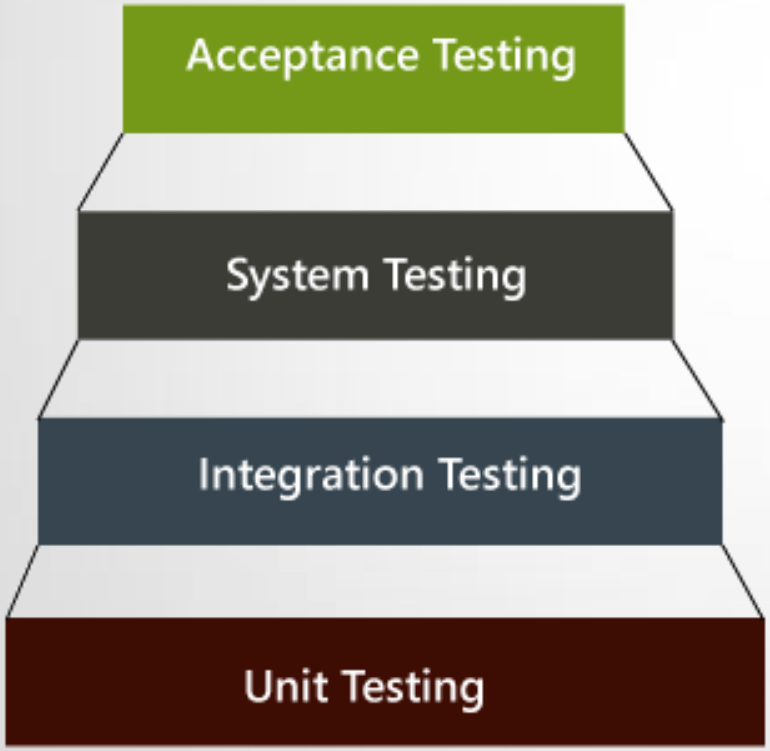
- **Security & Portability** - These tests are done when the software is meant to work on various platforms and accessed by number of persons.

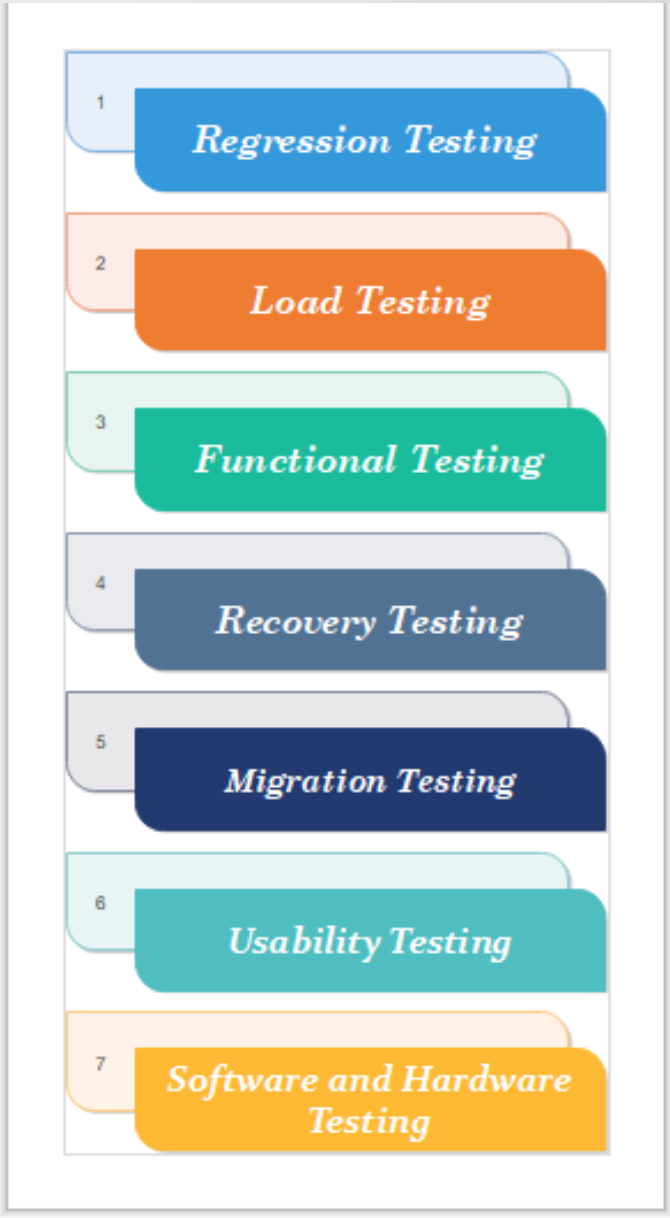


System Testing

System Testing

- The software is compiled as product and then it is tested as a whole.
- System Testing includes testing of a fully integrated software system.
- System testing is divided into more than 50 types.





Acceptance Testing : User testing

Acceptance Testing

When the software is ready to hand over to the customer it has to go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if user does not like the way it appears or works, it may be rejected.

- **Alpha testing** - The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find out how user would react to some action in software and how the system should respond to inputs.

- **Beta testing** - After the software is tested internally, it is handed over to the users to use it under their production environment only for testing purpose. This is not as yet the delivered product. Developers expect that users at this stage will bring minute problems, which were skipped to attend.

Different types of testing

Regression Testing

Whenever a software product is updated with new code, feature or functionality, it is tested thoroughly to detect if there is any negative impact of the added code. This is known as regression testing.

Load Testing

Load testing is performed under system testing to clarify whether the system can work under real-time loads or not.

Stress testing

The stress testing is testing, which checks the behavior of an application by applying load greater than the desired load. Since it is non-functional testing, so we use this testing when the application is functionally stable.

Recovery Testing

Recovery testing of a system is performed under system testing to confirm reliability, trustworthiness, accountability of the system and all are lying on recouping skills of the system. It should be able to recover from all the possible system crashes successfully.
In this testing, we will test the application to check how well it recovers from the crashes or disasters.

Different types of testing

Migration Testing

Migration testing is performed to ensure that if the system needs to be modified in new infrastructure so it should be modified without any issue.

Usability Testing

The purpose of this testing to make sure that the system is well familiar with the user and it meets its objective for what it supposed to do.

Database Testing: Database testing is a type of testing which **checks the schema, tables, triggers, etc. of the database under test**. Database testing may involve creating complex queries to load/stress test the database and check its responsiveness. It checks the data integrity and consistency

Release testing the process of testing a particular release of a system that is intended for use outside of the development team. Release testing has a broad focus, since the entire functionality of the release is under test. The tests included in release testing is strongly dependent on the software itself.

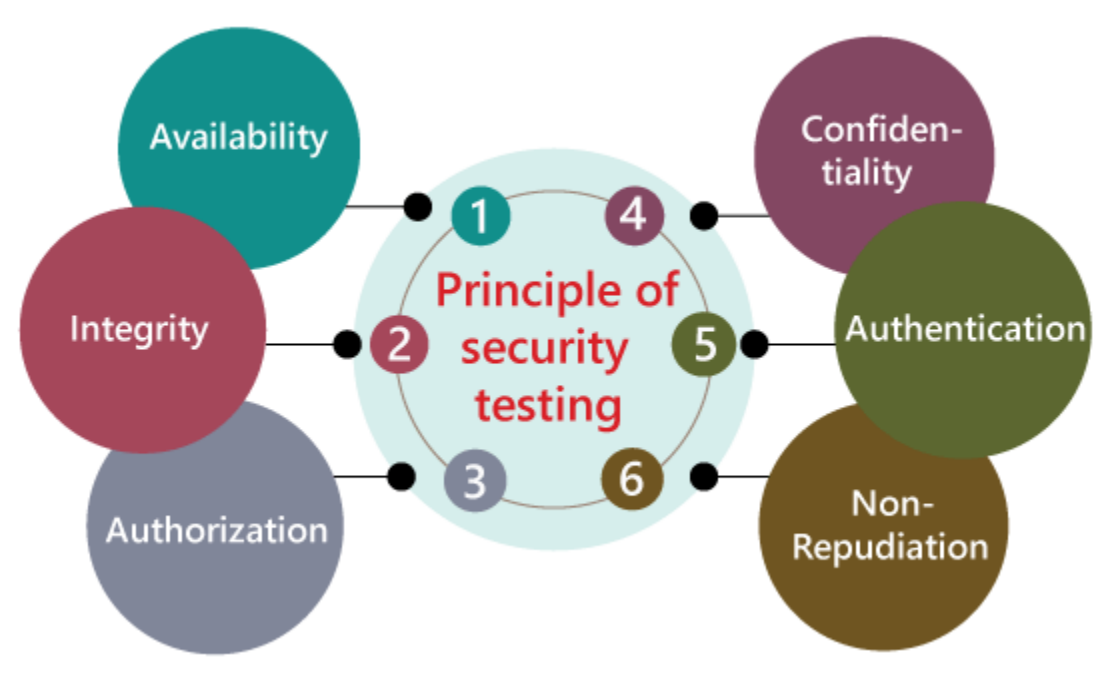
- **Requirements-based testing**
- **Scenario testing**

Different types of testing

Security testing

Security testing is an integral part of software testing, which is used to **discover the weaknesses, risks, or threats** in the software application and also help us to stop the nasty attack from the outsiders and make sure the security of our software applications.

The primary objective of security testing is to find all the potential ambiguities and vulnerabilities of the application so that the software does not stop working. If we perform security testing, then it helps us to identify all the possible security threats and also help the programmer to fix those errors.

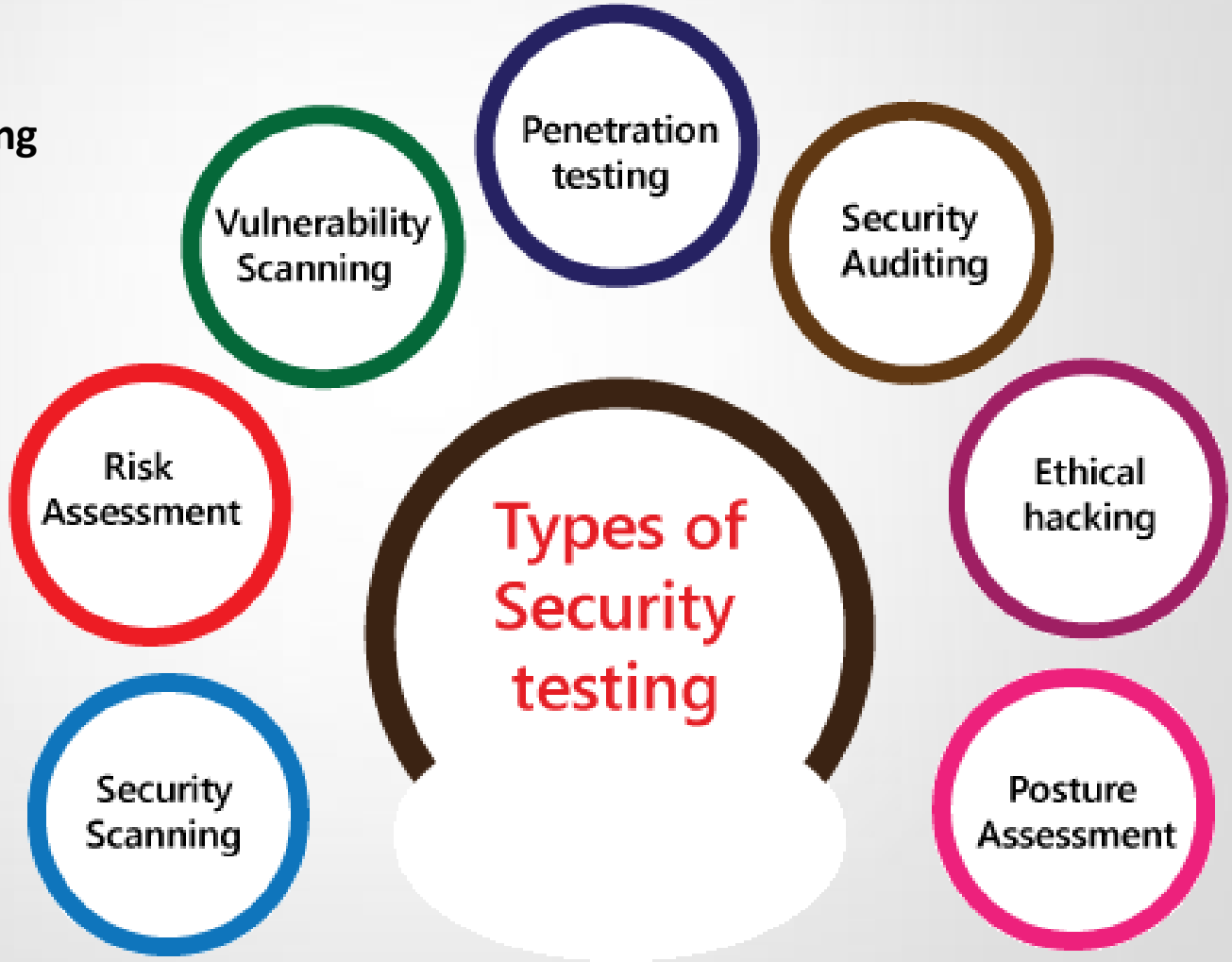


Security testing

Types of Security testing

As per Open Source Security Testing techniques, we have different types of security testing which as follows:

- **Security Scanning**
- **Risk Assessment**
- **Vulnerability Scanning**
- **Penetration testing**
- **Security Auditing**
- **Ethical hacking**
- **Posture Assessment**



Alpha testing vs Beta testing

Alpha Testing	Beta Testing
Alpha testing performed by a team of highly skilled testers who are usually the internal employee of the organization.	Beta testing performed by clients or end-users in a real-time environment, who is not an employee of the organization.
Reliability or security testing not performed in-depth in alpha testing.	Reliability, security, and robustness checked during beta testing.
Alpha testing involves both white box and black-box techniques.	Beta testing uses only black-box testing.
Long execution cycles maybe require for alpha testing.	Only a few weeks are required for the execution of beta testing.
Alpha testing performed before the launch of the product into the market.	At the time of software product marketing.
Alpha testing focuses on the product's quality before going to beta testing.	Beta testing concentrates on the quality of the product, but gathers users input on the product and ensures that the product is ready for real-time users.
Alpha testing performed nearly the end of the software development.	Beta testing is a final test before shipping a product to the customers.
Alpha testing is conducting in the presence of developers and the absence of end-users.	Beta testing reversed of alpha testing.

Re-testing Vs Regression Testing

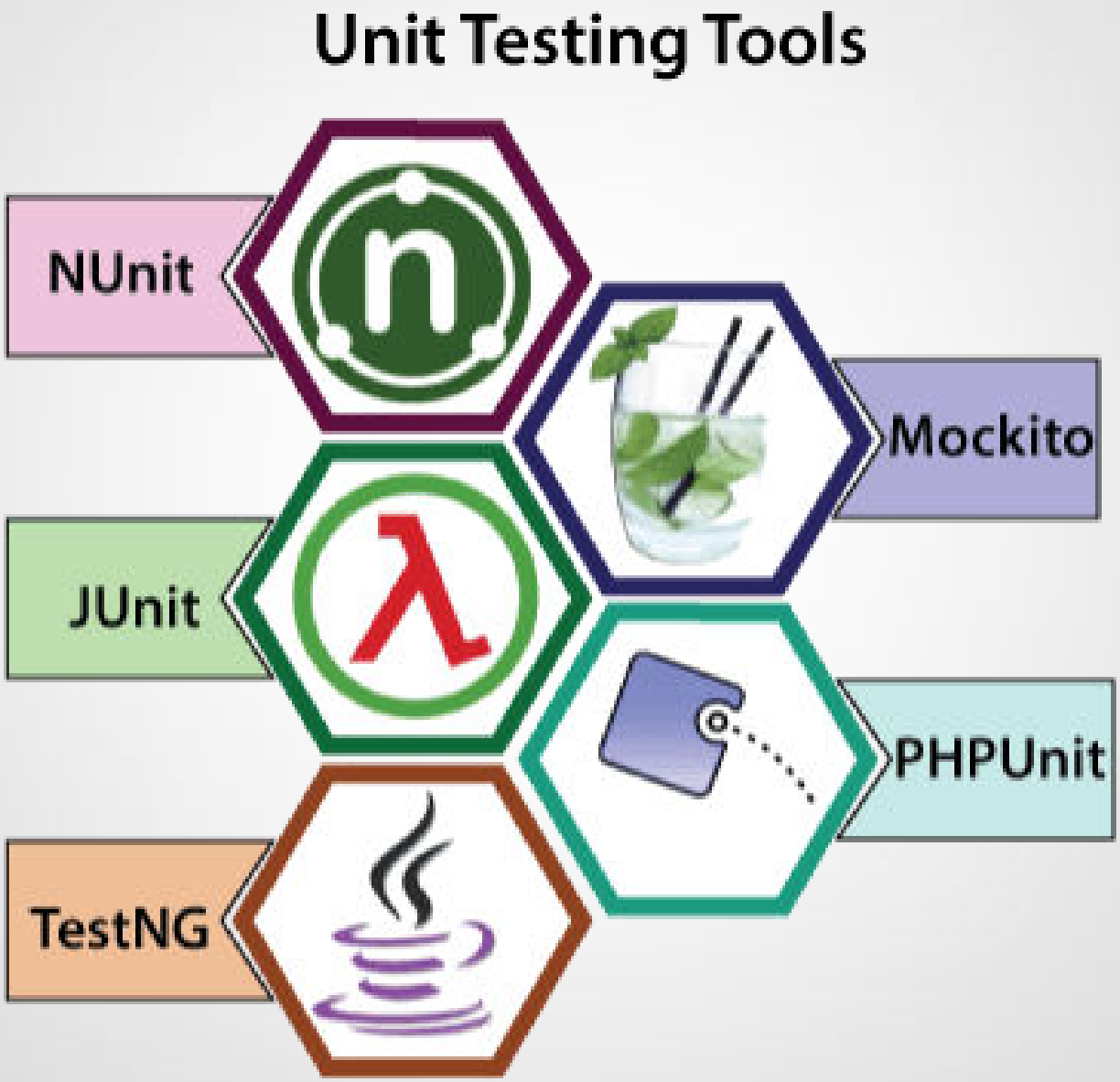
Re-testing	Regression Testing
Re-testing is performed to ensure that the test cases that are failed in the final execution are passing after the defects fixed.	Regression Testing is done to confirm whether the code change has not affected the existing features.
Re-Testing works on defect fixes.	The purpose of regression testing is to ensure that the code changes adversely not affect the existing functionality.
Defect verification is the part of the Retesting.	Regression testing does not include defect verification
The priority of Retesting is higher than Regression Testing, so it is done before the Regression Testing.	Based on the project type and availability of resources, regression testing can be parallel to Retesting.
Re-Test is a planned Testing.	Regression testing is a generic Testing.
We cannot automate the test-cases for Retesting.	We can do automation for regression testing; manual testing could be expensive and time-consuming.
Re-testing is for failed test-cases.	Regression testing is for passed Test-cases.
Re-testing make sure that the original fault is corrected.	Regression testing checks for unexpected side effect.
Retesting executes defects with the same data and the same environment with different input with a new build.	Regression testing is when there is a modification or changes become mandatory in an existing project.

System testing vs Acceptance testing

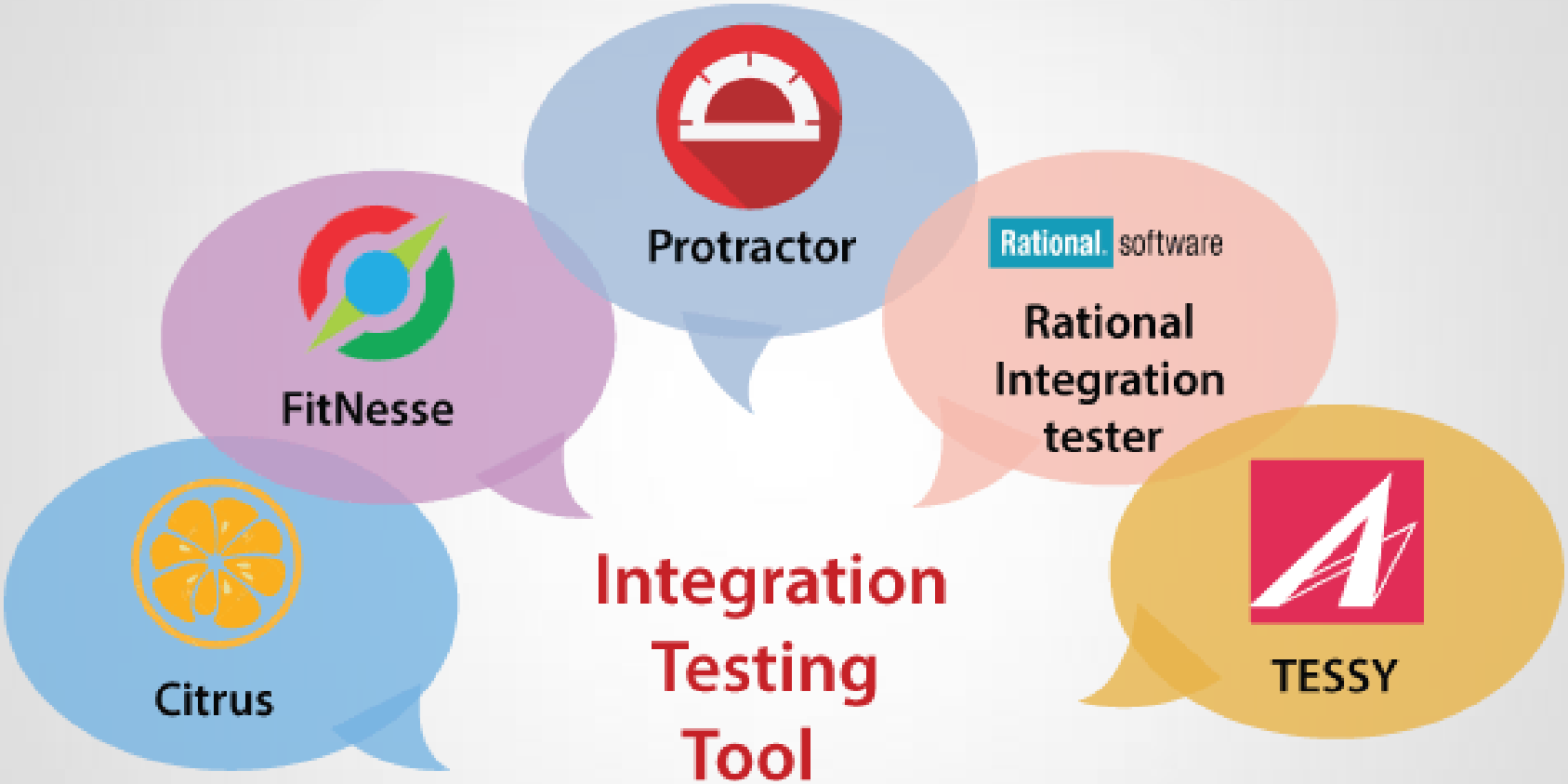
System Testing	Acceptance Testing
System testing is performed to test end to end functionality of the software.	Acceptance testing is performed to test whether the software is conforming specified requirements and user requirements or not.
Only developers and testers can perform System testing.	It can be performed by testers, stakeholders and costumers.
It can be both non-functional and functional testing.	It can be only functional testing.
In System testing, we test the performance of the whole system.	In Acceptance testing, we test whether the system is conforming requirements or not.
System testing uses demo input values that are selected by the testing team.	Acceptance testing uses the actual real-time input values provided by the user.
System Testing is a combination of System part testing and Integration testing.	Acceptance Testing is a combination of alpha testing and beta testing.
It is performed before the Acceptance testing.	It is performed after the System testing.
System testing involves load and stress testing under non-functional testing.	Acceptance testing involves boundary value analysis, equivalence portioning and decision table under functional testing.
The defects found in system testing are considered to be fixed.	The defects found in acceptance testing are considered as product failure.

Testing Tools

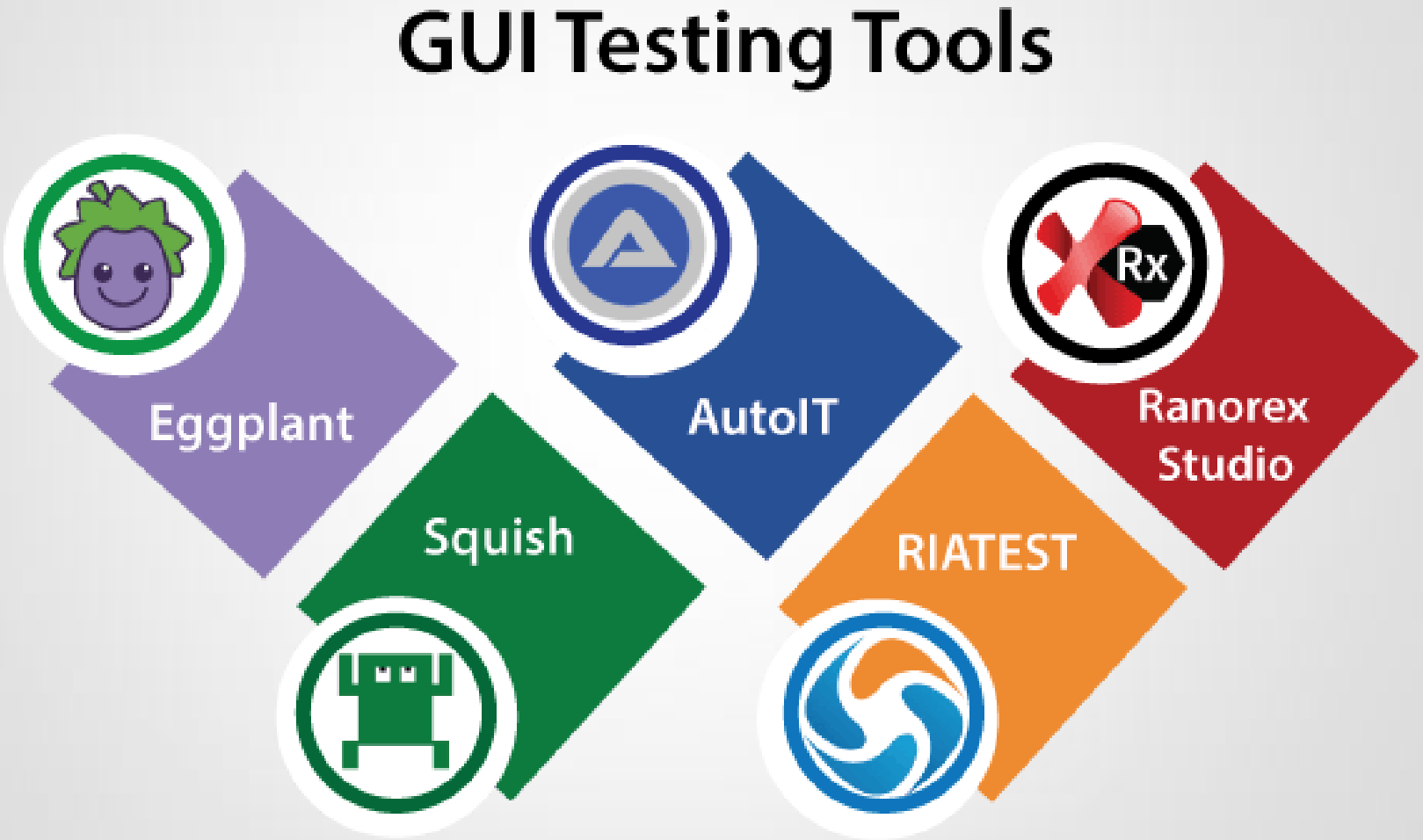
Testing Tools



Testing Tools

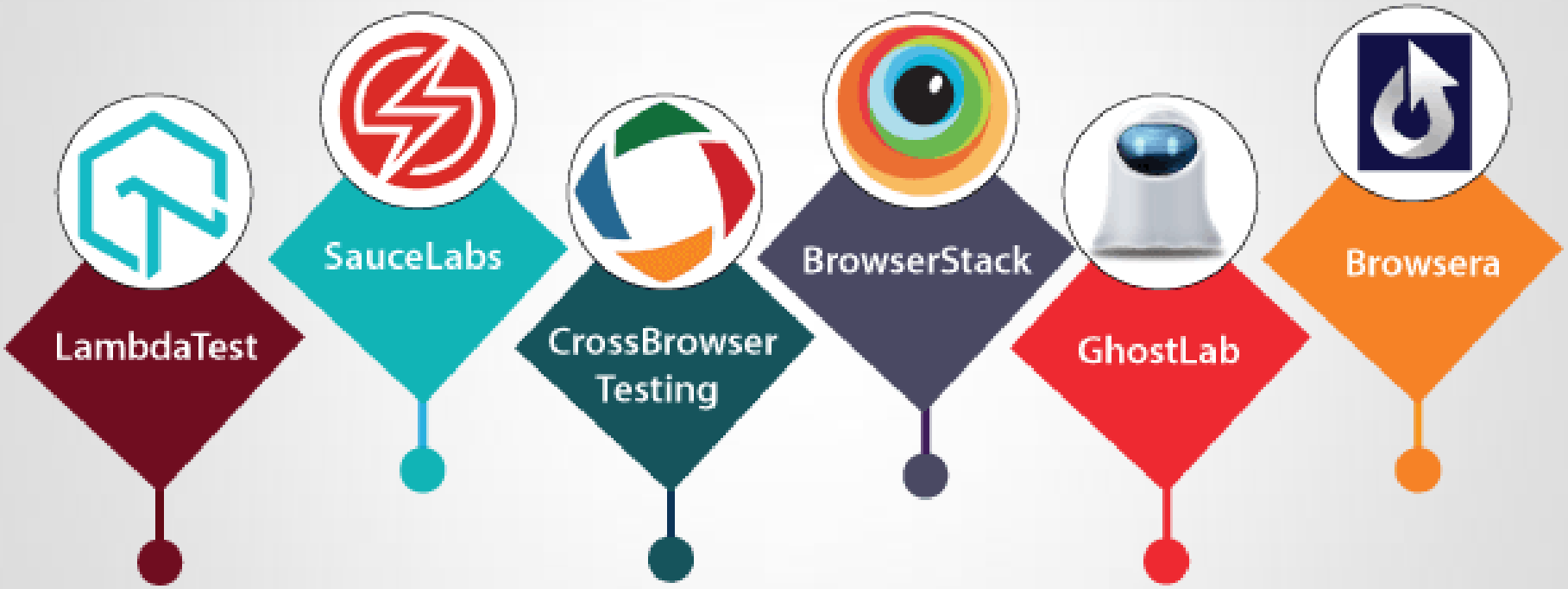


Testing Tools



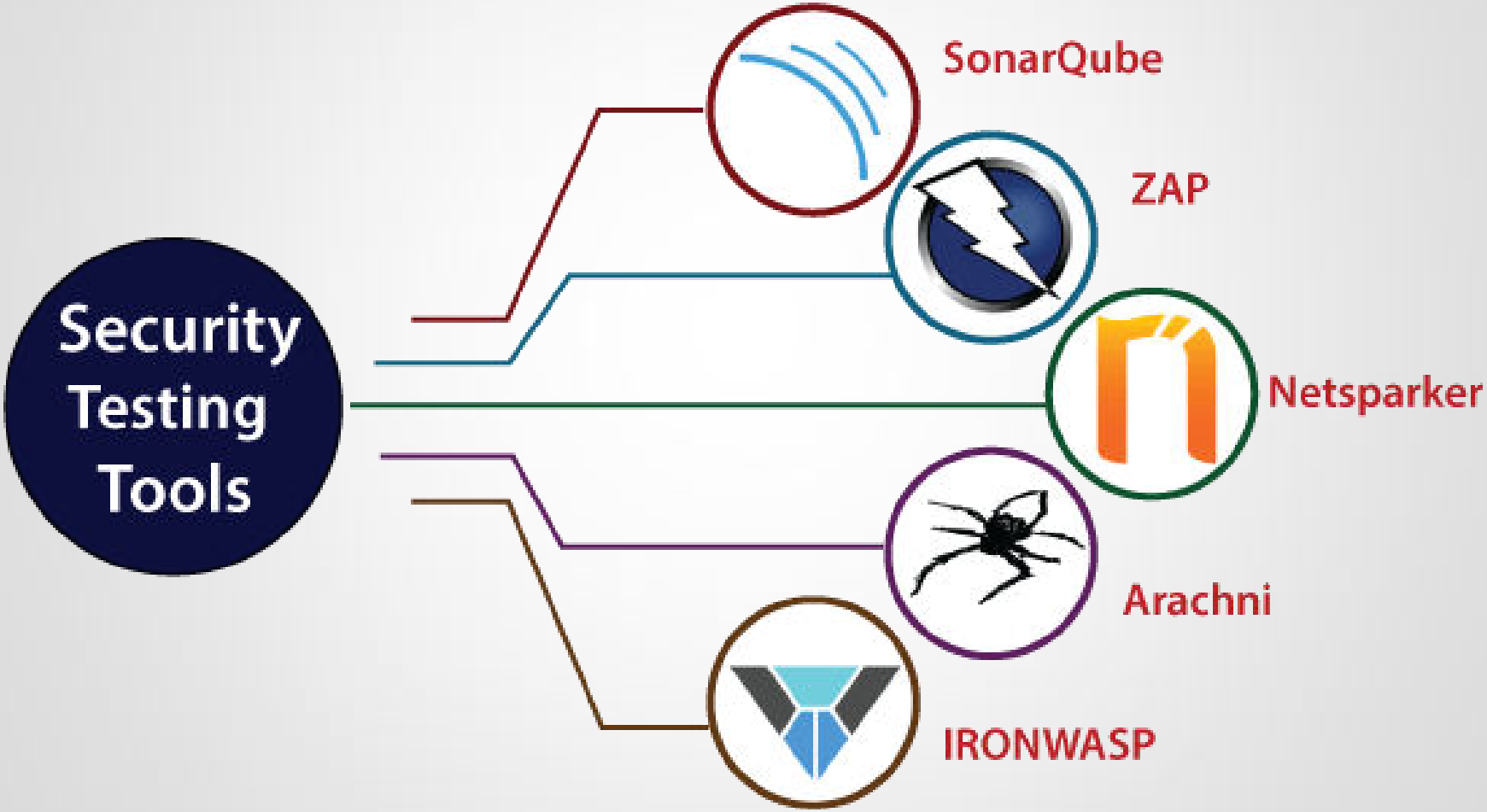
Testing Tools

Testing Tools



Testing Tools

Testing Tools



Testing Tools

Testing Documentation

Testing Documentation

Before Testing

Testing starts with test cases generation. Following documents are needed for reference –

- **SRS document** - Functional Requirements document
- **Test Policy document** - This describes how far testing should take place before releasing the product.
- **Test Strategy document** - This mentions detail aspects of test team, responsibility matrix and rights/responsibility of test manager and test engineer.
- **Traceability Matrix document** - This is SDLC document, which is related to requirement gathering process. As new requirements come, they are added to this matrix. These matrices help testers know the source of requirement. They can be traced forward and backward.

Testing Documentation

While Being Tested

The following documents may be required while testing is started and is being done:

- **Test Case document** - This document contains list of tests required to be conducted. It includes Unit test plan, Integration test plan, System test plan and Acceptance test plan.

- **Test description** - This document is a detailed description of all test cases and procedures to execute them.

- **Test case report** - This document contains test case report as a result of the test.

- **Test logs** - This document contains test logs for every test case report.

Testing Documentation

After Testing

The following documents may be generated after testing :

- **Test summary** - This test summary is collective analysis of all test reports and logs. It summarizes and concludes if the software is ready to be launched. The software is released under version control system if it is ready to launch.

Software Quality

Software Quality

The quality of software can be defined **as the ability of the software to function as per user requirement**. When it comes to software products it must **satisfy all the functionalities** written down in the SRS document.

Software Quality

The modern view of a quality associated with a software product several quality methods such as the following:

Portability: A software device is said to be portable, if it can be freely **made to work in various operating system environments**, in multiple machines, with other software products, etc.

Usability: A software product has better usability if various categories of users can easily invoke the functions of the product.

Reusability: A software product has excellent reusability if different modules of the product can quickly be reused to develop new products.

Correctness: A software product is correct if various requirements as specified in the SRS document have been correctly implemented.

Maintainability: A software product is maintainable if bugs can be easily corrected as and when they show up, new tasks can be easily added to the product, and the functionalities of the product can be easily modified, etc.

Testing vs. QA, QC and Audit

Software quality assurance (QA) -
These are software development process monitoring means, by which it is assured that **all the measures are taken as per the standards of organization**. This monitoring is done to make sure that **proper software development methods were followed.**

The responsibility of quality assurance is not of any specific team, but it is a responsibility of each member of the development team.

- Quality assurance **prevents defects.**
- Quality assurance is **process** oriented.
- Quality assurance is proactive in a process and preventive in nature.
- Quality assurance is a managerial tool.
- Each developer is responsible for quality assurance.

Testing vs. QA, QC and Audit

Software quality control (QC) - This is a system to maintain the quality of software product. It may include **functional and non-functional** aspects of software product, which enhance the goodwill of the organization. This system makes sure that the customer is receiving quality product for their requirement and the product certified as '**fit for use'.**

The responsibility of quality control is of a specific team which is known as a testing team that tests the defects of software by validation and corrective tools.

- Quality Control provides **identification of defects**.
- Quality Control is **product** oriented.
- Quality Control is a corrective tool.
- Testing team is responsible for Quality control.
- Quality Control is a reactive process.

Testing vs. QA, QC and Audit

Points	Quality Assurance (QA)	Quality Control (QC)
Definition	QA is a group of activities which ensures **that the quality of processes which is used during the development of the software always be maintained.**	QC is a group of activities to **detect the defects** in the developed software.
Focus	The focus of QA is **to prevent defects** in the developing software by paying attention to processes.	The focus of QC is to **identify defects** in the developed software by paying attention to testing processes.
How	Establishment of **the high-quality management system and periodic audits** for conformance of the operations of the developing software.	Detecting and eliminating the quality problem elements by **using testing techniques** and tools in the developed software.
What	QA ensures **prevention of quality problem elements** by using systematic activities including documentation.	QC ensures **identification and elimination of defects by using processes** and techniques to achieve and maintain high quality of the software.
Orientation	QA is **process** oriented.	QC is **product** oriented.
Type of process	QA is a proactive process. It concerns to **improve development** so; **defects do not arise** in the testing period.	QC is a **reactive process** because it concerns to **identify defects after the development of product** and before its release.
Responsibility	Each and every member of the **development team** is responsible for QA	Only the **specific testing team** is responsible for QC
Example	**Verification** is the example of QA	**Validation** is the example of QC

Testing vs. QA, QC and Audit

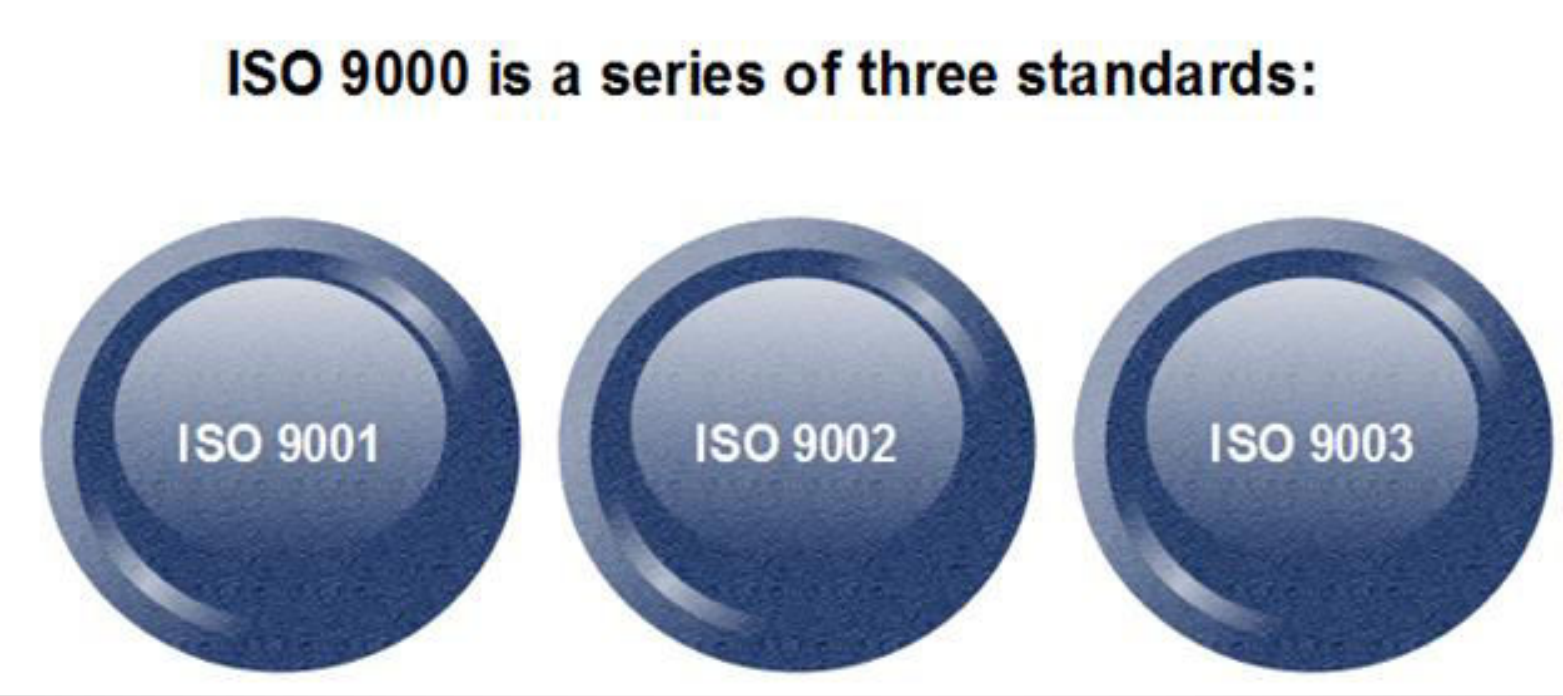
Software audit - This is a **review of procedure** used by the organization to develop the software. A team of auditors, **independent of development team examines the software process, procedure, requirements and other aspects of SDLC**. The purpose of software audit is to check that software and its development process, both conform standards, rules and regulations.

ISO 9000 Certification

ISO (International Standards Organization) is a group or consortium of **63 countries** established to plan and fosters standardization. ISO declared its 9000 **series of standards in 1987**. It serves as a reference for the contract between independent parties. The ISO 9000 standard determines the **guidelines for maintaining a quality system**. The ISO standard mainly addresses operational methods and organizational methods such as responsibilities, reporting, etc. ISO 9000 defines a set of guidelines for the production process and is not directly **concerned about the product itself**.

ISO 9000 Certification

Types of ISO 9000 Quality Standards



ISO 9000 Certification

How to get ISO 9000 Certification?

ISO 9000 Certification

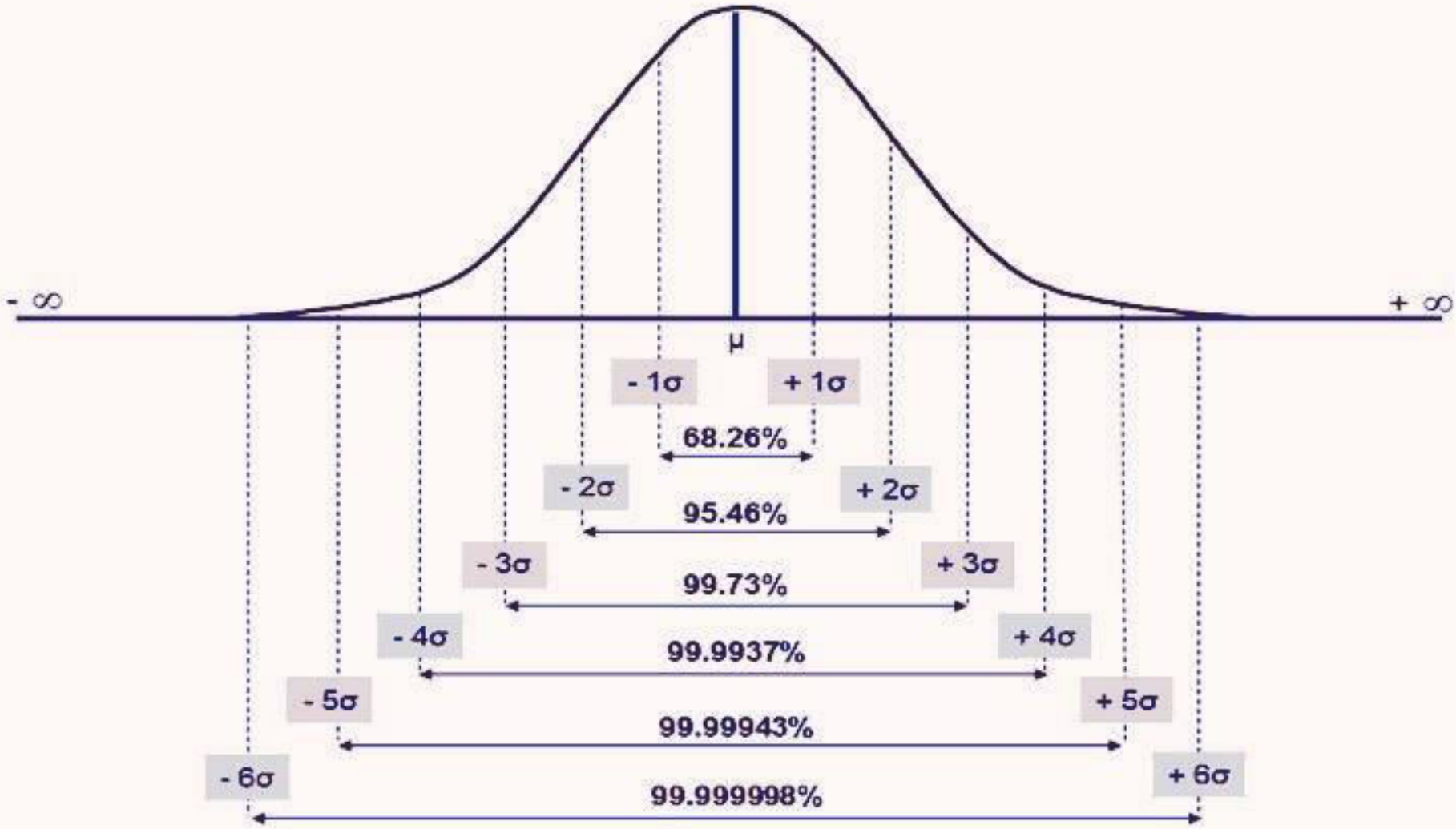
How to get ISO 9000 Certification?

- **Application:** Once an organization decided to go for ISO certification, it applies to the registrar for registration.
- **Pre-Assessment:** During this stage, the registrar makes a rough assessment of the organization.
- **Document review and Adequacy of Audit:** During this stage, the registrar reviews the document submitted by the organization and suggest an improvement.
- **Compliance Audit:** During this stage, the registrar checks whether the organization has compiled the suggestion made by it during the review or not.
- **Registration:** The Registrar awards the ISO certification after the successful completion of all the phases.
- **Continued Inspection:** The registrar continued to monitor the organization time by time.

Six Sigma

Six Sigma is the process of **improving the quality of the output** by **identifying and eliminating the cause of defects and reduce variability** in **manufacturing and business processes**. The maturity of a manufacturing process can be defined by a **sigma rating** indicating its percentage of defect-free products it creates. A six sigma method is one in which **99.99966%** of all the opportunities to produce some features of a component are statistically expected to be free of defects (**3.4 defective features per million opportunities**).

Six Sigma


- ∞
+ ∞
μ
- 1σ
+ 1σ
68.26%
- 2σ
+ 2σ
95.46%
- 3σ
+ 3σ
99.73%
- 4σ
+ 4σ
99.9937%
- 5σ
+ 5σ
99.99943%
- 6σ
+ 6σ
99.999998%

Six Sigma

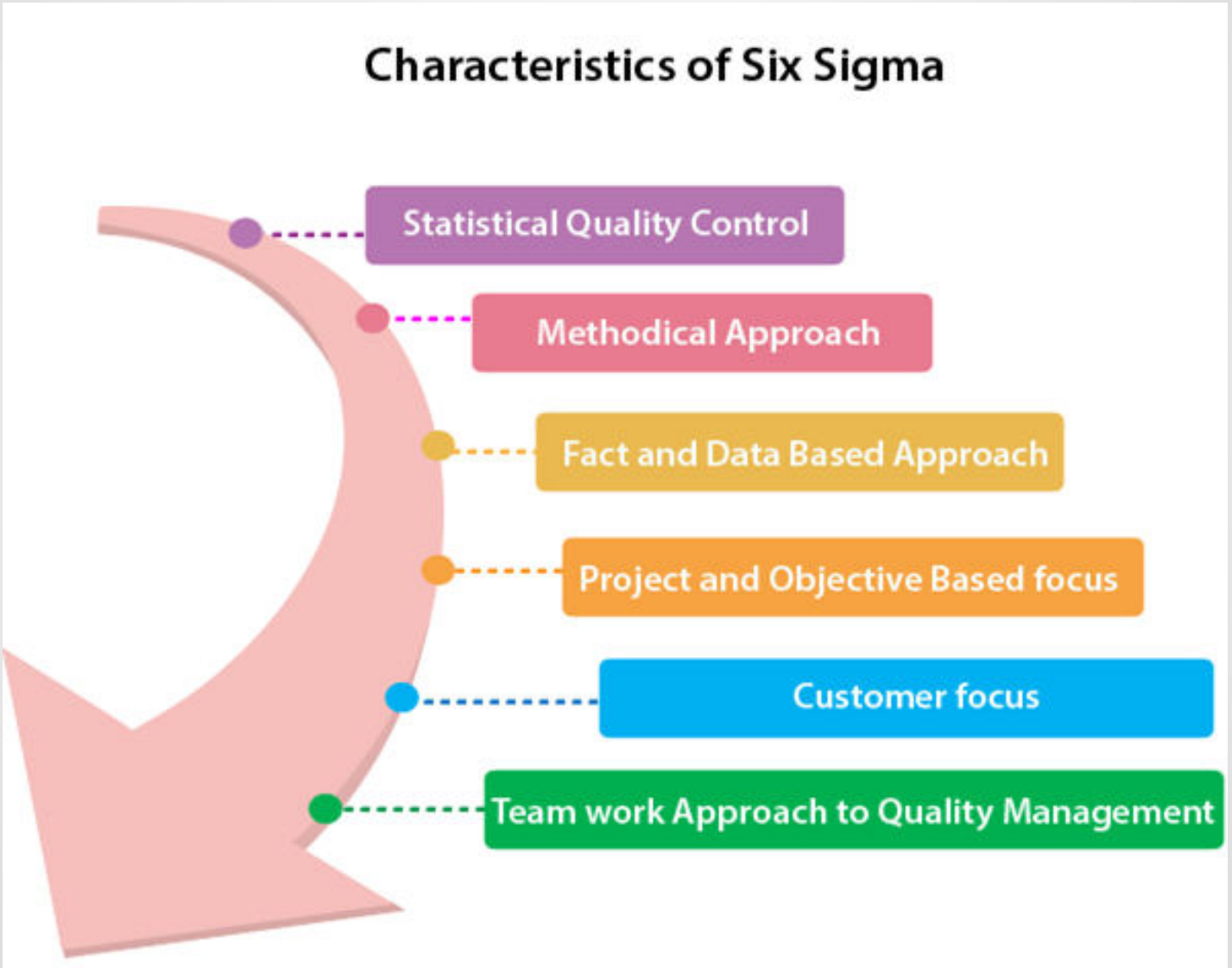
Origin of Six Sigma

- Six Sigma originated at Motorola in the early 1980s, in response to achieving 10X reduction in product-failure levels in 5 years.

- Engineer **Bill Smith** invented Six Sigma, but died of a heart attack in the Motorola cafeteria in 1993, never knowing the scope of the craze and controversy he had touched off.

- Six Sigma is based on various quality management theories (e.g. Deming's 14 point for management, Juran's 10 steps on achieving quality).

Characteristics of Six Sigma



Six Sigma

Characteristics of Six Sigma

- **Statistical Quality Control:** Six Sigma is derived from the Greek Letter σ (Sigma) from the Greek alphabet, which is used to denote Standard Deviation in statistics. Standard Deviation is used to measure variance, which is an essential tool for measuring non-conformance as far as the quality of output is concerned.

- **Methodical Approach:** The Six Sigma is not a merely quality improvement strategy in theory, as it features a well defined systematic approach of application in DMAIC and DMADV which can be used to improve the quality of production. DMAIC is an acronym for Design-Measure- Analyze-Improve-Control. The alternative method DMADV stands for Design-Measure- Analyze-Design-Verify.

- **Fact and Data-Based Approach:** The statistical and methodical aspect of Six Sigma shows the scientific basis of the technique. This accentuates essential elements of the Six Sigma that is a fact and data-based.

Characteristics of Six Sigma

- **Project and Objective-Based Focus:** The Six Sigma process is implemented for an organization's project tailored to its specification and requirements. The process is flexed to suits the requirements and conditions in which the projects are operating to get the best results.

- **Customer Focus:** The customer focus is fundamental to the Six Sigma approach. The quality improvement and control standards are based on specific customer requirements.

- **Teamwork Approach to Quality Management:** The Six Sigma process requires organizations to get organized when it comes to controlling and improving quality. Six Sigma involving a lot of training depending on the role of an individual in the Quality Management team.

Benefits of Six Sigma

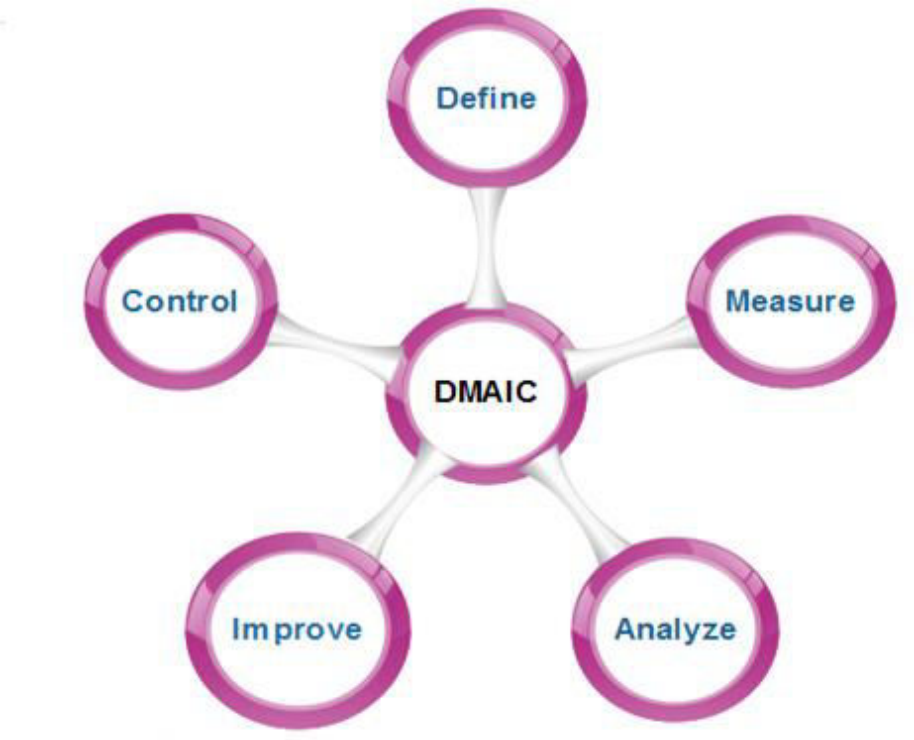
Six Sigma offers six major benefits that attract companies −

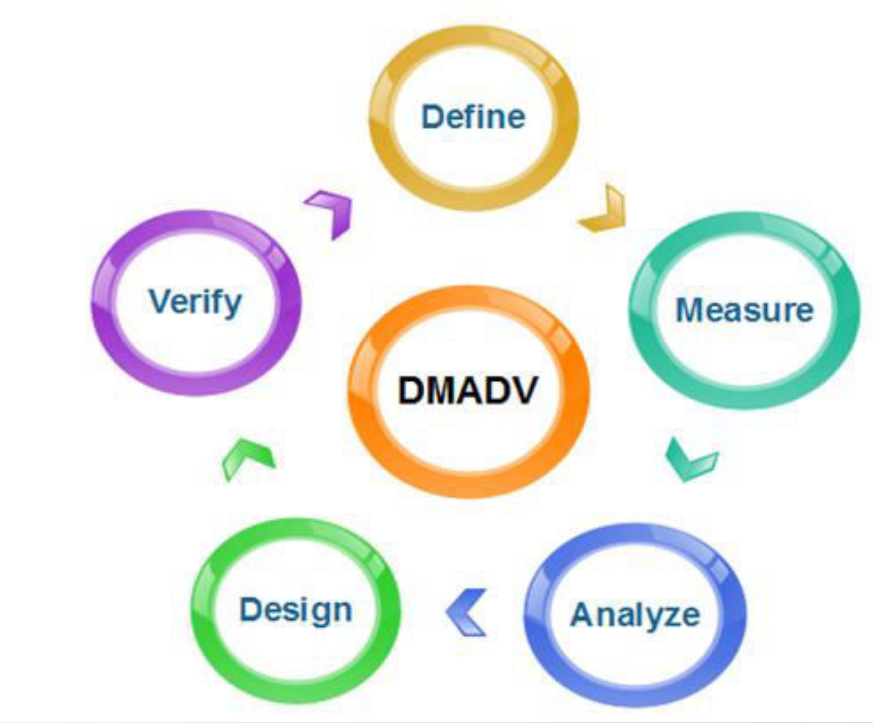
- Generates sustained success
- Sets a performance goal for everyone
- Enhances value to customers
- Accelerates the rate of improvement
- Promotes learning and cross-pollination
- Executes strategic change

Six Sigma Methodologies

Six Sigma projects follow two project methodologies:

1. **DMAIC:** is used to enhance an existing business process
2. **DMADV:** is used to create new product designs or process designs





Software Maintenance

Outlines

- Software Maintenance
- Need for Maintenance
- Types of Software Maintenance
- Causes of Software Maintenance Problems
- Software Maintenance Cost Factors
- Software Re-engineering

Software Maintenance

Software Maintenance is the process of **modifying a software product after it has been delivered** to the customer.

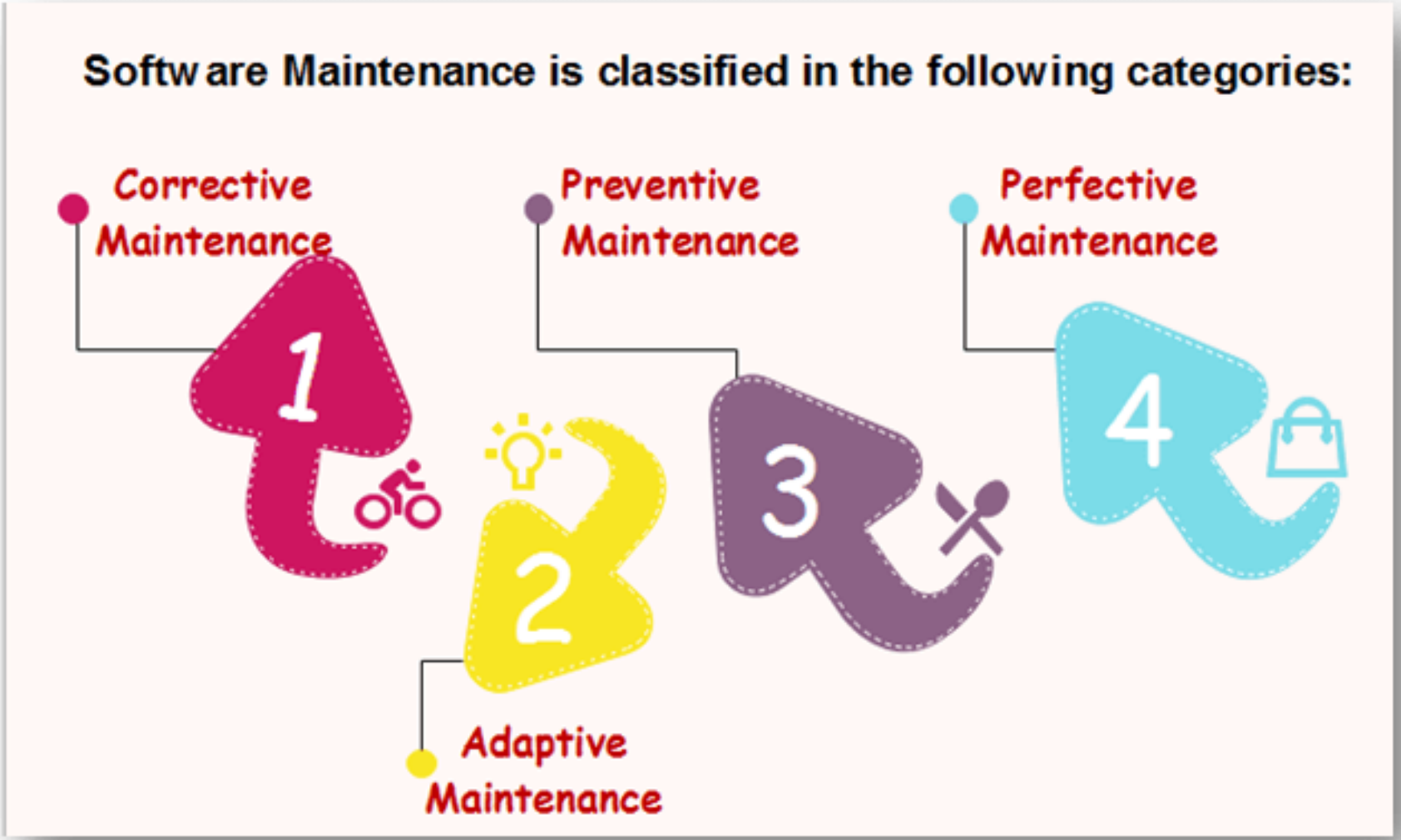
The main purpose of software maintenance is to **modify and update software application** after delivery to correct faults and to improve performance.

Need for Maintenance

Software Maintenance is required to ensure that the system continues to satisfy user requirements:-

- Correct errors
- Change in user requirement with time
- Changing hardware/software requirements
- To improve system efficiency
- To optimize the code to run faster
- To modify the components
- To reduce any unwanted side effects.

Types of Software Maintenance



Types of Software Maintenance

Maintenance can be divided into the following:

1.Corrective maintenance:

Corrective maintenance of a software product may be essential either to **rectify some bugs observed** while the system is in use, or to enhance the performance of the system.

2.Adaptive maintenance:

This includes modifications and updates when the customers need the product to run on **new platforms, on new operating systems**, or when they need the product to interface with new hardware and software.

3.Perfective maintenance:

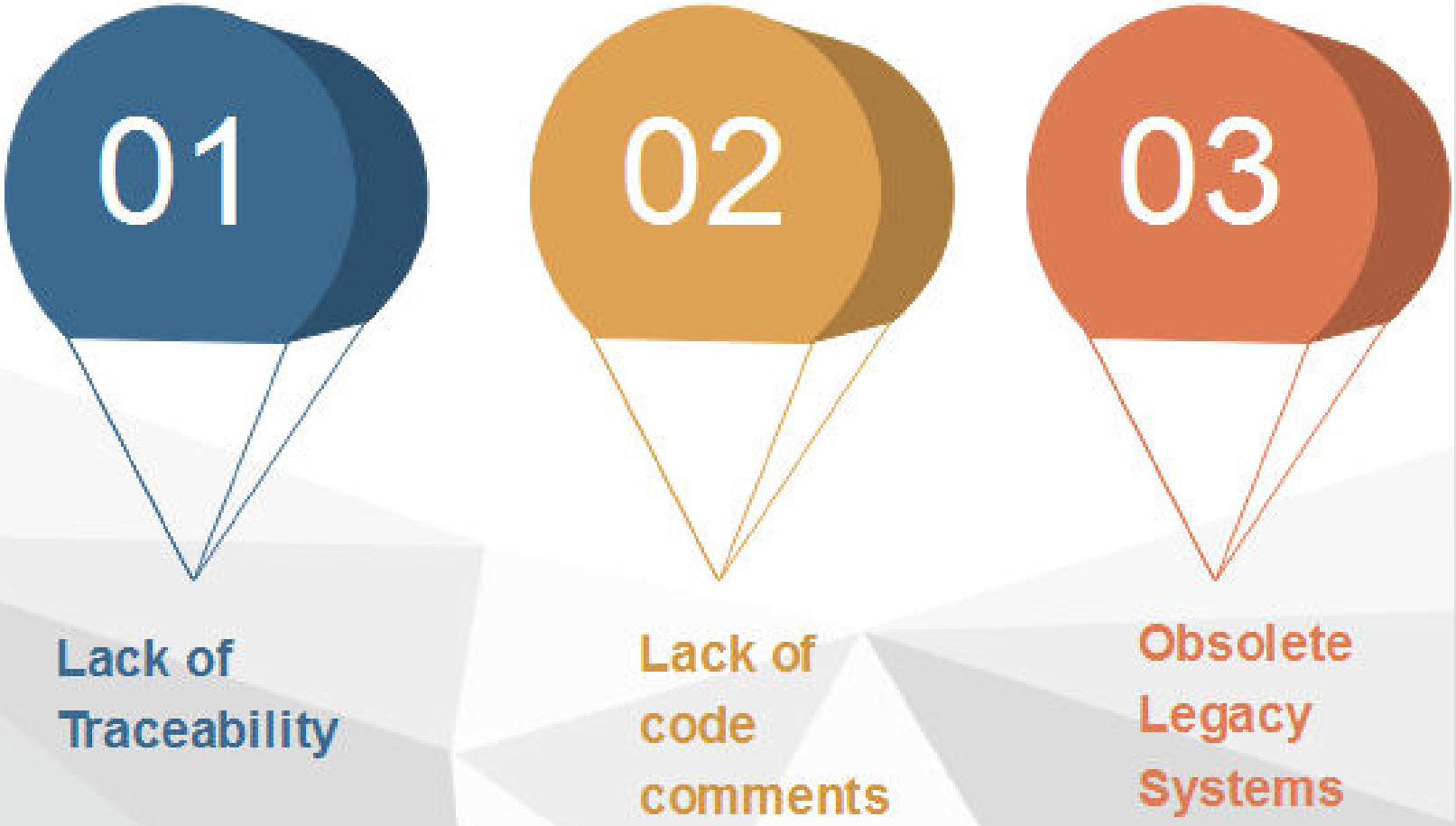
A software product needs maintenance to **support the new features** that the users want or to change different types of functionalities of the system according to the customer demands.

4.Preventive maintenance:

This type of maintenance includes modifications and updates **to prevent future problems** of the software. It goals to attend problems, **which are not significant at this moment but may cause serious issues in future.**

Causes of Software Maintenance Problems

Following are the major causes of software maintenance problems:



Causes of Software Maintenance Problems

Lack of Traceability

- Codes are rarely traceable to the requirements and design specifications.
- It makes it **very difficult for a programmer to detect and correct a critical defect affecting customer operations**.
- Like a detective, the programmer pores over the program looking for clues.
- Life Cycle documents are not always produced even as part of a development project.

Lack of Code Comments

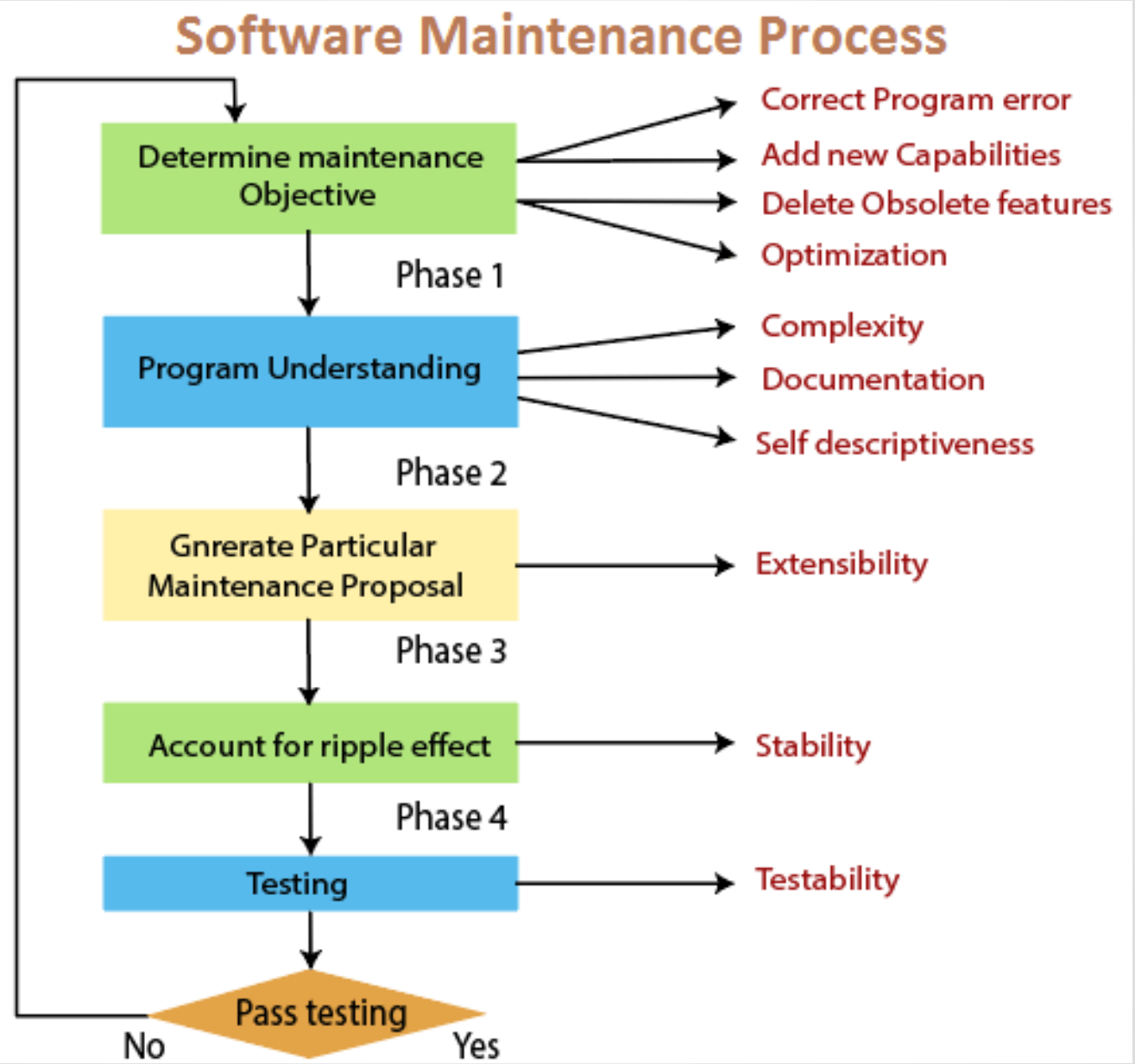
- Most of the software system codes lack adequate comments. Lesser comments may not be helpful in certain situations.

Causes of Software Maintenance Problems

Obsolete Legacy Systems

- In most of the countries worldwide, the legacy system that provides the backbone of the nation's critical industries, e.g., telecommunications, medical, transportation utility services, were not designed with maintenance in mind.
- They were not expected to last for a quarter of a century or more.
- As a consequence, the code supporting these systems is devoid of traceability to the requirements, compliance to design and programming standards and often includes dead, extra and uncommented code, which all make the maintenance task next to the impossible.

Causes of Software Maintenance Problems



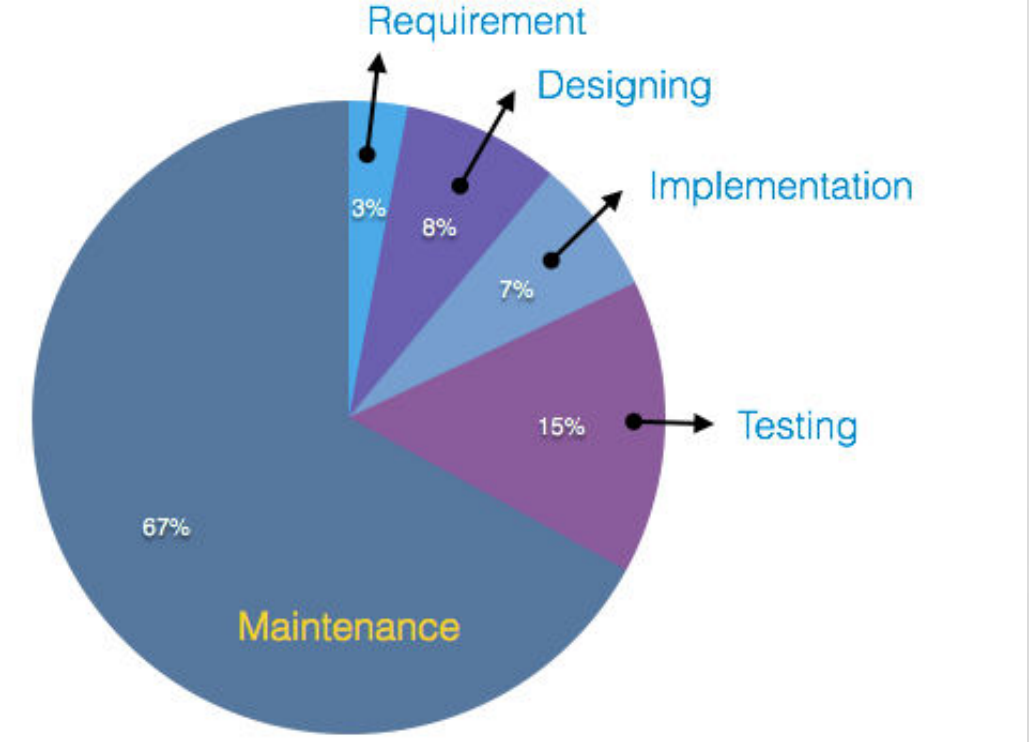
Software Maintenance Cost Factors

A study on estimating software maintenance found that the cost of maintenance is as high as **67% of the cost of entire software process cycle**.

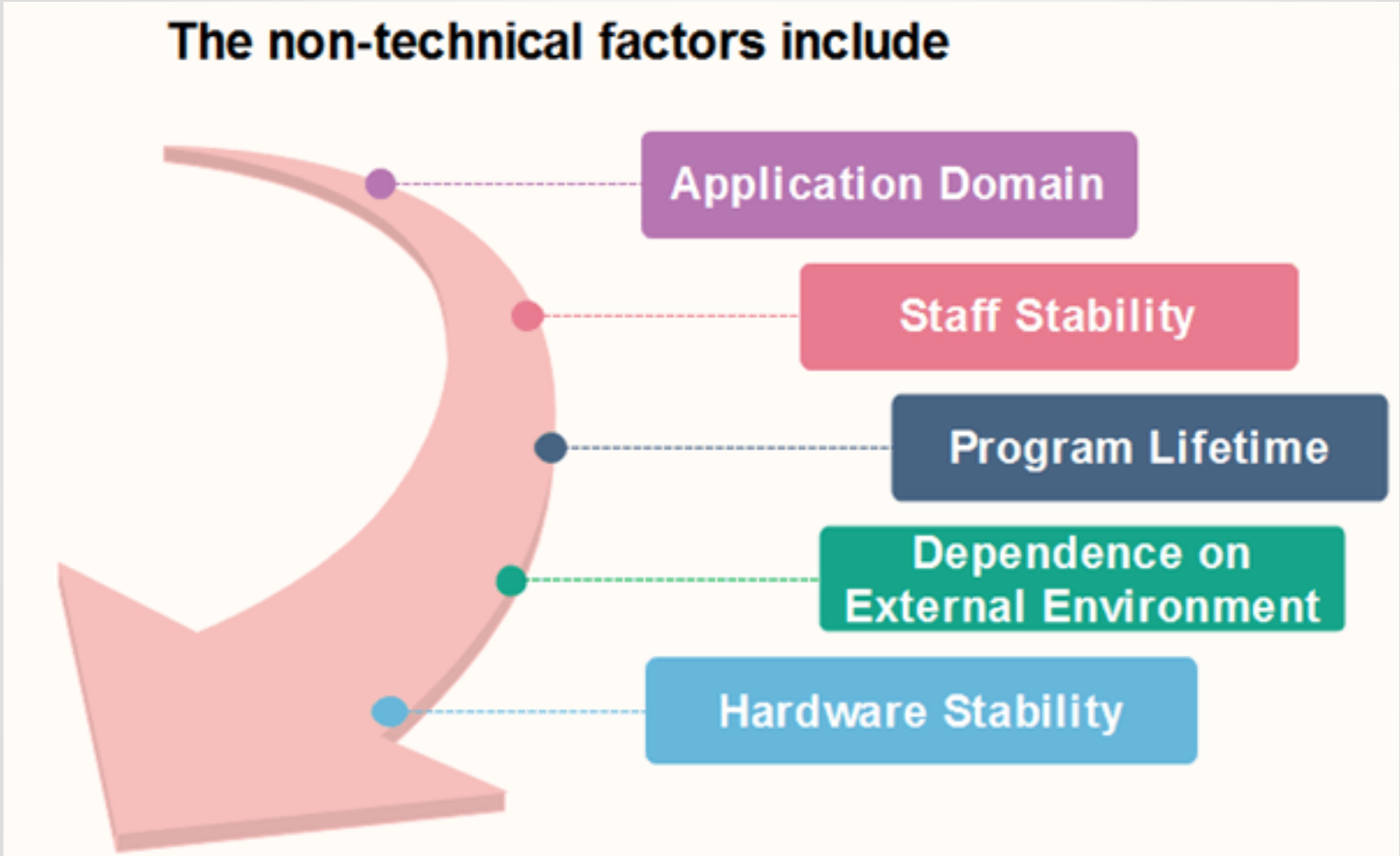
There are two types of cost factors involved in software maintenance.

These are

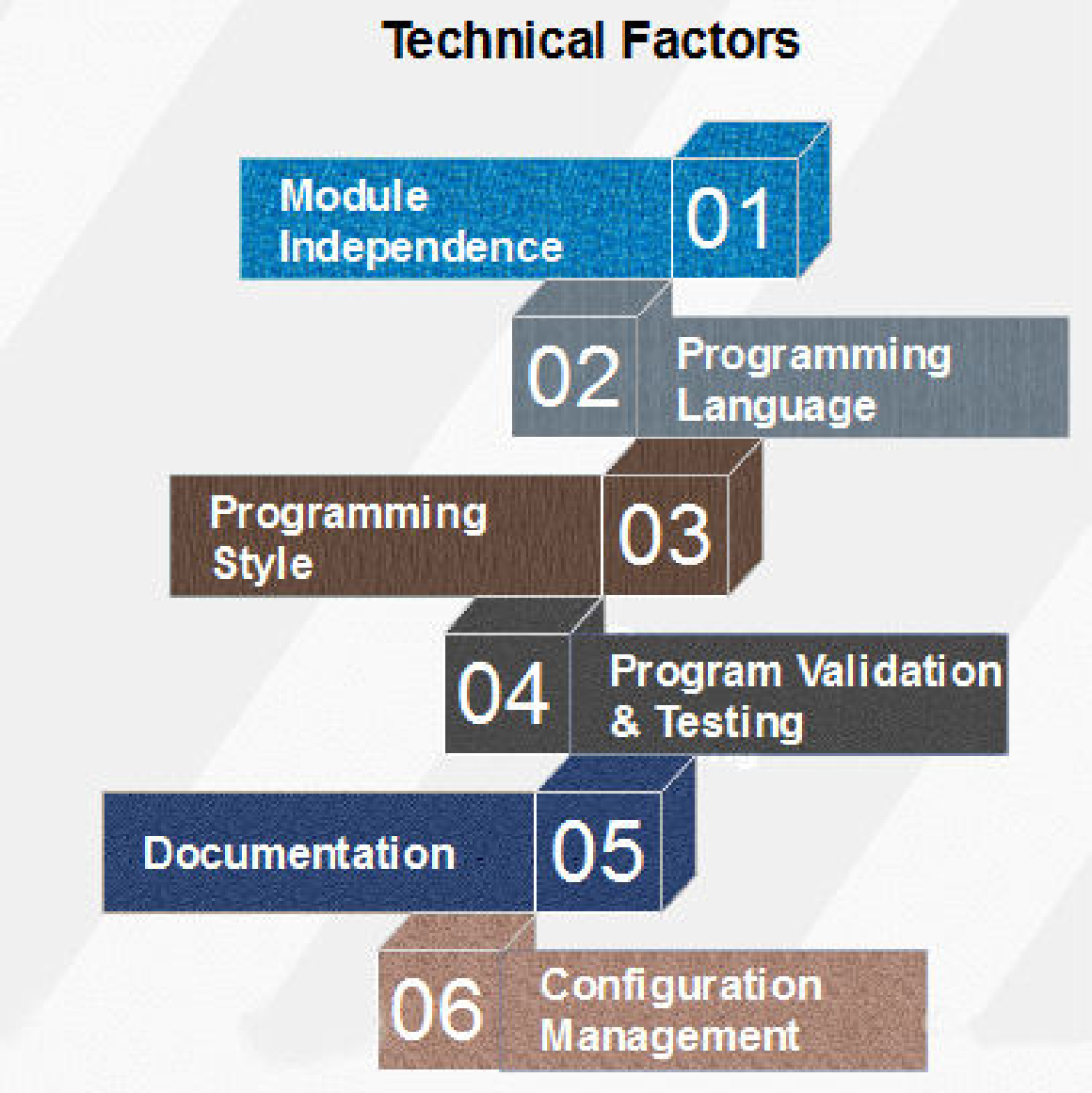
- Non-Technical Factors
- Technical Factors



Software Maintenance Cost Factors

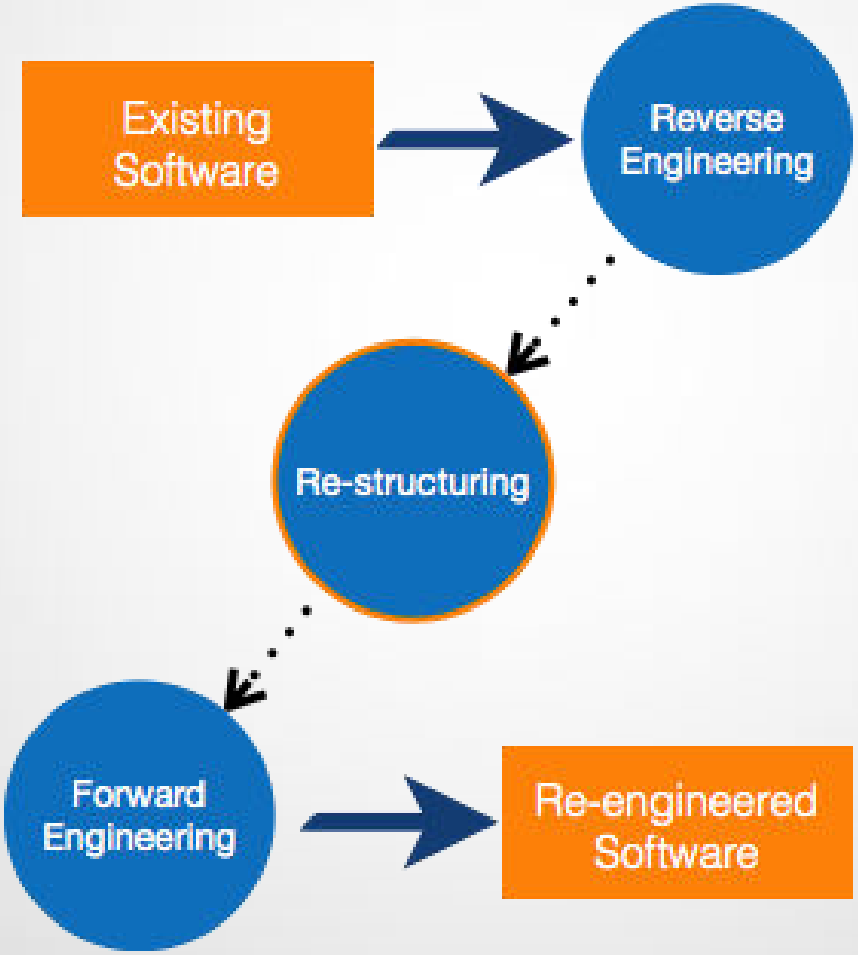


Software Maintenance Cost Factors



Software Re-engineering

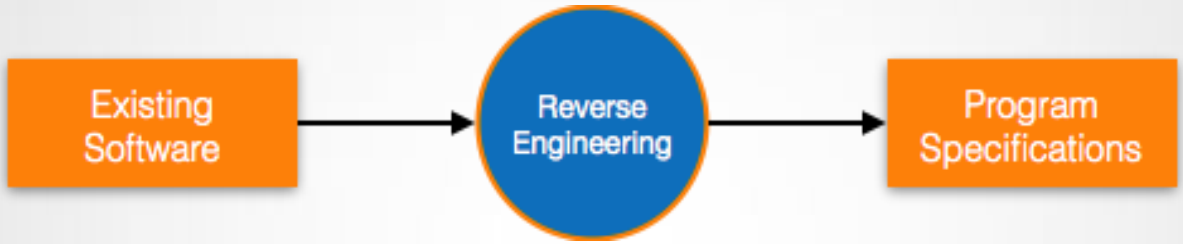
When we need to update the software to keep it to the current market, without impacting its functionality, it is called software re-engineering. It is a thorough process where the design of software is changed and programs are re-written.



Re-Engineering Process

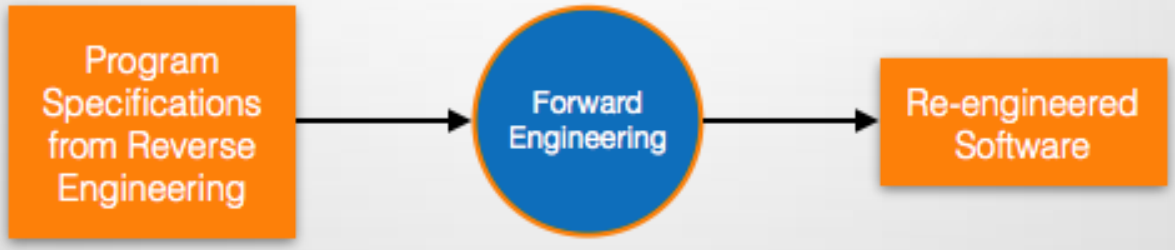
Reverse Engineering

Reverse Engineering is processes of extracting knowledge or design information from anything man-made and reproducing it based on extracted information. It is also called back Engineering.



Forward Engineering

Forward engineering is a process of obtaining desired software from the specifications in hand which were brought down by means of reverse engineering. It assumes that there was some software engineering already done in the past.



Re-Engineering Process

Why Reverse Engineering?

- Providing proper system documentation.
- Recovery of lost information.
- Assisting with maintenance.
- Facility of software reuse.
- Discovering unexpected flaws or faults

Used of Software Reverse Engineering –

- Software Reverse Engineering is used in software design, reverse engineering enables the developer or programmer to add new features to the existing software with or without knowing the source code.

- Reverse engineering is also useful in software testing, it helps the testers to study the virus and other malware code .