Unit 6

Software Testing & Quality Assurance

What is Testing?

Many people understand many definitions of testing:

1. **Testing is the process of demonstrating that errors are not present.**

2. **The purpose of testing is to show that a program performs its intended functions correctly.**

3. **Testing is the process of establishing confidence that a program does what it is supposed to do.**

These definitions are incorrect

Software Testing

A more appropriate definition is:

"Testing is the process of executing a program with the intent of finding errors".

Software Testing

- Once source code has been generated, software must be tested to allow errors to be identified and removed before delivery to the customer.
- Testing verifies that the system meets the different requirements including, functional, performance, reliability, security, usability and so on. This verification is done to ensure that we are building the system right.
- In addition, testing validates that the system being developed is what the user needs. In essence, validation is performed to ensure that we are building the right system.
- Software testing contributes to improving the quality of the product.
- It is important for software testing to verify and validate that the product meets the stated requirements / specifications.

The followings are some commonly used terms associated with testing:

A **failure** is a manifestation of an error (or defect or bug). But , the mere presence of an error may not necessarily lead to a failure.

A **fault** is an incorrect intermediate state that may have been entered during program execution, e.g., a variable value is different from what it should be. A fault may or may not lead to a failure.

A **test data** is the inputs which have been devised to test the system.

A **test case** is the triplet [I,S,O], where I is the data input to the system, S is the state of the system at which data is input, and O is the expected output of the system or Inputs to test the system and the predicted outputs from these inputs if the system operates according to its specification

Software Testing Objectives

- Testing is the process of executing a program with the intent of finding errors.
- A good test case is one with a high probability of finding an as-yet undiscovered error.
- A successful test is one that discovers an as-yet-undiscovered error.

Software Testing Principles

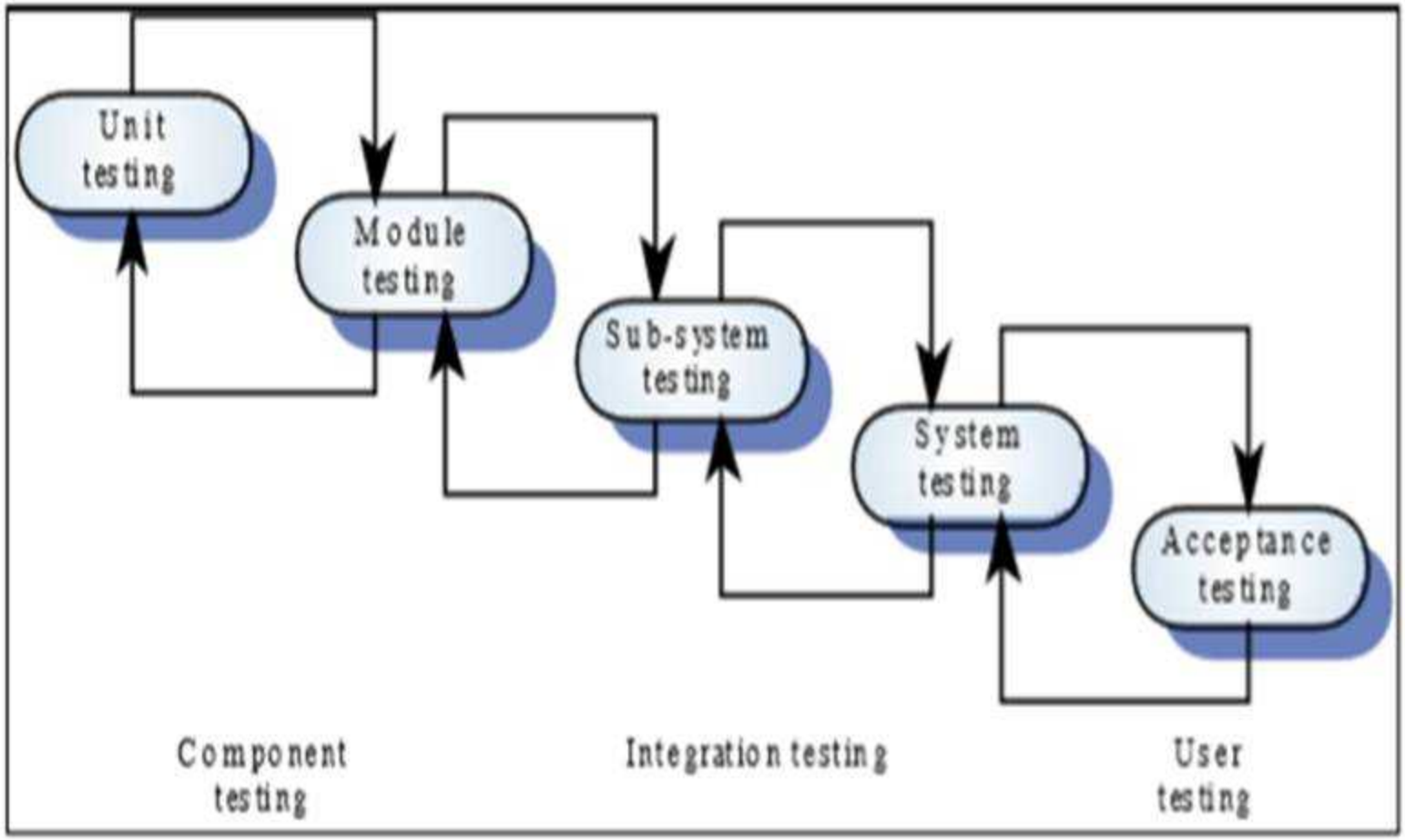
- All tests should be traceable to customer requirements.
- Tests should be planned long before testing begins.
- The Pareto principle (80% of all errors will likely be found in 20% of the code) applies to software testing.
- Testing should begin in the small and progress to the large.
- Exhaustive testing is not possible.
- To be most effective, testing should be conducted by an independent third party.

Software Testability Checklist

- **Operability-** the better it works the more efficiently it can be tested
- **Observability** - what you see is what you test
- **Controllability-** the better software can be controlled the more testing can be automated and optimized
- **Decomposability-** by controlling the scope of testing, the more quickly problems can be isolated and retested intelligently
- **Simplicity-** the less there is to test, the more quickly we can test
- **Stability-** the fewer the changes, the fewer the disruptions to testing
- **Understandability-**the more information known, the smarter the testing

Software Testing Process

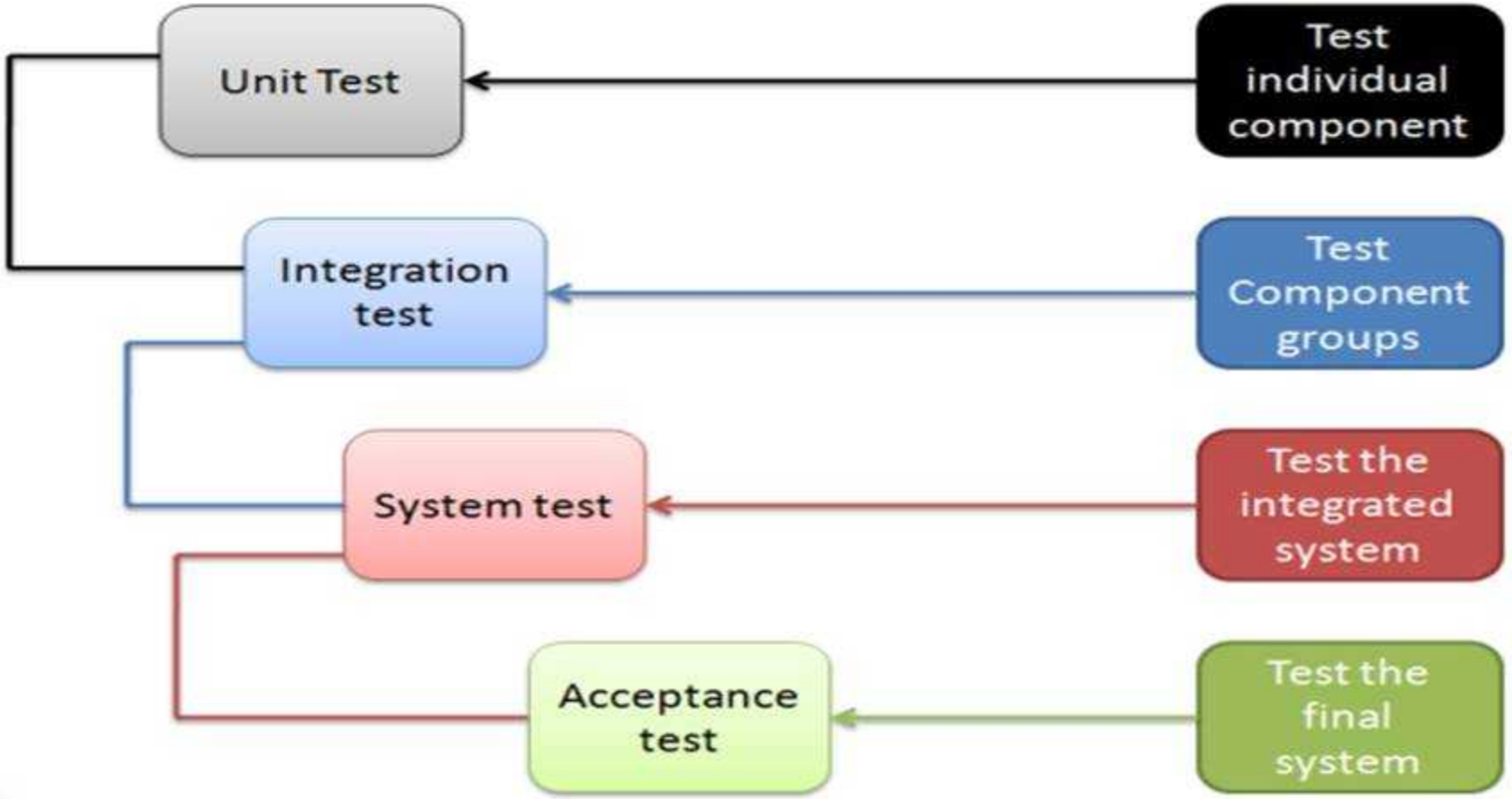
Software Testing Process



Software Testing Process [CTD...]

- **Unit testing** - Individual components are tested independently, without other system components
- **Module testing** - Related collections of dependent components(class, procedures & functions) are tested, without other system module.
- **Sub-system testing-** Modules are integrated into sub-systems and tested. The focus here should be on interface testing to detect module interface errors or mismatches.
- **System testing** - Testing of the system as a whole. Validating functional and non-functional requirements & Testing of emergent system properties.
- **Acceptance testing**-Testing with customer data to check that it is acceptable.
- Making sure the software works correctly for intended user in his or her normal work environment.
    - **Alpha test**-version of the complete software is tested by customer/developer under the supervision of the developer at the developer's site.
    - **Beta test**-version of the complete software is tested by customer at his or her own site without the developer being present

SOFTWARE TESTING LEVELS are the different stages of the software development lifecycle where testing is conducted. There are four levels of software testing:



Overview of Software Testing Levels

Level of Testing [CTD...]

1. **Unit Testing:** A level of the software testing process where individual units of a software are tested. The purpose is to validate that each unit of the software performs as designed.
2. **Integration Testing:** After unit testing is integration testing. In this stage, all units are integrated together and tested. It is a form of testing in the testing process performed to detect defects in the interactions and the interfaces between the integrated units.
3. **System Testing:** A level of the software testing process where a complete, integrated system is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. After integration testing, the fully integrated application is tested to check that whether the system meets its software requirements specifications (SRS).

There are various types of system testing:

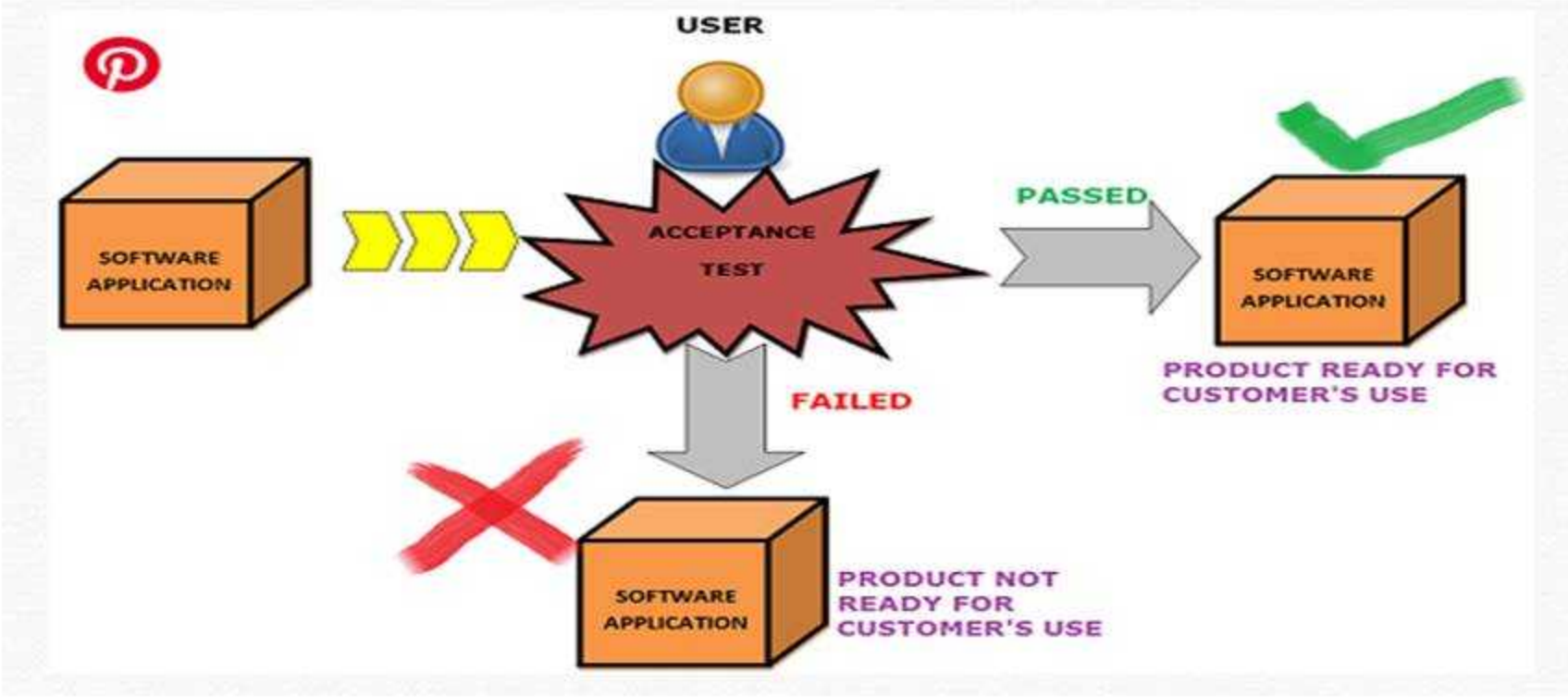
- **Usability Testing:** Focuses on the user's ease to operate and use and checks that the user interface is user-friendly.

CTD...

- **Regression Testing:** Type of software testing that checks the performance of existing functionalities when a change is made.
- **Functional Testing:** Type of black box testing that bases its test cases on the specifications of the software component under test.
- **Recovery Testing:** to test that how well a system recovers from crashes, hardware failures, or other catastrophic problems.
- **Security Testing:** To test that the system is secured enough to protect it from unintended users.
- **Performance Testing:** Performance testing is a type of testing that is performed to determine how fast and stable the system performs under a particular workload.
- **Load Testing:** Load testing is a kind of testing which determines the performance of the system under real-life load conditions. it measures the maximum load capacity of the system when multiple users access the system at the same time

CTD...

4. **Acceptance Testing:** A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery



CTD...

Broadly, there are two main types of Acceptance Testing

Alpha Testing:

- It is an on-site acceptance testing executed at developer's site usually by the group of skilled testers. This is carried out at the end of the software development process and before the handing over of the software product to clients/users. The feedback or the collected result from the alpha testing is significantly used to fix bugs or defects, and it is feasible to improve the usability of the product.

Beta Testing:

- Also known by the name field testing, Beta Testing is an off-site acceptance testing, carried out at client's site by the stakeholder or the end-users. This testing involves the evaluation of the software in the real environment by its potential users before its actual release in the market.

Integration Testing Types or Approaches

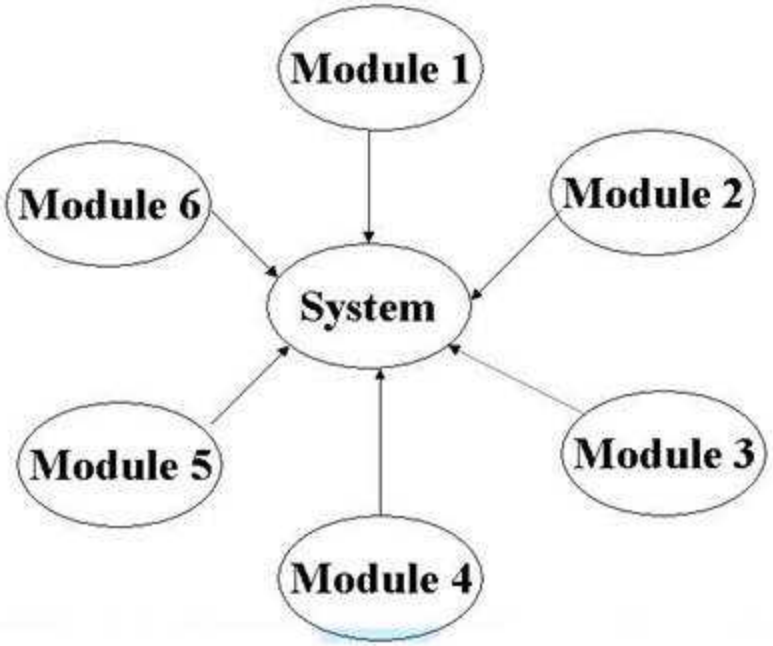
- There are many different types or approaches to integration testing. The most popular and frequently used approaches are Big Bang Integration Testing, Top Down Integration Testing, Bottom Up Integration testing and Incremental integration testing. The choice of the approach depends on various factors like cost, complexity, criticality of the application etc.

- There are many lesser known types of integration testing like distributed services integration, sandwich integration testing, backbone integration, high frequency integration, layer integration etc.
- Commonly used integration testing are described below [**Next Slide**]

Integration Testing Types or Approaches [Ctd...]

❖ Big Bang Integration Testing

- In **Big Bang Integration Testing** all components or modules are integrated simultaneously, after which everything is tested as a whole. As per the below image all the modules from 'Module 1' to 'Module 6' are integrated simultaneously then the testing is carried out



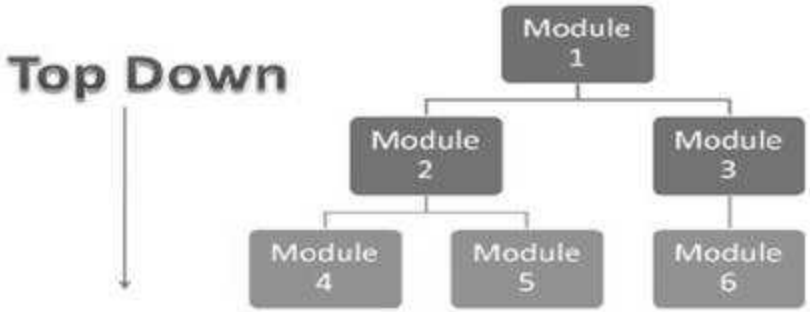
Advantage: Big Bang testing has the advantage that everything is finished before integration testing starts.

Disadvantage: The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration

Integration Testing Types or Approaches [Ctd...]

❖Top-down Integration Testing

- Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu). Components or systems are substituted by stubs. Below is the diagram of 'Top down Approach':



Advantages of Top-Down approach:

- The tested product is very consistent because the integration testing is basically performed in an environment that almost similar to that of reality

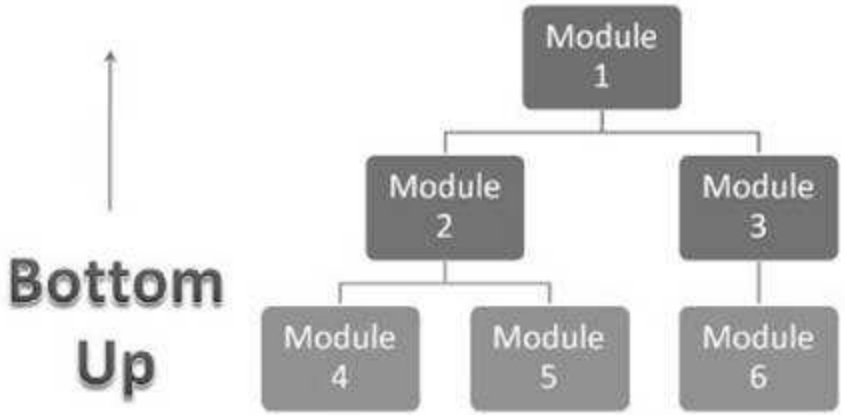
Disadvantages of Top-Down approach:

- Basic functionality is tested at the end of cycle

Integration Testing Types or Approaches [Ctd...]

❖ Bottom up Integration Testing

- Testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers. Below is the image of 'Bottom up approach':



Advantage of Bottom-Up approach:

- In this approach development and testing can be done together so that the product or application will be efficient and as per the customer specifications.

Disadvantages of Bottom-Up approach:

- We can catch the Key interface defects at the end of cycle
- It is required to create the test drivers for modules at all levels except the top control

Integration Testing Types or Approaches [Ctd...]

❖Incremental Integration Testing

- Another approach is that all programmers are integrated one by one, and a test is carried out after each step.
- The incremental integration testing approach has the advantage that the defects are found early in a smaller assembly when it is relatively easy to detect the cause.
- A disadvantage is that it can be time-consuming since stubs and drivers have to be developed and used in the test.
- Within incremental integration testing a range of possibilities exist, partly depending on the system architecture.

Integration Testing Types or Approaches [Ctd...]

❖Sandwich Integration Testing

- Sandwich integration testing is a combination of both top down and bottom up approaches. It is also called as hybrid integration testing or mixed integration testing.

- In sandwich integration testing, the system is considered to be made up of three layers.

- A layer in the middle which will be the target of testing
- A layer above the target layer and a layer below the target layer
- Testing begins from the outer layer and converges at the middle layer
- Advantage: The benefit of this approach is that top and bottom layers can be tested in parallel
- Disadvantage: Extensive testing of the sub-systems is not performed before the integration

Integration Testing Types or Approaches [Ctd...]

❖Functional Incremental Testing

- Integration and testing takes place on the basis of the functions and functionalities, as documented in the functional specification.

Steps – How to do Integration Testing

- Choose the module or component to be tested based on the strategy or approach
- Unit testing of the component is to be completed for all the features
- Deploy the chosen modules or components together and make initial fixes that are required to get the integration testing running
- Perform functional testing and test all the use cases for the components chosen
- Perform structural testing and test the components chosen
- Record the results of the above testing activities
- Repeat the above steps until the complete system is fully tested

Difference Between – Integration Testing and/vs Unit Testing

Unit Testing	Integration Testing
Unit testing is done to confirm if the unit of code works as expected	Integration testing is done to confirm if the different modules work as expected, when integrated together
In unit testing, the unit is not dependent on anything outside the unit being tested	In Integration testing, the components may have inter-dependency on each other or external systems
Unit testing is done by developer	Integration testing is done by the testing team
In the Software Testing Life Cycle (STLC), Unit testing is the first test to be executed	Integration testing is usually done before system testing and it comes after unit testing.

Smoke Testing & Sanity Testing

Smoke Testing Vs Sanity Testing - Key Differences

Smoke Testing	Sanity Testing
Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine	Sanity Testing is done to check the new functionality/bugs have been fixed
The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing	The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing
This testing is performed by the developers or testers	Sanity testing is usually performed by testers
Smoke testing is usually documented or scripted	Sanity testing is usually not documented and is unscripted
Smoke testing is a subset of Acceptance testing	Sanity testing is a subset of Regression Testing
Smoke testing exercises the entire system from end to end	Sanity testing exercises only the particular component of the entire system
Smoke testing is like General Health Check Up	Sanity Testing is like specialized health check up

Types of Functional & Non-functional Testing

Functional testing types include:

- Unit testing
- Integration testing
- System testing
- Sanity testing
- Smoke testing
- Interface testing
- Regression testing
- Beta/Acceptance testing

Non-functional testing types include:

- Performance Testing
- Load testing
- Stress testing
- Volume testing
- Security testing
- Compatibility testing
- Install testing
- Recovery testing
- Reliability testing
- Usability testing
- Compliance testing
- Localization testing

Test Case Design Strategies

1. Black-box or behavioral testing
2. White-box or glass-box testing

Black Box testing

- Also known as Behavioral or Functional testing.
- The system is a " Black-box" whose behavior is determined by studying its input and related the outputs.
- It is techniques of knowing the specified function a product is to perform and demonstrating correct operation based solely on its specification without regard for its internal logic.
- Focus on the functional requirements of the software i.e., information domain not the implementation part of the software.
- Different categories of errors include incorrect or missing functions, interface error, errors in data structures or external database access, behavior or performance errors, and initiation and termination errors.
- It is performed during later stages of testing process, like in the acceptance testing or beta testing.

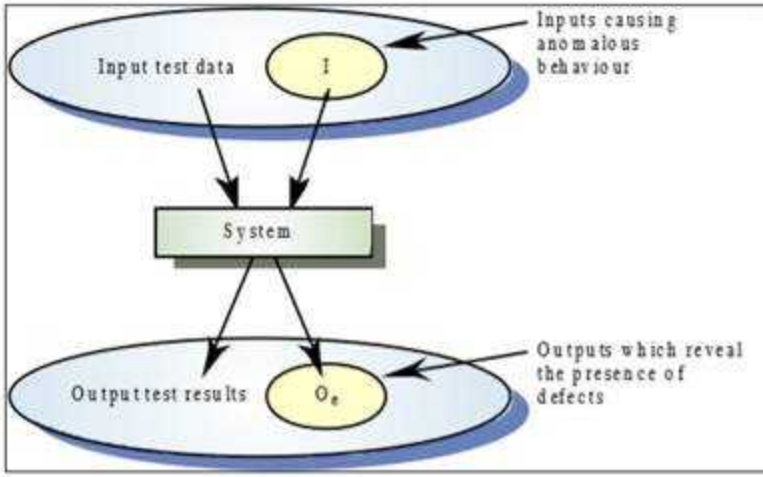


Fig: Black box testing

White-box testing

- Also known as Structural or Glass-box testing.
- Knowing the internal workings of a product, tests are performed to check the workings of all independent logic paths.
- It uses the control structure of the procedural design to derive test cases that:
    - Guarantee that all independent paths within a module have been exercised at least once.
    - Exercise all logical decisions on their true and false sides.
    - Execute all loops at their boundaries and within their operational bounds
    - Exercise internal data structures to ensure their validity

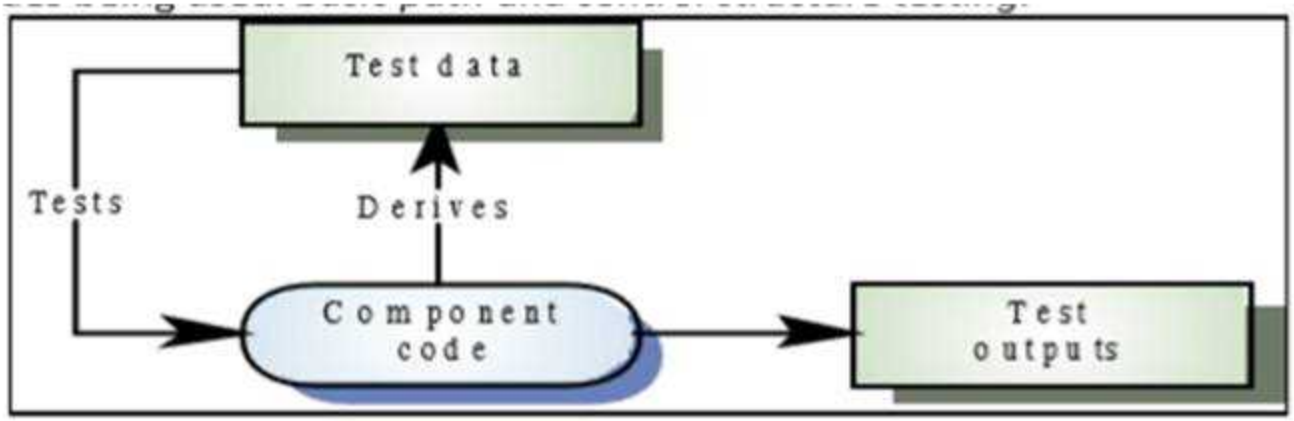


Fig: Structural testing or while box testing

Differences between Black Box Testing & White Box Testing

Black-box testing	White-box testing
Also referred as Function testing as it test the functionality of the system	Also referred as Structural testing as it checks the workings of all independent logic paths.
Also known as behavioral testing as system behavior is determined by studying its input and related the outputs.	Also known as glass-box testing as its internal coding part should be tested and verified.
Called " testing in the large"	Called "testing in the small" as applied to small program components.
Knowledge of internal program logic is not important.	Knowledge of internal program logic is an essential
Objective is to check it meets the user requirements as per the specification documents.	Objective is to check all program statement as per the design specification.
Discover ***faults of omission***, indicating that part of the specification has not been fulfilled.	Discover ***faults of commission***, indicating that part of the implementation is faulty.
Mostly performed by the user's side.	Mostly performed by the programmer's side
Back-box testing is done at the later stage of the SWSDLC.	While-box testing is done at the earlier stage of the SWDLC.
Performed as per the acceptance test plan.	Performed as per the Component and Integration test plan.
Techniques include Equivalence Partitioning, Boundary Value Analysis, Comparison Testing, and Orthogonal Array Testing.	Techniques include basic path and control structure testing
Errors include incorrect or missing functions, interface error, errors in data structures or external database access, behavior or performance errors, and initiation and termination errors.	Errors include logical errors in the coding statement and typing error.

Differences between Validation & Verification

Distinction between Verification and Validation

Software testing is one type of a broader domain that is known as verification and validation (V&V). Verification and validation is intended to show that a system conforms to its specification and meets the requirements of the system customer. It involves checking and review processes and system testing. The following table shows the distinction between verification and validation.

Verification	Validation
A set of operations that the software correctly implements a particular function.	a different set of activities that ensures that the software that has been produced is traceable to customer real needs.
Software verification is achieved through a series of while box tests that shows conformity to its design and statement of codes.	Software validation is achieved through a series of black box tests that show conformity with requirements.
Ensure the correct implementation of a specific function.	Ensure that the software that has been built is traceable to customer requirements.
It focuses on "are we building the product right?"	It focuses on "are we building the right product?"
It is performed at the each and every stages of the software coding like unit, module and subsystem testing.	It is performed at the later stages of the software development like system testing, acceptance testing.

Software Testing Plan

- Preparation and planning for the test phase should begin early in the development cycle and be a constant concern throughout the development process.

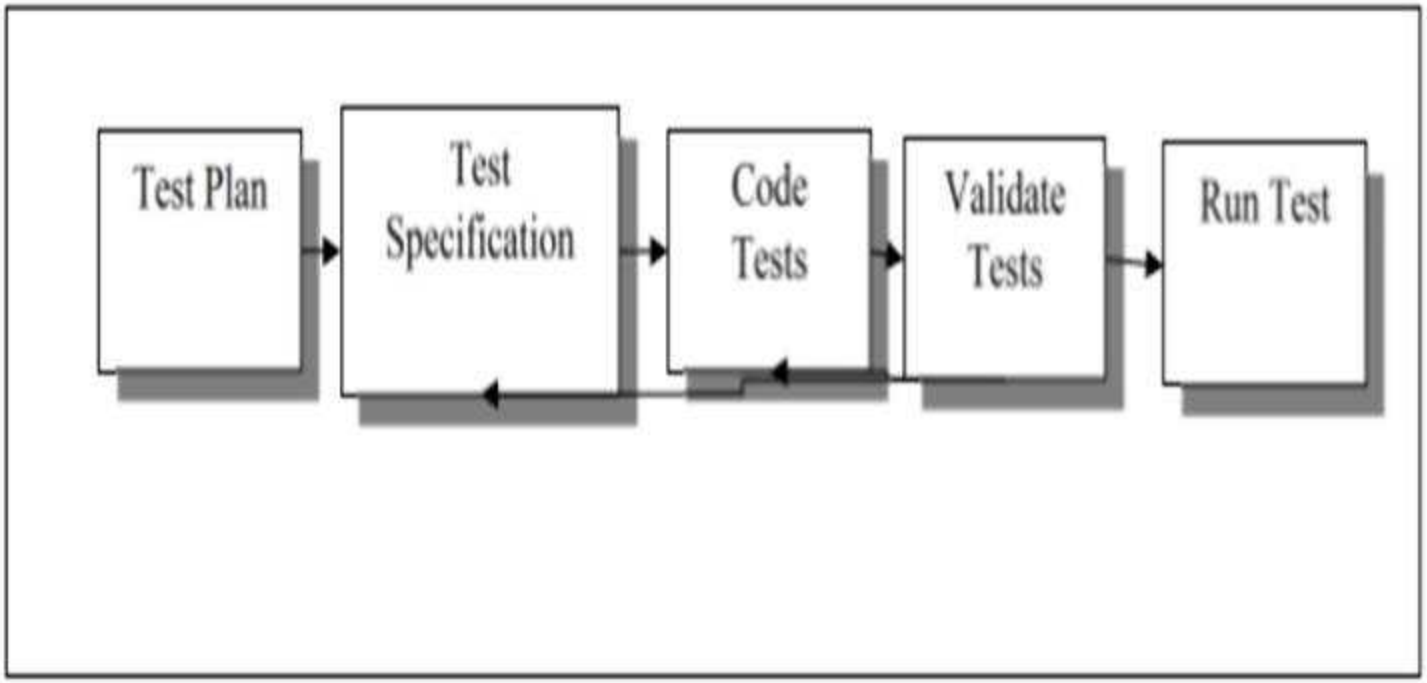


Fig: Test development cycle

Debugging

- Debugging (removal of a defect) occurs as a consequence of successful testing.
- Testing establishes the existence of defects where as debugging is concerned with locating and correcting these defect

Debugging Process

Figure illustrates a possible debugging process. Defects in the code must be located and the program modified to meet its requirements. Testing must then be repeated to ensure that the change has been made correctly. Thus debugging process is a part of both software development and software testing.

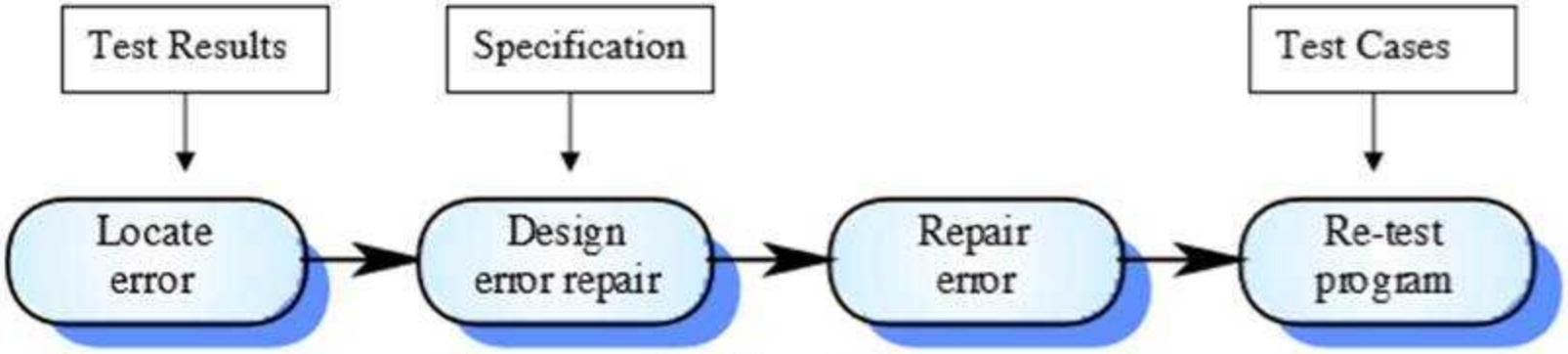


Fig: debugging process

Software Quality Assurance

- The aim of SQA is to help organizations develop high quality software.
- Conformance to software requirements is the foundation from which software quality is measured.
- Specified standards are used to define the development criteria that are used to guide the manner in which software is developed.
- Software must conform to implicit requirements as well as its explicit requirements.

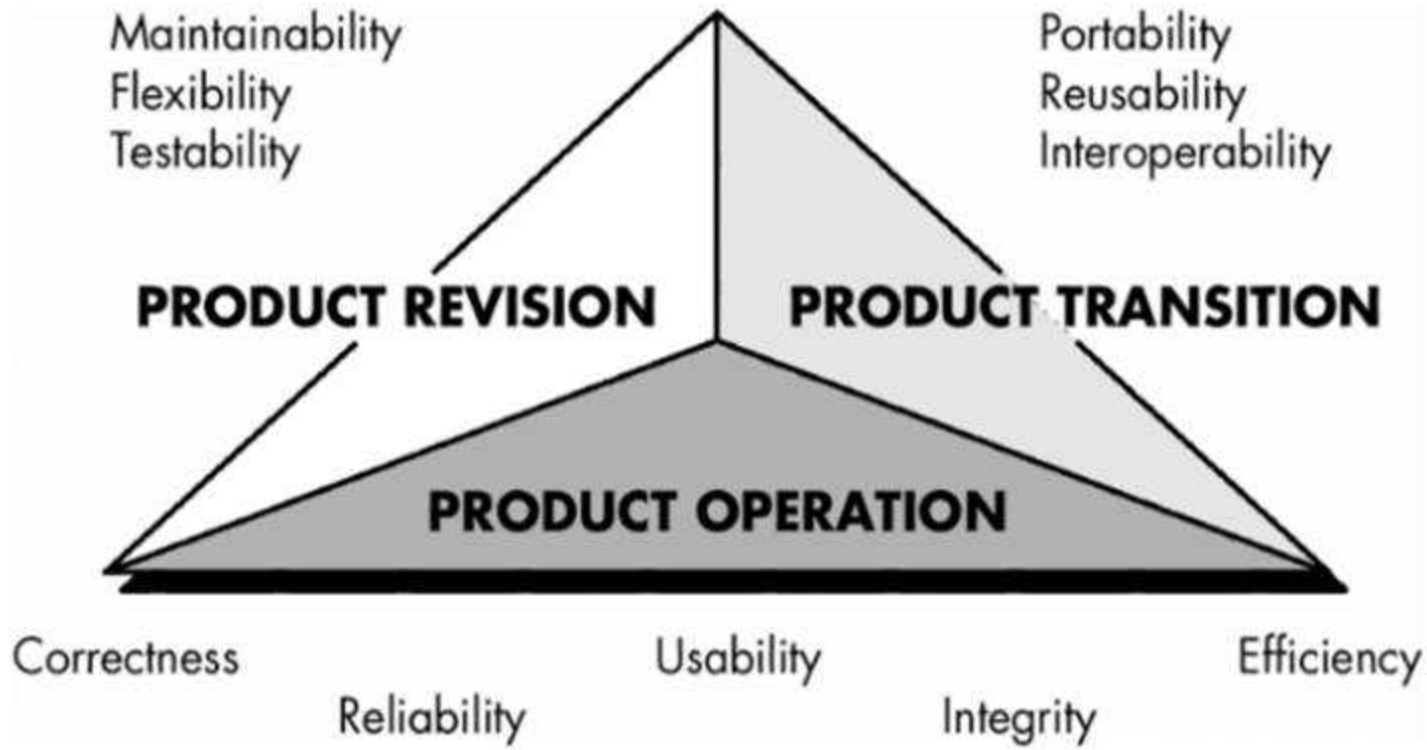
CTD...

- **Software Quality** is defined as "Conformance to explicitly stated functional and performance requirements, explicitly documented standards and implicit characteristics that are expected of all professionally developed software" .

Above definition serves three important points:

- ✓s/w requirements form a foundation for quality being measured. Lack of conformance to requirements is lack of quality.
- ✓If specified standards, that define a set of development criteria is not followed; lack of quality will almost surely result.
- ✓If s/w conform to its explicit requirements but fails to meet implicit requirements, s/w quality is suspect.

McCall's Quality Factors



McCall's Quality Factors

Software Quality Management

- Concerned with ensuring that the required level of quality is achieved in a software product.
- Involves defining appropriate quality standards and procedures and ensuring that these are followed.
- Should aim to develop a 'quality culture' where quality is seen as everyone's responsibility.

Quality Management Activities:

a) Quality Assurance

b) Quality Planning

c) Quality Control

CTD...

Quality Assurance

- Establishes a framework of organizational procedures and standards for high quality s/w.
- Consists of auditing and reporting procedures used to provide management with data needed to make proactive decisions.
- If data provided through QA identifies problems, management is responsible to address the problem and apply the necessary resources to resolve quality issues.

Quality Planning

- Selection of appropriate procedures and standards from this framework and adaptation of these for specific s/w projects and modify these as required.
- A quality plan sets out the desired product qualities and how these are assessed and the most significant quality attributes.

CTD...

Quality Control

- It ensures that the project quality procedures and standards are followed by the s/w development team.
- QC involves the series of inspection, reviews and test used throughout the s/w process to ensure conformance of a work product to its specifications.

Costs of Quality

a) Prevention cost
b) Appraisal cost
c) Failure cost

ISO 9000 Quality standards

- ISO 9000 describes quality assurance elements in generic terms that can be applied to any business.
- It treats an enterprise as a network of interconnected processes.
- To be ISO-compliant, processes should hold to the standards described.
- Elements include organizational structure, procedures, processes and resources.
- Ensures quality planning, quality control, quality assurance and quality improvement.
- **ISO 9001 standardizes the SQA activities**
- ISO 9000 is a generic standard - means that the standard can be applied to any organization, large or small, whatever its product in any sector of activity
- ISO is an auditable system – organization may be certified to a market standard by outside agency (Weigers, 2001)

ISO 9000 [CTD...]

An international standard which provides broad guidance to software developers on how to Implement, maintain and improve a quality software system capable of ensuring high quality software .**Consists of 20 requirements**:

- Management responsibility
- Quality system
- Contract review
- Design Control
- Document and data control
- Purchasing
- Control of customer supplied product
- Product identification and traceability
- Process control
- Inspection and testing
- Control of inspection, measuring and test equipment
- Inspection and test status
- Control of non-confirming product
- Corrective and preventive action
- Handling, storage, packaging, preservation and delivery
- Control of quality records
- Internal quality audits
- Training
- Servicing
- Statistical techniques

What is CMM?

- CMM: Capability Maturity Model
- Developed by the Software Engineering Institute of the Carnegie Mellon University
- Framework that describes the key elements of an effective software process.

What is CMM?

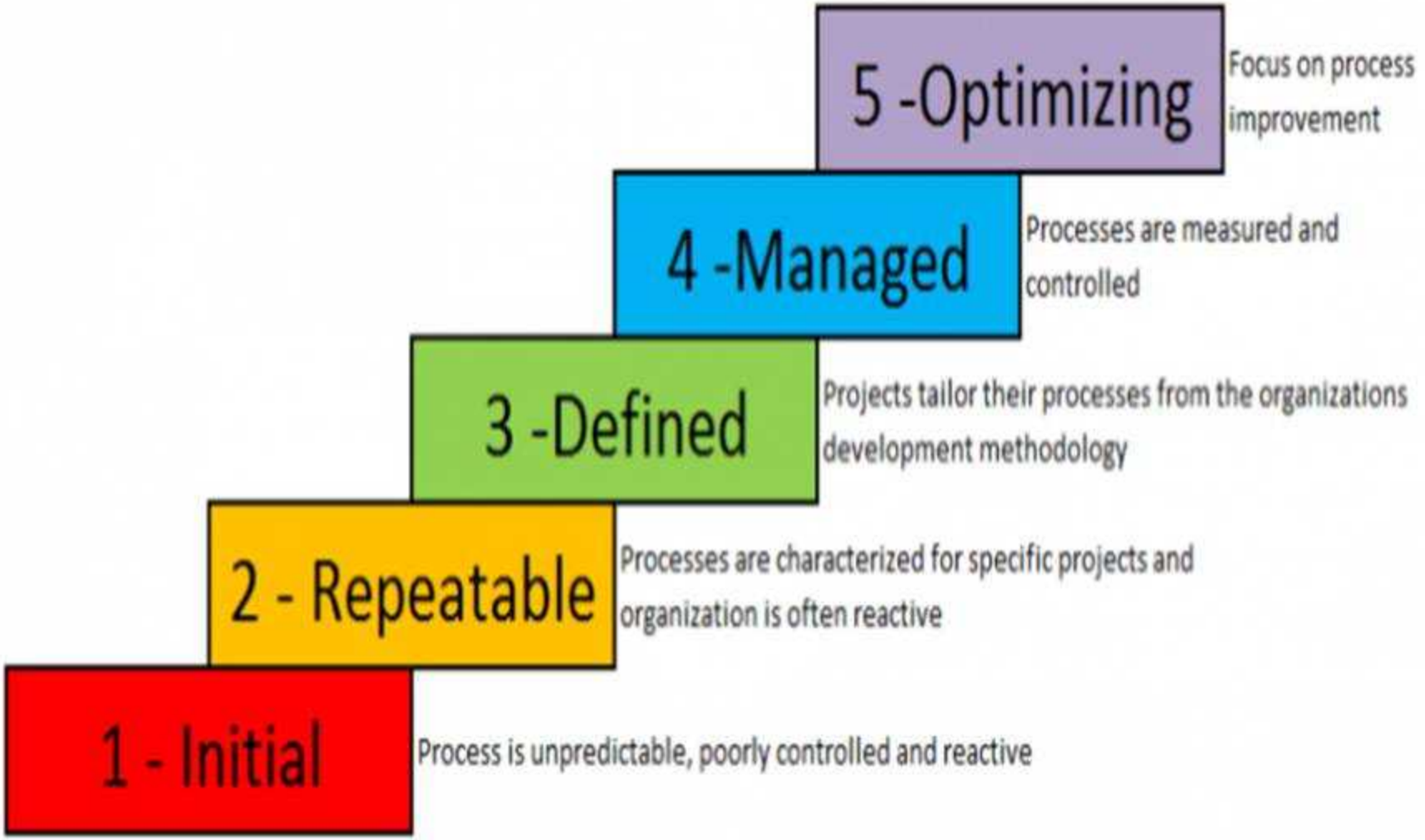
- Describes an evolutionary improvement path for software organizations from an ad hoc, immature process to a mature, disciplined one.
- Provides guidance on how to gain control of processes for developing and maintaining software and how to evolve toward a culture of software engineering and management excellence.

What are the CMM Levels?

(The five levels of software process maturity)

Maturity level indicates level of process capability:

- Initial
- Repeatable
- Defined
- Managed
- Optimizing


5 -Optimizing
Focus on process improvement
4 -Managed
Processes are measured and controlled
3 -Defined
Projects tailor their processes from the organizations development methodology
2 - Repeatable
Processes are characterized for specific projects and organization is often reactive
1 - Initial
Process is unpredictable, poorly controlled and reactive

Level 1: Initial

- Initial : The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.
  - Products developed are often over budget and schedule
  - Wide variations in cost, schedule, functionality and quality targets
  - Capability is a characteristic of the individuals, not of the organization
  - 80% of software organizations worldwide

Level 2: Repeatable

- Basic process management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
  - Realistic project commitments based on results observed on previous projects
  - Software project standards are defined and faithfully followed
  - Processes may differ between projects
  - Process is disciplined
  - earlier successes can be repeated
  - 15% of software organizations worldwide

Level 3: Defined

- The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved software process model for developing an maintaining software.
- 5% of software organizations worldwide

Level 4 - Managed

- Quality and performance is measured using metrics and can be predicted
- Quality and productivity *quantitative* goals are established
- Exceptional cases are identified and addressed
- Challenge of new domains can be managed
- *Process is measured and operates within limits*
- < 1% of software organizations worldwide

Level 5 - Optimizing

- Organization focuses on continuous process improvement
- Goal is to address and prevent problems by analyzing their cause in the process
- Process improvement is budgeted, planned, and part of the organization's process
- Identify and quickly transfer best practices
- Only a handful of organizations worldwide

CMM Level	Focus	Key Process Areas
1. Initial	Competent People	NO KPA'S
2. Repeatable	Project Management	Software Project Planning software Configuration Management
3. Defined	Definition of Processes	Process definition Training Program Peer reviews
4. Managed	Product and Process quality	Quantitiative Process Metrics Software Quality Management
5. Optimizing	Continuous Process improvement	Defect Prevention Process change management Technology change management

The focus of each SEI CMM level and the Corresponding Key process areas.

Software Reviews

- Purpose is to find defects (errors) before they are passed on to another software engineering activity or released to the customer.
- Software engineers (and others) conduct formal technical reviews (FTR) for software engineers.
- Using formal technical reviews (walkthroughs or inspections) is an effective means for improving software quality.

Formal Technical Reviews

Involves 3 to 5 people (including reviewers)

- Advance preparation (no more than 2 hours per person) required
- Duration of review meeting should be less than 2 hours
- Focus of review is on a discrete work product
- Review leader organizes the review meeting at the producer's request
- Reviewers ask questions that enable the producer to discover his or her own error (the product is under review not the producer)
- Producer of the work product walks the reviewers through the product
- Recorder writes down any significant issues raised during the review
- Reviewers decide to accept or reject the work product and whether to require additional reviews of product or not

Go through the Test case design, Boundary Value Analysis (BVA) and Equivalence Partitioning done in my lecture class