Software Engineering
[CACS253]
BCA 4th Sem

Er. Sital Prasad Mandal

Mechi Multiple Campus
Bhadrapur, Jhapa, Nepal

Unit 6

Software Testing and Quality Assurance

1. **Verification and Validation**
2. **Techniques of Testing: Black-box and White-box Testing**
3. **Software Inspections**
4. **Level of Testing: Unit Testing**
   Integration Testing
   Interface Testing
   System Testing
   Alpha and Beta Testing
   Regression Testing
5. **Design of Test Cases**
6. **Quality Management Activities**
7. **Product and Process Quality**
8. **Standards: IS09000**
9. **Capability Maturity Model (CMM)**

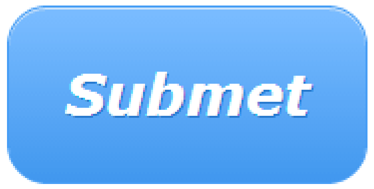
Verification and Validation

Verification in Software Testing is a process of checking documents, design, code, and program in order to check if the software has been built according to the requirements or not. The main goal of verification process is to ensure quality of software application, design, architecture etc. The verification process involves activities like reviews, walk-throughs and inspection.

Validation in Software Engineering is a dynamic mechanism of testing and validating if the software product actually meets the exact needs of the customer or not. The process helps to ensure that the software fulfills the desired use in an appropriate environment. The validation process involves activities like unit testing, integration testing, system testing and user acceptance testing.

Verification and Validation

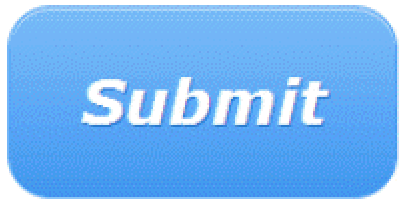
A clickable button with name Submet
Verification would check the design doc and correcting the spelling mistake.
Otherwise, the development team will create a button like



Example of Verification

Are we building the product right?

A clickable button with name Submit
Once the code is ready, Validation is done. A Validation test found –



Example of Validation

Are we building the right product?

2 . Techniques of Testing

Testing Techniques is the method applied to evaluate a system or a component with a purpose to find if it satisfies the given requirements. Testing of a system helps to identify gaps, errors, or any kind of missing requirements differing from the actual requirements.

These techniques ensure the overall quality of the product or software including performance, security, customer experience, and so on.

2 . Techniques of Testing

Black Box Testing

Black box testing is a type of software testing, which checks for the functionality of a software or an application without knowing the design, internal components, or structure of an application to be tested. It is also referred to as Specifications-based testing.
The black box testing method is mainly used to find missing functions, performance errors, initialization errors, and errors while accessing the external database.

2 . Techniques of Testing

White Box Testing

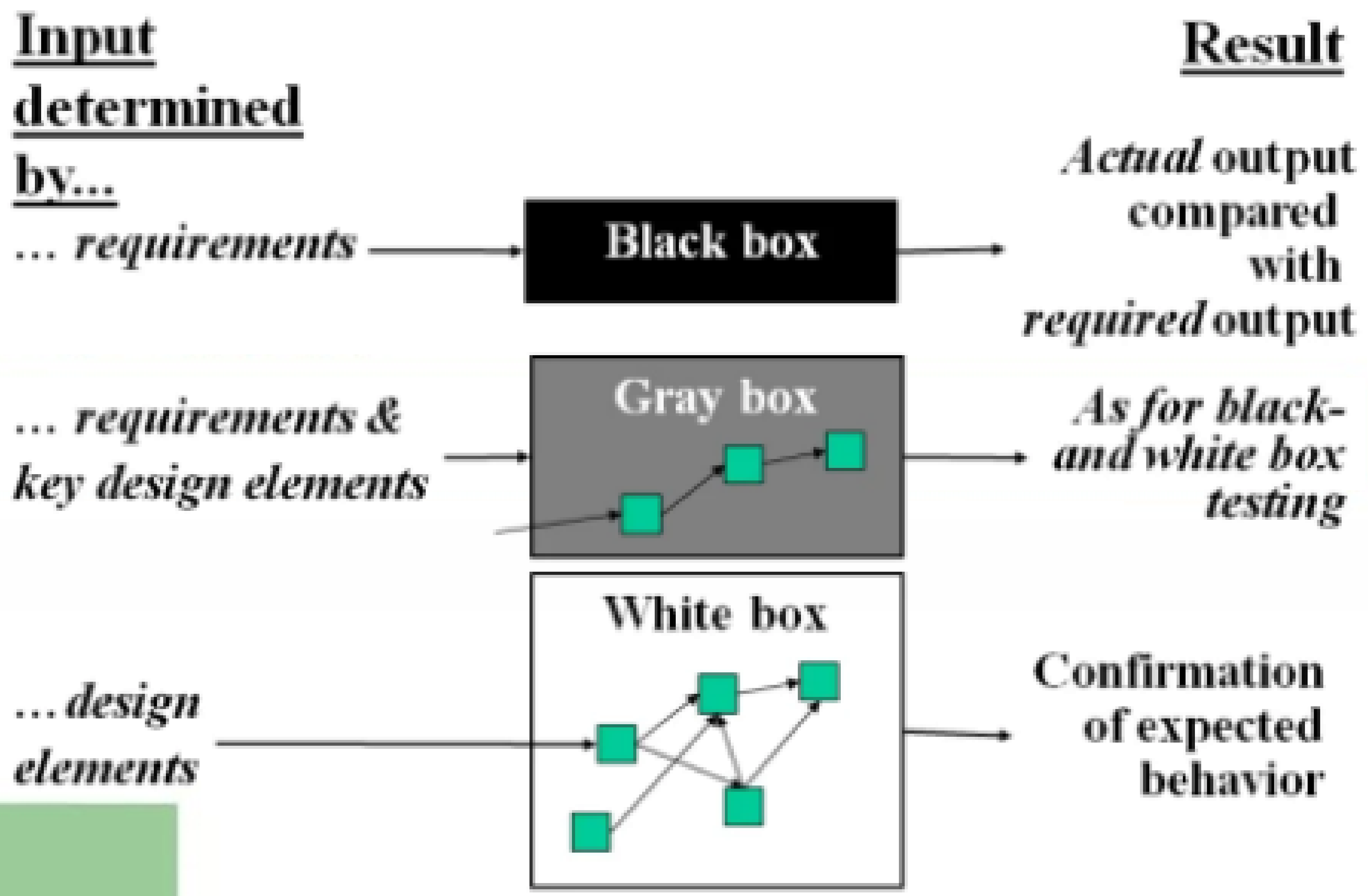
White box testing is a method of software testing that tests internal programming structures of an application. This type of testing technique is known as clear box testing, open box testing, structural testing, and transparent box testing. Its operation is opposite to black-box testing and is used at unit, integration, and system levels of the testing process.

2 . Techniques of Testing

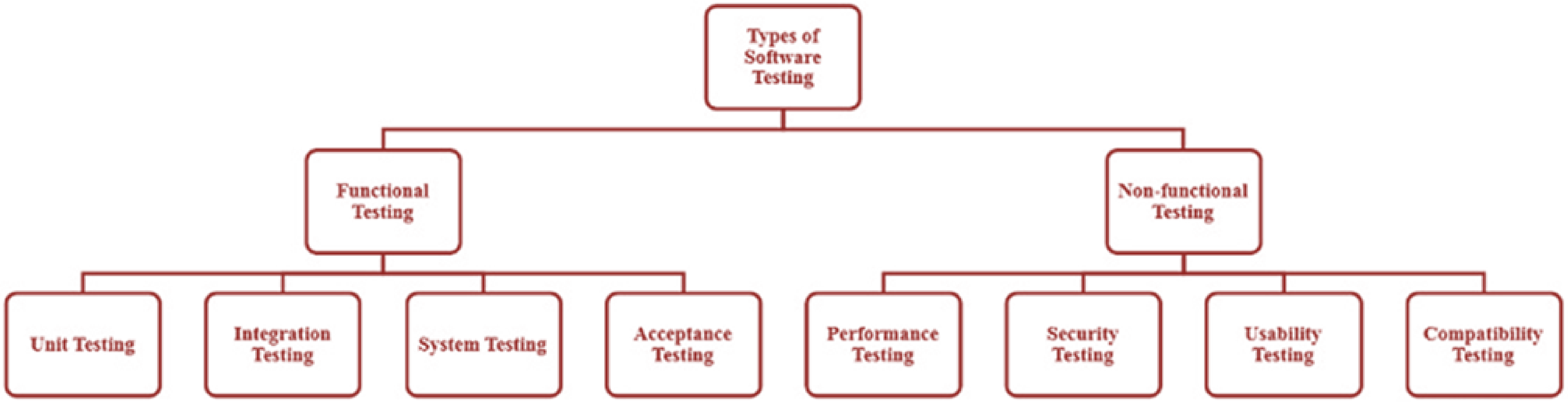
Black Box Testing – (application interface, internal module interface and input/output description)

White Box Testing- function executed and checked

Gray Box Testing - test cases, risks assessments and test methods


Input determined by…
… requirements
Black box
Result
Actual output compared with required output
… requirements & key design elements
Gray box
As for black- and white box testing
… design elements
White box
Confirmation of expected behavior

2 . Techniques of Testing



3. Software Inspections

The term software inspection was developed by IBM in the early 1970s, when it was noticed that the testing was not enough sufficient to attain high quality software for large applications.

Inspection is used to determine the defects in the code and remove it efficiently. This prevents defects and enhances the quality of testing to remove defects.

This software inspection method achieved the highest level for efficiently removing defects and improving software quality.

A small group of programmers who follow a specific procedure to review the source code developed by a peer with the intent of identifying defects and improving maintainability.

3. Software Inspections

Participants and Roles:

During software inspection, all the participants have a set role. However, more than one role can be assigned to the same person, in which case the team size may vary from one inspection to another. The roles of these participants are:

Author	Programmer or designer who is responsible for producing the program/document that is being inspected.
Moderator	Leader of the inspection who is responsible for planning the inspection & coordinating it.
Reader	Present the code at an inspection meeting, where they read the document one by one.
Recorder/ Scribe	A participant who is responsible for documenting the defects found during the inspection process.
Inspector	Inspectors are allotted the task of examining the work product to identify possible defects.

3. Software Inspections

Software Inspection Process : The inspection process was developed in the mid 1970s, later extended and revised. The process must have an entry basis that determines whether the inspection process is ready to begin. this prevents incomplete products from entering the inspection process. Entry criteria can be substitutional with items such as “The Spell-Document Check”.

There are some of the stages in the software inspection process such as-

1. **Planning**: The inspection is planned by the moderator.
2. **Overview** meeting: The author describes the background of the work product.
3. **Preparation**: Each inspector examines the work product to identify possible defects.
4. **Inspection** meeting: During this meeting the reader reads through the work product, part by part and the inspectors point out the defects for every part.
5. **Rework**: The author makes changes to the work product according to the action plans from the inspection meeting.
6. **Follow-up**: The changes by the author are checked to make sure everything is correct.

3. Software Inspections

Advantages of Software Inspection:

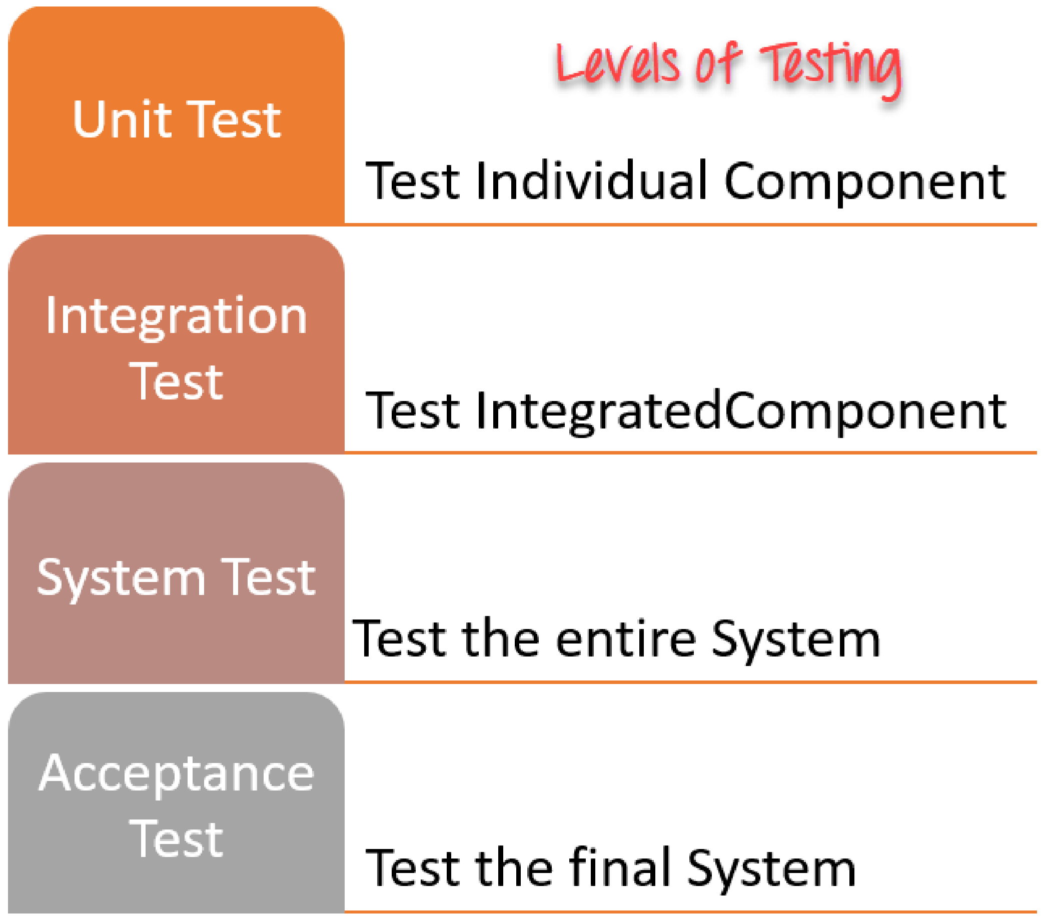
- Helps in the Early removal of major defects.
- This inspection enables a numeric quality assessment of any technical document.
- Software inspection helps in process improvement.
- It helps in staff training on the job.
- Software inspection helps in productivity.

Disadvantages of Software Inspection:

- It is a time-consuming process.
- Software inspection requires discipline.

4. Level of Testing

1. **Unit/ Component Testing**
2. **Integration Testing**
3. **Interface Testing**
4. **System Testing**
5. **Regression Testing**
6. **Alpha and Beta Testing**



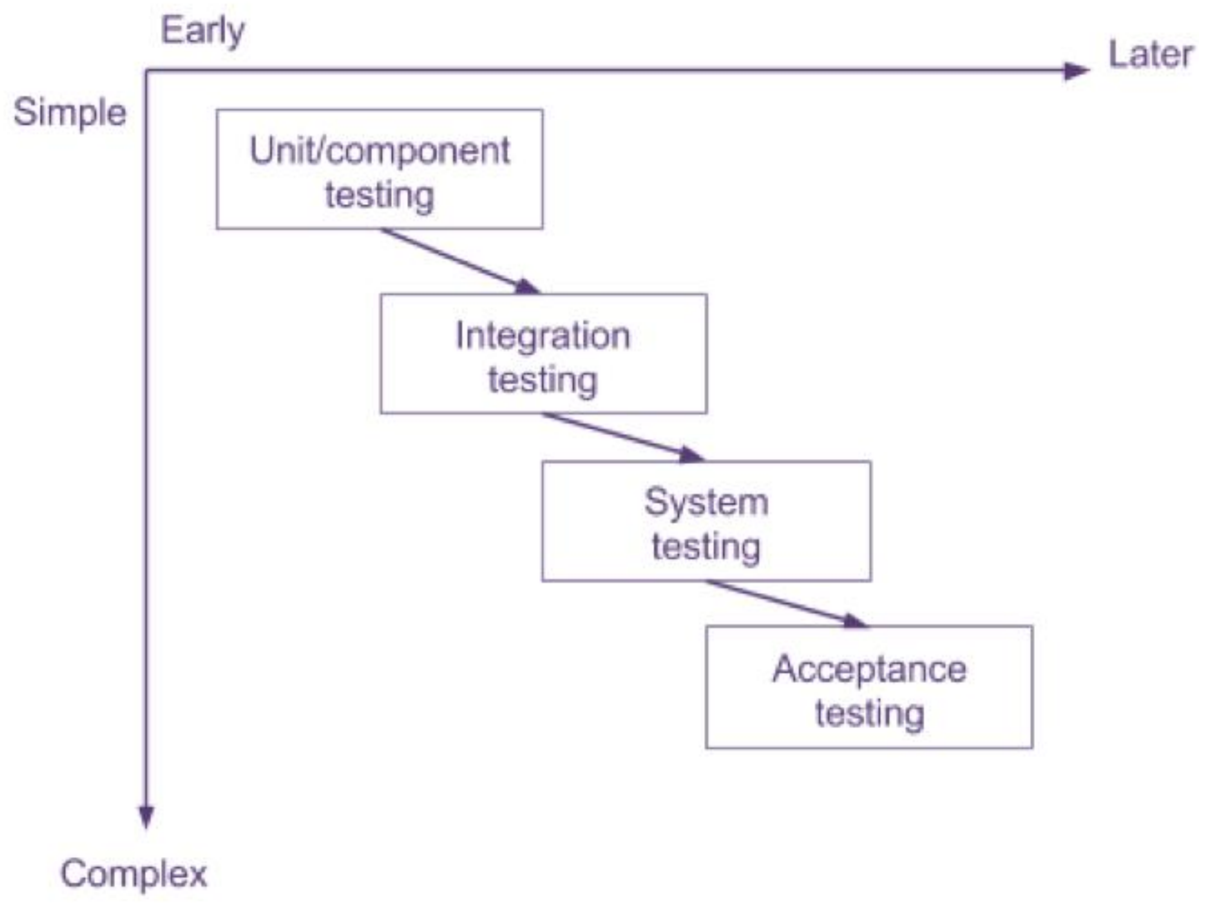
4. Level of Testing

Unit Testing : checks if software components are fulfilling functionalities or not.

Integration Testing : checks the data flow from one module to other modules.

System Testing : evaluates both functional and non-functional needs for the testing.

Acceptance Testing : checks the requirements of a specification or contract are met as per its delivery.



4. Level of Testing

1) **Unit testing:**

A Unit is a smallest testable portion of system or application which can be compiled, liked, loaded, and executed. This kind of testing helps to test each module separately.

The aim is to test each part of the software by separating it. It checks that component are fulfilling functionalities or not. This kind of testing is performed by developers.

2) **Integration testing:**

The second level of software testing is the **integration testing.** The integration testing process comes after **unit testing**. It is mainly used to test the **data flow from one module or component to other modules.** In integration testing, the **test engineer** tests the units or separate components or modules of the software in a group. When each component or module works separately, we need to check the data flow between the dependent modules, and this process is known as **integration testing**. In simple words, we can say that **integration testing** aims to evaluate the accuracy of communication among all the modules.

3) **System testing:**

The third level of software testing is **system testing**, which is used to test the software's functional and non-functional requirements. It is **end-to-end testing** where the testing environment is parallel to the production environment. In the third level of software testing, **we will test the application as a whole system.** To check the end-to-end flow of an application or the software as a user is known as **System testing**.

4. Level of Testing

4) **Acceptance testing:**

Acceptance testing is a test conducted to find if the requirements of a specification or contract are met as per its delivery. Acceptance testing is basically done by the user or customer. However, other stockholders can be involved in this process.

5. Regression Testing

Whenever a change in a software application is made, it is quite possible that other areas within the application have been affected by this change. Regression testing is performed to verify that a fixed bug hasn't resulted in another functionality or business rule violation. The intent of regression testing is to ensure that a change, such as a bug fix should not result in another fault being uncovered in the application.

4. Level of Testing

Alpha Testing

This test is the first stage of testing and will be performed amongst the teams (developer and Quality Assurance teams). Unit testing, integration testing and system testing when combined together is known as alpha testing.

During this phase, the following aspects will be tested in the application –

- Spelling Mistakes
- Broken Links
- Cloudy Directions
- The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

4. Level of Testing

Beta Testing

This test is performed after alpha testing has been successfully performed. In beta testing, a sample of the intended audience tests the application. Beta testing is also known as **pre-release testing**. Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program a "real-world" test and partly to provide a preview of the next release.

In this phase, the audience will be testing the following –

- Users will install, run the application and send their feedback to the project team.
- Typographical errors, confusing application flow, and even crashes.
- The more issues you fix that solve real user problems, the higher the quality of your application will be.
- Having a higher-quality application when you release it to the general public will increase customer satisfaction.

5. Design of Test Cases

Test case design refers to how you set-up your test cases. It is important that your tests are designed well, or you could fail to identify bugs and defects in your software during testing.

There are many different test case design techniques used to test the functionality and various features of your software. Designing good test cases ensure that every aspect of your software gets tested so that you can find and fix any issues.

<u>A basic example of test case design:</u>

Title: *Login to the website or app*

Description: *User should be able to successfully log in to their account on the website/app*

Preconditions: *User must already be registered and use their correct login details*

Assumptions: *They are using a supported device or browser to log in*

Test Steps:

Open website or app

Enter the username and password in the appropriate fields

Click "login"

Expected Result: *The user should log in successfully.*

5. Design of Test Cases

Test case design refers to how you set-up your test cases. It is important that your tests are designed well, or you could fail to identify bugs and defects in your software during testing.

There are many different test case design techniques used to test the functionality and various features of your software. Designing good test cases ensure that every aspect of your software gets tested so that you can find and fix any issues.

<u>A basic example of test case design:</u>

Title: *Login to the website or app*
Description: *User should be able to successfully log in to their account on the website/app*
Preconditions: *User must already be registered and use their correct login details*
Assumptions: *They are using a supported device or browser to log in*
Test Steps:

Open website or app
Enter the username and password in the appropriate fields
Click "login"

Expected Result: *The user should log in successfully.*

5. Design of Test Cases

What are the types of test case design techniques?

The main purpose of test case design techniques is to test the functionalities and features of the software with the help of effective test cases. The test case design techniques are broadly classified into three major categories.

1. Specification-Based techniques
2. Structure-Based techniques
3. Experience-Based techniques

5. Design of Test Cases

Boundary Value Analysis (BVA)

This technique is applied to explore errors at the boundary of the input domain. BVA catches any input errors that might interrupt with the proper functioning of the program.

Equivalence Partitioning (EP)

In Equivalence Partitioning, the test input data is partitioned into a number of classes having an equivalent number of data. The test cases are then designed for each class or partition. This helps to reduce the number of test cases.

Decision Table Testing

In this technique, test cases are designed on the basis of the decision tables that are formulated using different combinations of inputs and their corresponding outputs based on various conditions and scenarios to different business rules.

5. Design of Test Cases

State Transition Diagrams

In this technique, the software under test is perceived as a system having a finite number of states of different types. The transition from one state to another is guided by a set of rules. The rules define the response to different inputs. This technique can be implemented on the systems which have certain workflows within them.

Use Case Testing

A use case is a description of a particular use of the software by a user. In this technique, the test cases are designed to execute different business scenarios and end-user functionalities. Use case testing helps to identify test cases that cover the entire system.

5. Design of Test Cases

2. Structure-Based or White-Box techniques

The structure-based or white-box technique design test cases based on the internal structure of the software. This technique complete tests of developed code. Developers who have complete information of the software code, its internal structure, and design help to design the test cases. This technique is further divided into five categories.

1. **Statement Testing & Coverage**
2. **Decision Testing Coverage**
3. **Condition Testing**
4. **Multiple Condition Testing**
5. **All Path Testing**

5. Design of Test Cases

Statement Testing & Coverage

This technique involves execution of all the executable statements in the source code at least once. The percentage of the executable statements is calculated as per the given requirement. This is the least preferred metric for checking test coverage.

Decision Testing Coverage

This technique is also known as branch coverage is a testing method in which each one of the possible branches from each decision point is executed at least once to ensure all reachable code is executed. This helps to validate all the branches in the code. This helps to ensure that no branch leads to unexpected behavior of the application.

5. Design of Test Cases

Condition Testing

Condition testing also is known as Predicate coverage testing, each Boolean expression is predicted as TRUE or FALSE. All the testing outcomes are at least tested once. This type of testing involves 100% coverage of the code. The test cases are designed as such that the condition outcomes are easily executed.

Multiple Condition Testing

The purpose of Multiple condition testing is to test the different combination of conditions to get 100% coverage. To ensure complete coverage, two or more test scripts are required which requires more efforts.

All Path Testing

In this technique, the source code of a program is leveraged to find every executable path. This helps to determine all the faults within a particular code.

5. Design of Test Cases

Condition Testing

Condition testing also is known as Predicate coverage testing, each Boolean expression is predicted as TRUE or FALSE. All the testing outcomes are at least tested once. This type of testing involves 100% coverage of the code. The test cases are designed as such that the condition outcomes are easily executed.

Multiple Condition Testing

The purpose of Multiple condition testing is to test the different combination of conditions to get 100% coverage. To ensure complete coverage, two or more test scripts are required which requires more efforts.

All Path Testing

In this technique, the source code of a program is leveraged to find every executable path. This helps to determine all the faults within a particular code.

5. Design of Test Cases

3. Experience-Based techniques

These techniques are highly dependent on tester's experience to understand the most important areas of the software. The outcomes of these techniques are based on the skills, knowledge, and expertise of the people involved. The types of experience-based techniques are as follows:

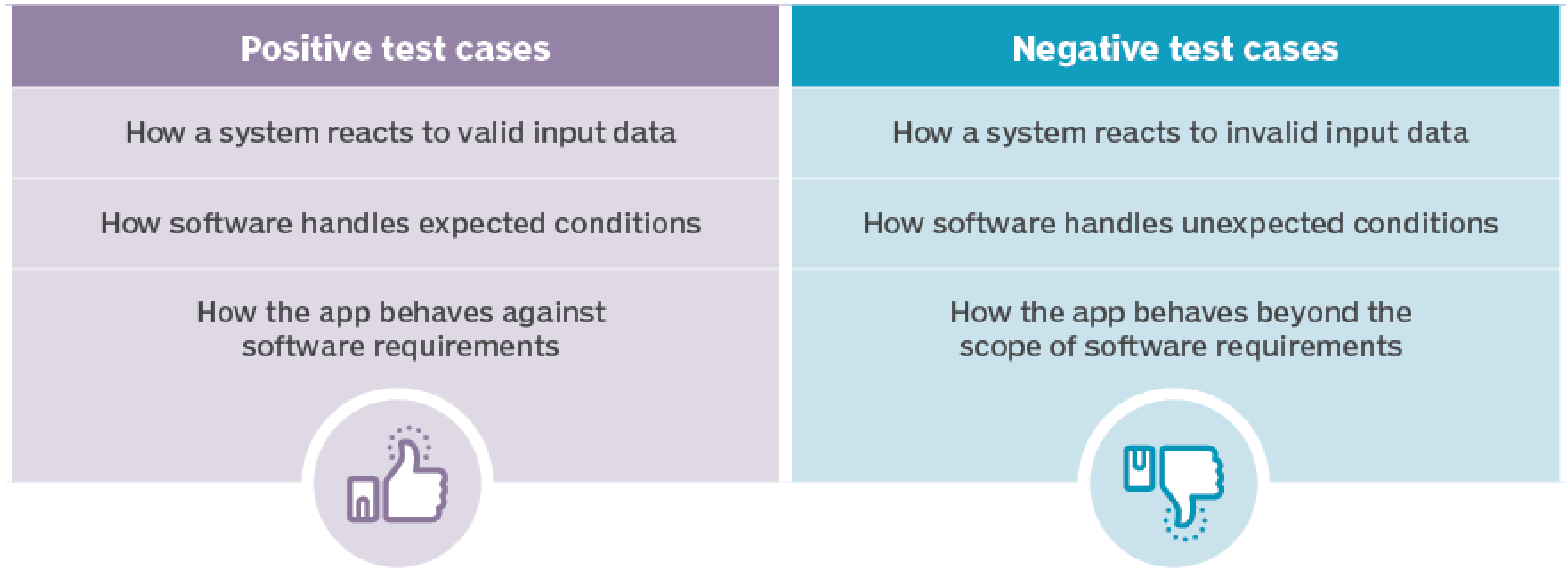
Error Guessing

In this technique, the testers errors based on their experience, availability of data and their knowledge of product failure. Error guessing is dependent on the skills and experience of the testers.

Exploratory Testing

This technique is used to test the application without any formal documentation. There is minimum time available for testing and maximum for test execution. In exploratory testing, the test design and test execution are performed concurrently.

5. Design of Test Cases



Positive test cases	Negative test cases
How a system reacts to valid input data	How a system reacts to invalid input data
How software handles expected conditions	How software handles unexpected conditions
How the app behaves against software requirements	How the app behaves beyond the scope of software requirements

6. Quality Management Activities

Software Quality Management ensures that the required level of quality is achieved by submitting improvements to the product development process. SQA aims to develop a culture within the team and it is seen as everyone's responsibility.

Software Quality management should be independent of project management to ensure independence of cost and schedule. It directly affects the ***process quality*** and indirectly affects the ***product quality***.

Activities of Software Quality Management:

Quality Assurance - QA aims at developing Organizational procedures and standards for quality at Organizational level.

Quality Planning - Select applicable procedures and standards for a particular project and modify as required to develop a quality plan.

Quality Control - Ensure that best practices and standards are followed by the software development team to produce quality products.

6. Quality Management Activities

Software Quality Attributes

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

7. Product and Process Quality

- ✧The quality of a developed product is influenced by the quality of the production process.
- ✧This is important in software development as some product quality attributes are hard to assess.
- ✧However, there is a very complex and poorly understood relationship between software processes and product quality.
  - The application of individual skills and experience is particularly important in software development;
  - External factors such as the novelty of an application or the need for an accelerated development schedule may impair product quality.

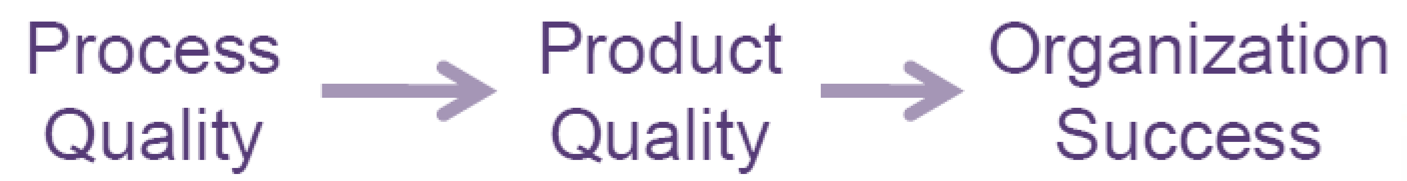
7. Product and Process Quality

Focus on product quality

- –Defects as a quality indicator
- –Other quality attributes
- –Customer perspectives

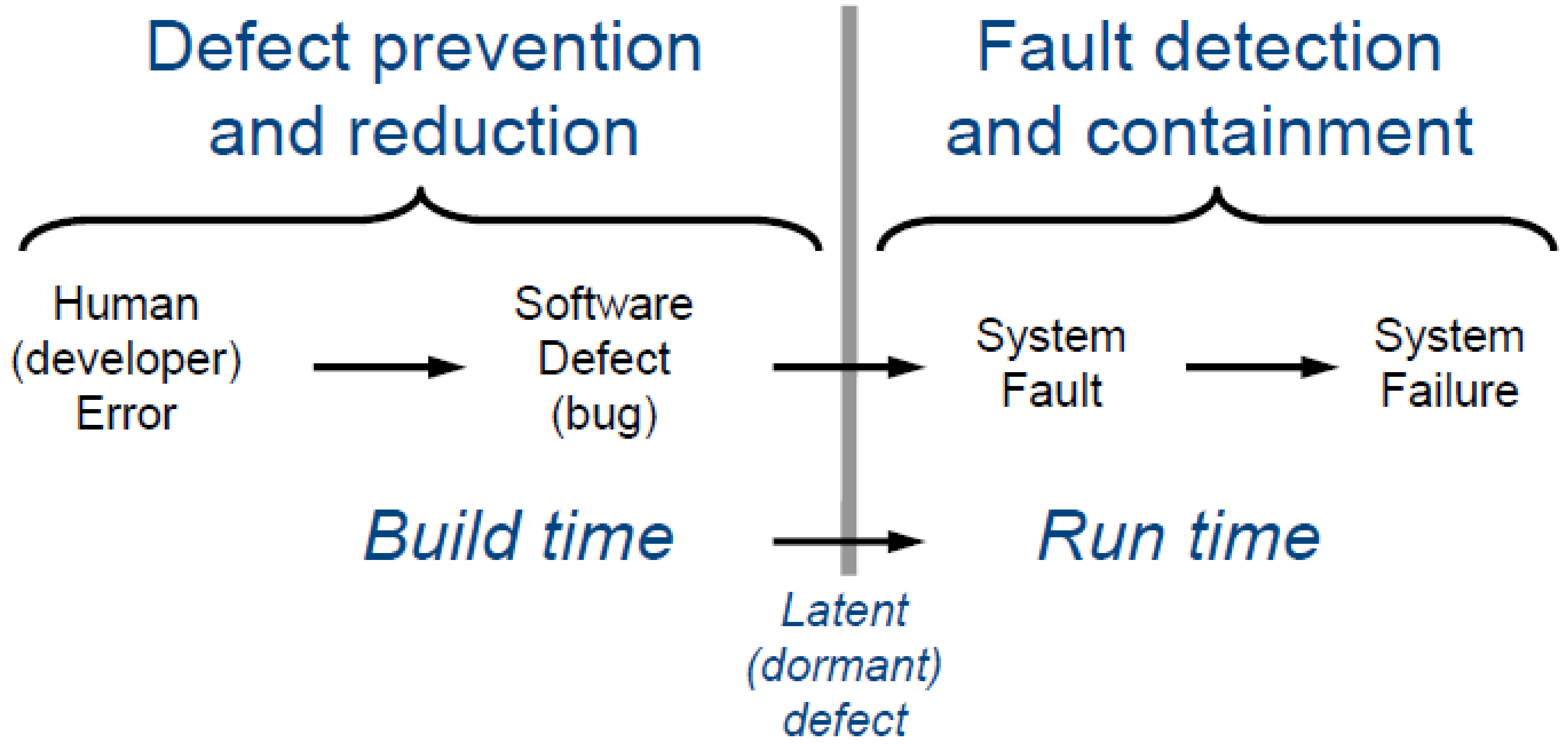
- **Focus on process quality**
  - –Project-level and activity-level
  - –Process capability assessment and improvement
    - ▪ Quality system frameworks

Premise of a Quality Focus



7. Product and Process Quality

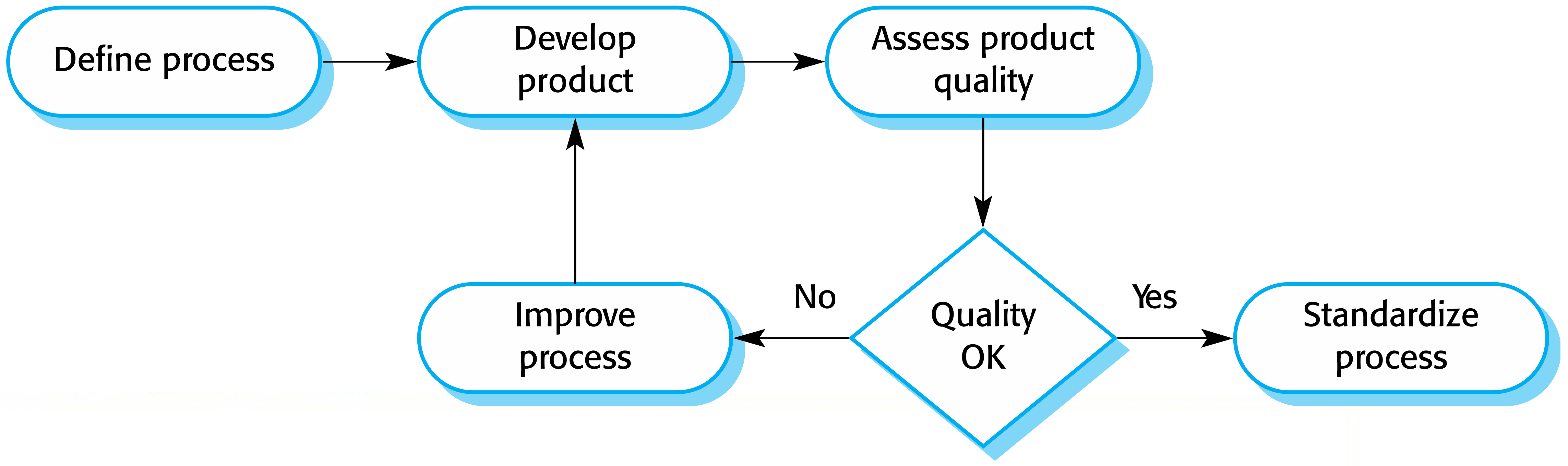
Product Quality



Defects and Failures

7. Product and Process Quality

Process-based Quality



7. Product and Process Quality

Quality Culture

- Quality managers should aim to develop a ‘quality culture’ where everyone responsible for software development is committed to achieving a high level of product quality.
- They should encourage teams to take responsibility for the quality of their work and to develop new approaches to quality improvement.
- They should support people who are interested in the intangible aspects of quality and encourage professional behavior in all team members.

Software standards

- Standards define the required attributes of a product or process.
- They play an important role in quality management.
- Standards may be international, national, organizational or project standards.

Software standards

Importance of standards

- Encapsulation of best practice- avoids repetition of past mistakes.
- They are a framework for defining what quality means in a particular setting i.e. that organization’s view of quality.
- They provide continuity - new staff can understand the organisation by understanding the standards that are used.

Software standards

Product standards

Apply to the software product being developed. They include document standards, such as the structure of requirements documents, documentation standards, such as a standard comment header for an object class definition, and coding standards, which define how a programming language should be used.

Process standards

These define the processes that should be followed during software development. Process standards may include definitions of specification, design and validation processes, process support tools and a description of the documents that should be written during these processes.

Product standards	Process standards
Design review form	Design review conduct
Requirements document structure	Submission of new code for system building
Method header format	Version release process
Java programming style	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

Software standards

Product standards

Apply to the software product being developed. They include document standards, such as the structure of requirements documents, documentation standards, such as a standard comment header for an object class definition, and coding standards, which define how a programming language should be used.

Process standards

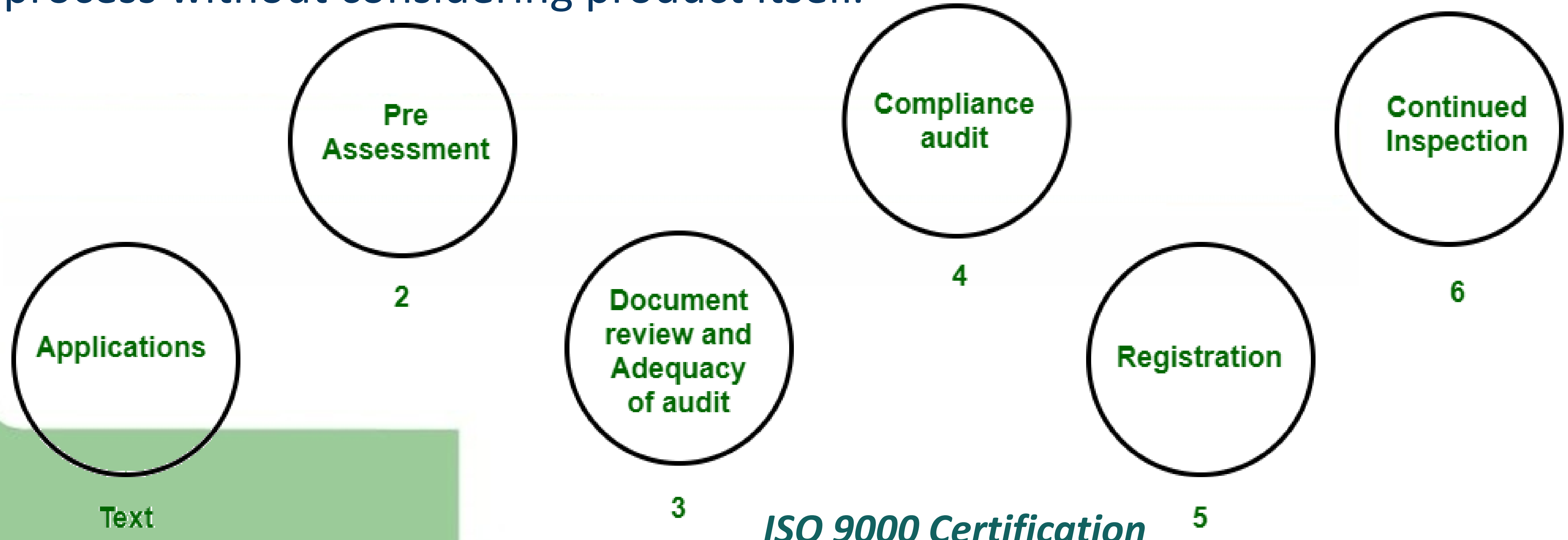
These define the processes that should be followed during software development. Process standards may include definitions of specification, design and validation processes, process support tools and a description of the documents that should be written during these processes.

Product standards	**Process standards**
Design review form	Design review conduct
Requirements document structure	Submission of new code for system building
Method header format	Version release process
Java programming style	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

Standards: ISO 9000

The International organization for Standardization is a world wide federation of national standard bodies. The **International standards organization (ISO)** is a standard which serves as a for contract between independent parties. It specifies guidelines for development of **quality system**.

Quality system of an organization means the various activities related to its products or services. Standard of ISO addresses to both aspects i.e. operational and organizational aspects which includes responsibilities, reporting etc. An ISO 9000 standard contains set of guidelines of production process without considering product itself.



ISO 9000 Certification

Standards: IS0 9000

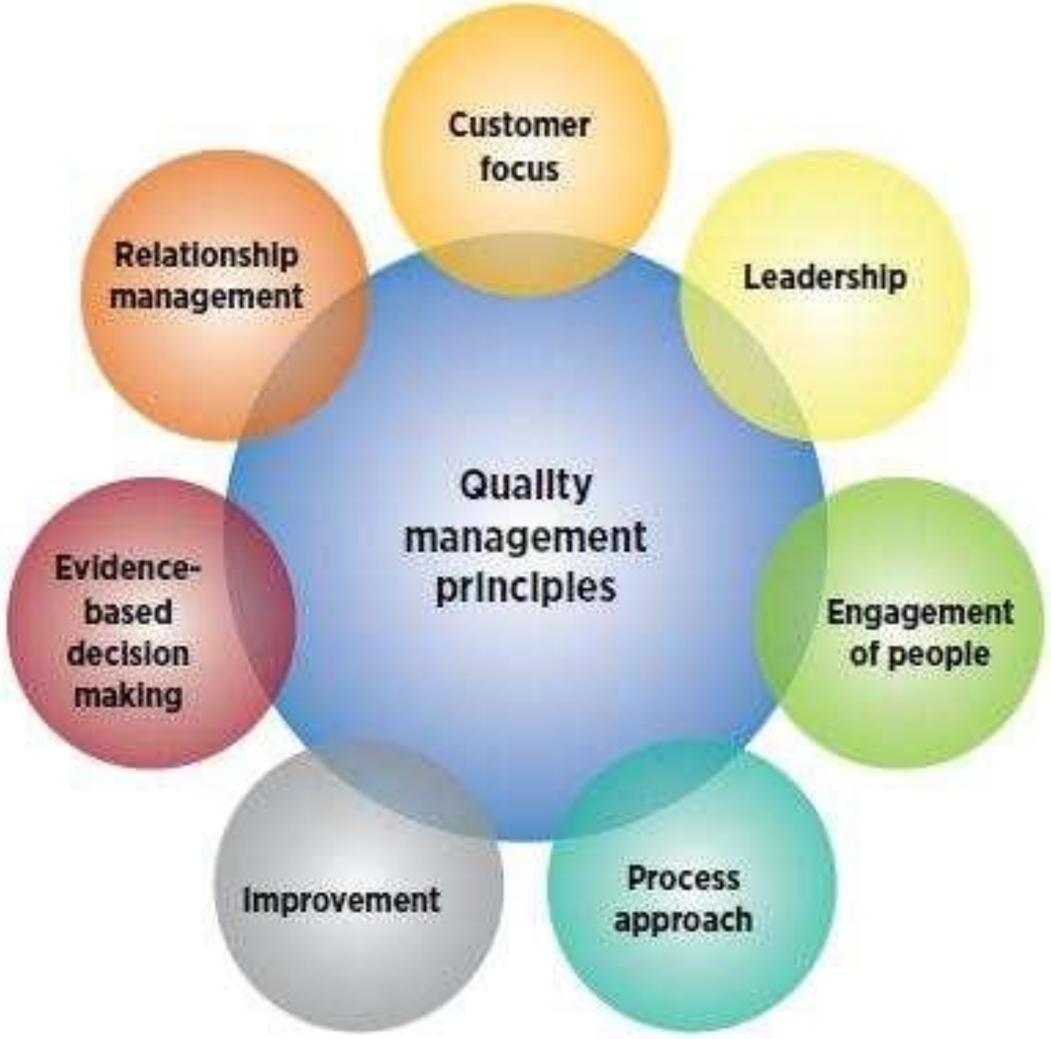
Why ISO Certification required by Software Industry?

Some of reasons are as follows :

- This certification has become a standards for international bidding.
- It helps in designing high-quality repeatable software products.
- It emphasis need for proper documentation.
- It facilitates development of optimal processes and totally quality measurements.

Features of ISO 9001 Requirements :

1. Document control
2. Planning
3. Review
4. Testing
5. Organizational Aspects



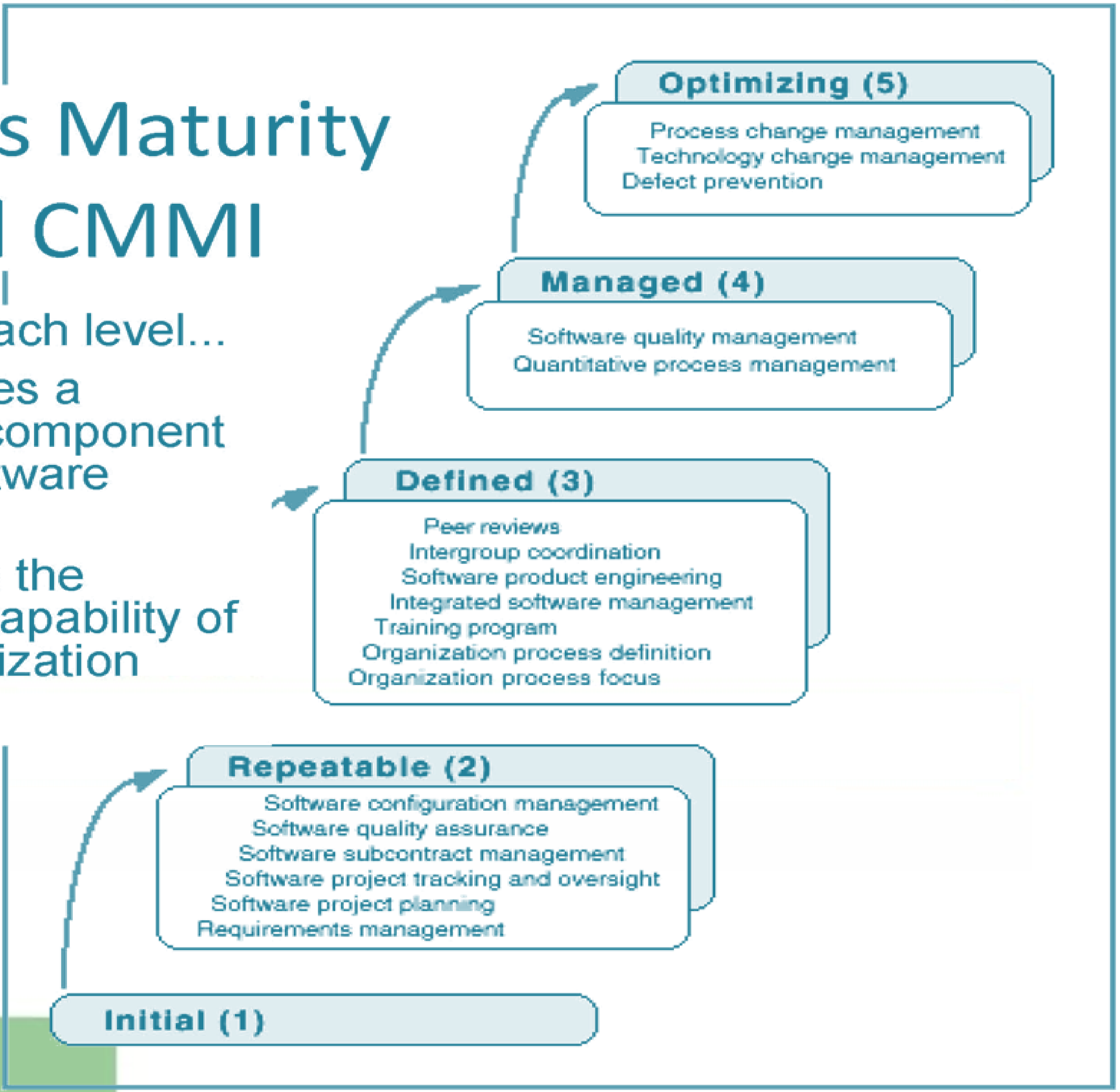
ISO 9000 Quality Management Principles

Capability Maturity Model (CMM)

The Software Engineering Institute (SEI) Capability Maturity Model (CMM) specifies an increasing series of levels of a software development organization. The higher the level, the better the software development process, hence reaching each level is an expensive and time-consuming process.

- It is not a software process model. It is a framework that is used to analyze the approach and techniques followed by any organization to develop software products.
- It also provides guidelines to further enhance the maturity of the process used to develop those software products.
- It is based on profound feedback and development practices adopted by the most successful organizations worldwide.
- This model describes a strategy for software process improvement that should be followed by moving through 5 different levels.
- Each level of maturity shows a process capability level. All the levels except level-1 are further described by Key Process Areas (KPA’s).

Capability Maturity Model (CMM)



Optimizing (5)
Process change management
Technology change management
Defect prevention
Managed (4)
Software quality management
Quantitative process management
Defined (3)
Peer reviews
Intergroup coordination
Software product engineering
Integrated software management
Training program
Organization process definition
Organization process focus
Repeatable (2)
Software configuration management
Software quality assurance
Software subcontract management
Software project tracking and oversight
Software project planning
Requirements management
Initial (1)

Process Maturity and CMMI

- Achieving each level…
  - Establishes a different component in the software process
  - Increases the process capability of the organization

Figure The Key Process Areas by Maturity Level

Capability Maturity Model (CMM)

The entire CMM level is divided into five levels:

Level 1 (Initial): Where requirements for the system are usually uncertain, misunderstood and uncontrolled.

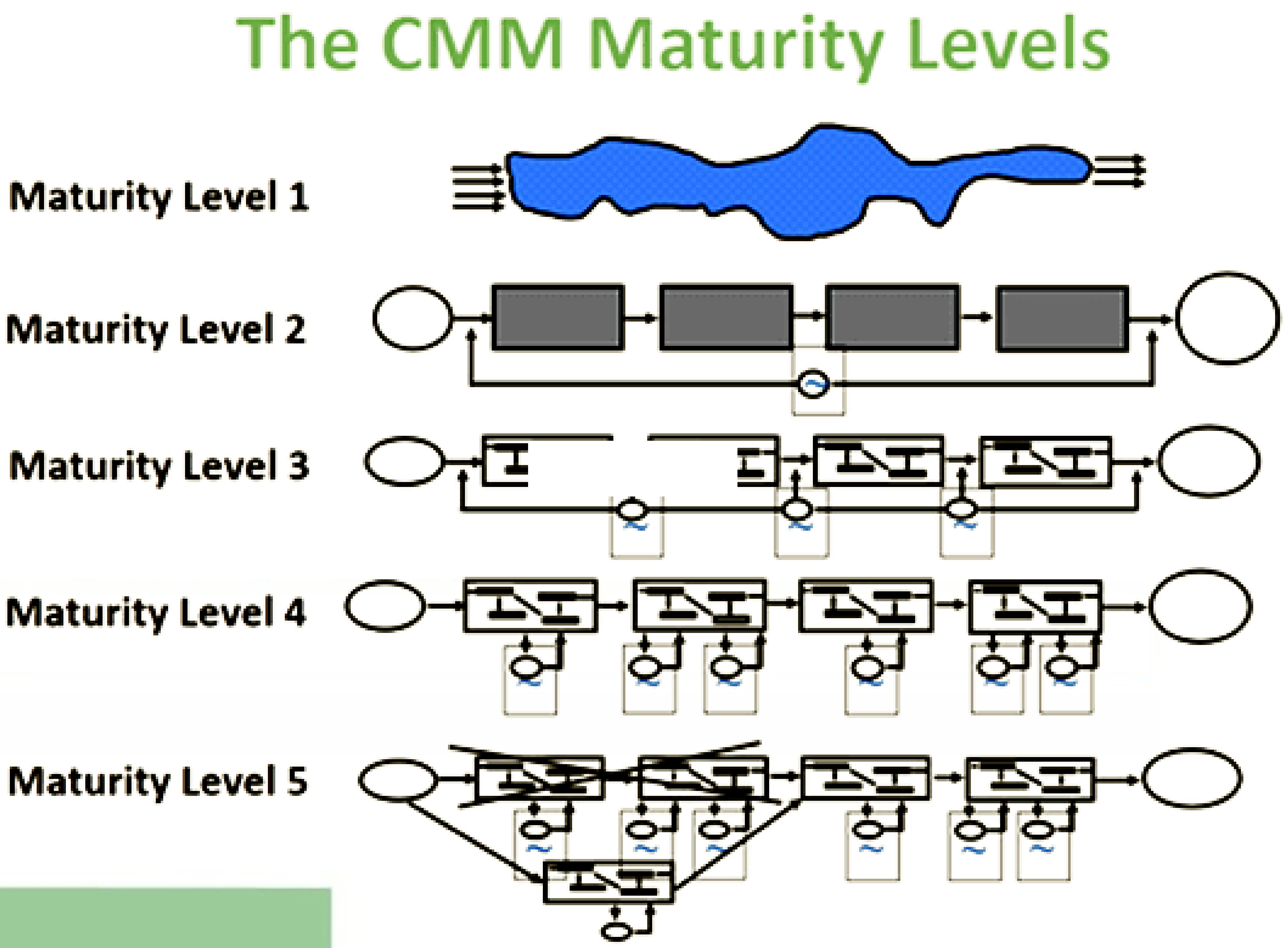
Level 2 (Managed): Estimate project cost, schedule, and functionality. Software standards are defined

Level 3 (Defined): Makes sure that product meets the requirements and intended use

Level 4 (Quantitatively Managed): Manages the project's processes and sub-processes statistically

Level 5 (Maturity): Identify and deploy new tools and process improvements to meet needs and business objectives

Capability Maturity Model (CMM)



Capability Maturity Model (CMM)

Key Process Areas (KPA’s):

Each of these KPA’s defines the basic requirements that should be met by a software process in order to satisfy the KPA and achieve that level of maturity.

Conceptually, key process areas form the basis for management control of the software project and establish a context in which technical methods are applied, work products like models, documents, data, reports, etc. are produced, milestones are established, quality is ensured and change is properly managed.

Reference

1. https://www.techtarget.com/searchsoftwarequality/tip/Efficient-test-case-design-techniques-to-boost-coverage#:~:text=This%20test%20case%20design%20technique%20organizes%20test%20data%20into%20groups,the%20group%20needs%20a%20test.

2. https://www.guru99.com/capability-maturity-model-cmm-cmm-levels-a-fool-s-guide.html