ISTQB Ch 1

TESTING

- Do you test
- What you test
- How you test
- Why you test
- When do you stop testing

Software failures

- https://www.cigniti.com/blog/37-software-failures-inadequate-software-testing/ -need of testing
- Above link listing failures with sources
- https://www.bugraptors.com/blog/top-software-failures-due-to-lack-of-testing
- https://twitter.com/rajaharia/status/1345659284160356352

- Every Software Failure occurs when it doesn't meet desired requirements
- Requirements can be in terms of any Functionality'
- Calculation , Calibration, Security, Resource Management,
- Example

Why to Test Software ?

- Testing is necessary because we all make mistakes.
- Some of those mistakes are unimportant, but some of them are expensive or dangerous.
- We need to check everything that we produce because things can go wrong
- To Err is Human !
- Most important - We should produce QUALITY product !
- What is QUALITY ?

What is Quality ?

- Quality : The degree to which a component, system or process meets specified requirements and/or user/customer needs and expectations.
- Quality is Consistently meeting the
  - Requirements
  - Cost
  - Delivery Schedule
  - Service commitments

Quality is not meeting customer expectations

but

beating customer expectations !

Who is interested in Quality of product ?

We should develop Quality Software !

What is the definition of Software ?

Definition of Software

- **Software :**
  - Computer programs,
  - procedures, associated with
  - documentation and
  - data pertaining to the operation of a computer system. [IEEE 610]

What is Testing ?

- **Testing :**
  - The process consisting of all lifecycle activities, both static and dynamic
    - Concerned with planning, preparation and evaluation of software products and related work products
    - To determine that the software satisfies specified requirements,
    - To demonstrate that it is fit for purpose
    - To detect defects.

TESTING…

- Software testing, when done correctly, can increase overall software quality of conformance by testing that the product conforms to its requirements.
- Testing begins with the software engineer in early stages, but later specialists may be involved in the testing process.
- After generating source code, the software must be tested to uncover and correct as many errors as possible before delivering it to the customer.
- It's important to test software because a customer, after running it many times would finally uncover an error.

SOFTWARE TESTING FUNDAMENTALS

- During software engineering activities, an engineer attempts to build software, where as during testing activities, the intent is to demolish the software.
- Hence testing is psychologically destructive rather constructive.
- A good and successful test is one that uncovers an undiscovered error.
- All the test that are carried out should be traceable to customer requirements at the sometime exhaustive testing is not possible.

What is the Objective of Testing ?

- To check whether the application works as per requirement specifications.
- To find as many defects as possible before the software is delivered
- To provide input for process improvement
- To ensure that best quality product is delivered to customer

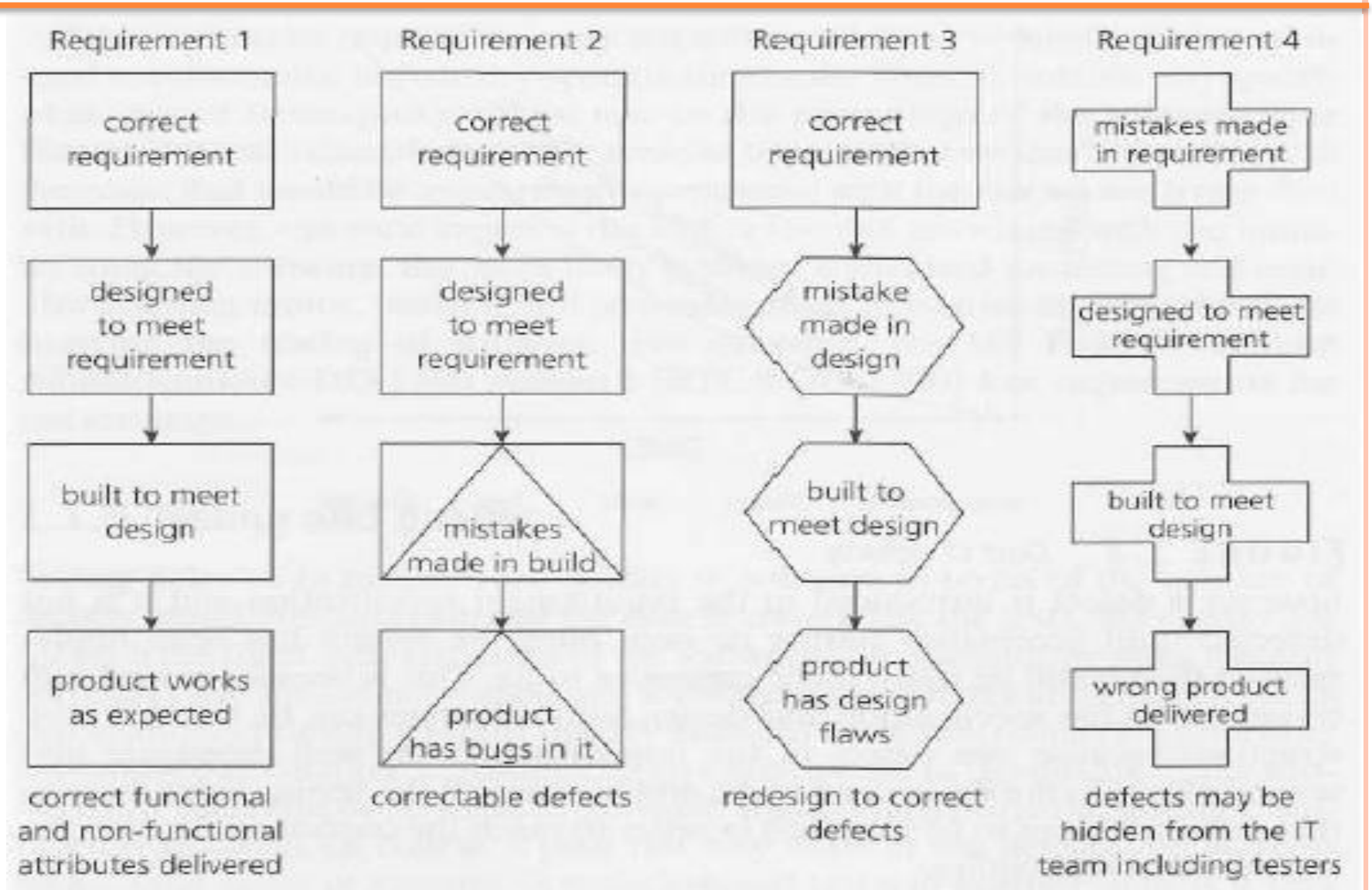
COMMON TERMS:

- **Error**:- difference between the actual output of a software and the correct output.
  - A mistake made by programmer. OR Anything that stops the execution of application.
- **Fault**: is a condition that cause a system to fail in performing its required function.
- **Bug:** A Bug is a Application Error, in Software doesn't affect the whole application in terms of running. It doesn't stop the execution of application. It produces wrong output.
- **Failure:** occurs when a fault is executed. The inability of a system or component to perform its required functions within specified performance requirements.

What is the cause of defect ?

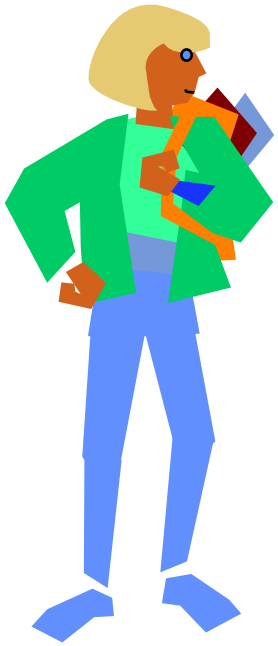
- Errors in the specification
- Errors in design
- Errors in implementation of the software and system
- Complex domain and technology
- Errors in use of the system
- Environmental conditions
- Intentional damage
- Potential consequences of earlier errors, intentional damage, defects and failures.

When Defects are Introduced ?

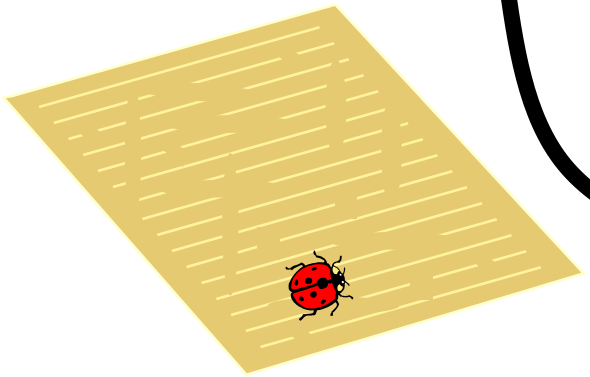


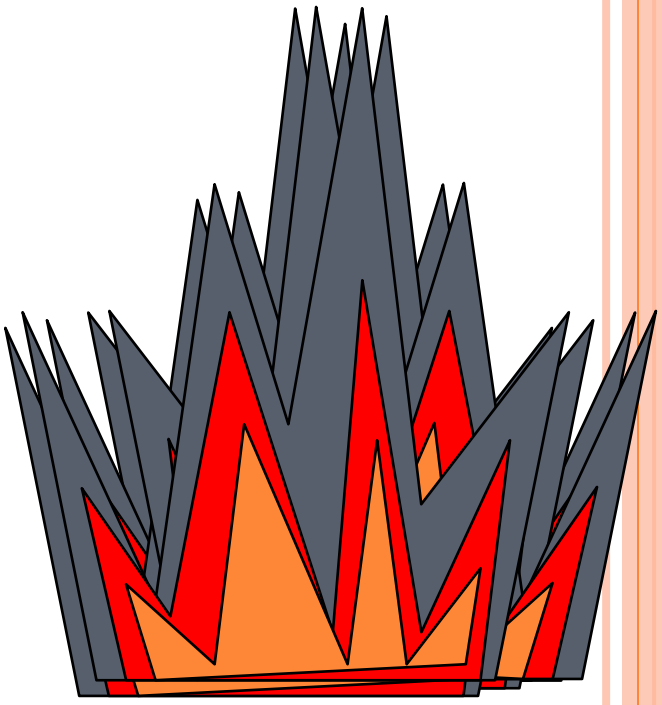
Error - Fault - Failure

A person makes an error ...



... that creates a fault in the software ...





... that can cause a failure in operation

Error Prone Construct

- *Unconditional branch (goto) statements*
- *Floating-point numbers*
- *Pointers*
- *Dynamic memory allocation*
- *Parallelism*
- *Recursion*
- *Interrupts*
- *Inheritance*
- *Aliasing*
- *Unbounded arrays*

Error, Defect and Failure

- **Error** : A human action that produces an incorrect result. [After IEEE 610]

- **Defect (Bug, Fault) :** A flaw in a component or system that can cause the component or system to fail to perform its required function, e.g. an incorrect statement or data definition.

  A defect, if encountered during execution, may cause a failure of the component or system.

- **Failure :** Deviation of the component or system from its expected delivery, service or result

What is the Cost of Defects ?

- ***A stitch in time saves nine !***
- If an error is made at requirement stage and the defect is detected in the requirement stage itself, then it is relatively cheap to find and fix.
- The specifications can be corrected and re-issued.
- If an error is made at requirement stage and the consequent defect is detected at the design stage then both requirements and design should be corrected and re-issued.
- This is more expensive.
- The cost of defect increases if it is not detected early.

Cost of Defects

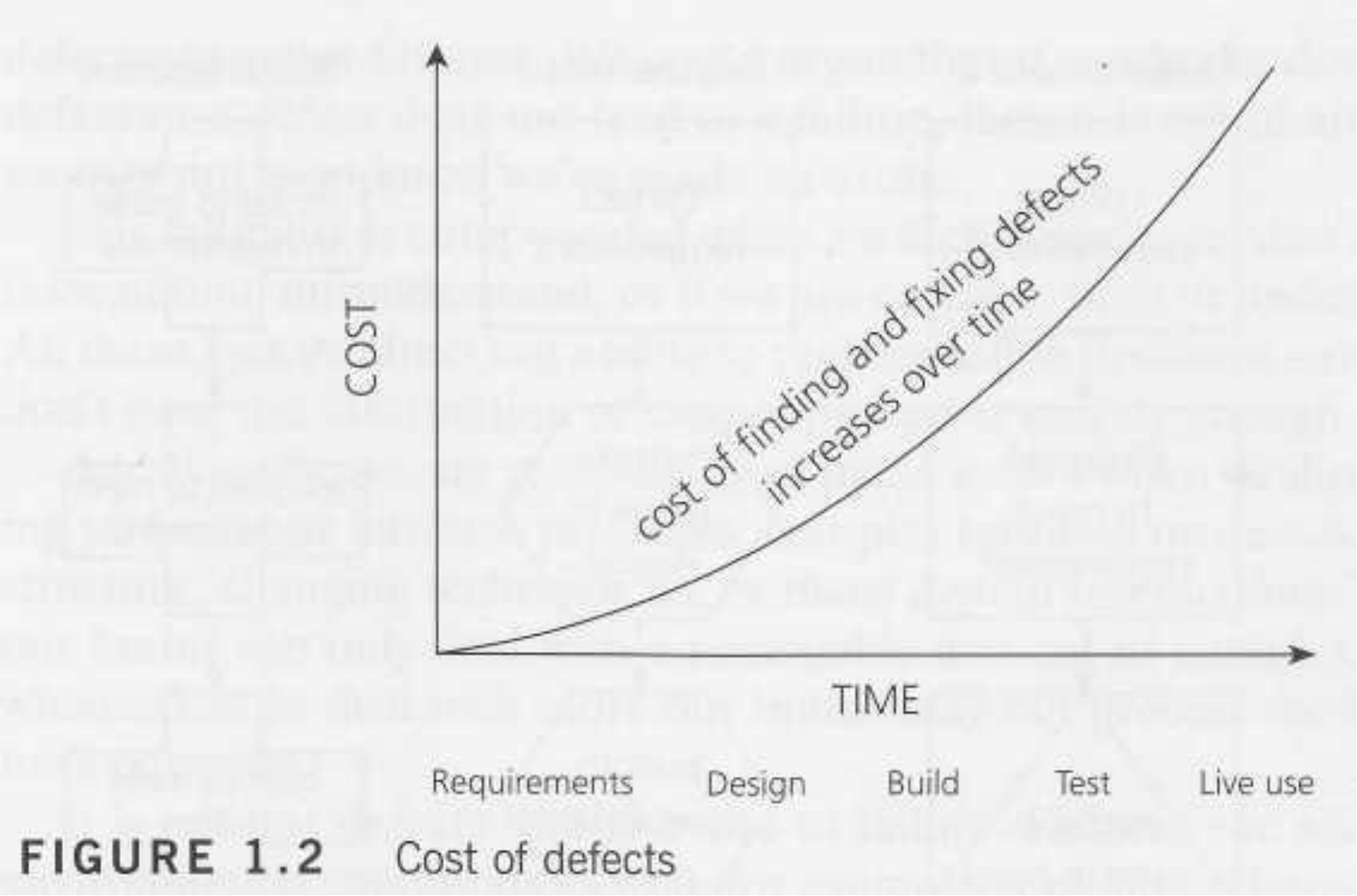


FIGURE 1.2 Cost of defects

Exhaustive Testing

- A test approach in which the test suite comprises of all combinations of input values and preconditions.
- Exhaustive testing requires
  - Infinite time
  - Infinite resources

How much testing is Enough ?

- It's never enough
- When you have done what you planned
- When your customer/user is happy and confident to use software
- When you have proved that the system works correctly
- It depends on the **risks** for your system

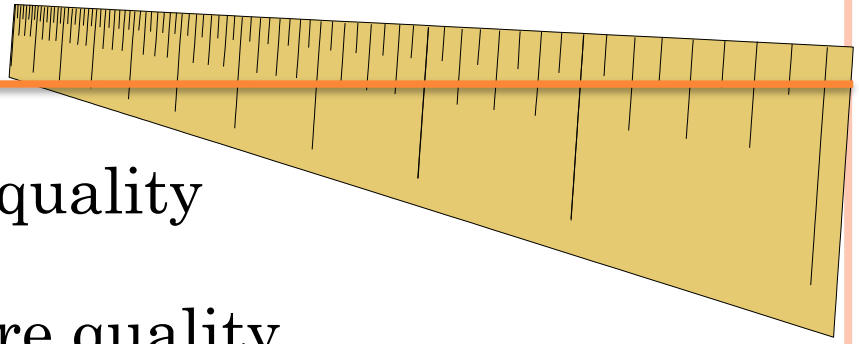
How much testing?

- It depends on **RISK**
- What is RISK ?
  - A factor that could result in future negative consequences
  - Usually expressed as impact and likelihood.
- There can be a risk of
  - missing important functions
  - incurring failure costs … loosing business
  - releasing untested or under-tested software
  - losing credibility, reputation and market

Most important principle

Prioritise tests
so that,
whenever you stop testing,
you have done the best testing
in the time available.

Testing and quality



- Testing measures software quality
- Testing can improve software quality
- What does testing include ?
  - system function, correctness of operation
  - non-functional qualities:
    - reliability,
    - usability,
    - maintainability,
    - reusability,
    - security, etc.

TESTING PRINCIPLES

- "All tests should be traceable to customer requirements." For a customer, the most severe defect are the one that causes the program to fail to meet the requirements.
- "Tests should be planned long before testing begins." All test can be planned and designed before any code has been generated
- "Testing should begin 'in the small' and progress toward testing in the large." The first test planned and executed generally focus on individual component.

TESTING PRINCIPLES

- "As testing progresses focus shifts on integrated clusters of components and then the entire system."
- "Exhaustive testing is not possible." to test exhaustively, even a small program could take years.
- "To be more effective, testing should be conducted by an independent third party.".
  - Software engineer who constricts the software is not always the best person to test it. So a third party deems to be more efficient.

Principles of testing

- **Principle 1 – Testing shows presence of defects**
- **Principle 2 – Exhaustive testing is impossible**
- **Principle 3 – Early testing**
- **Principle 4 – Defect clustering**
- **Principle 5 – Pesticide paradox**
- **Principle 6 – Testing is context dependent**
- **Principle 7 – Absence-of-errors fallacy**

General testing principles

- A number of testing principles have been suggested over the past 40 years and offer general guidelines common for all testing.

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

General testing principles

- **Principle 2 – Exhaustive testing is impossible**
  - Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases. Instead of exhaustive testing, risk analysis and priorities should be used to focus testing efforts.

- **Principle 3 – Early testing**
  - Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.

General testing principles

- **Principle 4 – Defect clustering**
  - A small number of modules contain most of the defects discovered during pre-release testing, or are responsible for the most operational failures.

- **Principle 5 – Pesticide paradox**
  - If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects. To overcome this "pesticide paradox", the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.

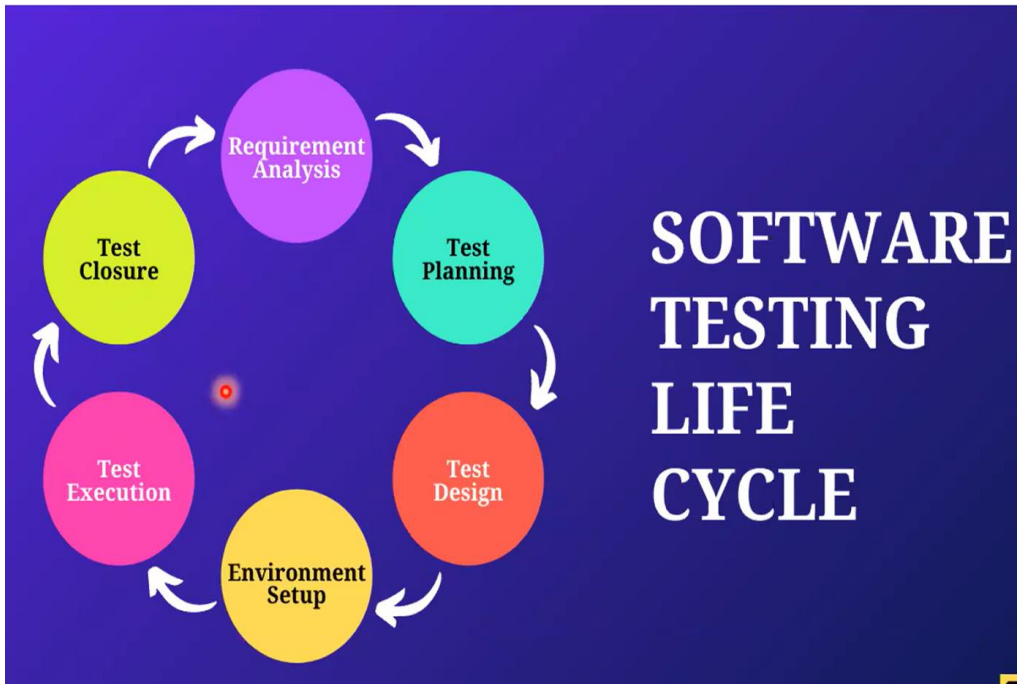
General testing principles

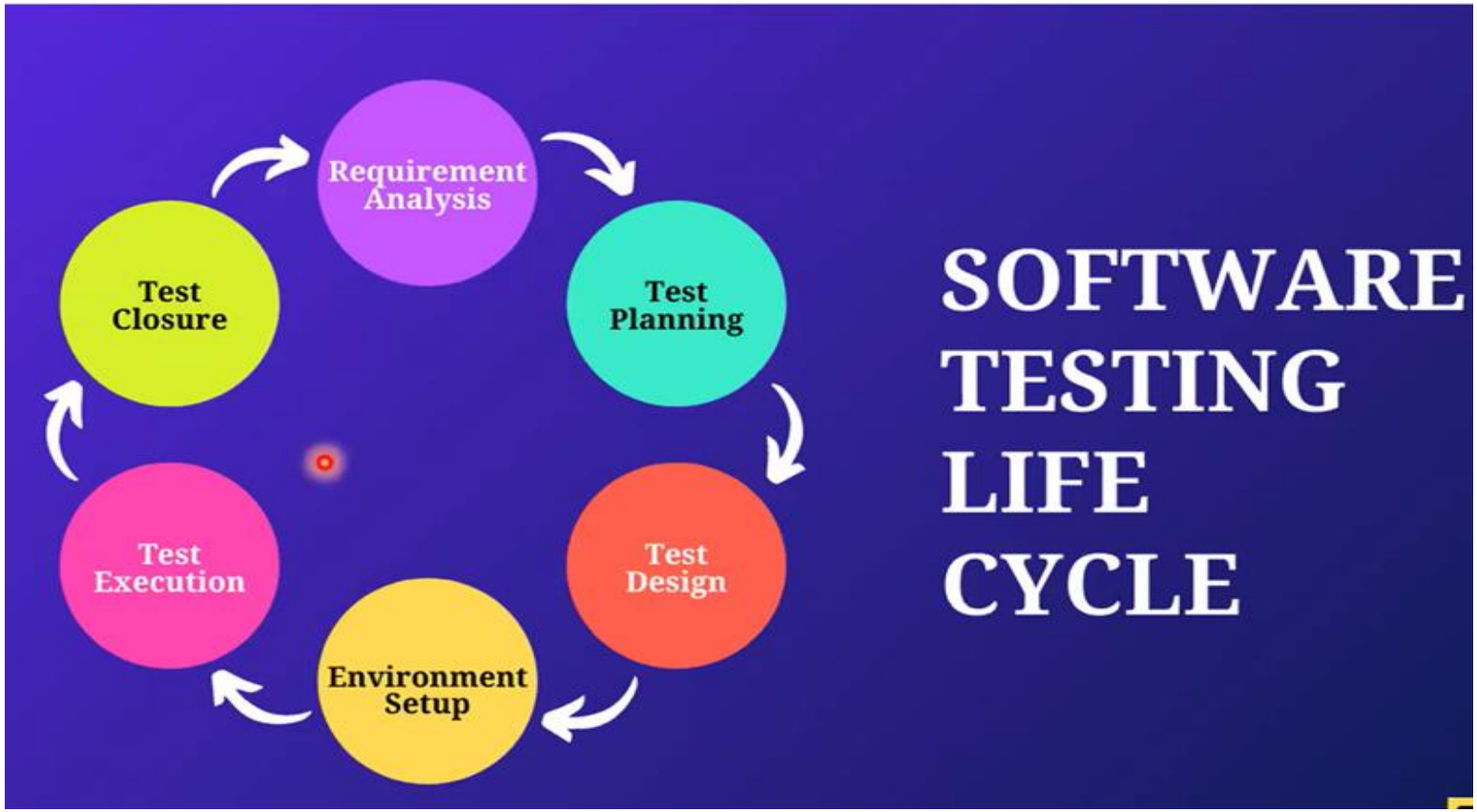
- **Principle 6 – Testing is context dependent**
  - Testing is done differently in different contexts. For example, safety-critical software is tested differently from an e-commerce site.

- **Principle 7 – Absence-of-errors fallacy**
  - Finding and fixing defects does not help if the system built is unusable and does not fulfill the users' needs and expectations.

Fundamental Test Process (Test Life Cycle)

- Test Planning and Control
- Test Analysis and Design
- Test Implementation
- Test Execution
- Defect Reporting
- Test Closure




Requirement Analysis
Test Planning
Test Design
Environment Setup
Test Execution
Test Closure
SOFTWARE TESTING LIFE CYCLE

Who should participate in Testing ?

Developer	→	Unit Testing
Developer / Tester	→	Integration Testing
Tester	→	System Testing
Customer	→	User Acceptance Testing

- ✓ End User
- ✓ Management
- ✓ Auditors
- ✓ Consultants

How long one should continue testing ?

- There is no single criteria
- It depends on many factors
  - Till all the test cases are executed
  - Till all the requirements (specified) are working properly
  - Till budget / time is over
  - Till customer is confident in using the system
- How long one should continue testing depends on the risk factor of application.

Quality Assurance Vs. Quality Control

Quality Assurance

1. **Process Driven Approach :** It is a process to monitor and improve existing quality processes.
2. Quality Assurance ensures that the processes designed for the product development and services are effective enough to meet the objectives.
3. Quality Assurance focuses on defect prevention.

Quality Control

1. **Concerned with Product**: It measures and controls the quality of the software as it is being developed.
2. Quality Control ensures that the final product is error free and satisfactory.
3. Quality Control finds product defects.

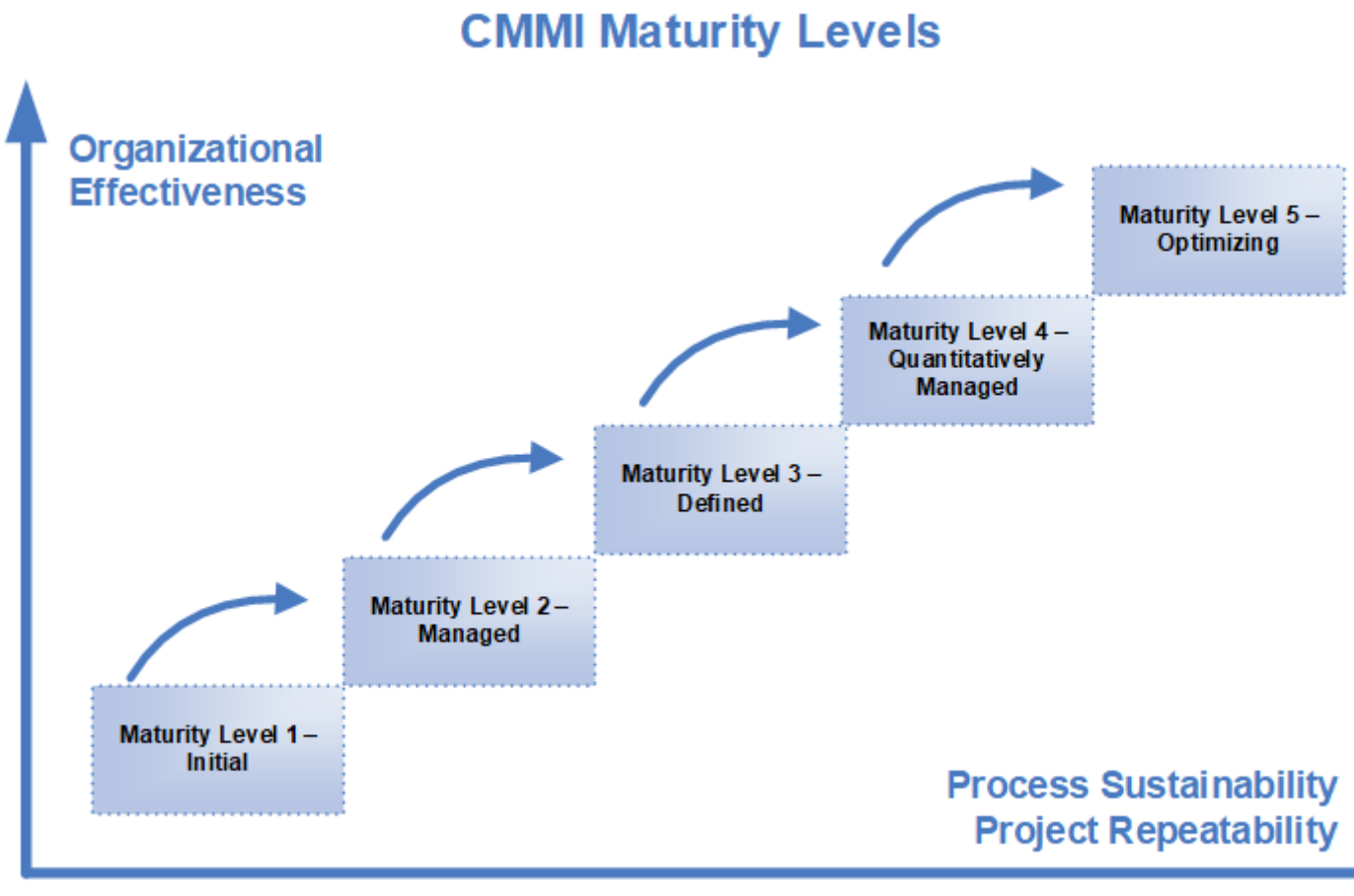
Quality Control Vs Quality Assurance

QC	QA
Operational Issue	Strategic Issue
Correction(Reactive)	Prevention(Proactive)
Find Defects	Prevent Defects

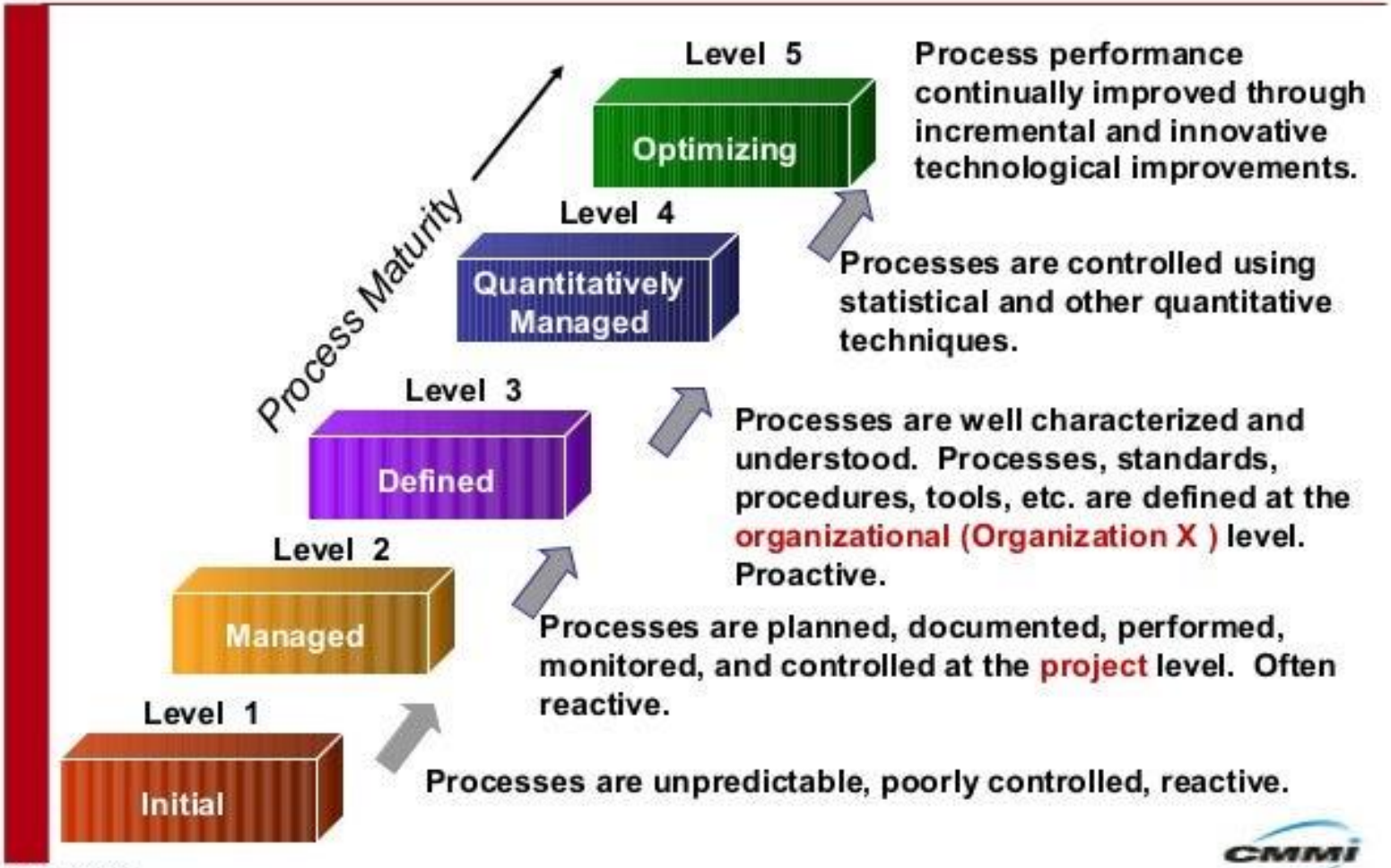
Audits

- **1. ISO 9000 Quality Standards :**
- **https://www.geeksforgeeks.org/iso-standards-in-software-engineering**

- 2. CMMI(Capability Maturity Model Integration)-LEVEL-5
- https://www.99corporates.com/Company-List/CERTIFICATI
- https://www.bmc.com/blogs/cmmi-capability-maturity-model-integration/


CMMI Maturity Levels
Organizational Effectiveness
Maturity Level 1 – Initial
Maturity Level 2 – Managed
Maturity Level 3 – Defined
Maturity Level 4 – Quantitatively Managed
Maturity Level 5 – Optimizing
Process Sustainability
Project Repeatability

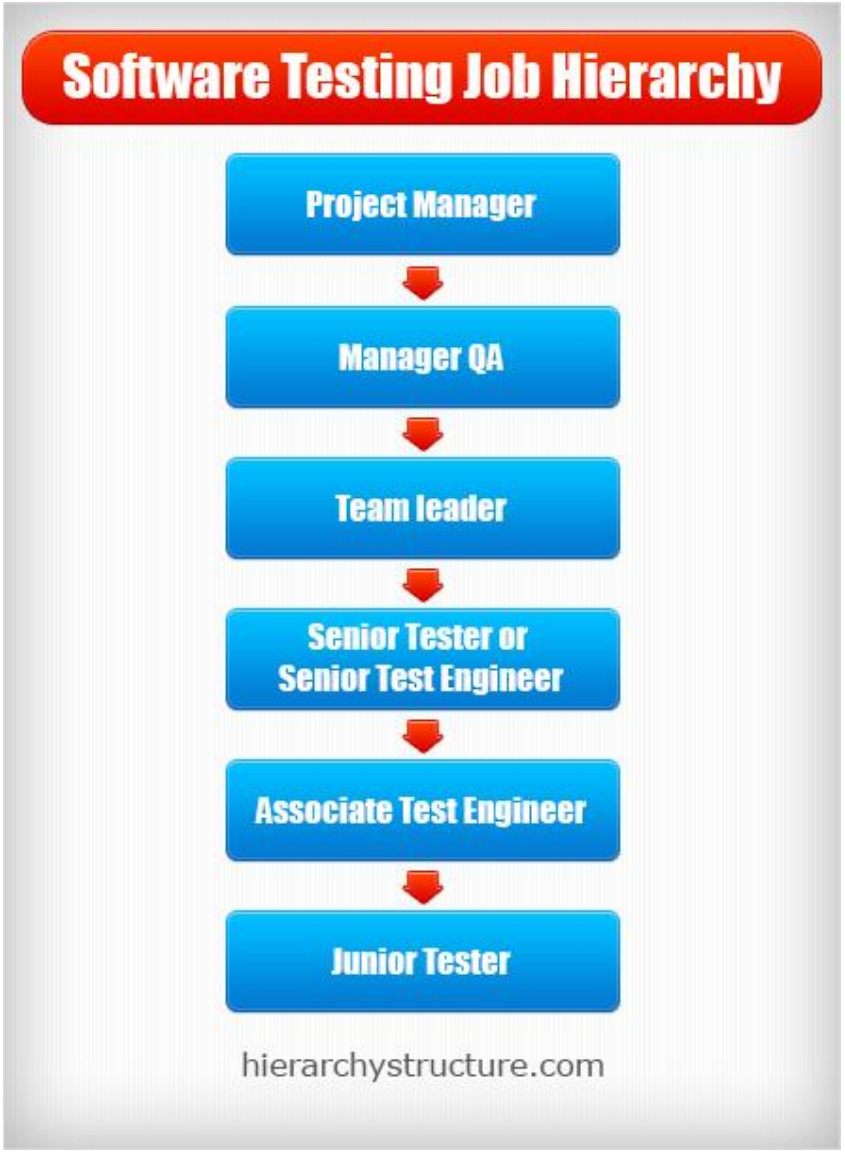
CMMI Staged Representation - 5 Maturity Levels



- This certificates are given to Company Process framework for achieving Quality Sotware Product

CONCLUSION

- What is a good test for you
- What is your strategy for testing
- What is your definition of Quality


Software Testing Job Hierarchy
Project Manager
Manager QA
Team leader
Senior Tester or
Senior Test Engineer
Associate Test Engineer
Junior Tester
hierarchystructure.com