What is Software Testing

Software Testing is the process of identifying the ***correctness and quality*** of software programs. The purpose is to check whether the software satisfies the specific requirements, needs, and expectations of the customer.

Difference between a Bug and a defect

A Bug is the result of a coding Error or 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.

A Defect is a deviation from the Requirements. A Software Defect is a condition in a software product which does not meet a software requirement (as stated in the requirement specifications) or end-user expectations. Defect could arise due to many reasons:- Invalid requirement, coding error,

- A bug is the result of a coding error
- A defect is a deviation from the requirements

Principles of Software Testing:-

Exhaustive Testing not Possible*:-* Software Testing shows presence of Defects, and we can reduce defects in software but end of software testing doesn't guarantee all defects removed from the system as Exhaustive testing is not possible

Early Testing:- It is much cheaper to change an incorrect requirement than having to change a functionality in a large system that is not working as requested or as designed!

Defect Clustering:- If a pattern is showing of defects in a smaller number of module, then focusing on the cluster for testing is advised

Pesticide Paradox:- Test cases to be reviewed and updated time by time

Testing is Context Dependent:- same tests should not be applied across the board because different software products have varying requirements, *For example, a software application in a medical device software needs more testing than a games software.*

Black Box and 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.

Smoke and Sanity Testing

Smoke Testing is a kind of Software Testing performed after software build, to ascertain that the critical functionalities of the program are working fine. It is executed "before" any detailed functional or regression tests are executed on the software build. The purpose is to reject a badly broken application so that the QA team does not waste time installing and testing the software application.

In Smoke Testing, the test cases chose to cover the most important functionality or component of the system.

For Example, a typical smoke test would be - Verify that the application launches successfully, Check that the GUI is responsive ... etc.

Sanity Testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If a sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.

The objective is "not" to verify thoroughly the new functionality but to determine that the developer has applied some rationality (sanity) while producing the software. For instance, if your scientific calculator gives the result of 2 + 2 =5! Then, there is no point testing the advanced functionalities like sin 30 + cos 50.

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

Level of Testing

- Unit
    - Since the bugs are found early in unit testing hence it also helps in reducing the cost of bug fixing.
    - Debugging is easy. When a test fails, only the latest changes need to be debugged.

- Component Integration:-
    - Integration testing may be carried out by the developers, but can be done by a separate team of specialist integration testers, or by a specialist group of developers/integrators including non-functional specialists.
    - Like in the leave management system, after applying the leave table in the database is updated as expected.

    Big Bang Approach
    - Convenient for small systems.
    - Here all component are integrated together at once and then tested.
    - Fault Localization is difficult.
    - Since the Integration testing can commence only after "all" the modules are designed, the testing team will have less time for execution in the testing phase.

    Incremental Approach

    Stubs and Driver are dummy programs

    - Top-down: 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.
    - Bottom-up: testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers

- Functional
- System Integration
  Leave management system should integrate with the billing system i.e. if someone took paid leave, billing should reflect that.
- User Acceptance

Software Testing LifeCycle

https://www.guru99.com/software-testing-life-cycle.html

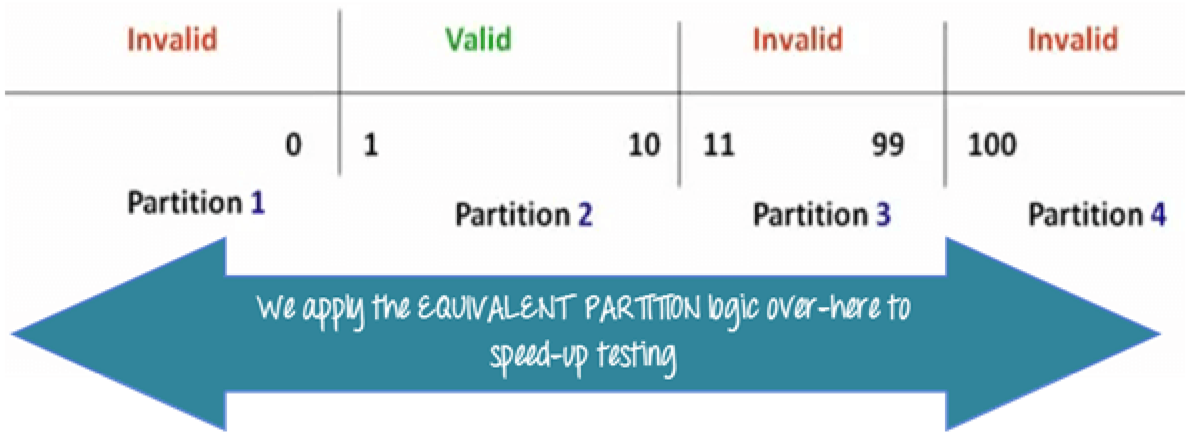
- KT and Requirement Analysis
  Exit Criteria
    - System Understanding Document/ Reverse KT Doc
    - Identified Testable and Non-Testable Requirements
    - Requirements Traceability Matrix:- Components of RTM Requiremnet, Test Scenario, Test case, defect
- Test Planning:-
  Create Test Plan
    - Background of the application
    - Scope of the testing
    - Test Inclusions
    - Test Exclusions
    - Test Approach (Types of Testing)
    - Entry & Exit Criteria of Test Cycle

- Test Environment (Software & Hardware)
- Roles & Responsibilities
- Milestones (Schedule)
- Risk & Risk Mitigation

KEY DIFFERENCE Test Plan is a document that describes the scope, objective and weight on software testing task whereas Test Strategy describes how testing needs to be done. Test Plan is used at the project level whereas Test Strategy is used at the organization level.

Exit Criteria:- Test plan/strategy document, Effort estimation document.

- Test Design
  - Equivalence partition:- The idea behind the technique is to divide (i.e. to partition) a set of test conditions into groups or sets that can be considered the same (i.e. the system should handle them equivalently). The equivalence-partitioning technique then requires that we need test only one condition from each partition.



  - Boundary value Analysis:-
  - Sate Transition Diagram:- **State transition testing** is used where some aspect of the system can be described in what is called a 'finite state machine'. This simply means that the system can be in a (finite) number of different states, and the transitions from one state to another are determined by the rules of the 'machine'. This is the model on which the system and the tests are based. Any system where you get a different output for the same input, depending on what has happened before,

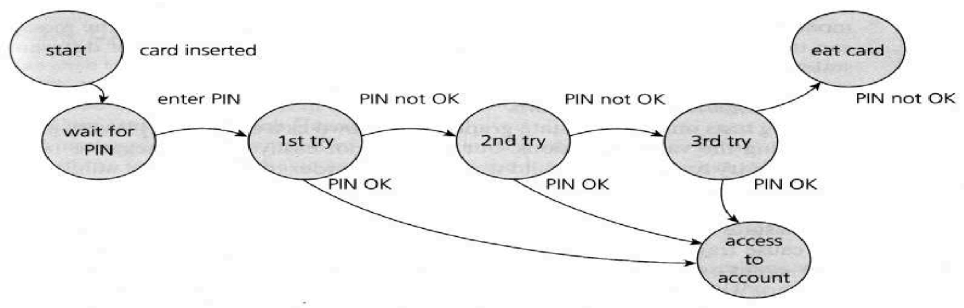


FIGURE 4.2 State diagram for PIN entry

Decision table testing

Why use decision tables?.

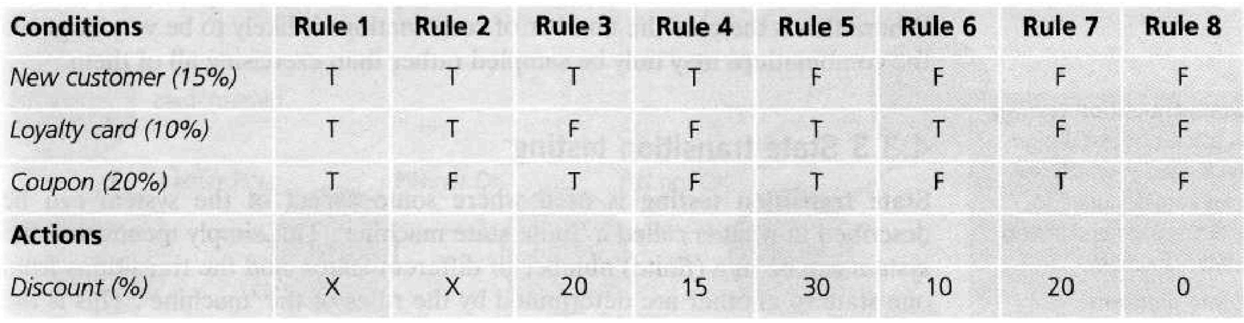
The techniques of equivalence partitioning and boundary value analysis are
often applied to specific situations or inputs. However, if different combinations

of inputs result in different actions being taken, this can be more difficult to show using equivalence partitioning and boundary value analysis, which tend to be more focused on the user interface. The other two specification-based tech-niques, decision tables and state transition testing are more focused on business logic or business rules.

A **decision table** is a good way to deal with combinations of things (e.g. inputs). This technique is sometimes also referred to as a 'cause-effect' table.

Credit card worked example

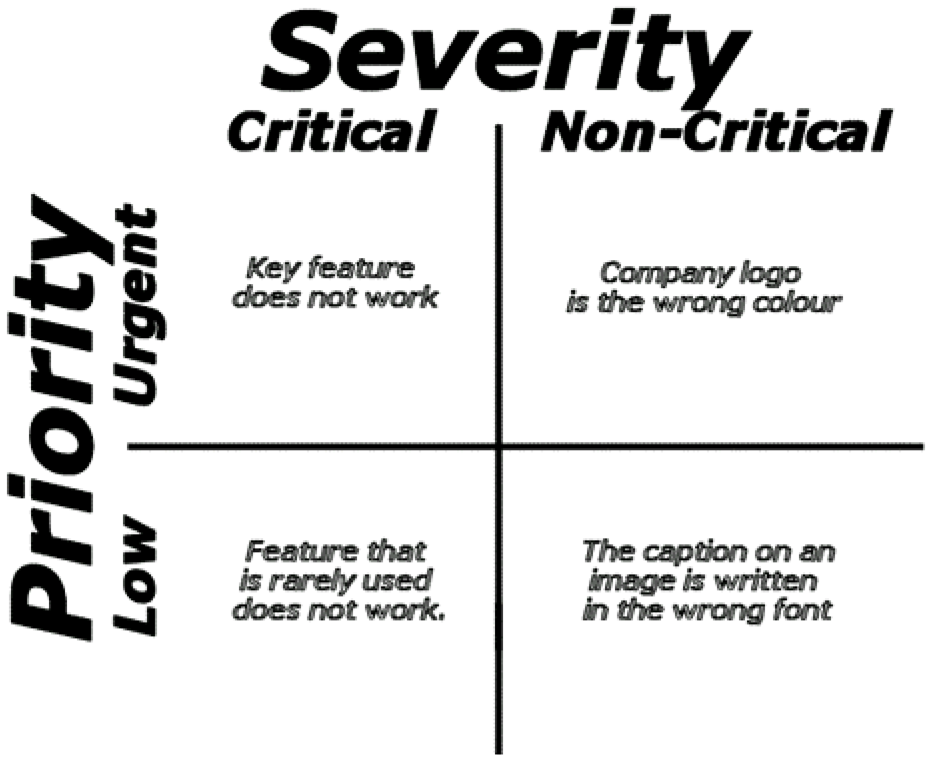
Let's look at another example. If you are a new customer opening a credit card account, you will get a 15% discount on all your purchases today. If you are an existing customer and you hold a loyalty card, you get a 10% discount. If you have a coupon, you can get 20% off today (but it can't be used with the 'new customer' discount). Discount amounts are added, if applicable. This is shown in Table 4.8.



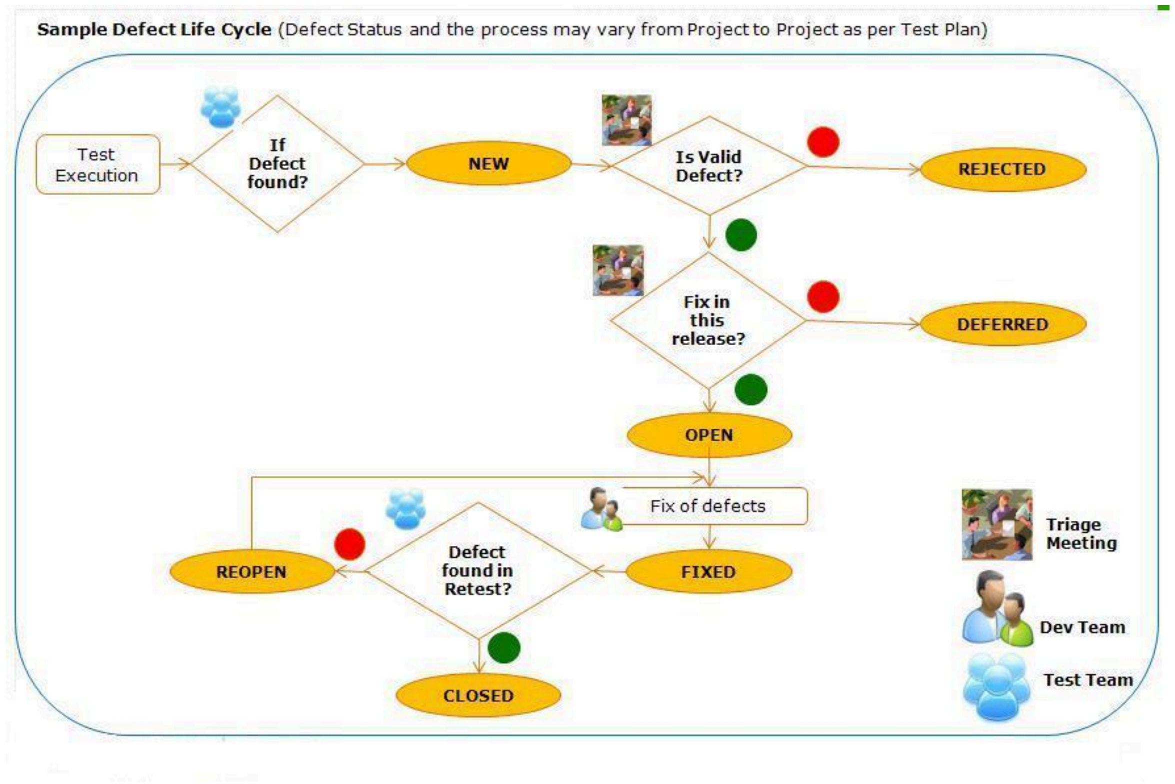
Conditions	**Rule 1**	**Rule 2**	**Rule 3**	**Rule 4**	**Rule 5**	**Rule 6**	**Rule 7**	**Rule 8**
New customer (15%)	T	T	T	T	F	F	F	F
Loyalty card (10%)	T	T	F	F	T	T	F	F
Coupon (20%)	T	F	T	F	T	F	T	F
Actions								
Discount (%)	X	X	20	15	30	10	20	0

- Test Execution
  Defect
  Priority Level:- Low, Medium, High, Urgent
  Severity:- Minor, Major, Critical
  https://www.guru99.com/defect-severity-in-software-testing.html

- Test Closure


Severity
Critical
Non-Critical
Priority
Urgent
Low
Key feature does not work
Company logo is the wrong colour
Feature that is rarely used does not work.
The caption on an image is written in the wrong font

Defect Life Cycle:-



- Test Closure
    - Test Summary Report

- Defect Causal Analysis

  Step #3) Testing Scope, Metrics(Defect Summary, Defects distribution – module wise, Test Case execution), Lessons Learned, Recommendations

  Fishbone diagram

  http://www.onestoptesting.com/testing-interview-questions/manual-testing/details/what-is-fish-bone-diagram-or-explain-ishikawa-diagram-599.asp

Fish Bone Diagram is also called Ishikawa Diagram or Cause and Effect Diagram.
It is called Fish Bone Diagram because of its structure.
Dr. Kaoru Ishikawa invented it so it is called Ishikawa Diagram.
It performs the job of analyzing the causes and their effects pertaining to the Project. So, it is called Cause and Effect Diagram.

It helps in identifying the potential causes of problems and finally the root cause which make a difference on the performance of the project. This helps in finding the solutions to the problems which have affected the performance of the last project.

Pareto Chart

Entry and Exit criteria for each phase

Agile, Scum, Kanban

Agile:- Agile Development is a development process based on iterative development, where requirement and solution evolve over time.

It promotes disciplined project management

https://www.cprime.com/resources/what-is-agile-what-is-scrum/

Scrum:- It is an Agile framework

It works on the philosophy that no matter how complex the problem you got, it can always be broken down into easily manageable smaller chunks. When we manage smaller pieces of work, we have better control, and we can deliver products incrementally with better quality. *Release products and enhancements, as fr*

- ***Sprint Planning** – Meeting to plan what needs to be done in a Sprint*
  Sprint Planning answers the following: • What can be delivered in the Increment resulting from the upcoming Sprint? • How will the work needed to deliver the Increment be achieved?
  The input to this meeting is the Product Backlog, the latest product Increment, projected capacity of the Development Team during the Sprint, and past performance of the Development Team.
  If the Development Team determines it has too much or too little work, it may renegotiate the selected Product Backlog items with the Product Owner.
- ***Daily Scrum** – Meeting to check the progress and find if there are any issues*
- ***Sprint Review** – Meeting at the end of Sprint to make any adjustments to the next Sprint objectives*

The Sprint Review includes the following elements: • Attendees include the Scrum Team and key stakeholders invited by the Product Owner; • The Product Owner explains what Product Backlog items have been “Done” and what has not been “Done”; • The Development Team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved; • The entire group collaborates on what to do next, so that the Sprint Review provides valuable input to subsequent Sprint Planning; • Review of how the marketplace or potential use of the product might have changed what is the most valuable thing to do next; and,

- ***Sprint Retrospective*** *– Meeting to find out what went well and what can be improved in the next Sprints.*

Scrum master:- Accountable remove obstacles, Process of Scrum

The Product Owner is the sole person responsible for managing the Product Backlog. Product Backlog management includes: • Clearly expressing Product Backlog items; • Ordering the items in the Product Backlog to best achieve goals and missions; • Optimizing the value of the work the Development Team performs; • Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next; and, • Ensuring the Development Team understands items in the Product Backlog to the level needed.

The Scrum Master is responsible for promoting and supporting Scrum as defined in the Scrum Guide. Scrum Masters do this by helping everyone understand Scrum theory, practices, rules, and values.

Kanban:- It is an Agile framework