MANUAL TESTING MODULE 1

Session 1
SOFTWARE TESTING

Software testing is the process of ensuring that a developed software is performing its functionalities as expected. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements. Testing is a process of evaluating a system by manual or automatic and verifying that it satisfies that specified requirements or identifying the difference between expected result and actual result.

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

Two ways of Testing

1. Manual Testing

It is the process of manually testing a software application for defects. It requires a tester to play the role of an end user and use most of all the features of the application to ensure correct behaviour. The main goal of manual testing is to make sure that the application under test is defect free.

2.Automation Testing

It is the process of testing a software application by using a software tool. It is the process in which a tool executes pre-scripted tests on a software application before it is released into production.

Why do we need testing?

- Ensure that software is bug free
- Ensure that the system meets customer requirements and software specifications.
- Ensure that the system meets end user expectations.
- Fixing the bugs identified after release, is expensive.

What are the roles and responsibilities of a Tester?

- In the **test planning** and preparation phases of the testing, testers should review and contribute to test plans, as well as analyzing, reviewing and assessing requirements and design specifications.
- They may be involved in or even be the primary people identifying test conditions and **creating test designs**, test cases, test procedure specifications and test data, and may automate or help to automate the tests.
- They often set up the test environments or assist system administration and network management staff in doing so.
- As test execution begins, the number of testers often increases, starting with the work required to implement tests in the test environment.
- Testers execute and log the tests, evaluate the results and document problems found.
- They monitor the testing and the test environment, often using tools for this task, and often gather performance metrics.
- Throughout the **software testing life cycle**, they review each other's work, including test specifications, **defect reports** and test results.

Seven principles of testing

- Testing Shows the presence of defects not their absence : Testing shows the presence of defects in the software. The goal of testing is to make the software fail. Sufficient testing reduces the presence of defects. In case testers are unable to find defects after repeated regression testing doesn’t mean that the software is bug-free. Testing talks about the presence of defects and doesn't talk about the absence of defects.
- Exhaustive Testing is Impossible : Testing everything (i.e. all combinations of input and preconditions) is not feasible for trivial cases.
- Testing should start as early as possible.
- Defect Clustering: In software testing means that a small module or functionality contains most of the bugs or it has the most operational failures.
- Pesticide Paradox: Pesticide Paradox in software testing is the process of repeating the same test cases again and again, eventually, the same test cases will no longer find new bugs. So to overcome this Pesticide Paradox, it is necessary to review the test cases regularly and add or update them to find more defects.
- Testing is more context dependent: Testing approach depends on the context of the software we develop. We do test the software differently in different contexts. For example, an online banking application requires a different approach to testing compared to an e-commerce site.
- Absence of error-Fallacy : Finding and fixing errors does not help if the system built is unusable and does not fulfil the users needs and expectations.

Why Software Testing is important

1. Helps in saving money
2. Security
3. Quality of the product
4. Satisfaction of the customer
5. Enhancing the development process
6. Easy while adding new features
7. Determining the performance of the software

Session 2

MANUAL TESTING

How to do manual testing?

- Analyze requirements from the software requirement specification document
- Create a clear test plan
- Write test cases that cover all the requirements defined in the document
- Get test cases reviewed by the QA lead
- Execute test cases and detect any bugs
- Report bugs, if any, and once fixed, run the failed tests again to re-verify the fixes.

Why we need manual testing

Whenever an application comes into the market, and it is unstable or having a bug or issues or creating a problem while end-users are using it.If we don't want to face these kinds of problems, we need to perform one round of testing to make the application bug free and stable and deliver a quality product to the client, because if the application is bug free, the end-user will use the application more conveniently.If the test engineer does manual testing, he/she can test the application as an end-user perspective and get more familiar with the product, which helps them to write the correct test cases of the application and give the quick feedback of the application.

Advantages

- Human Intelligence
- Adaptability
- Personal Touch
- Early Detection
- Simplicity

Disadvantages

- Time-consuming
- Human Error
- Expensive
- Difficult to Measure

Session 3

Defect

A defect is an undesirable state.

Categories of Defect :

● Extra - An extra requirement in corporate into the product that was not requested by the client.

● Wrong - Requirements have been implemented in the wrong way. This defect is variance from the given specification.

● Missing - A requirement which has been requested by the client but not implemented in the application.

Error It refers to the difference between actual output and expected output.

Or

The mismatch between the application and its specification.

Types:

● *Error*: Mistake made by people ; usually reported by developers.

● *Fault* : It is an incorrect step, process or data definition in a computer program which causes the program to behave in an unintended or unanticipated manner.

● *Bug*: It is a fault in the program which causes the program to behave in an unintended or unanticipated manner reported by engineer(s).

● *Defect* : An undesirable state is an error in coding or logic that causes a program to malfunction or to produce unexpected or incorrect results.

● *Failure* :It is the inability of a system or component to perform a required function, according to its specifications or specified performance requirements. Failure occurs when a fault is executed.

Defect Masking

Defect masking is a terminology in software testing that refers to a scenario where a critical defect goes undetected due to another defect or issue that attracts more attention during the testing process.

Session 4

QUALITY

Quality is defined as meeting the customers requirements first time and every time. Quality is much more than the absence of defects which allows us to meet customers expectations.

Five perspectives of quality

- ❖ Transcendent : I know it when I see it.
- ❖ Product based: Possesses desired features.
- ❖ User based: Fitness for use.
- ❖ Development & manufacturing based : conforms the requirements
- ❖ Value based: At an acceptable cost.

SOFTWARE QUALITY FACTORS (SQF)

SQF -factors/attributes which improve the quality of software product.

➢ ***Correctness*** : extend to which a program satisfies its specification and fullfils-the user's mission objectives.

➢***Reliability*** :extent to which a program can be expected to perform its intended function with required precision.

➢ ***Usability*** : Effort required learning,operating,preparing input and interpreting the output of a program.

➢ **Maintainability : Effort required locationg ande fixing an error in an operational program.**

➢ ***Flexibility*** :Effort required to modifying an operational program.

➢ ***Testability*** :Effort required testing a program to ensure that it performs its intended function.

➢ ***Reusability*** : Extent to which a program can be used in other applications.

➢ ***Interoperability***: Effort required to couple one system with another.

- **Integrity** :Extent to which access to software or data by unautorised persons can be controlled.
- **Efficiency**: The amount of computing resources and code required by a program to perform a function.

Principles of Quality

- Accept Responsibility-by management for the quality of the product.
- Establishment of standard procedures.
- Customer satisfaction.

Session 5

Quality Assurance

QA includes activities that ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. Focuses on processes and procedures rather than conducting actual testing on the system.

Roles of QA

1. To establish a process to fix any proven defect.
2. To establish new processes and monitor existing processes.
3. Define how defects are proactively mitigated.
4. Train the organisation and shareholders on general quality issues.
5. Ensure compliance throughout the product life cycle.

Activities

1. Audting
2. Monitoring
3. Testing

Objectives

1. **Goals**: Focusing on goals verifies that the system achieves the objectives of both the user and total organization.
2. **Methods** : Method verification proves that the structured methodologies are adhered to and the policies,procedures,standards and guidelines that are in place are being followed.
3. **Performance** :is monitored to prove that the most optimal use of computer hardware and software is in place when implementing the application. Quality Control It includes activities that ensure the verification of a developed software with respect to documented (or not in some cases) requirements. Focuses on actual testing by executing the software with an aim to identify bugs/defects through implementation of procedures and processes.

Session 6

Quality Control

QC testing is a function of software quality that checks that the project follows standards, processes, and procedures laid down, and that the project produces the required internal and external deliverables. The QC team's job is to identify defects after a product is developed but before it is released. QC aims to identify (and correct) defects in the finished product.

Quality Testing

The IEEE defines testing as a process of operations under specified conditions, observing or recording the result and making an evaluation of some aspect of a small component.

QT=VERIFICATION+VALIDATION

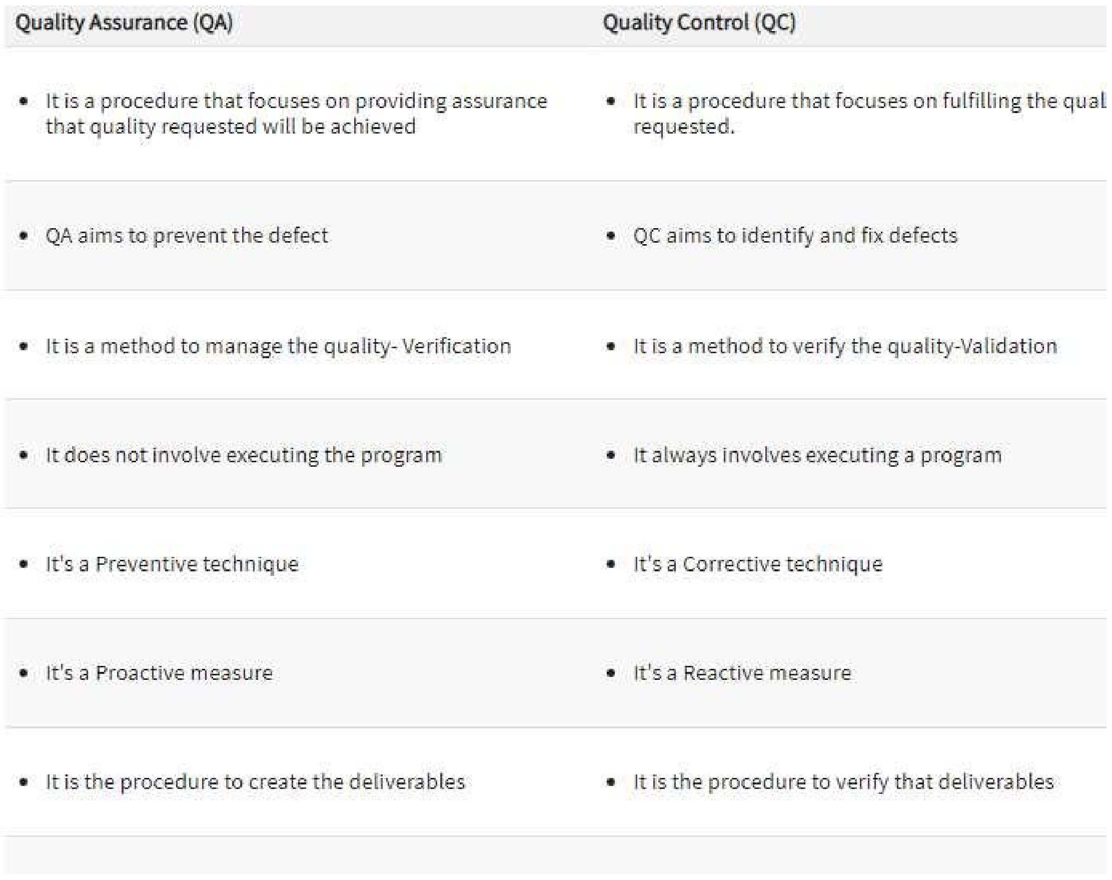
Verification: The process of evaluating a s/m or component or determining whether the product as a given development phase satisfies the conditions imposed and status of the phase.

ARE WE BUILD THE PRODUCT RIGHT? -Ensure that the software is developed in the right manner.

Validation: The process of evaluating a s/m or component during or at the end of the development process to determine whether it satisfies specified requirements

ARE WE BUILD THE RIGHT PDT? -Ensure you build one right things or product

Difference between QA & QC



Quality Assurance (QA)	Quality Control (QC)
• It is a procedure that focuses on providing assurance that quality requested will be achieved	• It is a procedure that focuses on fulfilling the qual requested.
• QA aims to prevent the defect	• QC aims to identify and fix defects
• It is a method to manage the quality- Verification	• It is a method to verify the quality-Validation
• It does not involve executing the program	• It always involves executing a program
• It's a Preventive technique	• It's a Corrective technique
• It's a Proactive measure	• It's a Reactive measure
• It is the procedure to create the deliverables	• It is the procedure to verify that deliverables

Session 7

COST OF QUALITY

All the costs that occur beyond one cost of producing the product right at the first time.

1. *Prevention cost*

Money required to prevent errors and to do the job right the first time .The category includes money spent on establishing methods and producers, training workers, acquiring tools and planning or quality. Preventing money is all spent before the product is actually built.

2.Appraisal cost

Money spent to review completed products against requirement appraisal includes the cost of inspections, testing and reviews. The money spent after the product is built but before it is shipped to the user.

2. Failure cost

This cost is associated with defective products that have been delivered to the user or moved into production. Some failure costs involve repairing products to meet these requirements..

1.Internal Failure Cost

2.External Failure Cost

Internal failure cost Quality cost associated with defects discovered before the product has been delivered to customer or moved into production .These internal costs are detected through the firm's inspection and appraisal activities and may include cost of rework scrapping of rejected products downtime caused by quality problems.

Alpha testing Customer tests the application at developers site with/without the presence of a tester .It implies an initial meeting between software vendor and client to ensure that client's requirements are properly met by the developer in terms of performance ,functionality and durability of the software program.

Beta testing

>takes place at customer's site

>also known as yield testing

>its goal is to place the application in the hands of real users to discover any flows and issues from the user's perspective that would not want to have in the final released version of the application.

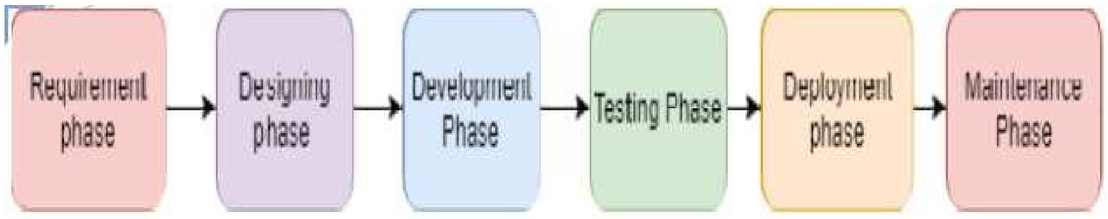
External failure cost

External failure cost arise from the rejection of products or services by the customers due to poor quality of the product shipped .liability from legal actions /penalties repairs and replacement warranty or work product recall loss of market share loss of business.

Session 8

SDLC [Software Development Life Cycle]

SDLC is a process that creates a structure of software development. There are different phases within SDLC, and each phase has its various activities. It makes the development team able to design, create, and deliver a high-quality product. SDLC describes various phases of software development and the order of execution of phases. Each phase requires deliverables from the previous phase in a life cycle of software development. Requirements are translated into design, design into development and development into testing; after testing, it is given to the client.



1. Requirement Phase This is the most crucial phase of the software development life cycle for the developing team as well as for the project states requirements, specifications, expectations, and any other special requirement related to the product or software. All these are gathered by the business manager or project manager or analyst of the service pro The requirement includes how the product will be used and who will use the product to determine the load of operations. All information gathered from this phase is critical to developing the product as per the customer requirements.

3. Design Phase The design phase includes a detailed analysis of new software according to the requirement phase. This is the high priority phase in the development life cycle of a system because the logical designing of the system is converted into physic designing. The output of the requirement phase is a collection of things that are required, and the design phase gives the way to accomplish these requirements.

3. Build /Development Phase After the successful completion of the requirement and design phase, the next step is to implement the design into the development of a software system. In this phase, work is divided into small units, and coding starts by the team of developers according to the design discussed in the previous phase and according to the requirements of the client discussed in the requirement phase to produce the desired result. Front-end developers develop easy and attractive GUI and necessary interfaces to interact with back-end operations and back-end developers do back-end coding according to the required operations. All is done according to the procedure and guidelines demonstrated by

the project manager. Since this is the coding phase, it takes the longest time and more focused approach for the developer in the software development life cycle.

4. Testing Phase Testing is the last step of completing a software system. In this phase, after getting the developed GUI and back-end combination, it is tested against the requirements stated in the requirement phase. Testing determines whether the software is actually giving the result as per the requirements addressed in the requirement phase or not. 5.

5.Deployment/ Deliver Phase

When software testing is completed with a satisfying result, and there are no remaining issues in the working of the software, it is delivered to the customer for their use. As soon as customers receive the product, they are recommended first to do the beta testing. In beta testing, customers can require any changes which are not present in the software but mentioned in the requirement document or any other GUI changes to make it more userfriendly. Besides this, if any type of defect is encountered while a customer is using the software; it will be informed to the development team of that particular software to sort out the problem. If it is a severe issue, then the development team solves it in a short time; Otherwise, if it is less severe, then it will wait for the next version. After the solution of all types of bugs and changes, the software finally deployed to the end-user.

6. Maintenance

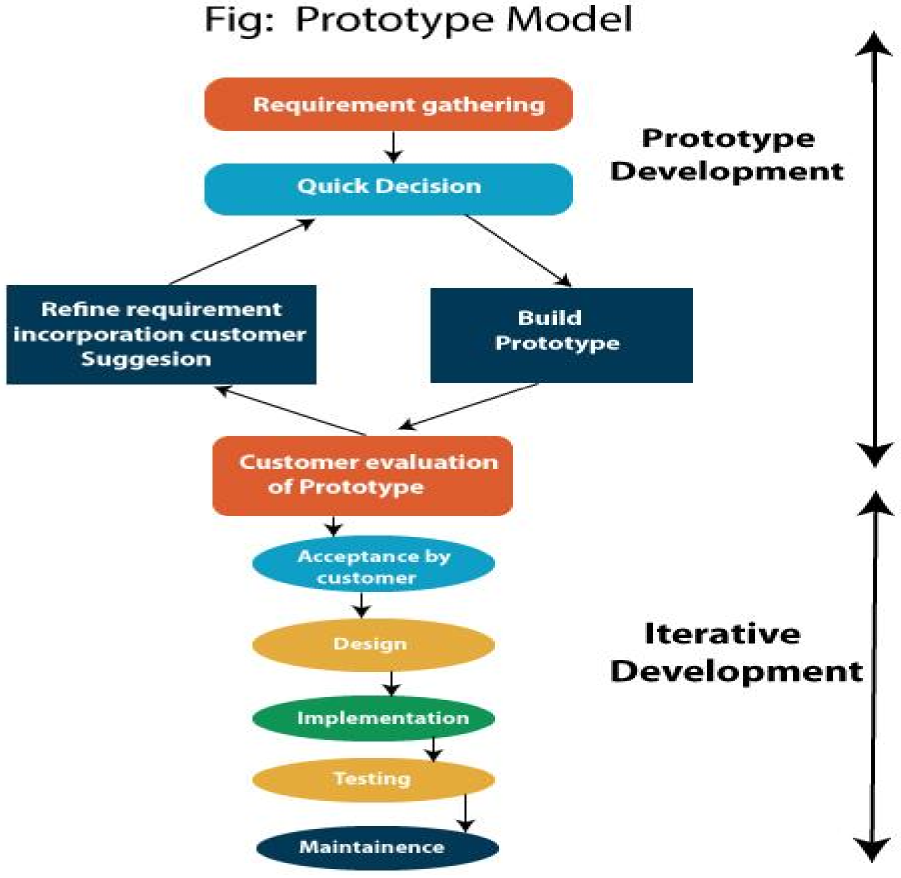
The maintenance phase is the last and long-lasting phase of SDLC because it is the process which continues until the software's life cycle comes to an end. When a customer starts using software, then actual problems start to occur, and at that time there's a need to solve these problems. This phase also includes making changes in hardware and software to maintain its operational effectiveness like to improve performance, enhance security features and according to customer's requirements with upcoming time. This process to take care of the product from time to time is called maintenance.

SDLC MODELS

Prototype Model

- A prototype of the application is created to understand the requirements.
- This prototype is developed based on the currently known requirements.
- The prototype is then shown to the customer to get the feedback. This helps in better requirement understanding.
- From the prototype we learn and try to build a better final product.

Fig: Prototype Model



Steps of Prototype Model

1. Requirement Gathering and Analyst
2. Quick Decision
3. Build a Prototype
4. Assessment or User Evaluation
5. Prototype Refinement
6. Engineer Product

Advantage of Prototype Model

1. Reduce the risk of incorrect user requirement
2. Good where requirement are changing/uncommitted
3. Regular visible process aids management
4. Support early product marketing
5. Reduce Maintenance cost.
6. Errors can be detected much earlier as the system is made side by side.

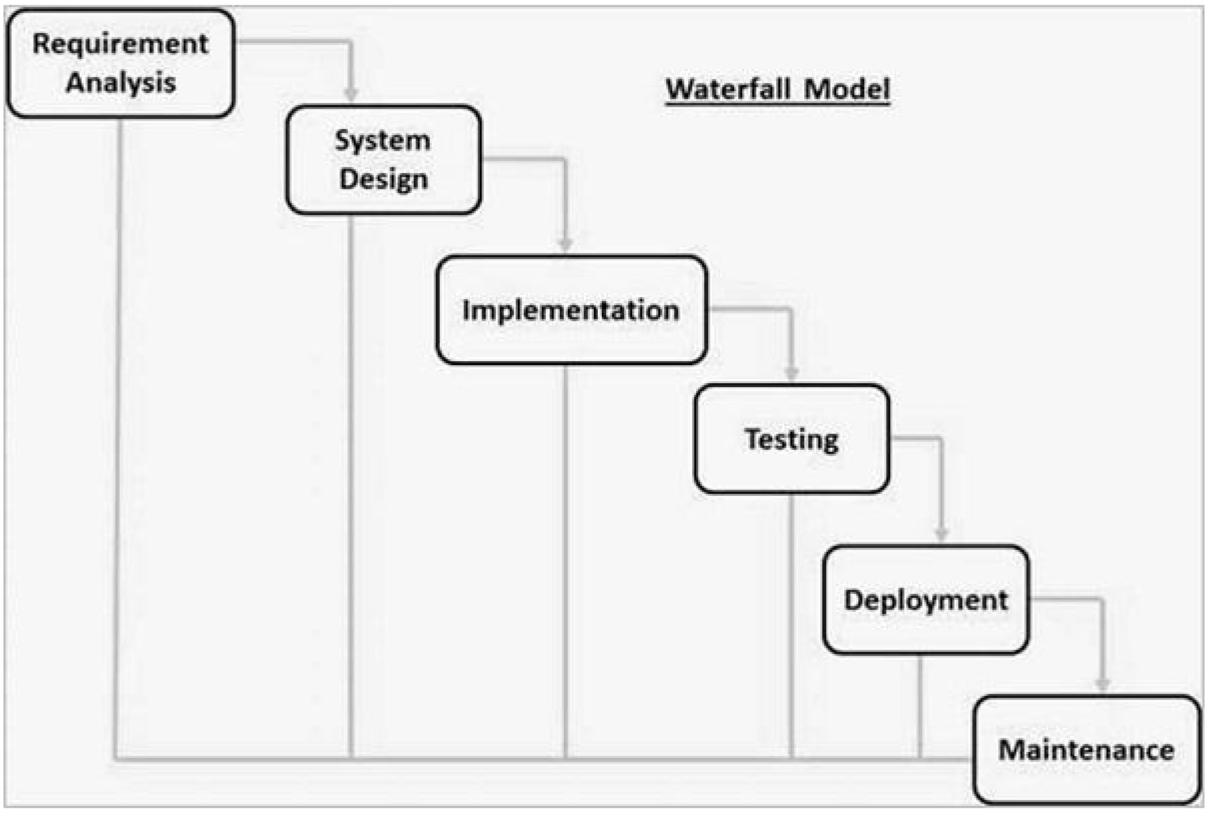
Disadvantage of Prototype Model

1. An unstable/badly implemented prototype often becomes the final product.
2. Require extensive customer collaboration
    - Costs customer money
    - Needs committed customer
    - Difficult to finish if customer withdraw
    - May be too customer specific, no broad market

3. Difficult to know how long the project will last.
4. Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback.
5. Prototyping tools are expensive.
6. Special tools & techniques are required to build a prototype.
7. It is a time-consuming process.

Waterfall Model

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.



Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Requirement Gathering and analysis − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

System Design − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

Implementation − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

Integration and Testing − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

Deployment of system − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

Maintenance − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Advantages

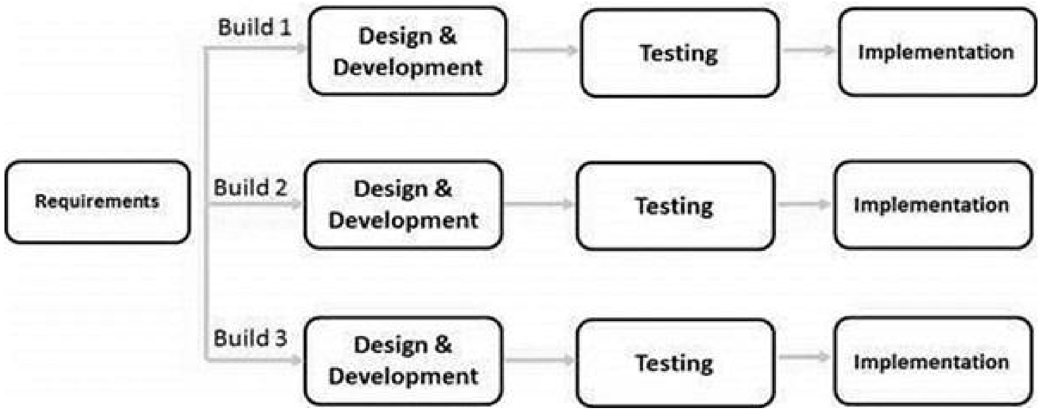
- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Disadvantages

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.

Iterative Model

In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.



Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."

Advantages

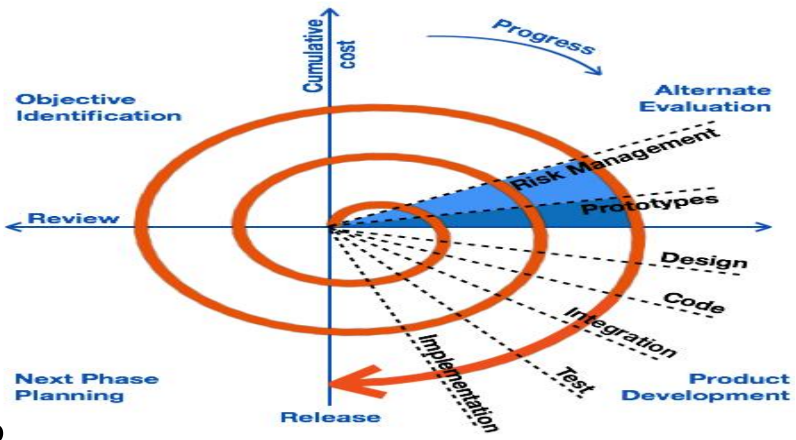
- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.

Disadvantages

- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which is a risk.
- Highly skilled resources are required for risk analysis.
- Projects progress is highly dependent upon the risk analysis phase.

Spiral Model

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the



sp iral.

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

Identification

This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

Design

The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

Construct or Build

The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

Evaluation and Risk Analysis

Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

Hybrid Model

The hybrid model is the combination of two or more primary (traditional) models and modifies them as per the business requirements. This model is dependent on the other SDLC models, such as spiral, V and V, and prototype models. The hybrid model is mainly used for small, medium, and large projects. It focuses on the risk management of the product.We go for the hybrid model whenever we want to obtain the features of two models in a single model. And when the model is dependent and the customer is new to the industry.The most commonly used combination of two models is as follows:

- **Spiral and prototype**
- **V & V and Prototype**

Advantages

- The hybrid model is highly flexible.
- In this model, the customer rejection is less because of the Prototype.
- It is easy to implement because it has the flexibility of synchronization.
- It is easy to use and apply, especially with small and medium projects.

- In this, the development process will be smooth and quick because here we follow only the relevant process cycles.Disadvantages

- Every hybrid model is different from each other.
- It does not follow the usual standards.

Incremental Model

Incremental Model is a process of software development where requirements are divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system is achieved.

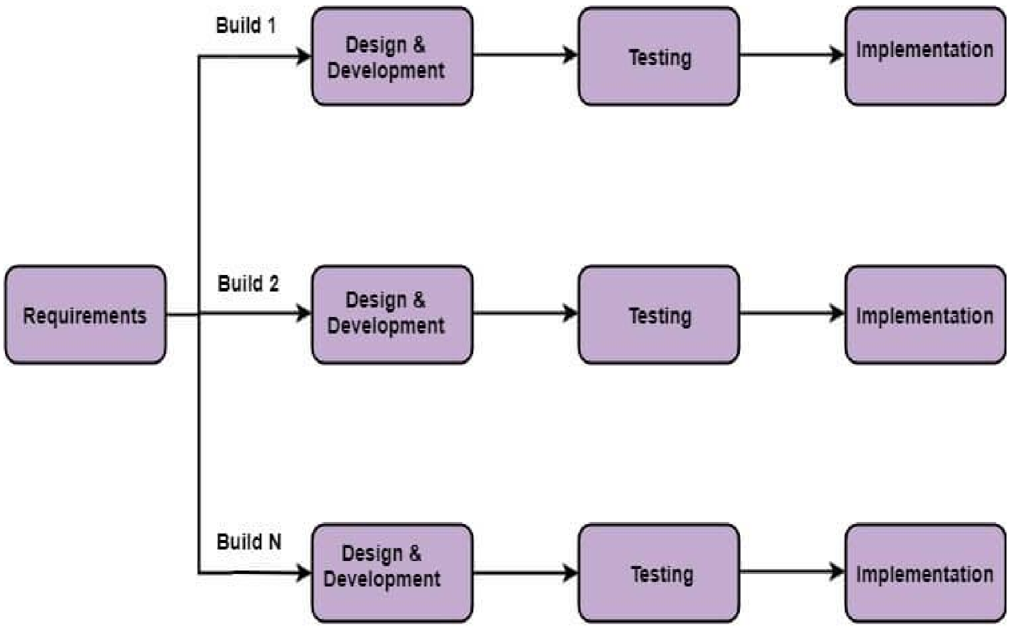


Fig: Incremental Model

Th e various phases of incremental model are as follows:

1. **Requirement analysis:** In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement

analysis team. To develop the software under the incremental model, this phase performs a crucial role.

2. Design & Development: In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

3. Testing: In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

4. Implementation: Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

- When we use the Incremental Model?

- When the requirements are superior.
- A project has a lengthy development schedule.
- When Software team are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.
- Advantage of Incremental Model

- Errors are easy to be recognized.
- Easier to test and debug

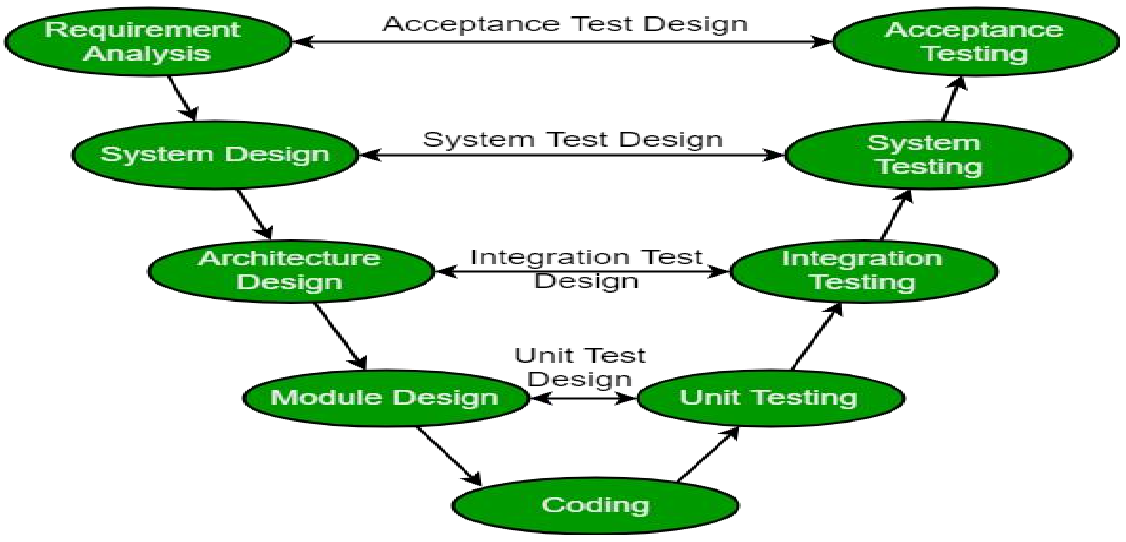
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

Disadvantage of Incremental Model

- Need for good planning
- Total Cost is high.
- Well defined module interfaces are needed.

V-Model

- V Model is also known as the Verification and Validation model.
- In this model, the testing phase goes in parallel with the development phase.
- For every single phase in the development cycle there is a directly associated testing phase.



Advantages

- Testing activities like planning, designing happens well before coding, this saves time.
- Simple and easy to understand and use.
- Works well for smaller projects where requirements are very well understood.

Disadvantages

- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.
- No working software is produced until late during the life cycle.

Session 9

Agile Methodology

Agile sdlc method is a combination of iterative and incremental process model with focus on process adaptability and customer satisfaction by rapid delivery of working software product
agile methods breaks the product into small incremental builds .Every interaction involves cross functional teams working simultaneously or various area like ,
1.planning
2.requirement analysis
3.design
4.coding
5.unit testing

6.acceptance testing

Agile manifesto principles

> Individuals and interactions

> working software

> customer collaboration

> responding to change

Advantages

> Customer satisfaction by rapid ,continuous delivery of useful software

> Face to face conversation is best form of communication

> Continues attention to technical excellence is good design

> Regular adaptation to changing circumstances

> Even late changes in requirements are welcomed

12Soften Technologies

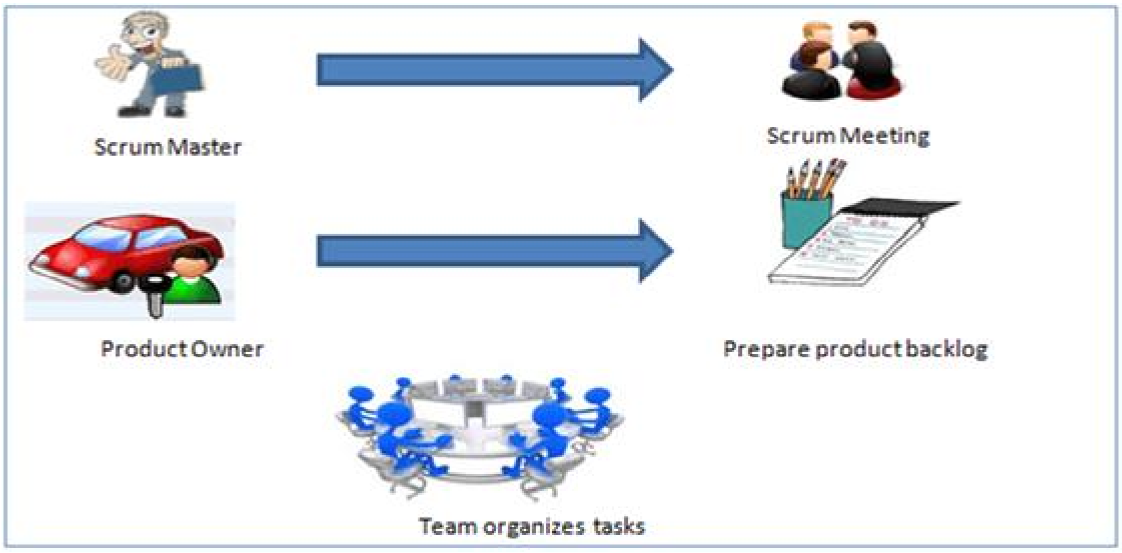
12 principles agile methodology

1. Welcome change
2. Satisfy the customer
3. Deliver frequently
4. Work together
5. Build projects
6. Face to face time
7. Measure of progress
8. Sustainable development
9. Continuous attention
10. Keep it simple
11. Organised team
12. Reflect for effectiveness

Scrum Methodology

Scrum

SCRUM is an agile development method which concentrates specifically on how to manage tasks within a team-based development environment. Basically, Scrum is derived from activity that occurs during a rugby match. Scrum believes in empowering the development team and advocates working in small teams (say- 7 to 9 members). Agile and Scrum consist of three roles, and their responsibilities are explained as follows:



Scrum Method

Scrum Master

Scrum Master is responsible for setting up the team, sprint meeting and removes obstacles to progress

Product owner

The Product Owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration

Scrum Team

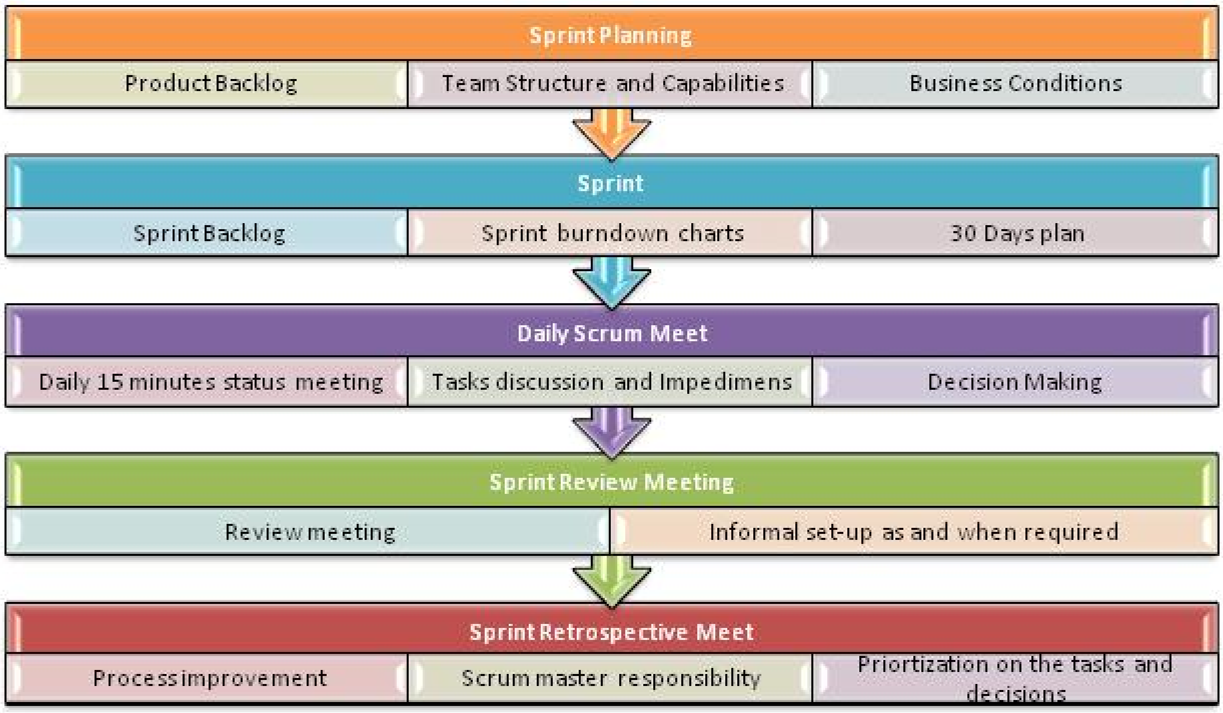
Team manages its own work and organizes the work to complete the sprint or cycle

Product Backlog

This is a repository where requirements are tracked with details on the no of requirements(user stories) to be completed for each release. It should be maintained and prioritized by Product Owner, and it should be distributed to the scrum team. Team can also request for a new requirement addition or modification or deletion

Scrum Practices

Practices are described in detailed:


Sprint Planning
Product Backlog
Team Structure and Capabilities
Business Conditions
Sprint
Sprint Backlog
Sprint burndown charts
30 Days plan
Daily Scrum Meet
Daily 15 minutes status meeting
Tasks discussion and Impedimens
Decision Making
Sprint Review Meeting
Review meeting
Informal set-up as and when required
Sprint Retrospective Meet
Process improvement
Scrum master responsibility
Priortization on the tasks and decisions

Scrum Practices

Process flow of Scrum Methodologies:

Process flow of scrum testing is as follows:

- Each iteration of a scrum is known as Sprint

- Product backlog is a list where all details are entered to get the end-product
- During each Sprint, top user stories of Product backlog are selected and turned into Sprint backlog
- Team works on the defined sprint backlog
- Team checks for the daily work
- At the end of the sprint, team delivers product functionality

What is Agile Testing?

Agile Testing is a testing practice that follows the rules and principles of agile software development. Unlike the Waterfall method, Agile Testing can begin at the start of the project with continuous integration between development and testing. Agile Testing methodology is not sequential (in the sense it’s executed only after coding phase) but continuous.

Principles of Agile Testing

Here are the essential Principles of Agile Testing:

- In this Agile testing model, working software is the primary measure of progress.
- The best result can be achieved by the self-organizing teams.
- Delivering valuable software early and continuously is our highest priority.
- Software developers must act to gather daily throughout the project.
- Enhancing agility through continuous technical improvement and good design.
- Agile Testing ensures that the final product meets the business’s expectations by providing continual feedback.

Session 10

PROCESS

A process can be defined as a set of activities that represents the way work is performed.

The outcome from a process is usually a product or service. These are two ways to visually
portray a process:

a) PDCA Cycle

b) Workbench

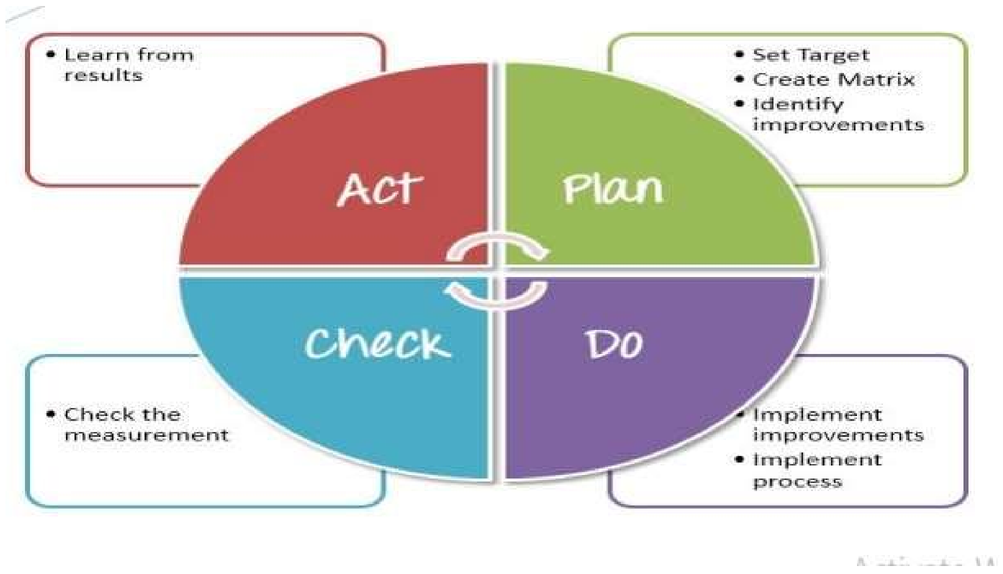
PDCA Cycle

PLAN :- Define goal and plan for achieving goal.

DO :- Depending on the plan strategy decided during the plan stage we do execution according to this phase.

CHECK :- Check or test to ensure that we are moving according to plan and are getting the desired result.

ACT :-During the check cycle , if any error/issue occurs then we take appropriate action accordingly and revise our plan again.

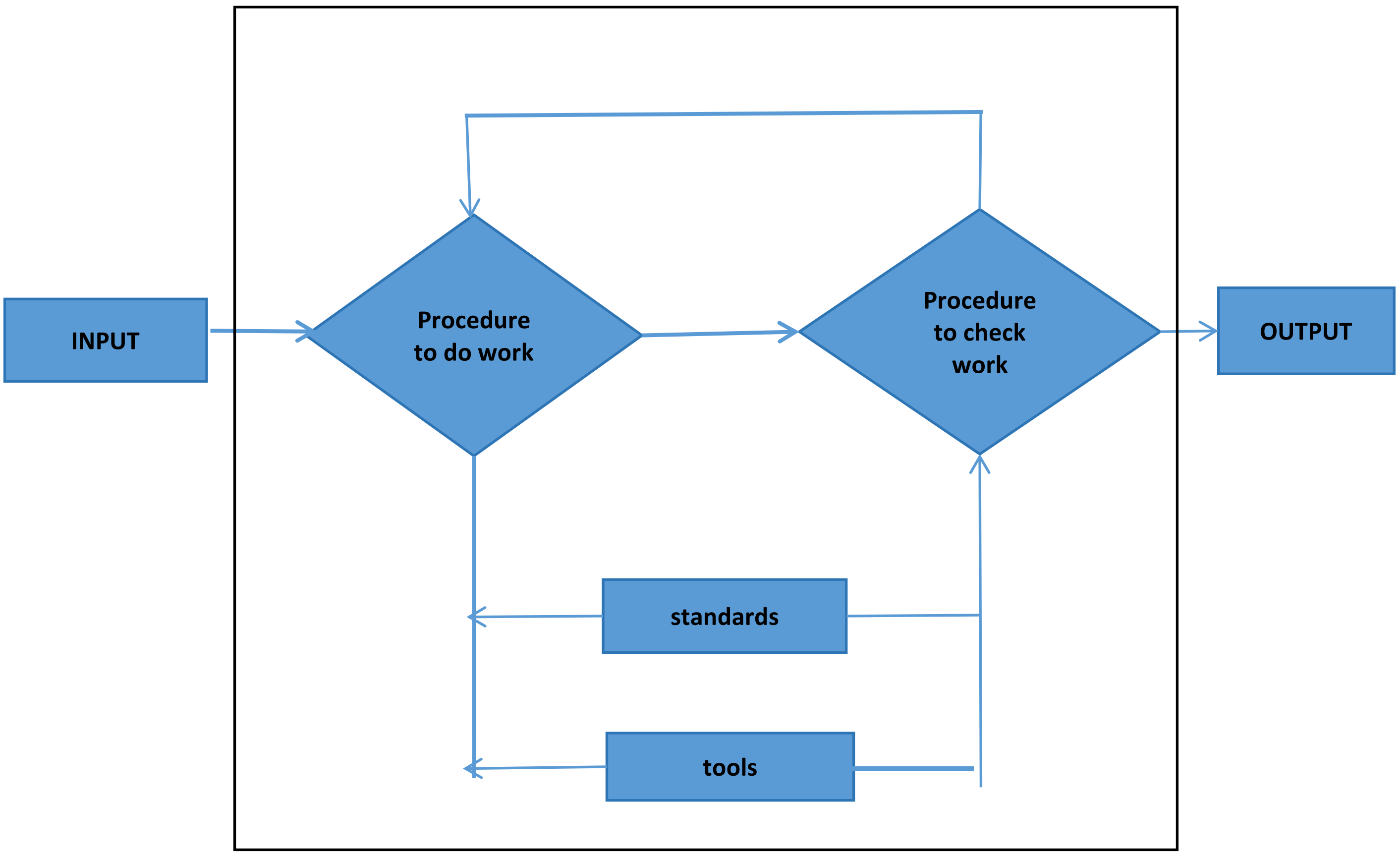


Workbench

It is a more practical illustration of a process. The testers workbench is a pictorial representation of how a specific test task is performed.

Workbench is built on the following components.

1. Objective –Purpose of the process
2. People skills-Roles, responsibilities and associated skills needed to execute a process.
3. Input –Entrance criteria or deliverable needed to perform testing.
4. Procedure –Do and check procedure.



Session 11

Certifications

CMMI (Capacity Maturity Model integration)

CMM is a method to evaluate and measure the maturity of the software development process of an organisation. Certifications are given to organisations by Carnegie Mellon University.for getting this we have to satisfy 18 KPA.

There are 5 maturity levels designated by the number 1 through 5.

CMM 1 -Initial(Lowest quality/Highest risk)

-Any newly satisfied software organisation will get this certification.

CMM 2-Repeatable(Lowest quality/high risk)

-if any organisation managing the existing process or following a new process to develop a software or organisation following trial and error method to develop a software will get the second level certification.

CMM 3-Defined(Medium quality/medium risk)

-if any organisation having some standards and procedures to develop a software will get the third level of certifications.

CMM 4-managing(Higher quality/Lowest risk)

-If any organisation having some target and policy to develop a software will get the fourth level of certifications.

CMM 5- Optimized

-if any organisation develops a system by reducing the resources and developing the software using new tools and techniques and achieves more will get the fifth level of certifications.

ISO- International Organisation for standard

PCMM- People capability maturity model

-it is given to a particular department of an organisation based on their performance and progress.

TMM-testing maturity model

-it is given to the testing department of an organisation.

5 levels of TMM

1.Initialisation- to make sure that the software is running fine & no defect.

2.Definition- Test strategy,test plan,test cases are developed according to the requirement.

3.Integration- Testing is integrated with software life cycle & becomes a part of it.

4.Management & measurement-Test becomes the part of all activities in the software life cycle.

5.Optimization-carried out by the help of different tools

Six Sigma Certifications-Zero Defect Orientation

Session 12

Test Scenario

A TEST SCENARIO is defined as any functionality that can be tested. It is also called *Test Condition* or *Test Possibility*. As a tester, you should put yourself in the end user's shoes and figure out the real-world scenarios and use cases of the Application Under Test.

The test scenarios are those derived from the **use case** and give **the one-line information about what to test**.

Advantages of Test Scenario

- Covers the entire functionality of the software
- Simulates a real-life situation from an end-user point of view
- Helps testing teams control the testing processes
- Easy to create and maintain
- Significant time saver especially for agile teams

Examples:
Test scenarios of a pen
1. Verify that the length and the diameter of the pen are as per the specifications.
2. Verify the outer body material of the pen. {Check if it is metallic, plastic, or
any other material specified in the requirement specifications.}
3. Check the color of the outer body of the pen. It should be as per the specifications.
4. Verify that the brand name and/or logo of the company creating the pen should
be clearly visible.
5. Verify that any information displayed on the pen should be legible and clearly
visible.
6. Verify the type of pen, whether it is a ballpoint pen, ink pen, or gel pen.

7. Verify that the user is able to write clearly over different types of papers.
8. Check the weight of the pen. It should be as per the specifications. In case not mentioned in the specifications, the weight should not be too heavy to impact its smooth operation.
9. Verify if the pen is with a cap or without a cap.
10. Verify over a surface.
11. Verify the surfaces over which the pen is able to write smoothly color of the ink of the pen.
12. Check the odor of the pen's ink on writing apart from paper e.g. cardboard, rubber surface, etc.
13. Verify that the text written by the pen should have consistent ink flow without leaving any blob.
14. Check that the pen's ink should not leak in case it is tilted upside down.
15. Verify if the pen's ink should not leak at higher altitudes.

Write test scenarios of the following objects:

- Pencil
- Fan
- Lift
- Door
- Keyboard
- Mouse
- Atm machine
- Google search page
- Gmail login page
- Chair
- Table

- Book
- Wrist watch
- Stapler
- Whatsapp
- Remote control