UNIT - 1

Software Engineering

- **Software** – Software is more than just a program code. A program is an executable code which serves some computational purpose.
Software is considered to be collection of executable programming code associated with libraries and documentation. Software well made for a specific recruitment is called software product.

- **Engineering** – Engineering on the other hand is all about developing products using well define scientific principles and methods.

1. Recruitment
2. System analysis
3. System design
4. Development
5. Testing
6. Maintainance
7. Updates
8. Code design

- **Software Engineering** – Software engineering is an engineering branch associated with development of software products using well define scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software products.

* **Generic view of Software Engineering** – The process of a software development has 3 generic views which are –

1.) **Definition phase** – It is the base of definition phase. The expert gets knowledge about what?

a) Information needed for processing.
b) Which functions are required.
c) Exceptation about the capacity.
d) Interface which is stablish.
e) Area of the validation.

This phase define all the exceptations depending on the standard of software engineering. It contains 3 steps –

DATE: / /

1- Analysis of system
2- Planning of project
3- Requirement analysis

2.) Development phase – Focus point of development phase is how after the explaination of work it turn how various type of questions arise in developer's mind, that how to design the data structure and architecture of software, procedural details how to implement and how design convert in a programming language and testing of software how to perform. Three special steps always taken in this phase which are-

(i) Designing of software
(ii) Coding
(iii) Testing of software system.

3.) Maintainance phase – The main focus of maintainance phase is change which cause correction of errors, adapting of new idea according to the needs of software after change in customer's mood.

UNIT - 2

PAGE NO :
DATE : / /

Requirements Analysis

* Statement of system scope :-
Project scope is the part of project planning that involves determining and documenting a list of specific project goals, deliverables, features, functions, task, dead lines and ultimately cost.
In other words, it is what needs to be achieved and the work that must be done to deliver a project.
It is important to pin down the scope early in a project life-cycle, as it can greatly impact the schedule or cost or both of the project down the track.

* Review :- A review is a systematic examination of a document by one or more people with the main aim of finding and removing errors early in the software development life-cycle. Review are used to verify for document such as requirements system, design code, test plan and test cases.
Reviews are usually perform manually while static analysis of the tools

PAGE NO:
DATE: / /

is performed using tools.
Importance of review process productivity of the team is improved and time sketch reduce repulse the correction in early stage and work products will help to ensure that those work products are clear and ambitious.
Reduction in cost because favour defects in final software. Testing costs and time is reduced as there is enough time ispend during the initial phase.

* Software Refinement:- Modern software development is a complicated process specially when a software system become large and complicated. Software developers must apply software refinement in order to procede from a high level abstract model to a final executable software system by adding more details over time. Class diagrams are important because they represent the static structure of a software system and therefore we design a UML profile extending the UML meta model to support class diagram refinement.
Refinement is an iterative process

working with a partner who is not on the design team to proof, read each other work.

* Analysing the problem — Analysing is the systematic process of reasoning about a problem and its constituent part to understand what is needed or what must be done. Analysis also involve communication with many people initially those who are most familier with the existing need and its surrounding that is the problem domain must be contacted. Requirement engineers must also communicate with whose who are working completeness checking requires examination of the problem as a prereceived by those who are most familiar with it so that the result will satisfy the real needs.

Key points of analysing the problem-

1- Problem analysis is the process of understanding real world problem and users needs and purposing solution and meets those needs.

2- The goal of problem analysing is to gain a better understanding before development being of the problem being solved.

3- To identify the root case of the problem behind the problem ask the people directly invoked.

4- Identify the actors on the system if a key step in problem analysis.

* Software Requirement Specification (SRS) :- A document used to describe the behaviour of software system, functional, non-functional requirements of the software system.

- Functional requirement – This is the list of the actual services which a system will provide. This is the list of the service / function which a user wants from the software.
For example – Business rules of the particular organisations for which developing software.

- Non-functional requirement – How a system should behave while performing the

DATE : / /

operations. These are the constants on the services which system is offering. For example - response time coverability.

* Users of the SRS -

1- Development team
2- Maintainance team
3- Client
4- Technical writers

* Contents of the SRS -

1- Category - What kind of your software.

2- Purpose - Describe what is the purpose of making this system (as a nothing is without any reason.

3- Scope - What is the area it covering, what is its range, in what limit it will help you.

4- Introduction - Define the existing system and purpose system.

5- Advantage - Define the advantage of the system.

* Review for correctness :- Correctness from software engineering prespective can be define as the adherence to the specification that determine how users can interact with the software and how the software should behave when it is used correctly. If the software behave incorrectly, it might take considerable amount.

1- Undetected errors remove.
2- Programming code write with programming rules.
3- Missing items correct.
4- Easy to involve other person.
5- Easy to implement.
6- Focus on system making error free.
7- Improve the quality of software.
8- Produce the result according to expectation.

* Software designing :- Software designing is the process of envisioning and design define software solution to one or more sets of problem. One of the main component of software design in the software requirement analysis (SRA) is a part of the software development process that list specification use in software

engineering. The design focus on capabilities and multiple design for the same problem can will exist.

The design process is a sequence of stack that enables the designers to describe all aspects of software for building. Creative skills, fast experience, a sense of what makes good software and an overall commitment to quality are examples of critical success for a component design.

Requirement Completeness

A quality demanded to the set of software requirements and to each requirement itself, in order to ensure that there is no information left aside.

The purpose of requirement completeness in software engineering is to detect the situation when the results of the program do not match the input data. Even for the simplest of programs the number of cases of input data is astronomical.

Requirement Consistency

Consistency checks requirements in the document should not conflict or different description of the same

function.
An important aspect in the software development process is the consistency between various parts of the software system being designed and implemented. During the development of a system we are aware of the consistency problems and we usually solve these by special arrangements developed as part of the development of the software system.

Example of Consistency Requirement

An example of consistency is a sauce that is easy to pour from a pitcher. An example of consistency is when all tests that students take are graded using the same grading scale.

* Software Design Levels :-

1- Software architecture design level - The architecture design is the highest abstract version of the system. It identify the software as a system with many components interacting with each other. At this level the designers gets the idea of purposed solution domain.

2- High level design - The high level design break the 'single-entity multiple component' concept of architecture design into less abstracted view of sub-system and modules and depicts their interaction with each other. High level design focus on how the system along with all of its components can be implemented in form of module. It recognise modular structure of each subsystem and their relation and interaction among each other.

3- Detailed design - Detailed design deals with the implementation part of what is seen as a system and its sub system in the previous ~~to~~ two design.

It is more detailed towards modulars and implementations. It define logical structure of each module and their interface to communicate with other modules.

* Object Oriented Design :-
Object oriented design works around the entities and their characteristics instead of functions involve in the software system. This design strategies focus on entities and its characteristics. The whole concept of software solution revolves around the engaged entities.
Important concepts of object oriented design -

1- Objects - All entities involve in the solution design are known as objects. For example - person, bank, company and customers are read as an object, associated to it and has some methods to perform on the attributes.

2- Class - A class is a generalization description of an object. An object is an instance of a class. Class define all the attributes which are object can have

and methods which define the functionalit of the objects. In the solution of design attributes are stored as a variable and functionality are define by means of method procedures.

3- Inheritance — Object oriented design allow similar classes to stack-up in hierarichal manner where the lower or subclasses can import, implement and reuse allow variables and methods from their immediate super-classes. These properties of object oriented design is known as inheritance. This make it easier to define specific class and to create generalize classes from specific one.

4- Polymorphism — Object oriented design languages provides a mechanism where methods performes similar tasks but vary in arguments can be assign same name. This is called polymorphism which allow a single interface performing task for different type. Depending upon how the function is invoke, respective portion of the code gets executed.

* Software Quality Requirement :-

Software quality requirement are specification of the quality of product, service, process or environment. Quality is any element, tangible or intangible that give things value beyond their functionality and features.

1- Availability - Is it available when and where I need to use it?
2- Installability - How easy is it to correctly install the product?
3- Integrity - Does it protect against unauthorize access and data loss?
4- Interoperability - How easily does it interconnect with other system?
5- Performance - How fast does it response or execute?
6- Recoverability - How quickly can the user recover from a failure?
7- Reliability - How long does it run before experiencing a failure?
8- Robustness - How will does it response to unaccepted operations?
9- Safety - How will does it protect against injure or damage?
10- Usability - How easy it for people to learn and use?

11- Effeciency — How will does it utilize processor capacity, disk space, memory and other resource?

12- Flexibility — How easily can it be update with new functionality?

13- Maintainability — How easy it to correct defect or may change?

14- Portability — How easily can it be made to work on other platforms?

15- Scalability — How easily can it add more users, servers or other extensions?

16- Reusability — How easily can it be use components in other system?

17- Supportability — How easily will it be to support after installation?

18- Testability — Can I verify that it was implemented correctly?

* **Relationship between design and implementation**

An implementation is the process of converting the design as a program. Software design and implementation is the stage of the software engineering process at which an executable software system is developed. Software design and implementation activities are invariably enterlevel.

* Software design concept –

1. Abstraction
2. Refinement
3. Modularity
4. Software architecture
5. Control hierarchy
6. Software procedure
7. Structure partition
8. Data structure
9. Information hiding

1- Abstraction – Abstraction is the process of generalising by reducing the information content of a concept. It is an act of representing essential features without including the background details and explaination.

2- Refinement – It is the process of elaboration one or several instructions of a given program are decomposed into more detailed instructions. Abstraction and refinement are complementry concepts.

3- Modularity – Software architecture is divided into components called modules.

4- Software architecture – The overall structure of the software provides conceptual integrity for a system.

5- Control hierarchy – A program structure

that represent the organisation of a program component and implies a hierarchy of control.

6- Software procedure — It focus on the processing of each module individually.

7- Structure Partition — The program structure can be divided both horizontal and vertically.
Horizontal partition define seperate branches of modular hierarchy for each major program function.
Vertical partitioning suggests that control and work should be distributed top-down in the program structure.

8- Data structure — It is a representation of the logical structure among individual elements of data.

9- Information hiding — Modules should be specified and design so that information contain within a module is inaccessible to other modules that have no need for such information.

★ Software Implementation :-

1. Programming style
2. Coding style
3. Selection of language and technology
4. Implementation means coding
5. There is so many challenges in implementation

1- **Programming Style** – Programming style also known as code style is a set of rules or guidelines used when writing the source code for a computer program. It is often claim that following a particular programming style will help programming read and understand source code confirming to the style and help to avoid introducing errors.

2- **Coding style** – Coding conventions are set of guidelines for a specific programming language that recommend programming style, practice and methods for each aspect of a program written comments, declaration, statements, white space, naming convention indentation, programming principles programming rules of thumb etc.
These are the guidelines for Software structure quality, software programmers are highly recommended to follow these

DATE : 9 / Mar/2019

guidelines to help improving the readability of their source code make software maintenance easy.

* **Software implementation process issues and challenges :-**

The process of new software implementation requires specialist, checklist and quality assurance. Software change management tracking solutions and a robust cooperation with the vendor. We are going to cover some of the most important challenges that need to be taken into account when creating a software implementation plan. After listening challenges and issues we are then going to present some general tips and ideas that will help to make sure your next roll-out is successful.

1- Software legacy - One of the main issues, company need to take during the software implementation planning phase is system of software legacy. A solid change management strategy always takes into account all the limitations connected to legacy issues

and already addresses the problem by defining implementation rules that generate compatibility and software compliance. In some extreme cases, the company might need to consider going through a massive legacy system modernisation to partially or completely solve an issue.

2- Software certification — Another important aspect that cannot be neglect and which usually is specifically connected to the negotiation and due phase is making sure that the software meets all the certifications necessary for diplomat.
Software certification are important when it comes to both compliance and security.

3- IT infrastructure and integrations — Sometimes it's hard to integrate new services or tools within an existing infrastructure. Creating a company structure that mirrors a smooth customer experience however requires data to be centralize and analyze across different units to avoid siloed structures and whole in the customer journey. Imagine implementing marketing

DATE : 12/ Mar/2019

automation software that integrate with the main interact with customer help storing solutions to triger action or cross reference data in order to implement a new technology.

4- Software training — One of the biggest risk connected to software implementation is the lack of proper software training. The project lead needs to set clear goals and deadlines. Sometimes a company wide software roll-out plan can be hindered by the lack of resource for software training. In this case organisation are often force to schedule different implementation mile stones across different units.

5- Software Portability and backward compatibility — There is nothing worse than creating a software implementation plan to roll-out a new platform or schedule a major company wide upgrade to find out that new system is not working properly across different computing platform. This issues might lead to very high cost link to either software customization if possible or

hardware updates. At the same time we also want to make sure that new update guarantee for interoperability with older legacy system.

6- Changes in the team structure – Changes are often reflected in the team structures and roles and this also often holds true when it comes to implementing new software A new software roll-out might need to be followed by small adjustments in roles and responsibilities. These can drive from the necessity of having an administrator or manager that work on the integration of the new solution or new reporting and data analyze figures within the team

7- Data migration – Connected to the preirious point data migration relies of backwards, downwards compatibility. This is essential to focus on the data migration very easily in the implementation process since this is a very time consuming and risk component of the implementation plan.

8- Security and stability – Every software implementation is a potential gate that

might lead to security branches rushin into a speedy roll-out might soon become an interact labyrinth of patch and work around that can jeopardize. The stability of the entire architecture. In some case, it is even recommended to carry-out vulnerability to verify that the structure is still robust and secure.

* Programming Support Environment :-

In computer program and software product development, the development environment is the set of process and programming tools used to create the program or software product. The term may sometime also employ the physical environment.

An integrated development environment is one in which the process and tools are coordinated to provide developers an orderly interface to add convenient view of the development process or at least the process of the writing code, testing it and packaging it for use.

An example of an IDE product is- Microsoft's visual studio .net, Net Beans, XCode, Web IDE's, edipse.

The term computer assisted software environment is generally used to describe a set of tools and practice that facilitate management of a software development project. Software environment is the term commonly used to refer to support an application. A software environment for a particular application could include the operating system, the data base system, specific development tool or compilers. development environment in which software coding is done. It include the technology use for front-end, back-end, computer configurations, memory disk space etc.

It also include the software development model following development tools, number of programs, the organisation policies or any kind of support needed in coding a software. production environment in which software is actually used by client. It comprise of operation system, computer configuration, browsers used by client, number of software users etc.

★ Good Programming Style :-

Proper programming style signification reduce maintenance cost and increase the lifetime and functionality of software Most software disasters are rooted in poor style of programming. There are many best rules that lead to a better programming style.

1- Readability — Good code is written to be easily understand by colleagues. It is properly and consistently formatted and clear meaningful name for functions and variables, concise and accurate comments describe a natural decomposition of the software functionality into simple and specific functions. Any tricky sections are clearly noted. It should be easy to see why the program will work and reason that it should work in all cases.

2- Maintenability — Code should be written so that it is straight forward for another programmer to fix bugs for make change to its functionality later. Function should be general and assume as little as possible about preconditions

All important values should be robust to handle any possible input and produce a responsible result without crashing. Clear message should be output for input which is not allow.

3- Comments — Comments are the first step towards making computer program human readable. Comments should explain clearly everything about a program which is not obvious to a fair programmer.
The volume of comments written is meaningless quality is all that counts. Block comments are written using comments style. They should go at the top of every source file and generally include your name, the data, your code was written an overall description of the purpose of that program. Block comments should also precede most functions with a description of the functions purpose. These can be omitted for every obvious function only.

4- Indentation — Indentation is used to clearly make mark control flow in a program within any bracket block. All code is identical any one tap. These includes the

class body itself. Each additional for, while, if or switch structure introduce a new block which is indented even if brackets are omitted for one line statement any corresponding else statement should line-up.

5- White space — White space is meaningless to compiler but should be used consistently to improve readability. Typically three blank lines are left in between functions.

6- Output — A final overlook aspect of good programming style is how your program output results and information to users part of writing professional looking programs is providing clear instructions and results to the user of your program. This means proper English with no spelling error conditions. One must always assume that writing program to be use by somebody.

* Function uses :-

Function should be short and clear specific task. As much as possible they should be consider black box which do not depend on anything except their parameters

and can be handle any possible input gracefully. A common rule of thumb is the tag line rules. Usually functions longer than tag lines are trying to do much and should be simplified.

* **Maintenance :-**

Software maintenance in software engineering is the modification of a software product after delivery, to correct fault to improve information of other attributes. A common perception of maintenance is that it merely involve fixing defects. However one study indicates that our 80% of maintenance effort is use for non-corrective actions. Software resolution is the term used in software engineering. Software maintenance refers to the process of developing software initially than repeated updating its various reasons. Software maintenance is widely accepted part of SDLC nowadays.

It stand for all the modification and updation done after the delievery of software product. There are number of reasons why modification are required. Some of them are ~~be~~ briefly mentioned here -

1- Market condition – Policies which change over the time such as newly introduced constants like how to maintain.

2- Host modification – If any of the hardware and platform such as operating system of the target host change, software change are needed to keep adaptability.

3- Client requirement – Over the time, customer may ask for new features or functions in the software.

4- Organisation change – If there is any business level change at client such as reduction of organisation strength, acquiring another company, organisation venture into new business need to modify in the original software may arise.

- Types of maintenance :-

In a software lifetime, type of maintenance may vary based on its nature. it may be just a routine. Maintenance task as some bugs discover by some users or it

may be a large event in itself. Based on maintenance size or nature, following are some types of maintenance based on their characteristics.

1. Corrective maintenance – This include modification and updations done in order to correct of fix problems which are either discovered by some users or it may be a large event in itself based on maintenance. Order to correct fix problem or concluded by users error reports.

2. Adaptive maintenance – This include modification and updation applied to keep the software product up to date and turn to the everchanging world of technology and business environment.

3. Perfective maintenance – This includes modification and updation done in order to keep the software usable over long period of time. It include new features, new users requirement for refining the software and improve its reliability and performance.

DATE : / /

4. Preventive maintenance — This include modification and updation to prevent future problems of the software. It aims to attend problems which are not significant at this moment but may cause series issues in future.

* Cost of Maintenance :-
Report suggest that the cost of maintenance is high. A steady on estimate software maintenance found that the cost of maintenance is as high as 67% of the cost of entire software process cycle on an average. The cost of software maintenance is more than 50% of on SDLC phases. There are various factors which trigger maintenance cost so high.

• Real world factor affecting maintenance cost — The standard age of any software is considered upto 10 to 15 years older the software which were where main to work on slow machines with less memory and storage capacity cannot kept themselves changing against newly coming inhance softwares or modern hardware. As a technology advanced it become costly to

maintain old software. Most maintenance engineers are newbie and use trial and errors method to rectified problems.
Often changes made can easily hurt the original structure of the software. Making it hard for any subsequence changes.
Changes are often poorly undocumented which may cause more conflict in future.

* **CASE Tools :-** CASE stands for Computer Aided Software Engineering. It means development and maintenance of software projects with the help of various automated software tools. Case tools are set of software application programs which are used to automated SDLC activities.

Case tools are used by software project managers, analyses and engineers to develop software system. There are number of case tools available to simplify various stages of software development life cycle such as analysis tools, design tools, project management tools, data base management tools, documentation tools are to name a few. Use of case tools accelerate the development of project to produce desired result and help to uncover flaw before moving ahead with

next stage in software development.

- Components of CASE tools — Case tools can serve as a source of common integra-ted and consistent information central repository is a central place of storage where product specification, requirements, documentations related supports and diagrams. Other useful information regarding management is stored central repository also serve as a data dictionary.

- Upper Case tools — Upper case tools are used in planning, analysis and design stages of SDLC.

- Lower case tools — Lower case tools are used in implementation, testing and maintenance.

- Integrated case tools — Integrated case tools are helpful in all the stages of SDLC from requirements gathering to testing and documentation. Case tools can be group together if they have similar functionality, process activities and capability of getting integrated with

other tools.

Case-tools types :- There are many types of case tools -

1. Diagram tools - These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example - flow chart maker tools for creating state of the art flow-charts.

2. Process modelling tools - Process model is the method to create software process model which is used to develop the software. Process modelling tools help the managers to choose a process model or modify it as per the requirement of software product.

3. Project management tools - These tools are used for project planning cost and effort estimate, project scheduling and resource planning. Project execution with every maintained step in software

project management. PMT helps in storing and sharing.
Example- creative project office, trace projects and base camp.

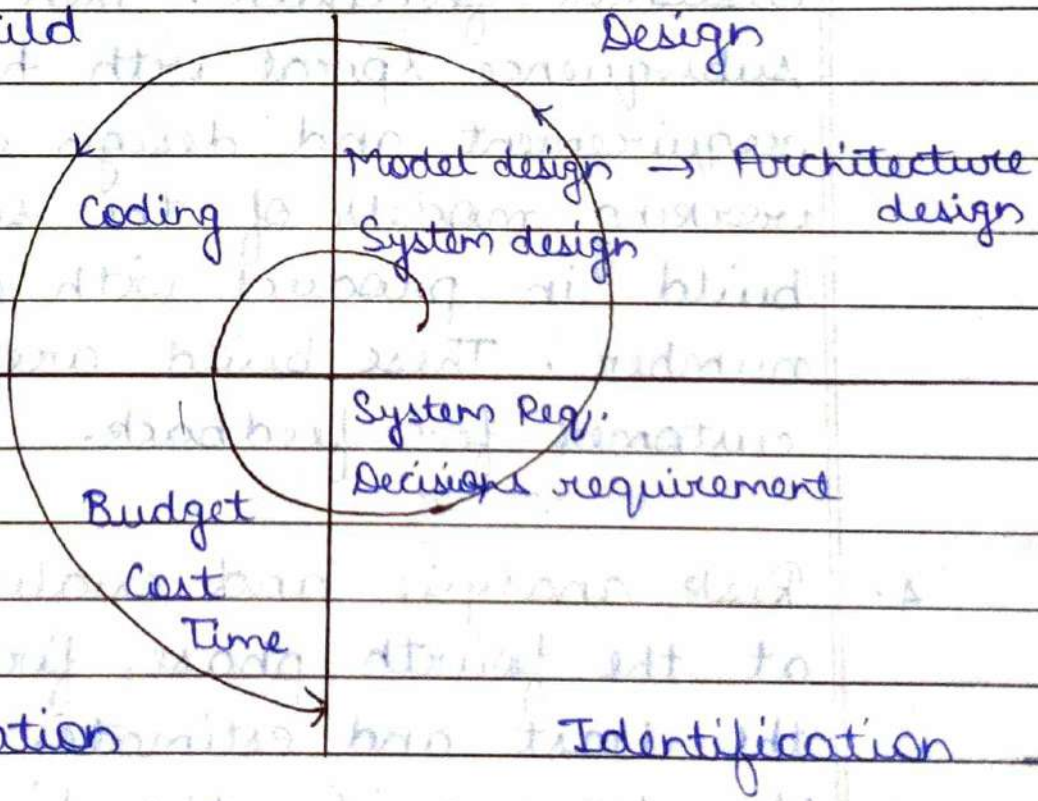
4. Quality Assurance tools- Quality assurance in a software organisation is monitoring the engineering syllabus process and methods adopted to develop the software product in order to ensure confirmance of quality as per organisation standard. Quality assurance tools consist of configuration and change control tool and software testing tools.

★ Spiral Models :-
The spiral model combine the idea of itterative development with the systematic control aspect of the water fall model. This spiral model is a combination of iterative development process model and sequential linear development model. The waterfall model with a very high emphasis or risk analysis. It allows incremental release of the product or increment-al refinement through iteration

PAGE NO :
DATE : / /

around the spiral. Spiral model designes the spiral model has 4 phases -

1. **Identification** – This phase start with gathering the business requirement in the baseline spiral. In the subsequence spiral as the product identification of system requirements and unit requirements are all done in this phase. This phase also include understanding the system requirements by continuous communication between the customer and the system analysis.



2. **Design**– The design phase starts with the conceptual design in the baseline spiral and involves architecture design, logical design of modules. Physical product design and the final design in the subsequence spiral.

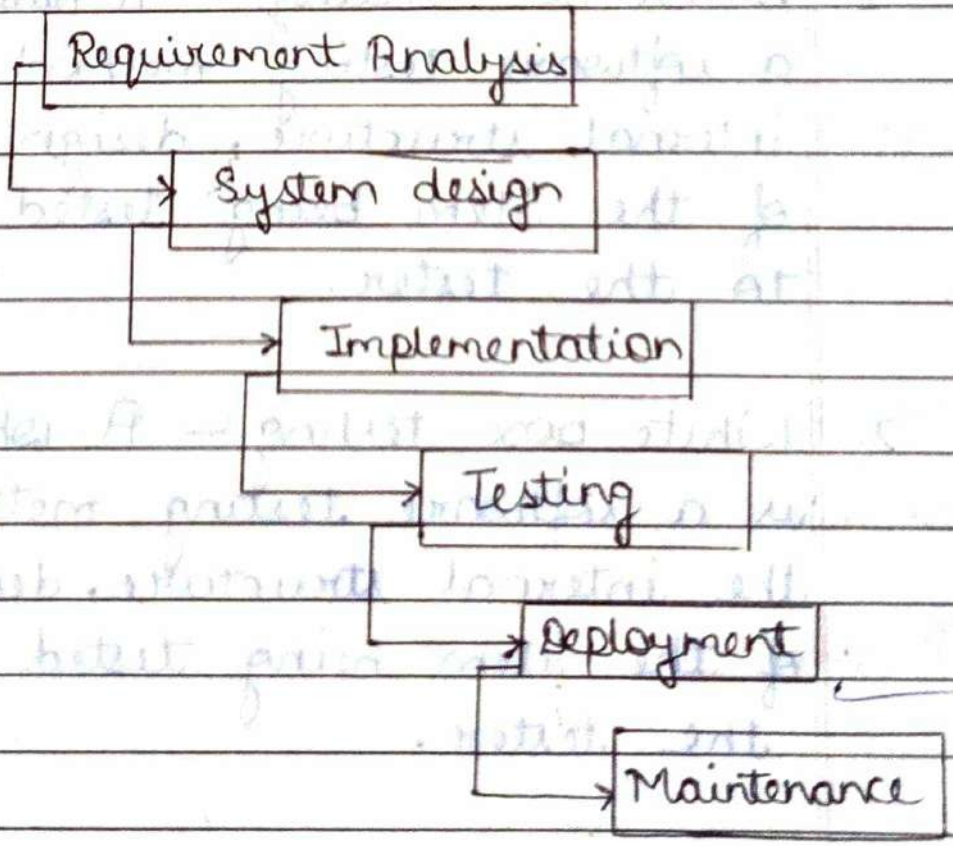
3. **Construct or build** – The construct phase refers to production of the actual software product at every spiral. In the baseline spiral when the product is just how the design is being developed a proof of concept is deploying in this phase to get customer feedback. Then in the software subsequence spiral with high clarity on requirement and design details a working models of the software. All build in product with a version number. These build are send to customer for feedback.

4. **Risk analysis and evaluation** – Here at the fourth phase, first we identify the least and estimates of the risk. We also provide the final builds of the project to the customer to get feedback from the customer.

★ Waterfall model :-

A model which contain series of phases that provide a common understanding of the software building process. It is very simple to understand and use. In a waterfall model each phase must be completed before the next phase can be and there is no overlapping in the phases.

The waterfall model is the earliest SDLC approach that was used for software development. The waterfall model illustreates the software development process in a linear sequential flow. This means that only phase in the development process being only if the previous phase is complete. In the waterfall model the phases do not overlap.



• Disadvantage of waterfall model –

1- Very difficult to go back to any stage after it finished.

2- A working version of the program will not be available until the date in the project time spend.

3- Costly and requires more time.

4- Linear nature can lead to blocking state.

★ Software testing :-

Software testing can be majorly classified into 2 category –

1- Black box testing

2- White box testing

1- Black box testing – A black box testing is a software testing method in which the internal structure, design, implementation of the item being tested is not known to the tester.

2- White box testing – A white box testing is a software testing method in which the internal structure, design, implementation of the item being tested is known to the tester.

* Difference between black box testing and white box testing :-

- Black box testing -

1- It is a way of software testing in which the internal structure of the program or the code is hidden. Nothing is known about it.
2- It is mostly done by software tester.
3- It can be refer as outer of external software testing.
4- It is functional test of the software.
5- This testing can be initiate on the basis of requirement specification documents.
6- No knowledge of programming is require.
7- It is the behaviour testing of the software.
8- It is applicaple on the higher level of testing of software.
9- It is also closed - testing.

- White box testing -

1- It is way of testing the software in which the tester has knowledge about the internal structure, the code or the
2- program of the software.