SOFTWARE ENGINEERING

- The term software engineering is composed of two words, software and engineering.
- Software is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be a collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called software product.
- Engineering on the other hand, is all about developing products, using well-defined, scientific principles and methods.

Generic software development:

- Generic software development is being undertaken by the company that owns the resulting product. This product has a reliable need in the market over a period of time and fit for most of the business needs.
- Software development companies develop generic software at own cost and sell it as a product for customers. Then generic software is ready, set to use and available in the market.

Custom software development :

Custom software development is being done as a work-for-hire project for a particular client (the group of people or another company). The list of requirements, product's idea, need and money on development come from the client/customer. Usually, it takes a lot of time to develop a custom product.

Types of softwares:

System Software:

- System software is necessary to manage the compiler resources and support the exectution of application programs.
- Ex:Compilers

Application software:

- It is designed to solve user problems as per the user's requirements. Application software can be generic or customized. Ex: MS Office

Scientific software:

- It deals with processing requirements in a specific application.
- These software are used in the field of mechanical, electrical, drafting, engineering and Analysis. They run on mainframes, general purpose workstation, and PCs (Personal computers).

Embedded software:

- This software is embedded into hardware as a part of larger systems to control its various functions.
- Ex:Keypad control software embedded in a microwave oven or washing machine

Product_line software:

- A software product line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

Web apps Software:

- It is an application that is accessed via web browsers over a network such as an internet or an intranet.
- Ex: Facebook

AI software:

- This software makes the use of non-numerical algorithms that use the data generated in the system to solve complex problems
- Ex: Robotics

• Software engineering principles use two important techniques to reduce problem

complexity:

- abstraction

- decomposition.

ABSTRACTION: The main purpose of abstraction is to consider only those aspects of the problem that are relevant for certain purpose and suppress other aspects that are not relevant for the given purpose.

Abstraction is a powerful way of reducing the complexity of the problem.

DECOMPOSITION: In decomposition, a complex problem is divided into several smaller problems and then the smaller problems are solved one by one.The problem has to be decomposed such that each component of thedecomposed problem can be solved independently and then the solution of the different components can be combined to get the full solution.

APPLICATIONS AND NEED OF SOFTWARE ENGINEERING

Software Development for various domains

• To perform various operations on the software like testing

• Maintenance of various software products

• To apply the knowledge, practices, and technologies to build high-quality

• software products that enhance productivity in every industry

CHARACTERISTICS OF GOOD SOFTWARE

Every software must satisfy the following attributes:

• Operational

• Transitional

• Maintenance

Operational

This tells us how well software works in operations. It can be measured on:

• Budget: Software must be developed in reasonable budget.

• Usability:It refers to the extent to which the software can be used with ease.

• Efficiency:It refers to the ability of the software to use system resources in the most effective and efficient manner.

• Correctness:The application should be correct in terms of its functionality, calculations

• Functionality:It refers to the degree of performance of the software against its intended purpose.

• Dependability:A set of attributes that bears on the capability of software to maintain its level of performance under the given condition for a stated period of time.

• SecurityL:System integrity or security should be sufficient to prevent unauthorized access to system functions,

• Safety:ensure that the software is protected from virus infection, and protect the privacy of data entered into the system.

Transitional

This aspect is important when the software is moved from one platform to another:

• Portability;A set of attributes that bears on the ability of software to be transferred from one environment to another, without or minimum changes.

• Interoperability:Interoperability of one system to another should be easy for the product to exchange data or services with other systems.

• Reusability:Different code library classes should be generic enough to be easily used in different application modules.

• Adaptability: it means software should be able to work in any condition

Maintenance

This aspect briefs about how well a software has the capabilities to maintain itself in the ever-changing

environment:

• Modularity: Any software is said to made of units and modules which are independent of each other. These modules are then integrated to make the final software.

• Maintainability:It refers to the ease with which the modifications can be made in a software system to extend its functionality

• Flexibility:Should be flexible enough to modify. Adaptable to other products with which it needs interaction.

• Scalability:**system to increase its capacity and functionalities based on its users' demand.**

SDLC : Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

SDLC PHASES

• Phase 1: Requirement collection and analysis:

• Phase 2: Feasibility study:

• Phase 3: Design:

• Phase 4: Coding:

• Phase 5: Testing:, • Phase 6: Installation/Deployment: ,• Phase 7: Maintenance:

1)Requirement collection and analysis:

- It is conducted by the senior team members with inputs from all the sales department and domain experts in the industry.
- Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage.
- This stage gives a clearer picture of the scope of the entire project.This helps companies to finalize the necessary timeline to finish the work of that system.

Feasibilty study

• Economic: Can we complete the project within the budget or not?

• Legal: Can we handle this project as cyber law and other regulatory compliances.

• Operation feasibility: Can we create operations which is expected by the client?

• Technical: Need to check whether the current computer system can support the software

• Schedule: Decide that the project can be completed within the given schedule or not.

Design

- the system and software design documents are prepared as per the requiremens specification document.
- **High-Level Design (HLD)**

  • Brief description and name of each module

  • An outline about the functionality of every module

- Interface relationship and dependencies between modules
- Database tables identified along with their key elements
- Complete architecture diagrams along with technology details

- **Low-Level Design(LLD)**
  - Functional logic of the modules
  - Database tables, which include type and size
  - Complete detail of the interface
  - Addresses all types of dependency issues
  - Listing of error messages
  - Complete input and outputs for every module.

Coding

- In this phase, developers start build the entire system by writing code using the chosen programming language.In the coding phase, tasks are divided into units or modules and assigned to the various developers.
- It is the longest phase of the Software Development Life Cycle process.In this phase, Developer needs to follow certain predefined coding guidelins.

Testing

- Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.
- During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system..

INSTALLATION/DEPLOYMENT

• Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts.

• Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

MAINTENANCE

Once the system is deployed, and customers start using the developed system, following 3 activities

occur

• Bug fixing - bugs are reported because of some scenarios which are not tested at all

• Upgrade - Upgrading the application to the newer versions of the Software

• Enhancement - Adding some new features into the existing software

The main focus of this SDLC phase is to ensure that needs continue to be met and that the system

continues to perform as per the specification mentioned in the first phase.

Life cycle models

1)WATERFALL MODEL

i] Requirement analysis and specification phase :-

- This phase aims to understand and properly document the exact requirements of the customer. This task is usually carried out in collaboration with the customer, as the aim is to record all specifications for the application.
- The SRS document may act as a contract between the developer and the customer. If the developer fails to implement a full set of requirements, it may fail to implement the contracted set of requirements.

ii]Design phase:-

• This step convert the requirement specification into a structure that is suitable for implementation in some programming languages.overall software architecture is defined, and the high level and detailed design work are performed.

• This work is documented and known as a software design description (SDD) document.

Implementation and unit testing phase:-

• During this phase, the design is implemented. If the SDD is complete, coding phase proceeds smoothly, because of all the information needed by the software developers contained in the SDD.

• During testing, the primary activities are centered around the examination and modification of the code. Small modules were initially tested in isolation from the rest of the software product.

• There are problems associated with the evaluation of unit tests in isolation. How do we run a module without anything to label it, to call it.

• These problems are resolved at this point, and the modules are checked after writing some overhead code.Integration and system testing:-

iii]Integration and system testing:-

• This is a very important phase. Good testing can help to deliver higher quality software products.

• It is a very expensive activity and consumes one-third to at least one half the value of a typical development project.

• The purpose of unit testing is to determine whether each independent module is implemented correctly.

• It gives very little chance to determine if the interface between modules is also correct and, therefore, an integration testing is performed. System testing involves the testing of the entire system, whereas software is a part of the system.

iv]Operation and maintenance phase:-

• Software maintenance is a challenge that every team has to face when the software is deployed and installed to the customer’s site.

• Software maintenance is a very broad activity that includes error correction, enhancement of capabilities, and optimizations.

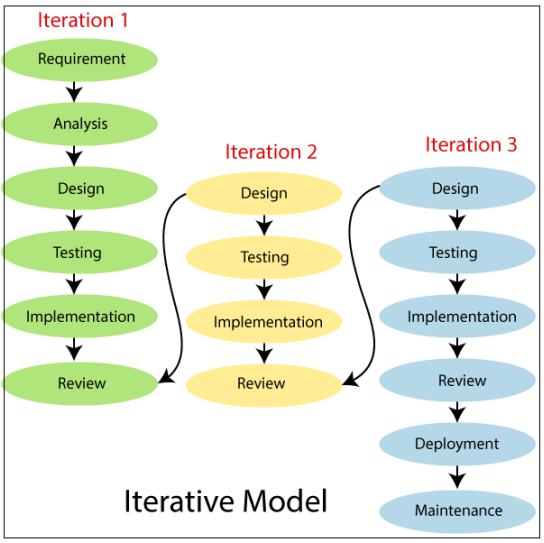
• The purpose of this phase is to maintain the value of the software over time. This stage can last from 5

to 50 years, while development can last from 1 to 3 years.

2)ITERATIVE MODEL

• This model consists of the same phases as the waterfall model, but with fewer restrictions and conducted in several cycles.

• A reusable product is released at the end of the cycle, with each release providing additional functionality.



Advantages of Iterative Model

• Many features can be developed quickly in the life cycle.

• Results are received quickly and periodically.

• Testing and debugging is easy during short iterations.

• Less expensive to change scope / requirements.

• Suitable for large projects.

Disadvantages of Iterative Model

• Requires more management attention.

• Risk analysis requires highly efficient resources.

• The project completion date has not been confirmed due to changing

requirements.

3) Prototyping Mode

• Prototyping Model is a software development model in which prototype is built, tested, and reworked until an acceptable prototype is achieved.

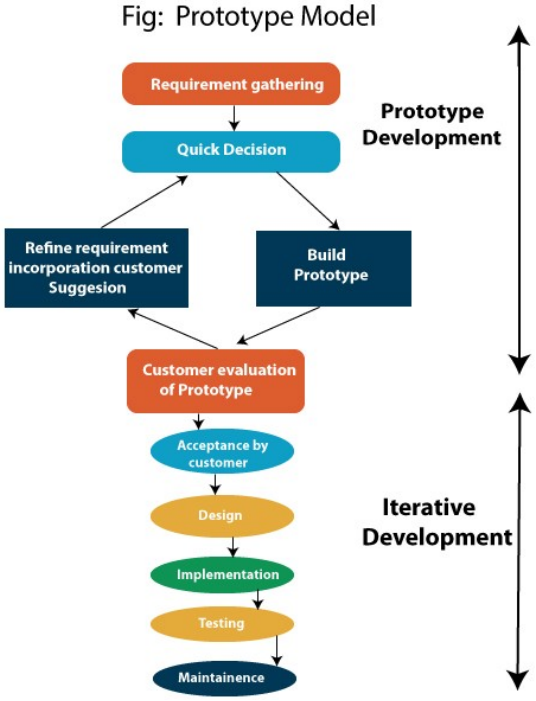
• It also creates base to produce the final system or software. It works best in scenarios where the project's requirements are not known in detail.

• It is an iterative, trial and error method which takes place between developer and client.

Steps of Prototype Model

- Requirement Gathering and Analyst
- Quick Decision
- Build a Prototype
- Assessment or User Evaluation
- Prototype Refinement
- Engineer Product

Fig: Prototype Model



Types of Prototyping Models
Four types of Prototyping models are:
• Rapid Throwaway prototypes
• Evolutionary prototype
• Incremental prototype
• Extreme prototype

Rapid Throwaway Prototype
It is quickly developed to show how the requirement will look visually. The customer's feedback helps drives changes to the requirement, and the prototype is again created until the requirement is baselined.This technique is useful for exploring ideas and getting instant feedback for customer requirements.

Evolutionary Prototyping
Here, the prototype developed is incrementally refined based on customer's feedback until it is finally accepted. It helps you to save time as well as effort. That's because developing a prototype from scratch for every interaction of the process can sometimes be very frustrating.It is helpful when the requirement is not stable or not understood clearly at the initial stage.

Incremental Prototyping
In incremental Prototyping, the final product is decimated into different small prototypes and developed individually. Eventually, the different prototypes are merged into a single product. This method is helpful to reduce the feedback time between the user and the application development team.

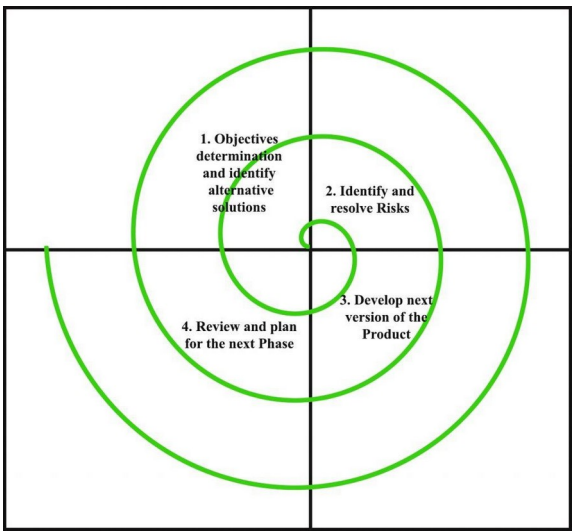
Extreme Prototyping
Extreme prototyping method is mostly used for web development. It is consists of three sequential phases.
• Basic prototype with all the existing page is present in the HTML format.
• You can simulate data process using a prototype services layer.
• The services are implemented and integrated into the final prototype.
The first phase is the static prototype, consisting of HTML pages and possibly a logical data model supporting those pages. The second phase is a coding process in your chosen web framework whereby the screens are fully functional using a simulated services layer. The third phase is where the services are implemented.

Best practices of Prototyping:
• You should use Prototyping when the requirements are unclear
• At a very early stage, you need to approve a prototype and only then allow the team to move to the next step.
• In software prototyping method, you should never be afraid to change earlier decisions if new ideas need to be deployed.
• You should select the appropriate step size for each version.
• Implement important features early on so that if you run out of the time, you still have a worthwhile system

Advantages of the Prototyping Model
• Helps team member to communicate effectively
• There will be hardly any chance of software rejection.
• It also identifies the complex or difficult functions.
• It is a straightforward model, so it is easy to understand.
• Prototypes may offer early training for future users of the software system.

Disadvantages of the Prototyping Model
• Prototyping is a slow and time taking process.
• Prototyping may encourage excessive change requests.
• Poor documentation because the requirements of the customers are changing.
• The client may lose interest in the final product when he or she is not happy with the initial prototype.
• Some times customers may not be willing to participate in the iteration cycle for the longer time duration.

4)SPIRAL MODEL
Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling.
• In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project.
• Each loop of the spiral is called a Phase of the software development process.
• The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks
• As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using spiral model.



•The Radius of the spiral at any point represents the expenses(cost) of the project so far.
• The angular dimension represents the progress

made so far in the
current phase.

•**Objectives determination and identify alternative solutions**: Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
• **Identify and resolve Risks:** During the second quadrant all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution is identified and the risks are resolved using the best possible strategy.
• **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
• **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

RISK HANDLING IN SPIRAL MODEL
• The most important feature of the spiral model is handling these unknown risks after the project has started. Such risk resolutions are easier done by developing a prototype.
• The spiral model supports coping up with risks by providing the scope to build a prototype at every phase of the software development.
• Prototyping Model also support risk handling, but the risks must be identified completely before the start of the development work of the project.

WHY SPIRAL MODEL IS C ALLED META MODEL ?
• The Spiral model is called as a Meta Model because it subsumes all the other SDLC models.
• For example, a single loop spiral actually represents the Iterative Waterfall Model.
• The spiral model incorporates the stepwise approach of the Classical Waterfall Model.
• The spiral model uses the approach of Prototyping Model by building a prototype at the start of each phase as a risk handling technique.

ADVANTAGES OF SPIRAL MODEL:
• Risk Handling: The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
• Good for large projects: It is recommended to use the Spiral Model in large and complex projects.
• Flexibility in Requirements: Change requests in the Requirements at later phase can be incorporated accurately by using this model.
• Customer Satisfaction: Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

DISADVANTAGES OF SPIRAL MODEL :
• Complex:The Spiral Model is much more complex than other SDLC models.
• Expensive: Spiral Model is not suitable for small projects as it is expensive.
• Too much dependable on Risk Analysis: The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced expertise, it is going to be a failure to develop a project using this model.
• Difficulty in time management: As the number of phases is unknown at the start of the project, so time estimation is very difficult.

5)AGILE METHOD
• AGILE methodology is a practice that promotes continuous iteration of development and

testing throughout the software development lifecycle of the project.
• In the Agile model, both development and testing activities are concurrent, unlike the Waterfall model.
• The Agile model was primarily designed to help a project to adapt to change requests quickly. So, the main aim of the Agile model is to facilitate quick project completion. To accomplish this task agility is required.
• Agility is achieved by fitting the process to the project, removing activities that may not be essential for a specific project.Also, anything that is wastage of time and effort is avoided.
• Agile model is the combination of iterative and incremental process models.
• The time to complete an iteration is known as a Time Box.
• Time-box refers to the maximum amount of time needed to deliver an iteration to customers.
• So, the end date for an iteration does not change. Though the development team can decide to reduce the delivered functionality during a Time-box if necessary to deliver it on time.
• The central principle of the Agile model is the delivery of an increment to the customer after each Time-box.

WHAT ARE THE 4 AGILE VALUES?
• Individuals and interactions over processes and tools
• Working software over comprehensive documentation
• Customer collaboration over contract negotiation
• Responding to change over following a plan

1)INDIVIDUALS AND INTERACTIONS OVER PROCESSES AND TOOLS
• No matter how well-researched your process and high-tech your tools are, it's the team you work with and the way you work together that determines success. Your team and their ability to communicate effectively and efficiently is more valuable than the processes they follow or the tools you use.
• This isn't to say that agile philosophies discourage formalized processes or tools. Both can be helpful in providing structure for your team and facilitating interactions. But at the end of the day, they come second.

2)WORKING SOFTWARE OVER COMPREHENSIVE DOCUMENTATION
Under the agile philosophy, getting software in the hands of customers is the highest priority. After all, how are you going to improve your product if you don't get it out in the wild and collect feedback from real users?
• While this value highlights the importance of shipping software over letting documentation be a bottleneck, it's important to note that documentation in itself is not a bad thing…as long as you don't overdo it.

3)CUSTOMER COLLABORATION OVER CONTRACT NEGOTIATION
• Under the agile philosophy, customer collaboration begins early in the development process and happens frequently throughout. This culture of close collaboration with real customers helps product people ensure they're delivering effective, useful solutions to customers. When you talk to customers often and build feedback into your development process, you reduce risk and eliminate guesswork.

4)RESPONDING TO CHANGE OVER FOLLOWING A PLAN
• An important benefit of the agile methodology is that it encourages frequent reviewing and retooling of current plans based on new information that the team is continually gathering and analyzing.
• The product roadmap, then, is no longer a static document, but a dynamic strategy Product managers in agile environments will need to learn to present their dynamic roadmaps to stakeholders in a transparent manner that reflects the likelihood of change based on new learnings.

PRINCIPLES OF AGILE MODEL:
- Agile project usually includes a customer representative on the team. At the end of each iteration stakeholders and the customer representative review, the progress made and re-evaluate the requirements.
- Frequent delivery of incremental versions of the software to the customer representative in intervals of few weeks.
- Requirement change requests from the customer are encouraged and efficiently incorporated
- It is recommended that the development team size should be kept small (5 to 9 peoples) to help the team members meaningfully engage in face-to-face communication and have collaborative work environment.

- • Agile development process usually deploy Pair Programming. In Pair programming, two programmers work together at one work-station. One does coding while the other reviews the code as it is typed in. The two programmers switch their roles every hour or so..

lists of different kinds of Agile SDLC models that are used by software development companies.
These are:
• Feature-driven development
• Crystal
• Atern
• Scrum
• Unified process
• Extreme programming (XP)
• Lean development

SOFTWARE RE-ENGINEERING
• When we need to update the software to keep it to the current market, without impacting its functionality, it is called software re-engineering. It is a thorough process where the design of software is changed and programs are re-written.
• Legacy software cannot keep tuning with the latest technology available in the market. As the hardware become obsolete, updating of software becomes a headache. Even if software grows old with time, its functionality does not.

RE-ENGINEERING PROCESS
• Decide what to re-engineer. Is it whole software or a part of it?
• Perform Reverse Engineering, in order to obtain specifications of existing software.
• Restructure Program if required. For example, changing function-oriented programs into object-oriented programs.
• Re-structure data as required.
• Apply Forward engineering concepts in order to get re-engineered software.

REVERSE ENGINEERING
• It is a process to achieve system specification by thoroughly analyzing, understanding the existing system. This process can be seen as reverse SDLC model
• An existing system is previously implemented design, about which we know nothing. Designers then do reverse engineering by looking at the code and try to get the design. With design in hand, they try to conclude the specifications. Thus, going in reverse from code to system specification.
• The purpose of reverse engineering is to facilitate the maintenance work by improving the understandability of a system and to produce the necessary documents for a legacy system.

REVERSE ENGINEERING GOALS:
• Cope with Complexity.
• Recover lost information.
• Detect side effects.
• Synthesise higher abstraction.
• Facilitate Reuse.

Advantages and Disadvantages of Waterfall-Model

Advantages

- Before the next phase of development, each phase must be completed.
- Suited for smaller projects where requirements are well defined .
- They should perform quality assurance test
- (Verification and Validation) before completing each stage .
- Elaborate documentation is done at every phase of the software's development cycle
- Project is completely dependent on project team with minimum client intervention
- Any changes in software is made during the process of the development.

Dis-Advantages

- Error can be fixed only during the phase.
- It is not desirable for complex project where requirement changes frequently.
- Testing period comes quite late in the developmental process.
- Documentation occupies a lot of time of developers and testers.
- Clients valuable feedback cannot be included with minimum client intervention ongoing development phase
- Small changes or errors that arise in the completed software may cause a lot of problems.

INCREMENTAL MODEL IN SDLC

• Incremental Model is a software development process where requirements are divided into several stand-alone software development modules.

• In this example, each module passes through the requirement, design, development, implementation, and testing phases.

• That subsequent release of the module adds a feature to the previous release.

• The process will continue until the whole software is achieved.

Characteristics of an Incremental module includes

• System development is broken down into many mini development projects

• Partial systems are successively built to produce a final total system

• Highest priority requirement is tackled first

• Once the requirement is developed, requirement for that increment are frozen

• **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.

• **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.

• **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.

• **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.