The Software Process Model

A software process model is a specified definition of a software process, which is presented from a particular perspective. Models, by their nature, are a simplification, so a software process model is an abstraction of the actual process, which is being described. Process models may contain activities, which are part of the software process, software product, and the roles of people involved in software engineering. Some examples of the types of software process models that may be produced are:

1. **A workflow model:** This shows the series of activities in the process along with their inputs, outputs and dependencies. The activities in this model perform human actions.
2. **2. A dataflow or activity model:** This represents the process as a set of activities, each of which carries out some data transformations. It shows how the input to the process, such as a specification is converted to an output such as a design. The activities here may be at a lower level than activities in a workflow model. They may perform transformations carried out by people or by computers.
3. **3. A role/action model:** This means the roles of the people involved in the software process and the activities for which they are responsible.

There are several various general models or paradigms of software development:

1. **The waterfall approach:** This takes the above activities and produces them as separate process phases such as requirements specification, software design, implementation, testing, and so on. After each stage is defined, it is "signed off" and development goes onto the following stage.
2. **Evolutionary development:** This method interleaves the activities of specification, development, and validation. An initial system is rapidly developed from a very abstract specification.
3. **Formal transformation:** This method is based on producing a formal mathematical system specification and transforming this specification, using mathematical methods to a program. These transformations are 'correctness preserving.' This means that you can be sure that the developed programs meet its specification.
4. **System assembly from reusable components:** This method assumes the parts of the system already exist. The system development process target on integrating these parts rather than developing them from scratch.

Software Crisis

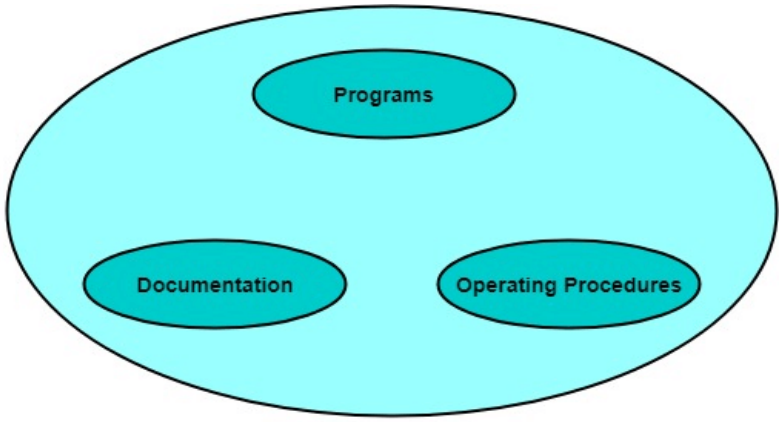
1. **Size:** Software is becoming more expensive and more complex with the growing complexity and expectation out of software. For example, the code in the consumer product is doubling every couple of years.
2. **Quality:** Many software products have poor quality, i.e., the software products defects after putting into use due to ineffective testing technique. For example, Software testing typically finds 25 errors per 1000 lines of code.

3. **Cost:** Software development is costly i.e. in terms of time taken to develop and the money involved. For example, Development of the FAA's Advanced Automation System cost over $700 per lines of code.
4. **Delayed Delivery:** Serious schedule overruns are common. Very often the software takes longer than the estimated time to develop, which in turn leads to cost shooting up. For example, one in four large-scale development projects is never completed.

Program vs. Software

Software is more than programs. Any program is a subset of software, and it becomes software only if documentation & operating procedures manuals are prepared.

There are three components of the software as shown in fig:

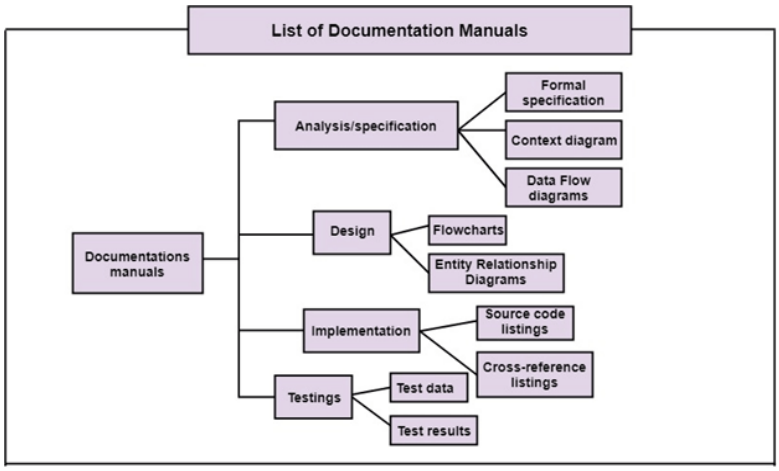


Software= Program + Documentation + Operating Procedures

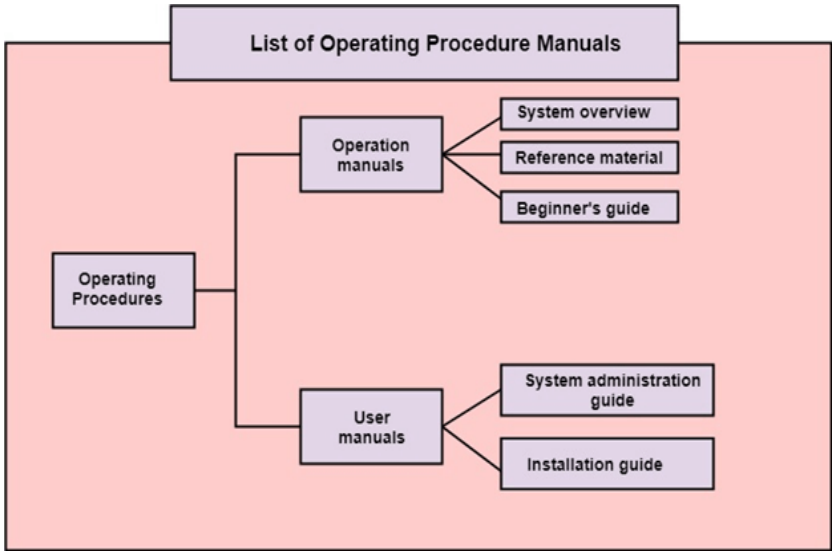
Fig:Components of Software

1. Program: Program is a combination of source code & object code.

2. Documentation: Documentation consists of different types of manuals. Examples of documentation manuals are: Data Flow Diagram, Flow Charts, ER diagrams, etc.



3. Operating Procedures: Operating Procedures consist of instructions to set up and use the software system and instructions on how react to the system failure. Example of operating system procedures manuals is: installation guide, Beginner's guide, reference guide, system administration guide, etc.



Software Development Life Cycle (SDLC)

A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.

In other words, a life cycle model maps the various activities performed on a software product from its inception to retirement. Different life cycle models may plan the necessary development activities to phases in different ways. Thus, no element which life cycle model is followed, the essential activities are contained in all life cycle models though the action may be carried out in distinct orders in different life cycle models. During any life cycle stage, more than one activity may also be carried out.

Need of SDLC

The development team must determine a suitable life cycle model for a particular plan and then observe to it.

Without using an exact life cycle model, the development of a software product would not be in a systematic and disciplined manner. When a team is developing a software product, there must be a clear understanding among team representative about when and what to do. Otherwise, it would point to chaos and project failure. This problem can be defined by using an example. Suppose a software development issue is divided into various parts and the parts are assigned to the team members. From then on, suppose the team representative is allowed the freedom to develop the roles assigned to them in whatever way they like. It is possible that one representative might start writing the code for his part, another might choose to prepare the test documents first, and some other engineer might begin with the design phase of the roles assigned to him. This would be one of the perfect methods for project failure.

A software life cycle model describes entry and exit criteria for each phase. A phase can begin only if its stage-entry criteria have been fulfilled. So without a software life cycle model, the entry and exit criteria for a stage cannot be recognized. Without software life cycle models, it becomes tough for software project managers to monitor the progress of the project.

SDLC Cycle

SDLC Cycle represents the process of developing software. SDLC framework includes the following steps:

The stages of SDLC are as follows:

Stage1: Planning and requirement analysis

Requirement Analysis is the most important and necessary stage in SDLC.

The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry.

Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.

Business analyst and Project organizer set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, what is the objective of the product. Before creating a product, a core understanding or knowledge of the product is very necessary.

For Example, A client wants to have an application which concerns money transactions. In this method, the requirement has to be precise like what kind of operations will be done, how it will be done, in which currency it will be done, etc.

Once the required function is done, an analysis is complete with auditing the feasibility of the growth of a product. In case of any ambiguity, a signal is set up for further discussion.

Once the requirement is understood, the SRS (Software Requirement Specification) document is created. The developers should thoroughly follow this document and also should be reviewed by the customer for future reference.

Stage2: Defining Requirements

Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted from the project stakeholders.

This is accomplished through "SRS"- Software Requirement Specification document which contains all the product requirements to be constructed and developed during the project life cycle.

Stage3: Designing the Software

The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.

Stage4: Developing the project

In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.

Stage5: Testing

After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.

During this stage, unit testing, integration testing, system testing, acceptance testing are done.

Stage6: Deployment

Once the software is certified, and no bugs or errors are stated, then it is deployed.

Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment.

After the software is deployed, then its maintenance begins.

Stage7: Maintenance

Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time.

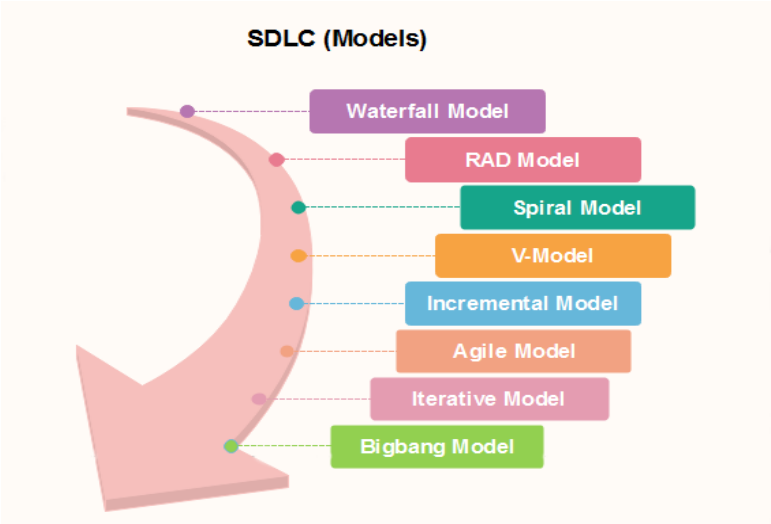
This procedure where the care is taken for the developed product is known as maintenance.

SDLC Models

Software Development life cycle (SDLC) is a spiritual model used in project management that defines the stages include in an information system development project, from an initial feasibility study to the maintenance of the completed application.

There are different software development life cycle models specify and design, which are followed during the software development phase. These models are also called "**Software Development Process Models**." Each process model follows a series of phase unique to its type to ensure success in the step of software development.

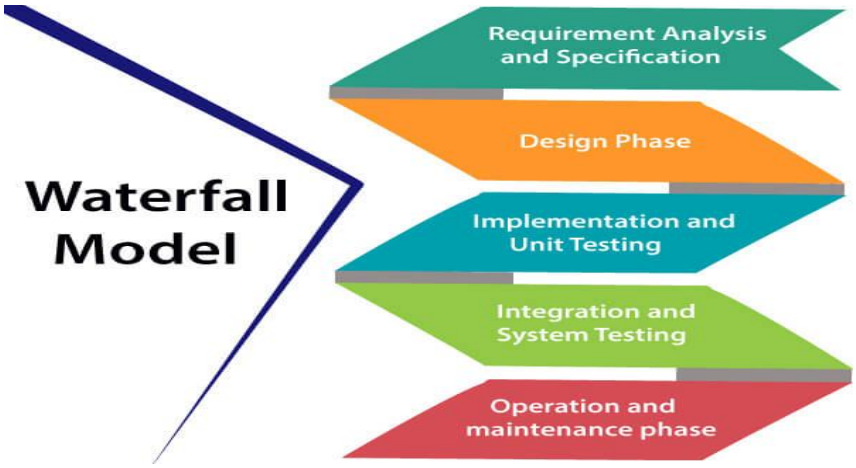
Here, are some important phases of SDLC life cycle:



Waterfall model

Winston Royce introduced the Waterfall Model in 1970.This model has five phases: Requirements analysis and specification, design, implementation, and unit testing, integration and system testing, and operation and maintenance. The steps always follow in this order and do not overlap. The developer must complete every phase before the next phase begins. This model is named "**Waterfall Model**", because its diagrammatic representation resembles a cascade of waterfalls.

1. Requirements analysis and specification phase: The aim of this phase is to understand the exact requirements of the customer and to document them properly. Both the customer and the software developer work together so as to document all the functions, performance, and interfacing requirement of the software. It describes the "what" of the system to be produced and not "how."In this phase, a large document called **Software Requirement Specification (SRS)** document is created which contained a detailed description of what the system will do in the common language.



2. Design Phase: This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language. It defines the overall software architecture together with high level and detailed design. All this work is documented as a Software Design Document (SDD).

3. Implementation and unit testing: During this phase, design is implemented. If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD.

During testing, the code is thoroughly examined and modified. Small modules are tested in isolation initially. After that these modules are tested by writing some overhead code to check the interaction between these modules and the flow of intermediate output.

4. Integration and System Testing: This phase is highly crucial as the quality of the end product is determined by the effectiveness of the testing carried out. The better output will lead to satisfied customers, lower maintenance costs, and accurate results. Unit testing determines the efficiency of

individual modules. However, in this phase, the modules are tested for their interactions with each other and with the system.

5. Operation and maintenance phase: Maintenance is the task performed by every user once the software has been delivered to the customer, installed, and operational.

When to use SDLC Waterfall Model?

Some Circumstances where the use of the Waterfall model is most suited are:

- When the requirements are constant and not changed regularly.
- A project is short
- The situation is calm
- Where the tools and technology used is consistent and is not changing
- When resources are well prepared and are available to use.

Advantages of Waterfall model

- This model is simple to implement also the number of resources that are required for it is minimal.
- The requirements are simple and explicitly declared; they remain unchanged during the entire project development.
- The start and end points for each phase is fixed, which makes it easy to cover progress.
- The release date for the complete product, as well as its final cost, can be determined before development.
- It gives easy to control and clarity for the customer due to a strict reporting system.

Disadvantages of Waterfall model

- In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
- This model cannot accept the changes in requirements during development.
- It becomes tough to go back to the phase. For example, if the application has now shifted to the coding phase, and there is a change in requirement, It becomes tough to go back and change it.
- Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare.

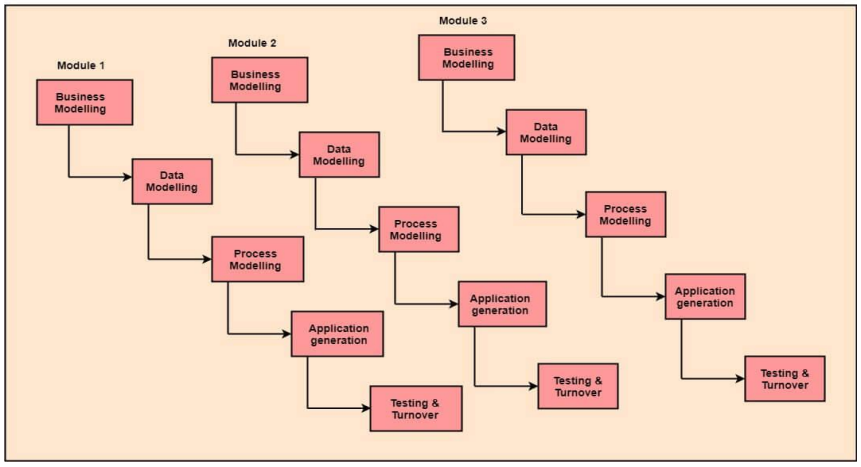
RAD (Rapid Application Development) Model

RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element based construction approach. If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a concise time period.

RAD (Rapid Application Development) is a concept that products can be developed faster and of higher quality through:

- Gathering requirements using workshops or focus groups
- Prototyping and early, reiterative user testing of designs
- The re-use of software components
- A rigidly paced schedule that refers design improvements to the next product version
- Less formality in reviews and other team communication

Fig: RAD Model



The various phases of RAD are as follows:

1.Business Modelling: The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.

2. Data Modelling: The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.

3. Process Modelling: The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

4. Application Generation: Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.

5. Testing & Turnover: Many of the programming components have already been tested since RAD emphasis reuse. This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised.

When to use RAD Model?

- When the system should need to create the project that modularizes in a short span time (2-3 months).
- When the requirements are well-known.
- When the technical risk is limited.
- When there's a necessity to make a system, which modularized in 2-3 months of period.
- It should be used only if the budget allows the use of automatic code generating tools.

Advantage of RAD Model

- This model is flexible for change.
- In this model, changes are adoptable.
- Each phase in RAD brings highest priority functionality to the customer.
- It reduced development time.
- It increases the reusability of features.

Disadvantage of RAD Model

- It required highly skilled designers.
- All application is not compatible with RAD.
- For smaller projects, we cannot use the RAD model.
- On the high technical risk, it's not suitable.
- Required user involvement.

Spiral Model

The spiral model, initially proposed by Boehm, is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model. It implements the potential for rapid development of new versions of the software. Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.

The Spiral Model is shown in fig:

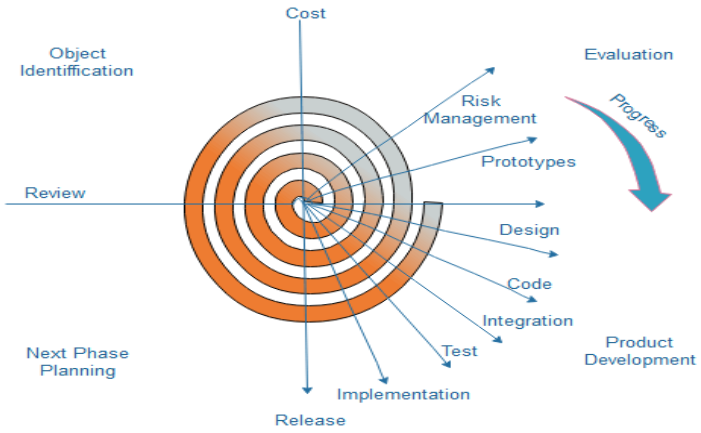


Fig. Spiral Model

Each cycle in the spiral is divided into four parts:

Objective setting: Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists.

Risk Assessment and reduction: The next phase in the cycle is to calculate these various alternatives based on the goals and constraints. The focus of evaluation in this stage is located on the risk perception for the project.

Development and validation: The next phase is to develop strategies that resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.

Planning: Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, if performance or user-interface risks are treated more essential than the program development risks, the next phase may be an evolutionary development that includes developing a more detailed prototype for solving the risks.

The **risk-driven** feature of the spiral model allows it to accommodate any mixture of a specification-oriented, prototype-oriented, simulation-oriented, or another type of approach. An essential element of the model is that each period of the spiral is completed by a review that includes all the products developed during that cycle, including plans for the next cycle. The spiral model works for development as well as enhancement projects.

When to use Spiral Model?

- When deliverance is required to be frequent.
- When the project is large
- When requirements are unclear and complex
- When changes may require at any time

- Large and high budget projects

Advantages of the Spiral Model

There are so many benefits of using the spiral model. These are as follows:

- The amount of risk should be reduced in this model.
- This model is best for large and critical projects.
- It provides the developer with solid approval and documentation control.
- The software production time is less in this spiral model.

Disadvantages of Spiral Model

It also has some disadvantages to using the spiral model. These are as follows.

- We need a lot of financial support to develop a project.
- We should not use this model for a very small project.

Incremental Model

Incremental Model is a process of software development where requirements 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 achieved.

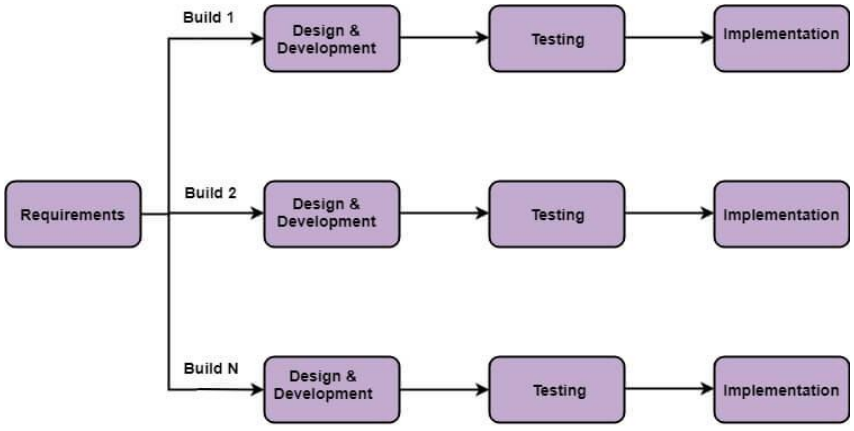


Fig: Incremental Model

The 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.

Iterative Model

In this Model, you can start with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.

The Iterative Model allows the accessing earlier phases, in which the variations made respectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.

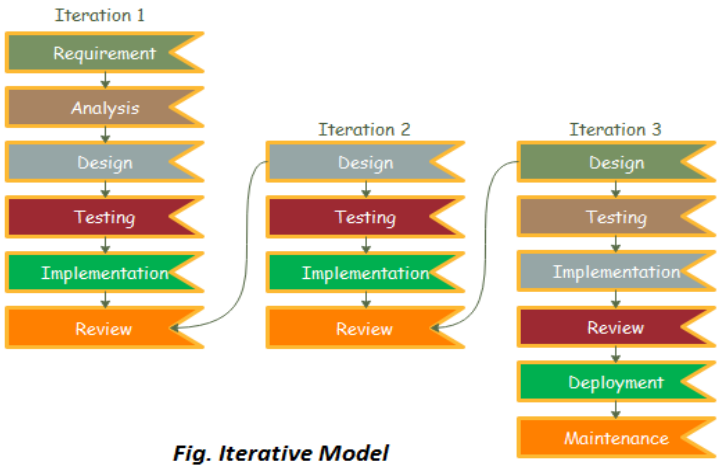


Fig. Iterative Model

The various phases of Iterative model are as follows:

1. Requirement gathering & analysis: In this phase, requirements are gathered from customers and check by an analyst whether requirements will fulfil or not. Analyst checks that need will achieve within budget or not. After all of this, the software team skips to the next phase.

2. Design: In the design phase, team design the software by the different diagrams like Data Flow diagram, activity diagram, class diagram, state transition diagram, etc.

3. Implementation: In the implementation, requirements are written in the coding language and transformed into computer programmes which are called Software.

4. Testing: After completing the coding phase, software testing starts using different test methods. There are many test methods, but the most common are white box, black box, and grey box test methods.

5. Deployment: After completing all the phases, software is deployed to its work environment.

6. Review: In this phase, after the product deployment, review phase is performed to check the behaviour and validity of the developed product. And if there are any error found then the process starts again from the requirement gathering.

7. Maintenance: In the maintenance phase, after deployment of the software in the working environment there may be some bugs, some errors or new updates are required. Maintenance involves debugging and new addition options.

When to use the Iterative Model?

1. When requirements are defined clearly and easy to understand.
2. When the software application is large.
3. When there is a requirement of changes in future.

Advantage(Pros) of Iterative Model:

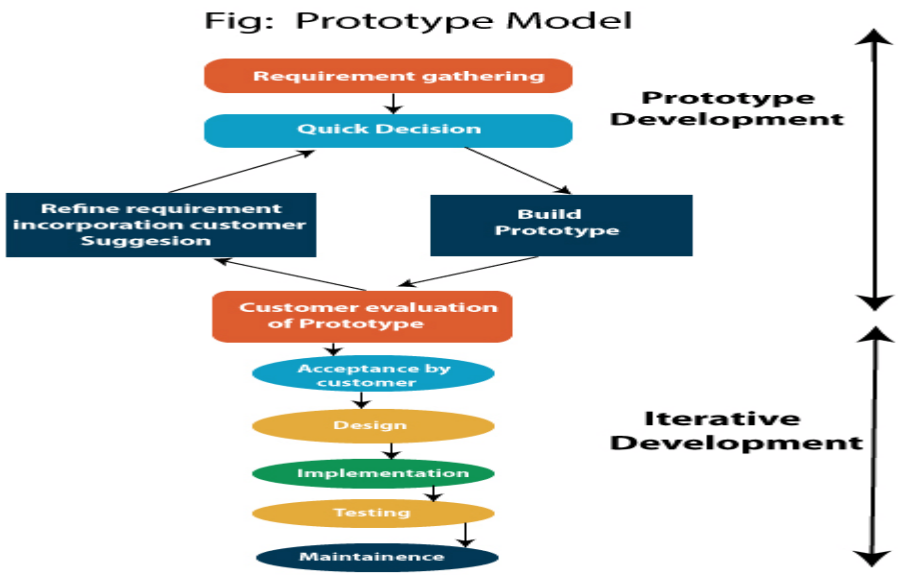
1. Testing and debugging during smaller iteration is easy.
2. A Parallel development can plan.
3. It is easily acceptable to ever-changing needs of the project.
4. Risks are identified and resolved during iteration.
5. Limited time spent on documentation and extra time on designing.

Disadvantage(Cons) of Iterative Model:

1. It is not suitable for smaller projects.
2. More Resources may be required.
3. Design can be changed again and again because of imperfect requirements.
4. Requirement changes can cause over budget.
5. Project completion date not confirmed because of changing requirements.

Prototype Model

The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possible exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software. In many instances, the client only has a general view of what is expected from the software product. In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs, and the output requirement, the prototyping model may be employed.



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.

Evolutionary Process Model

Evolutionary process model resembles the iterative enhancement model. The same phases are defined for the waterfall model occurs here in a cyclical fashion. This model differs from the iterative enhancement model in the sense that this does not require a useful product at the end of each cycle. In evolutionary development, requirements are implemented by category rather than by priority.

For example, in a simple database application, one cycle might implement the graphical user Interface (GUI), another file manipulation, another queries and another updates. All four cycles must complete before there is a working product available. GUI allows the users to interact with the system, file manipulation allow the data to be saved and retrieved, queries allow user to get out of the system, and updates allows users to put data into the system.

Benefits of Evolutionary Process Model

Use of EVO brings a significant reduction in risk for software projects.

EVO can reduce costs by providing a structured, disciplined avenue for experimentation.

EVO allows the marketing department access to early deliveries, facilitating the development of documentation and demonstration.

Better fit the product to user needs and market requirements.

Manage project risk with the definition of early cycle content.

Uncover key issues early and focus attention appropriately.

Increase the opportunity to hit market windows.

Accelerate sales cycles with early customer exposure.

Increase management visibility of project progress.

Increase product team productivity and motivations.

Characteristics of the Evolutionary Model

There are so many characteristics of using the evolutionary model in our project. These characteristics are as follows.

- We can develop the evolutionary model with the help of an iterative waterfall model of development.
- There are three types of evolutionary models. These are the Iterative model, Incremental model and Spiral model.
- Many primary needs and architectural planning must be done
- to implement the evolutionary model.
- When the new product version is released, it includes the new functionality and some changes in the existing product, which are also released with the latest version.
- This model also permits the developer to change the requirement, and the developer can divide the process into different manageable work modules.
- The development team also have to respond to customer feedback throughout the development process by frequently altering the product, strategy, or process.

Advantages of the Evolutionary Model

There are so many benefits of using the evolutionary model. These benefits are as follows.

- In the evolutionary model, the user can test partially developed software before the release of the final product in the market.
- In this model, we can test the core module, which reduces the chances of errors in the final product.
- We can use this module only for large products.
- We can use this model to develop a software product with some unique features. We can modify these specific features based on customer feedback and many factors throughout development.
- With the help of this evolutionary model, we can get to know the client's requirements during the delivery of different versions of the software.
- After completion of each cycle, the user can access the product.
- Using an evolutionary model removes the requirement to allocate significant resources simultaneously for system development.

Disadvantages of the Evolutionary Model

There are also many disadvantages to using this model. These are as follows.

- It is difficult to divide the problem into functionality units that the user accepts. After receiving the user, this problem should be incrementally implemented, and after that, it should be delivered.
- There will be a chance of being late in the delivery of the final product. There will be a chance that the market can go down due to different customer changes during development.
- In this module, the chances of risk factors are high. There is always a need to report customers continuously.

Comparison of the Evolutionary Model with Other Models

1. Evolutionary Model vs Incremental Model

Evolutionary	Incremental
The requirement of this evolutionary model is not a requirement. The development can be changed according to the development process.	The requirements for this incremental model are precise and based on the development team.
In this evolutionary model, the initial step is understanding the customer's requirements. Also, it involves the development of the core modules and functionality. After completing all the phases, they have to deliver the product to the customer for feedback.	In this incremental model, each module is defined with multiple iterations. We can also add new functionality during the development of the product. After completing all the steps, we have to release the deliverable product.
The activities of the complete cycle are repeated for each new product version.	Each module is developed, tested, and released at various intervals.
In this model, the development time is unknown for the developer. Also, the developer can not track the progress of the software product.	In this model, the development time is known to the developer. Also, the developer can track the progress of the software product.

2. Evolutionary Model vs Iterative Model

Evolutionary Model	Iterative Model
In this model, different modules are released in incremental order before releasing the final product.	It follows the process of the sequential development process. After completion of the development of the software product, it releases the final product.

There will be a chance of delivery of actual product be late.	In this model, the product will be delivered within the time.
In this module, the integration of the module will be complex.	This model makes it much easier to understand and implement the product.
We can use this module only for large products.	It is the most commonly used module in the software industry.

3. Evolutionary Model vs Classical Waterfall Model

Evolutionary Model	Classical Waterfall Model
In this model, we can deal with the different versions of the software.	In this model, we can deal with things according to the things mentioned in the testing manual.
In this model, integration is going to be complicated.	In this model, it is going to be simple to use, understand and implement.
This model accepts customer feedback during the development of the project.	In this model, there is no need for customer feedback.
In this model, we can detect the error in the core module of the project.	In this phase, we can detect the error in every development step.
If we need to add any new functionality, we have to release the latest product, and the existing functionalities may be updated.	After the completion of the development of the final product, we can not add any update to it.

4. Evolutionary Model vs Spiral Model

Evolutionary Model	Spiral Model
In this model, we can develop the software incrementally with the help of different modules.	In this model, all the modules are divided into many loops, further divided into four quadrants.

We can use this module only for large products.	We can use this model to develop technically difficult software products vulnerable to different threats.
In this model, Every version can fully function the mentioned functionalities.	In this module, each model contains a set of activities that the software performs.

OR

The Software Process Model

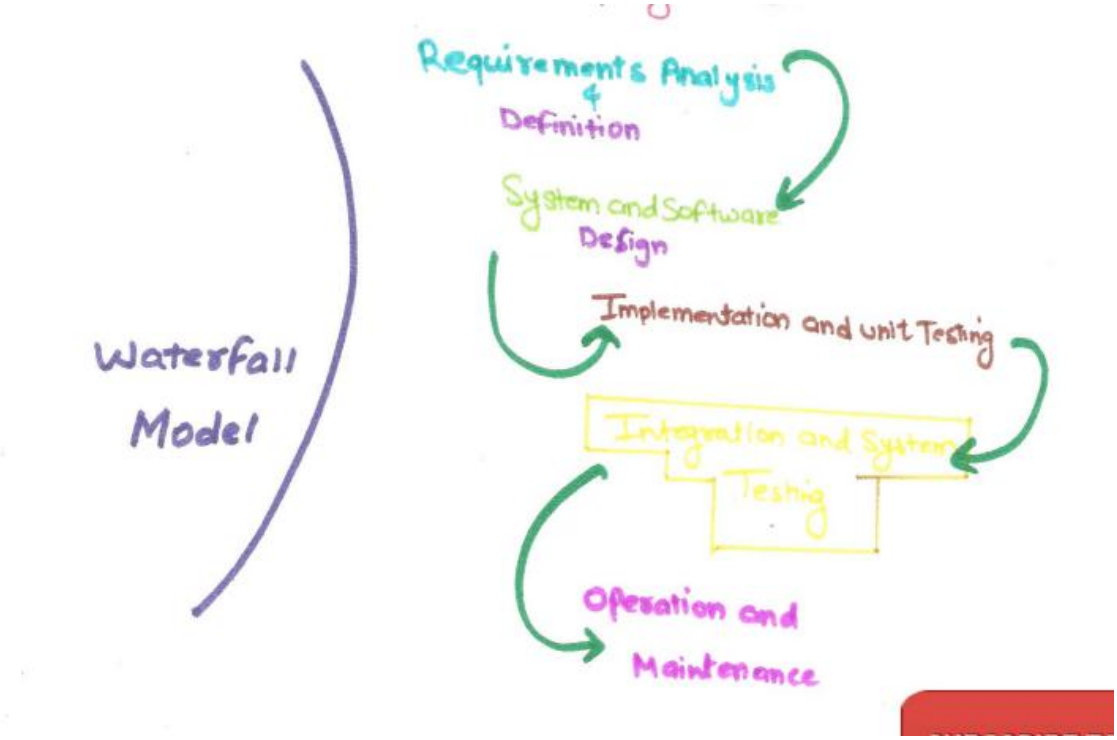
A software process model is a specified definition of a software process, which is presented from a particular perspective. Models, by their nature, are a simplification, so a software process model is an abstraction of the actual process, which is being described. Process models may contain activities, which are part of the software process, software product, and the roles of people involved in software engineering. Some examples of the types of software process models that may be produced are:

1. **A workflow model:** This shows the series of activities in the process along with their inputs, outputs and dependencies. The activities in this model perform human actions.
2. **2. A dataflow or activity model:** This represents the process as a set of activities, each of which carries out some data transformations. It shows how the input to the process, such as a specification is converted to an output such as a design. The activities here may be at a lower level than activities in a workflow model. They may perform transformations carried out by people or by computers.
3. **3. A role/action model:** This means the roles of the people involved in the software process and the activities for which they are responsible.

Waterfall Model OR Classical Waterfall Model:

- **Waterfall model is a simplest model of software development paradigm.**
- **All the phases of sdlc will function one after in linear manner. That is, when the first phase is finished then only the second phase will start and so on.**

The Waterfall model was first process model to be introduce and also called **"Linear-Sequential Life Cycle Model.**



This type of model is basically used for the project which is small and there are **"No Uncertain Requirements".**

At the end of a phase, a review take place to determine if the project is on the right path and whether or not to continue or discard the project.

In this testing starts only after the development is completed and phases do not overlap.

Advantages of WaterFall Model:

1) Discipline: This staged development cycle enforces discipline. Every phase has a start and end point, and progress can be conclusively identified(through the use of milestones) by both vendor and client.

2) minimal wastage:- the emphasis on requirement and design before writing a single line of code ensure **minimal wastage of time and efforts and reduces the risk of schedule slippage.**

3) improves quality:- It is much easier to catch and correct possible flows at the design stage then at the testing stage after all the component have been **integrated and tracking down specific error is more complex.**

Disadvantages of WaterFall Model:

Uncertain Nature of Customer Needs:- due to this estimating time and costs with any degree of accuracy(as the model subjects) is often extremely difficult.

Implicit Assumption to Roadblock: Implicit assumption that designs can be feasibly translated into real products, they sometimes runs into roadblock when developer begin implementation.

No Good Model: Not a good model for Complex and object oriented project as world changing fast as technology changes.

- **It haas High amount of risk and uncertainty.**
- **It is somewhere poor model for long and ongoing projects.**
- **It is Not suitable for the projects where requirements are at a moderate to higher risk of changing.**

Software Process Model: Spiral Model

Spiral Model was originally proposed by **"Boehm".** Rather than represent the software process as a **sequence of activities** with some **backtracking** from one activity to the another, the process is represented as a **SPIRAL.**

Each loop in spiral model represents a phase of the Software Process.

In this model, developers define and implement features in order of Decreasing Priority.

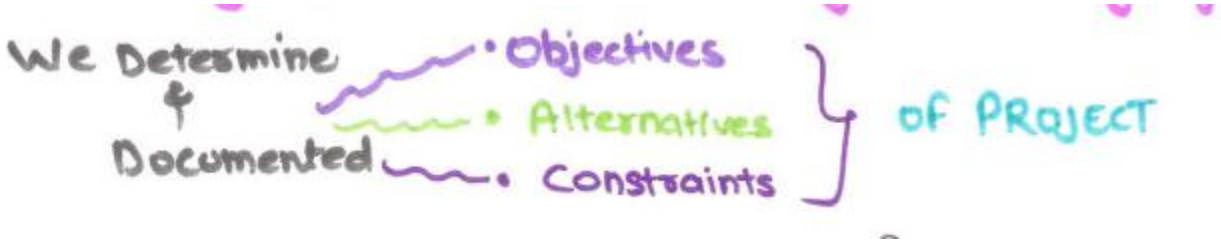
Exact numbers of loops in the spiral is **"Not fixed".**

This model considers **RISK,** which often goes **"UN-Noticed"** by most other models.

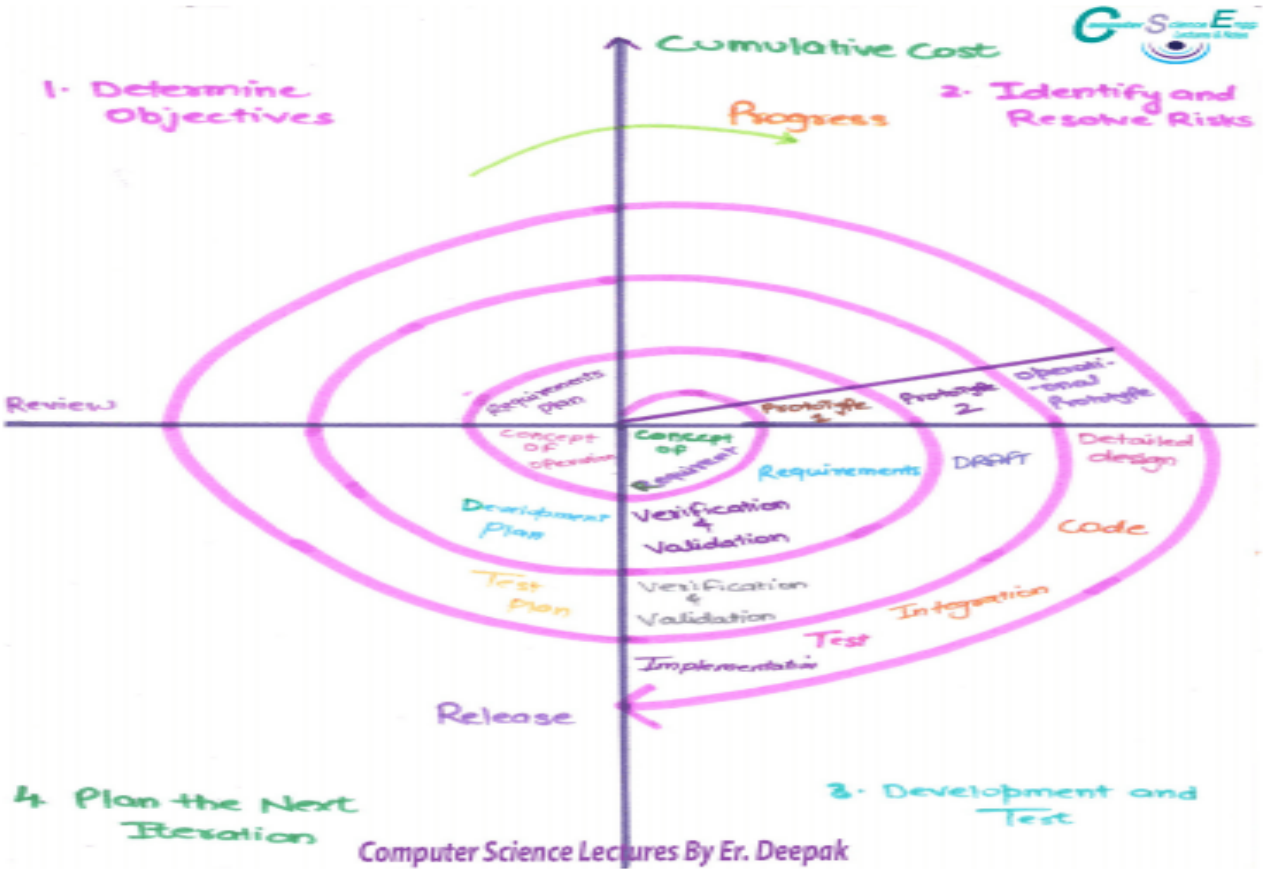
Each loop of the spiral represent a phase of the software process. And it has four phases.

The Four Phases of Spiral Model

1) Determining Object or **Objective Setting** or **Planning Object**



A detailed management plan is drawn up and "Project Risk" are identified.

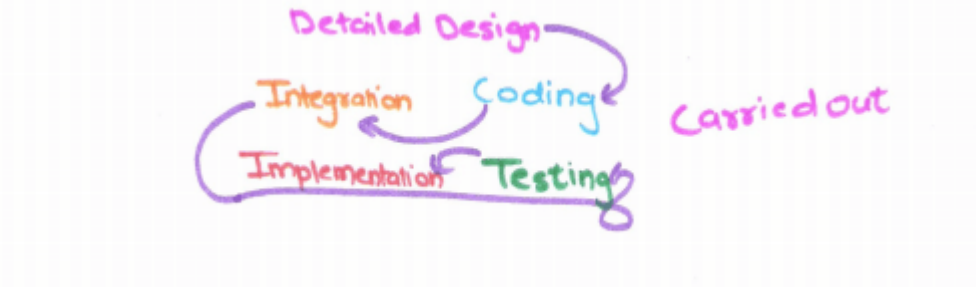


2) Identify And Resolve Risks or **Risk Assessment And Reduction** or **Risk Analysis:**

This phase has been added specially in order to identify and resolve all possible risks in project development. If RISKS indicate any kind of UNCERTAINLY in requirement, Prototyping may be used to proceed with the available data and find out solution in order to deal with the Potential changes in the Requirements.

3) Development and Test or **Engineering** or **Development And Validation:**

After resolving the identified risk, the actual development of the project is carried out in this..



4) Plan Next Iteration or **Planning** or **Reviewing And Planning:**

The prototype is reviewed and decision made whether to continue with a further loop of the spiral. It is decided to continue, plans are **drawn up for the next phase of the project.**

Advantages Of Spiral Model:

Later Stage Changes: **Addition functionality or changes can be done at a later stages as proper planning is done at each Iteration.**

Easy Cost Estimation: As **Prototype** building is done in **small fragments** by which cost estimation becomes easy.

Good Risk Management: As **Risk Analysis Stage** comes in very early stages and **continuous or repeated development** helps in **Risk Management.**

Systematic Development: **Delopment is fast and features can be added in a systematic way.**

Space For Customer Feedback: There is continuous feedback is taken at each **ITERATION.**

Large Project Oriented: It is very good model for handling large and **Mission Critical Projects.**

Better Project Monitoring: Project Monitoring is very "Easy And Effective". Each Phase as well as "Each Loop, requires a Review from concerned people. This makes people model more Transparent.

Disdvantages Of Spiral Model:

High Cost: Cost involved in this model is quite **"HIGH"** as more planning, prototype version, risk management has to be done.

Need well expertise: Skills required to evaluate and review project from time to time, need expertise.

No Reusablity of prototype: Due to various customization from client, using same prototype for the other project in future is difficult.

Not suitable for small project: It is quite difficult to follow this strategy for **Small Projects** due to

Low Budget

Time Constraint

Large Procedures.

What is the iterative process?

The iterative process is the practice of building, refining, and improving a project, product, or initiative. Teams that use the iterative development process create, test, and revise until they're satisfied with the end result. You can think of an iterative process as a trial-and-error methodology that brings your project closer to its end goal.

Iterative processes are a fundamental part of lean methodologies and Agile project management—but these processes can be implemented by any team, not just Agile ones. During the iterative process, you will continually improve your design, product, or project until you and your team are satisfied with the final project deliverable.

Example iterative processes

Engineering

Many engineering teams use the iterative process to develop new features, implement bug fixes, or A/B test new strategies. Often, an engineering team will create a few iterations that they think are equally promising, then test them with users. They'll note pain points and successes, and then continue building out the one that tested the best.

Product development

You might be surprised to realize that most product development is very iterative. Think of any personal technology you've ever purchased for yourself—there was likely a previous version before the one you bought, and maybe a version afterwards, as well. Think of the development of mobile phones throughout the years, how speakers have gotten smaller and more portable over time, or even the way refrigerators from the same brands have changed to adapt to new family needs. All of these are iterative processes.

Marketing

Some marketing teams embrace iterative processes, others not so much. But to a certain extent, a lot of marketing is iterative. For example, some marketing teams might test different advertising copy to see which one gets better engagement, or send out two versions of an email newsletter to compare click-through rates. Alternatively, a brand marketing team could use iterative design processes to identify the imagery that works best for their target audience.

Sales

Though most of a sales team's customer-facing work isn't iterative, some of their tasks can benefit from iterative processes. For example, a sales team might take an iterative approach to sending cold emails. They might have their reps send a few different email subject lines and analyze the results. Then, the team can implement the most successful subject lines moving forward.

The 5 steps of the iterative process

The iterative process can help you during the lifecycle of a project. During the steps of the iterative process, your goals and requirements will serve as the project's starting point. Then, your team will use testing, prototyping, and iteration to achieve the best possible result. Here's how:

1. Planning and requirements

During this step in the iterative process, you will define your project plan and align on your overall project objectives. This is the stage where you will outline any hard requirements—things that must happen in order for your project to succeed. Without this step, you run the risk of iterating but not hitting your goals.

2. Analysis and design

During this step, you and your team will focus on the business needs and technical requirements of your project. If step one was the process of outlining your goals, step two is when you brainstorm a design that will help you ultimately hit those goals.

3. Implementation

During the third step, your team will create the first iteration of your project deliverable. This iteration will be informed by your analysis and design, and should work to hit your ultimate project objective. The level of detail and time you spend on this iteration will depend on the project.

4. Testing

Now that you have an iteration, you will test it in whatever way makes the most sense. If you're working on an improvement to a web page, for example, you might want to A/B test it against your current web page. If you're creating a new product or feature, consider doing usability testing with a set of potential customers.

In addition to testing, you should also check in with your project stakeholders. Ask them to weigh in on the iteration, and provide any feedback.

Read: What is the Plan-Do-Check-Act (PDCA) cycle?

5. Evaluation and review

After testing, your team will evaluate the success of the iteration and align on anything that needs to change. Does this iteration achieve your project objectives? Why, or why not? If something needs to change, you can restart the iterative process by going back to step two to create the next iteration. Keep in mind that your initial planning and goals should remain the same for all iterations. Continue building upon the previous iteration until you get to a deliverable you're happy with.

If you restart the iterative process, make sure everyone is still aligned on your project goals. The iterative process can take weeks or months, depending on how many iterations you run through. Centering your iteration on your project objectives every time

you restart the iterative process can help you ensure you don't lose track of your north star.

Iterative Model

In this Model, you can start with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.

The Iterative Model allows the accessing earlier phases, in which the variations made respectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.

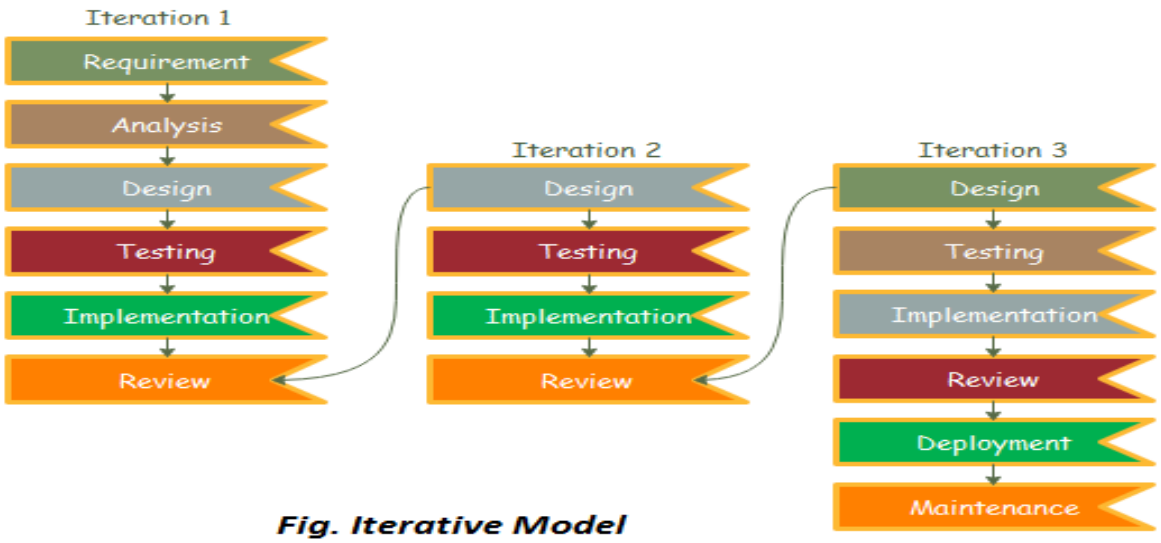


Fig. Iterative Model

The various phases of Iterative model are as follows:

1. Requirement gathering & analysis: In this phase, requirements are gathered from customers and check by an analyst whether requirements will fulfil or not. Analyst checks that need will achieve within budget or not. After all of this, the software team skips to the next phase.

2. Design: In the design phase, team design the software by the different diagrams like Data Flow diagram, activity diagram, class diagram, state transition diagram, etc.

3. Implementation: In the implementation, requirements are written in the coding language and transformed into computer programmes which are called Software.

4. Testing: After completing the coding phase, software testing starts using different test methods. There are many test methods, but the most common are white box, black box, and grey box test methods.

5. Deployment: After completing all the phases, software is deployed to its work environment.

6. Review: In this phase, after the product deployment, review phase is performed to check the behaviour and validity of the developed product. And if there are any error found then the process starts again from the requirement gathering.

7. Maintenance: In the maintenance phase, after deployment of the software in the working environment there may be some bugs, some errors or new updates are required. Maintenance involves debugging and new addition options.

When to use the Iterative Model?

1. When requirements are defined clearly and easy to understand.
2. When the software application is large.
3. When there is a requirement of changes in future.

Advantage(Pros) of Iterative Model:

1. Testing and debugging during smaller iteration is easy.
2. A Parallel development can plan.
3. It is easily acceptable to ever-changing needs of the project.
4. Risks are identified and resolved during iteration.
5. Limited time spent on documentation and extra time on designing.

Disadvantage(Cons) of Iterative Model:

1. It is not suitable for smaller projects.
2. More Resources may be required.
3. Design can be changed again and again because of imperfect requirements.
4. Requirement changes can cause over budget.
5. Project completion date not confirmed because of changing requirements.