UNIT-II

PROJECT LIFE CYCLE AND EFFORT ESTIMATION

Software process and process modes – Choice of process models – Rapid Application development – Agile methods – Dynamic System Development Method – Extreme Programming – Managing interactive processes – Basics of Software estimation – Effort and cost estimation techniques – COSMIC Full function points – COCOMO II.

Software process and process modes

Software is the set of instructions in the form of programs to govern the computer system and to process the hardware components. To produce a software product the set of activities is used. This set is called a software process.

Table of Content

What are Software Processes?

Software processes in software engineering refer to the methods and techniques used to develop and maintain software. Some examples of software processes include:

- **Waterfall**: a linear, sequential approach to software development, with distinct phases such as requirements gathering, design, implementation, testing, and maintenance.
- **Agile**: a flexible, iterative approach to software development, with an emphasis on rapid prototyping and continuous delivery.
- **Scrum**: a popular Agile methodology that emphasizes teamwork, iterative development, and a flexible, adaptive approach to planning and management.
- **DevOps**: a set of practices that aims to improve collaboration and communication between development and operations teams, with an emphasis on automating the software delivery process.

Each process has its own set of advantages and disadvantages, and the choice of which one to use depends on the specific project and organization.

Components of Software

There are three main components of the software:

1. **Program:** A computer program is a list of instructions that tell a computer what to do.
2. **Documentation:** Source information about the product contained in design documents, detailed code comments, etc.
3. **Operating Procedures:** Set of step-by-step instructions compiled by an organization to help workers carry out complex routine operations.

Other Software Components

Other Software Components are:

1. **Code**: the instructions that a computer executes in order to perform a specific task or set of tasks.

2. **Data**: the information that the software uses or manipulates.
3. **User** interface: the means by which the user interacts with the software, such as buttons, menus, and text fields.
4. **Libraries**: pre-written code that can be reused by the software to perform common tasks.
5. **Documentation**: information that explains how to use and maintain the software, such as user manuals and technical guides.
6. **Test cases**: a set of inputs, execution conditions, and expected outputs that are used to test the software for correctness and reliability.
7. **Configuration files**: files that contain settings and parameters that are used to configure the software to run in a specific environment.
8. **Build and deployment** scripts: scripts or tools that are used to build, package, and deploy the software to different environments.
9. **Metadata**: information about the software, such as version numbers, authors, and copyright information.

All these components are important for software development, testing and deployment.

Key Process Activities

There four basic key process activities are:

1. **Software Specifications:** In this process, detailed description of a software system to be developed with its functional and non-functional requirements.
2. **Software Development:** In this process, designing, programming, documenting, testing, and bug fixing is done.
3. **Software Validation:** In this process, evaluation software product is done to ensure that the software meets the business requirements as well as the end users needs.
4. **Software Evolution:** It is a process of developing software initially, then timely updating it for various reasons.

Software Crisis

The term "software crisis" refers to a set of problems that were faced by the software industry in the 1960s and 1970s, such as:

1. **Size and Cost:** Day to day growing complexity and expectation out of software. Software are more expensive and more complex.
2. **Quality:** Software products must have good quality.
3. **Delayed Delivery:** Software takes longer than the estimated time to develop, which in turn leads to cost shooting up.
4. **High costs and long development times:** software projects were taking much longer and costing much more than expected.
5. **Low quality**: software was often delivered late, with bugs and other defects that made it difficult to use.
6. **Lack of standardization**: there were no established best practices or standards for software development, making it difficult to compare and improve different approaches.
7. **Lack of tools and methodologies**: there were few tools and methodologies available to help with software development, making it a difficult and time-consuming process.

These problems led to a growing realization that the traditional approaches to software development were not effective and needed to be improved. This led to the development of new software development methodologies, such as the Waterfall and Agile methodologies, as well as the creation of new tools and technologies to support software development.

However, even today, software crisis could be seen in some form or the other, like for example software projects going over budget, schedule and not meeting the requirement.

Software Process Model

A software process model is an abstraction of the actual process, which is being described. It can also be defined as a simplified representation of a software process. Each model represents a process from a specific perspective.

Following are some basic software process models on which different type of software process models can be implemented:

1. **A workflow Model :** It is the sequential series of tasks and decisions that make up a business process.
2. **The Waterfall Model:** It is a sequential design process in which progress is seen as flowing steadily downwards.
   - Phases in waterfall model:
     - Requirements Specification
     - Software Design
     - Implementation
     - Testing
3. **Dataflow Model:** It is diagrammatic representation of the flow and exchange of information within a system.
4. **Evolutionary Development Model:** Following activities are considered in this method:
   - Specification
   - Development
   - Validation
5. **Role / Action Model:** Roles of the people involved in the software process and the activities.

Need for Process Model

The software development team must decide the process model that is to be used for software product development and then the entire team must adhere to it. This is necessary because the software product development can then be done systematically. Each team member will understand what is the next activity and how to do it. Thus process model will bring the definiteness and discipline in overall development process. Every process model consists of definite entry and exit criteria for each phase. Hence the transition of the product through various phases is definite.

If the process model is not followed for software development then any team member can perform any software development activity, this will ultimately cause a chaos and software project will definitely fail without using process model, it is difficult to monitor the progress of software product. Thus process model plays an important rule in software engineering.

Advantages or Disadvantages of Process Model

There are several advantages and disadvantages to different software development methodologies, such as:

Waterfall

Advantages of waterfall model are:

1. Clear and defined phases of development make it easy to plan and manage the project.
2. It is well-suited for projects with well-defined and unchanging requirements.

Disadvantages of waterfall model are:

1. Changes made to the requirements during the development phase can be costly and time-consuming.
2. It can be difficult to know how long each phase will take, making it difficult to estimate the overall time and cost of the project.

3. It does not have much room for iteration and feedback throughout the development process.

Agile

Advantages of Agile Model are:

1. Flexible and adaptable to changing requirements.
2. Emphasizes rapid prototyping and continuous delivery, which can help to identify and fix problems early on.
3. Encourages collaboration and communication between development teams and stakeholders.

Disadvantages of Agile Model are:

1. It may be difficult to plan and manage a project using Agile methodologies, as requirements and deliverables are not always well-defined in advance.
2. It can be difficult to estimate the overall time and cost of a project, as the process is iterative and changes are made throughout the development.

Scrum

Advantages of Scrum are:

1. Encourages teamwork and collaboration.
2. Provides a flexible and adaptive framework for planning and managing software development projects.
3. Helps to identify and fix problems early on by using frequent testing and inspection.

Disadvantages of Scrum are:

1. A lack of understanding of Scrum methodologies can lead to confusion and inefficiency.
2. It can be difficult to estimate the overall time and cost of a project, as the process is iterative and changes are made throughout the development.

DevOps

Advantages of DevOps are:

1. Improves collaboration and communication between development and operations teams.
2. Automates software delivery process, making it faster and more efficient.
3. Enables faster recovery and response time in case of issues.

Disadvantages of DevOps are:

1. Requires a significant investment in tools and technologies.
2. Can be difficult to implement in organizations with existing silos and lack of culture of collaboration.
3. Need to have a skilled workforce to effectively implement the devops practices.
4. Ultimately, the choice of which methodology to use depends on the specific project and organization, as well as the goals and requirements of the project.

Conclusion

Software processes provide structured methods for developing and maintaining software. They include approaches like Waterfall for linear projects, Agile for flexibility, Scrum for teamwork, and DevOps for automation and collaboration. Each has unique strengths, tailored to different project needs. Choosing the right process enhances efficiency and product quality.

Choice of process models

Software development models are various processes or methods that are chosen for project development depending on the objectives and goals of the project. Many development life cycle models have been developed to achieve various essential objectives. Models specify the various steps of the process and the order in which they are executed.

Table of Content

The choice of model has a great impact on the test to be conducted. This will define what aspects of the software are tested, where in the development process testing occurs, and when the testing activities are connected.

What is Software Modeling?

Software modeling is the process of creating abstract representations of a software system. These models serve as blueprints that guide developers, designers, and stakeholders through the system's structure, behavior, and functionality.

By using diagrams and various modeling languages, software modeling helps in visualizing and understanding the complex aspects of the software, making it easier to plan, develop, and manage the system.

Top 8 Software Development Models

Choosing the right model is very important for the development of a software product or application. Development and testing processes are carried out based on the model.

Popular-Software-Development-Models

Different companies, depending on the software application or product, choose the type of development model whichever is appropriate for their application. But these days 'Agile Methodology' is the most popular in the market.

Let's look at the every model in brief one by one:-

1. Waterfall Model

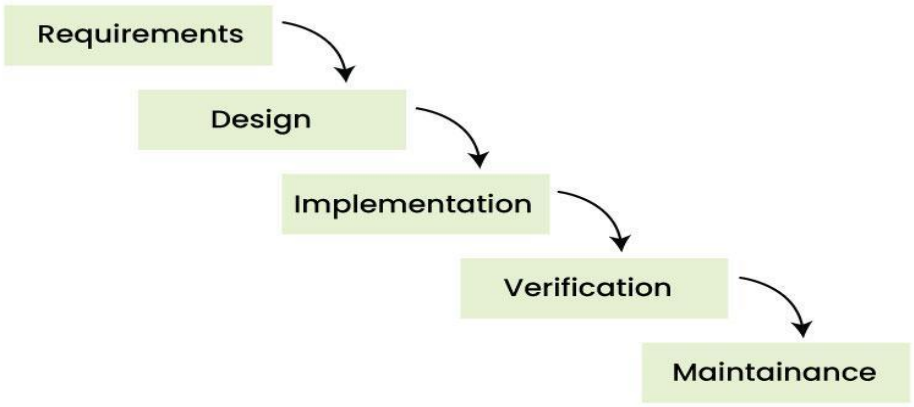
Waterfall model is a famous and good version of SDLC(System Development Life Cycle) for software engineering. The waterfall model is a linear and sequential model, which means that a development phase cannot begin until the previous phase is completed. We cannot overlap phases in waterfall model.

We can imagine waterfall in the following way

"Once the water starts flowing over the edge of the cliff, it starts falling down the mountain and the water cannot go back up."

Similarly waterfall model also works, once one phase of development is completed then we move to the next phase but cannot go back to the previous phase. In the waterfall model, the output of one phase serves as the input for the other phase.

Phases of Waterfall model



Waterfall Model

Waterfall Model

1. **Requirement phase:-** Requirement phase is the first phase of the waterfall model. In this phase the requirements of the system are collected and documented. This phase is very crucial because the next phases are based on this phase.
2. **Design phase:-** Design phase is based on the fact how the software will be built. The main objective of the design phase is to prepare the blueprint of the software system so that no problems are faced in the coming phases and solutions to all the requirements in the requirement phase are found.
3. **Implementation phase:-** In this phase, hardware, software and application programs are installed and the database design is implemented. Before the database design can be implemented, the software has to go through a testing, coding, and debugging process. This is the longest lasting phase in waterfall.
4. **Verification phase:-** In this phase the software is verified and it is evaluated that we have created the right product. In this phase, various types of testing are done and every area of the software is checked. It is believed that if we do not verify the software properly and there is any defect in it then no one will use it, hence verification is very important. One advantage of verification is that it reduces the risk of software failure.
5. **Maintenance phase:-** This is the last phase of waterfall. When the system is ready and users start using it, then the problems that arise have to be solved time-to-time. Taking care of the finished software and maintaining it as per time is called maintenance.

Advantages of Waterfall Model

- This model is simple and easy to understand.
- This is very useful for small projects.
- This model is easy to manage.
- The end goal is determined early.
- Each phase of this model is well explained.
- It provides a structured way to do things.
- This is a base model, all the SDLC models that came after this were created keeping this in mind, although they worked to remove its shortcomings.
- In this model, we can move to the next phase only after the first phase is successfully completed so that there is no overlapping between the phases.

Disadvantages of Waterfall Model

- In this model, complete and accurate requirements are expected at the beginning of the development process.
- Working software is not available for very long during the development life cycle.

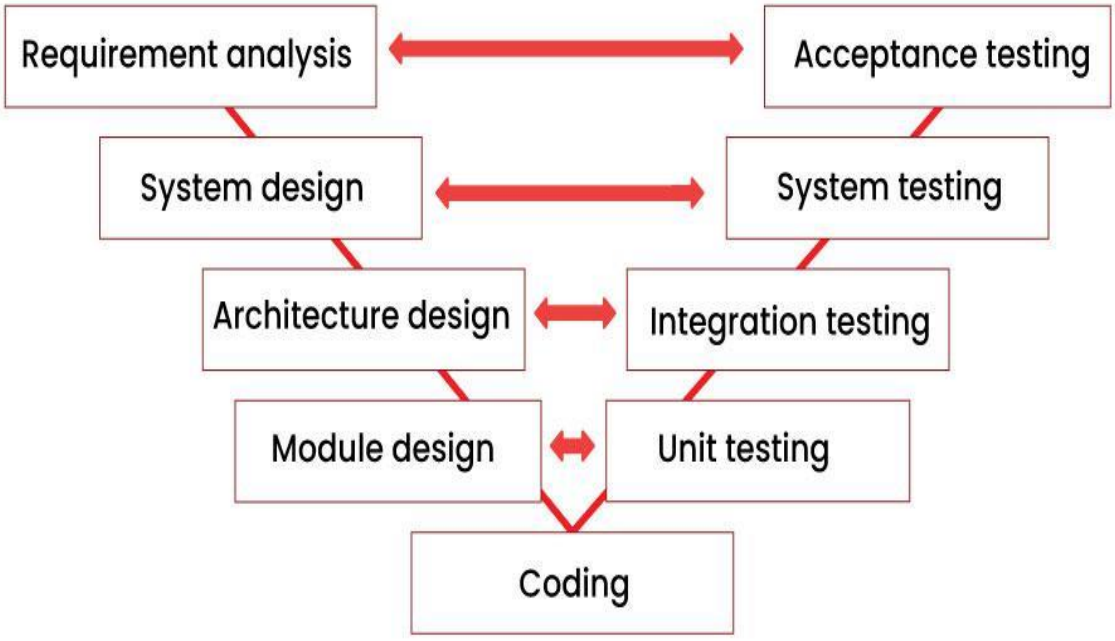
- We cannot go back to the previous phase due to which it is very difficult to change the requirements.
- Risk is not assessed in this, hence there is high risk and uncertainty in this model.
- In this the testing period comes very late.
- Due to its sequential nature this model is not realistic in today's world.
- This is not a good model for large and complex projects.

2. V-Model

V-Model is an SDLC model, it is also called Verification and Validation Model. V-Model is widely used in the software development process, and it is considered a disciplined model. In V-Model, the execution of each process is sequential, that is, the new phase starts only after the previous phase ends.

- It is based on the association of testing phase with each development phase that is in V-Model with each development phase, its testing phase is also associated in a V-shape in other words both software development and testing activities take place at the same time.
- So in this model, Verification Phase will be on one side, Validation Phase will be on the other side that is both the activities run simultaneously and both of them are connected to each other in V-Shape through Coding Phase, hence it is called V-Model.
- **V-Design:** In V-Design the left side represents the development activity, the right side represents the testing activity.

Phases of V-model



V-Model

V-Model

- **Requirements analysis:-** This is the first phase of the development cycle, in which the requirements of the product are analyzed according to the customer's needs. In this phase, product related requirements are thoroughly collected from the customer. This is a very important phase because this phase determines the coming phases. In this phase, acceptance tests are designed for later use.

- **System design:-** When we have the requirements of the product, after that we prepare a complete design of the system. In this phase, a complete description of the hardware and all the technical components required to develop the product .
- **Architectural design:-** In this phase architectural specifications are designed. It contains the specification of how the software will link internally and externally with all the components. Therefore this phase is also called high level design (HLD).
- **Module design:-** In this phase the internal design of all the modules of the system is specified. Therefore it is called low level design (LLD). It is very important that the design of all modules should be according to the system architecture. Unit tests are also designed in the module design phase.
- **Coding phase:-** In the coding phase, coding of the design and specification done in the previous phases is done. This phase takes the most time.

Validation Phases of V-Model

- **Unit testing:-** In the unit testing phase, the unit tests created during the module design phase are executed. Unit testing is code level testing, it only verifies the technical design. Therefore it is not able to test all the defects.
- I**ntegration testing:-** In integration testing, the integration tests created in the architectural design phase are executed. Integration testing ensures that all modules are working well together.
- **System testing:-** In system testing, the system tests created in the system design phase are executed. System tests check the complete functionality of the system. In this, more attention is given to performance testing and regression testing.
- **Acceptance testing:-** In acceptance testing, the acceptance tests created in the requirement analysis phase are executed. This testing ensures that the system is compatible with other systems. And in this, non-functional issues like:- load time, performance etc. are tested in the user environment.

Advantages of V-Model

- This is a simple and easy to use model.
- Planning, testing and designing tests can be done even before coding.
- This is a very disciplined model, in which phase by phase development and testing goes on.
- Defects are detected in the initial stage itself.
- Small and medium scale developments can be easily completed using it.

Disadvantages of V-Model

- This model is not suitable for any complex projects.
- There remains both high risk and uncertainty.
- This is not a suitable model for an ongoing project.
- This model is not at all suitable for a project which is unclear and in which there are changes in the requirement.

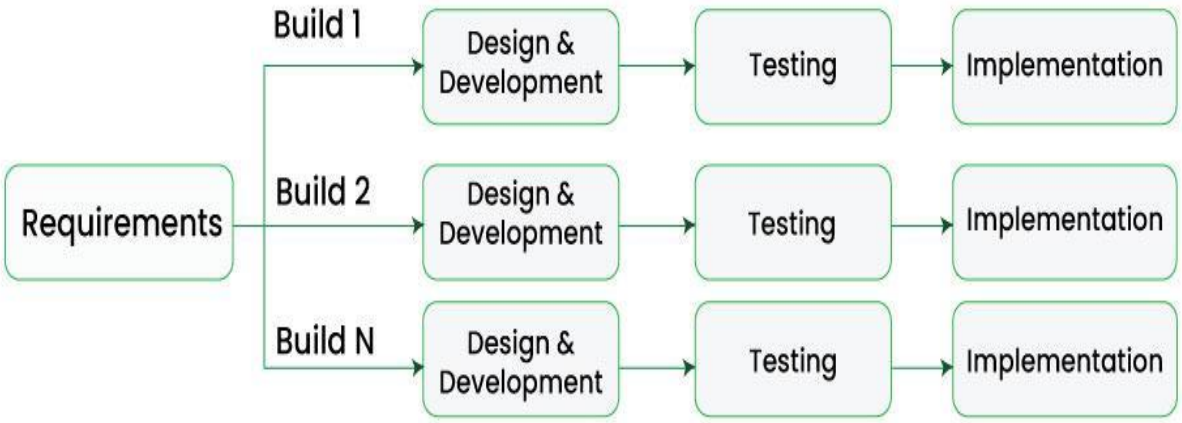
3. Incremental Model

In Incremental Model, the software development process is divided into several increments and the same phases are followed in each increment. In simple language, under this model a complex project is developed in many modules or builds.

- For example, we collect the customer's requirements, now instead of making the entire software at once, we first take some requirements and based on them create a module or function of the software and deliver it to the customer. Then we take some more requirements and based on them add another module to that software.
- Similarly, modules are added to the software in each increment until the complete system is created. However, the requirements for making a complex project in multiple iterations/parts should be clear.

- If we understand the entire principle of Incremental methodology, then it starts by developing an initial implementation, then user feedback is taken on it, and it is developed through several versions until an accepted system is developed. Important functionalities of the software are developed in the initial iterations.

Each subsequent release of a software module adds functions to the previous release. This process continues until the final software is obtained.



Incremental Model

Incremental Model

Phases of Incremental Model

- **Communication**: In the first phase, we talk face to face with the customer and collect his mandatory requirements. Like what functionalities does the customer want in his software, etc.
- **Planning**: In this phase the requirements are divided into multiple modules and planning is done on their basis.
- **Modeling**: In this phase the design of each module is prepared. After the design is ready, we take a particular module among many modules and save it in DDS (Design Document Specification). Diagrams like ERDs and DFDs are included in this document.
- **Construction**: Here we start construction based on the design of that particular module. That is, the design of the module is implemented in coding. Once the code is written, it is tested.
- **Deployment**: After the testing of the code is completed, if the module is working properly then it is given to the customer for use. After this, the next module is developed through the same phases and is combined with the previous module. This makes new functionality available to the customer. This will continue until complete modules are developed.

Advantages of Incremental Model

- Important modules/functions are developed first and then the rest are added in chunks.
- Working software is prepared quickly and early during the software development life cycle (SDLC).
- This model is flexible and less expensive to change requirements and scope.
- The customer can respond to each module and provide feedback if any changes are needed.
- Project progress can be measured.
- It is easier to test and debug during a short iteration.
- Errors are easy to identify.

Disadvantages of Incremental Model

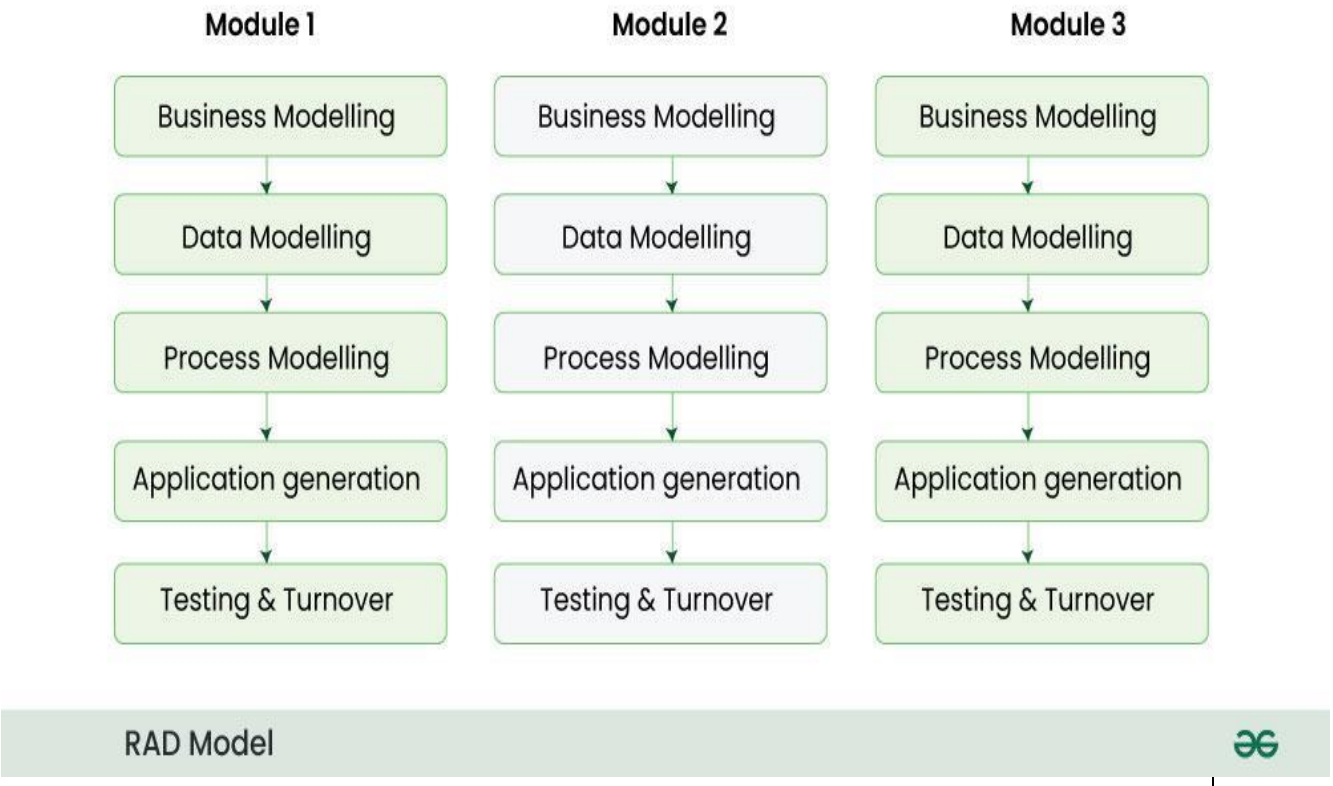
- Management is a continuous activity that must be handled.
- Before the project can be dismantled and built incrementally,
- The complete requirements of the software should be clear.
- This requires good planning and designing.
- The total cost of this model is higher.

4. RAD Model

RAD model stands for rapid application development model. The methodology of RAD model is similar to that of incremental or waterfall model. It is used for small projects.

- If the project is large then it is divided into many small projects and these small projects are planned one by one and completed. In this way, by completing small projects, the large project gets ready quickly.
- In RAD model, the project is completed within the given time and all the requirements are collected before starting the project. It is very fast and there are very less errors in it.

The main objective of RAD model is to reuse code, components, tools, processes in project development.



RAD Model

RAD Model

Phases RAD model

1. **Business modeling:** In this phase, the business model is designed on the basis of whatever functions the business has. If we speak in a little technical language, then we design the business model for the product on the basis of flow of information and distribution of information between different business channels. Here information flow means what type of information drives the business, where the information comes from and where it goes, who generates it, etc. This means that a complete business analysis is done in this phase.
2. **Data modeling:** Using the business model we had prepared, the data objects required for the business are defined.
3. **Process modeling:** The data objects that we defined in the data modeling phase are converted to establish the business information flow. It is necessary to achieve specific business objectives.
4. **Application generation:** In this phase we start building the software based on the output of the above three phases. For this we take the help of automation tools. However, in this phase we do not develop the actual software but make a working prototype.
5. **Testing and turnover:** Whatever prototype we have prepared or whatever components and interfaces we have, they are tested in this phase. Since prototypes are tested separately during each iteration, the overall testing time in rapid application development is reduced.

Advantage of RAD Model

- It reduces the time taken in development.
- In this the components are reused.
- It is flexible and it is easy to make any changes in it.
- It is easy to transfer like scripts because high level abstraction and intermediate codes are used in it.
- There are very few defects in it because it is a prototype by nature.
- In this, productivity can be increased in less time with less people.
- It is cost effective.
- It is suitable for small projects.

Disadvantages of RAD Model

- In this we need highly skilled developers and designers.
- It is very difficult to manage.
- It is not suitable for project that are complex and takes long time.
- In this, feedback from the client is required for the development of each phase.
- Automated code generation is very expensive.
- This model is suitable only for component based and scalable systems.

5. Iterative Model

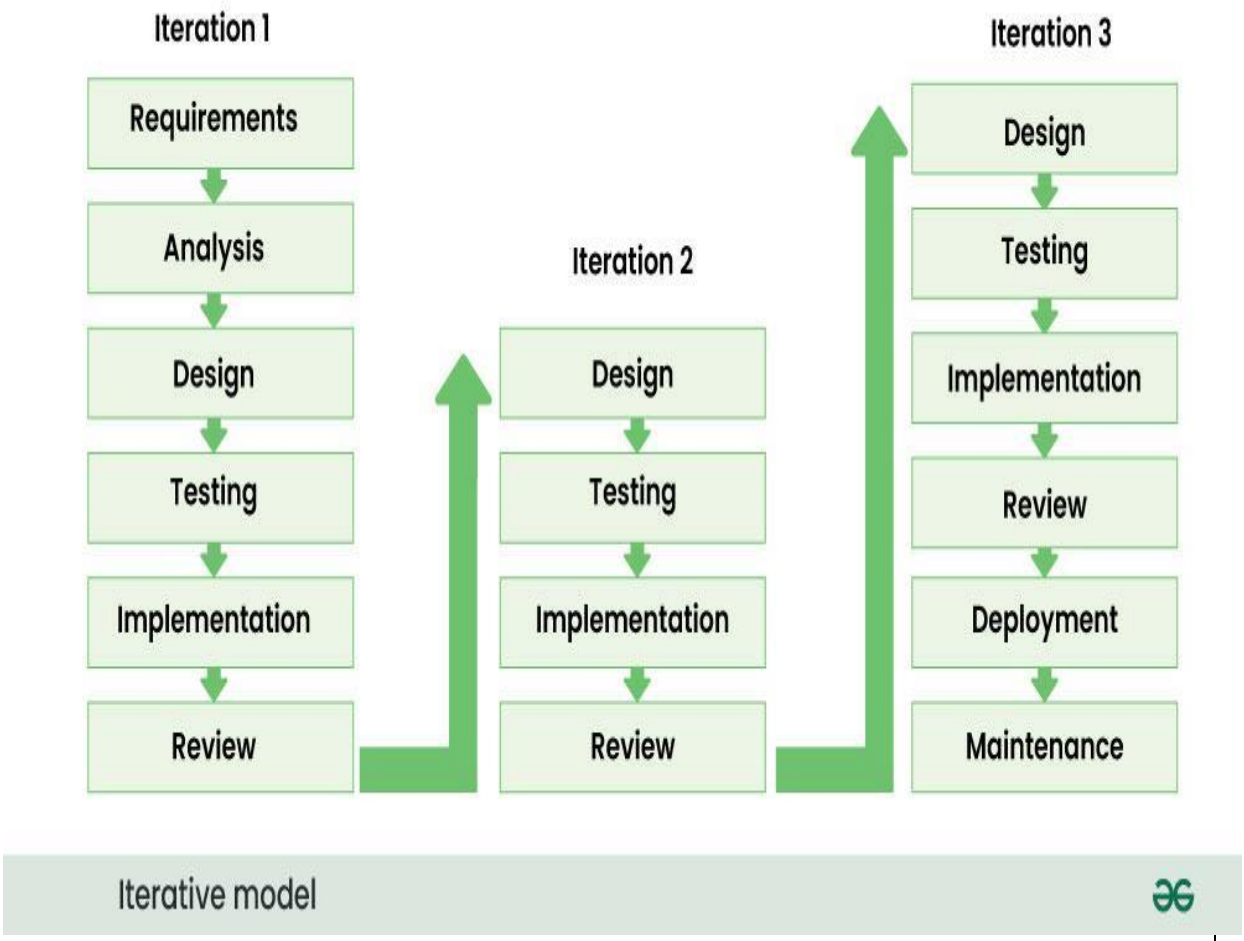
In Iterative model we start developing the software with some requirements and when it is developed, it is reviewed. If there are requirements for changes in it, then we develop a new version of the software based on those requirements. This process repeats itself many times until we get our final product.

- So, in Iterative model a software is developed by following several iterations. Iteration means that we are repeating the development process again and again. For example, we develop the first version of the software following the SDLC process with some software requirements.
- After the first version is developed, if there is a need to change the software , then a new version is developed with the second iteration. Now again we will see if the new

version is enough, if not then we will make changes in it with the third iteration. The iteration will be repeated until the complete software is ready.

- The basic concept of Iterative model is that the software should be developed through repeated cycles or what we also call iteration and only a small part of it should be developed at a time. This model was developed to overcome the drawbacks of the classical waterfall model.

Through this diagram you can understand the Interactive model.



Iterative model

Iterative Model

Phases of iterative model

1. **Requirement gathering & analysis:** In this phase, all the software requirements of the customer are collected and it is analyzed whether those requirements can be meet or not. Besides, it is also checked whether this project will not go beyond our budget.
2. **Design**: In this phase the design of software is prepared. For this, various diagrams like Data Flow diagram, class diagram, activity diagram, state transition diagram, etc. are used.
3. **Implementation**: Now the design of software is implemented in coding through various programming languages. We also call this coding phase.
4. **Testing**: After the coding of the software is done, it is now tested so that the bugs and errors present in it can be identified. To do this, various testing techniques like performance testing, security testing, requirement testing, stress testing, etc. are done.
5. **Deployment**: Finally the software is given to the customer. After this the customer starts using that software in his work environment.

6. **Review**: After the software is deployed in its work environment, it is reviewed. If any error/bug is found or any new requirements come in front of developer, then again these phases are repeated with new iteration and a new version is developed.
7. **Maintenance**: In this phase we look at customer feedback, solve problems, fix errors, update software, etc.

Advantage of Iterative model

- In iterative models, bugs and errors can be identified quickly.
- Under this model, software is prepared quickly with some specifications.
- Testing and debugging the software becomes easier during each iteration.
- We get reliable feedback from users along with blueprints.
- This model is easily adaptable to constantly changing needs.
- During the software development process, additional time is devoted to development and limited time to documentation.
- Risks are identified and resolved during iteration.

Disadvantage of Iterative model

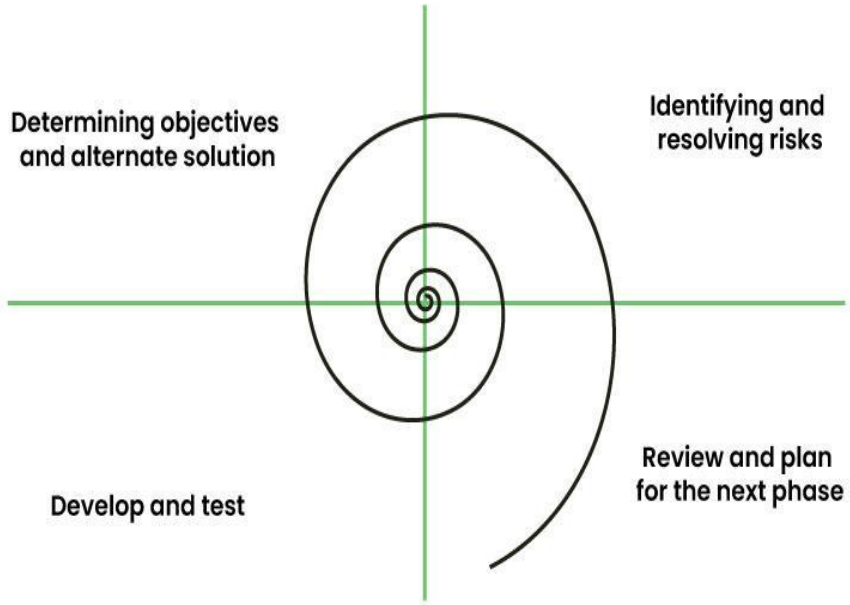
- Iterative model is not suitable for small projects.
- Since we have to repeat iterations many times in the software development process due to which we require more resources.
- Since the requirements are constantly changing, we have to make frequent changes in the software.
- Due to constantly changing requirements, the budget of the project also increases and it takes more time to complete it.
- In this model, it is complicated to control the entire process of software development.
- It is very difficult to tell by what date the complete software will be ready.

6. Spiral Model

Spiral model is a software development process model. This model has characteristics of both iterative and waterfall models. This model is used in projects which are large and complex. This model was named spiral because if we look at its figure, it looks like a spiral, in which a long curved line starts from the center point and makes many loops around it. The number of loops in the spiral is not decided in advance but it depends on the size of the project and the changing requirements of the user. We also call each loop of the spiral a phase of the software development process.

A software project goes through these loops again and again in iterations. After each iteration a more and more complete version of the software is developed. The most special thing about this model is that risks are identified in each phase and they are resolved through prototyping. This feature is also called Risk Handling.

Since it also includes the approaches of other SDLC models, it is also called Meta Model. It was first developed by Barry Boehm in 1986.



Spiral model

Spiral Model

In Spiral Model the entire process of software development is described in four phases which are repeated until the project is completed.

Those phases are as follows:-

- **Determining objectives and alternate solutions:** In the first phase, whatever requirements the customer has related to the software are collected. On the basis of which objectives are identified and analyzed and various alternative solutions are proposed.
- **Identifying and resolving risks:** In this phase, all the proposed solutions are assessed and the best solution is selected. Now that solution is analyzed and the risks related to it are identified. Now the identified risks are resolved through some best strategy.
- **Develop and test:** Now the development of software is started. In this phase various features are implemented, that is, their coding is done. Then those features are verified through testing.
- **Review and plan for the next phase:** In this phase the developed version of the software is given to the customer and he evaluates it. Gives his feedback and tells new requirements. Finally planning for the next phase (next spiral) is started.

Advantages of Spiral Model

- If we have to add additional functionality or make any changes to the software, then through this model we can do so in the later stages also.
- Spiral model is suitable for large and complex projects.
- It is easy to estimate how much the project will cost.
- Risk analysis is done in each phase of this model.
- The customer can see the look of his software only in the early stages of the development process.
- Since continuous feedback is taken from the customer during the development process, the chances of customer satisfaction increases.

Disadvantage of Spiral Model

- This is the most complex model of SDLC, due to which it is quite difficult to manage.

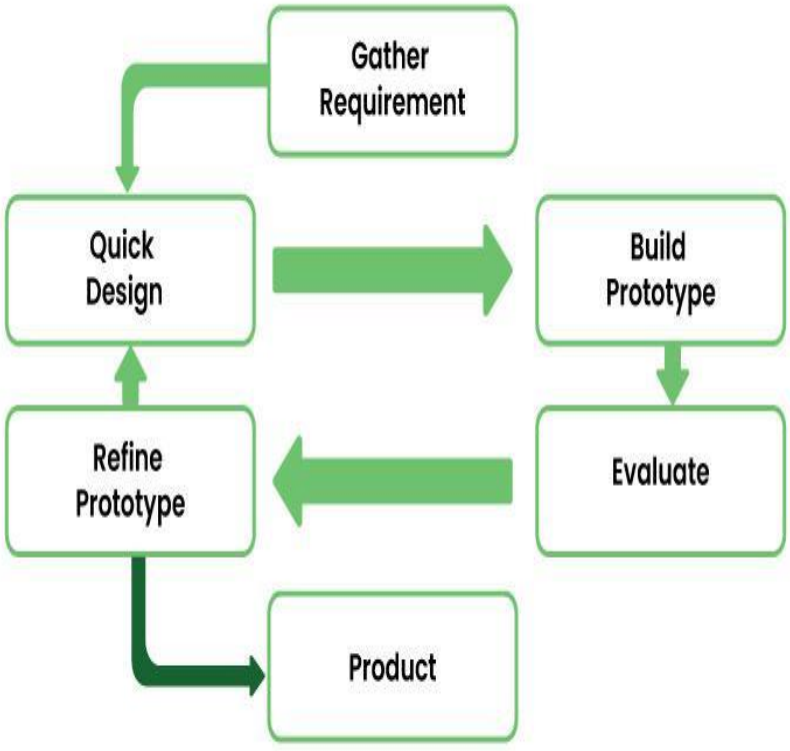
- This model is not suitable for small projects.
- The cost of this model is quite high.
- It requires more documentation than other models.
- Experienced experts are required to evaluate and review the project from time to time.
- Using this model, the success of the project depends greatly on the risk analysis phase.

7. Prototype model

Prototype model is an activity in which prototypes of software applications are created. First a prototype is created and then the final product is manufactured based on that prototype.

- The prototype model was developed to overcome the shortcomings of the waterfall model.
- This model is created when we do not know the requirements well.
- The specialty of this model is that this model can be used with other models as well as alone.

One problem in this model is that if the end users are not satisfied with the prototype model, then a new prototype model is created again, due to which this model consumes a lot of money and time.



Prototype model

Prototype Model

The prototype model has the following phases

- **Requirement gathering:** The first step of prototype model is to collect the requirements, although the customer does not know much about the requirements but the major requirements are defined in detail.

- **Build the initial prototype:** In this phase the initial prototype is built. In this some basic requirements are displayed and user interface is made available.
- **Review the prototype:** When the construction of the prototype is completed, it is presented to the end users or customer and feedback is taken from them about this prototype. This feedback is used to further improve the system and possible changes are made to the prototype.
- **Revise and improve the prototype:** When feedback is taken from end users and customers, the prototype is improved on the basis of feedback. If the customer is not satisfied with the prototype, a new prototype is created and this process continues until the customer gets the prototype as per his desire.

Advantages of Prototype model

- Prototype Model is suggested to create applications whose prototype is very easy and which always includes human machine interaction within it.
- When we know only the general objective of creating software, but we do not know anything in detail about input, processing and output. Then in such a situation we make it a Prototype Model.
- When a software developer is not very sure about the capability of an algorithm or its adaptability to an operating system, then in this situation, using a prototype model can be a better option.

Disadvantages of Prototype model

- When the first version of the prototype model is ready, the customer himself often wants small fixes and changes in it rather than rebuilding the system. Whereas if the system is redesigned then more quality will be maintained in it.
- Many compromises can be seen in the first version of the Prototype Model.
- Sometimes a software developer may make compromises in his implementation, just to get the prototype model up and running quickly, and after some time he may become comfortable with making such compromises and may forget that it is completely inappropriate to do so.

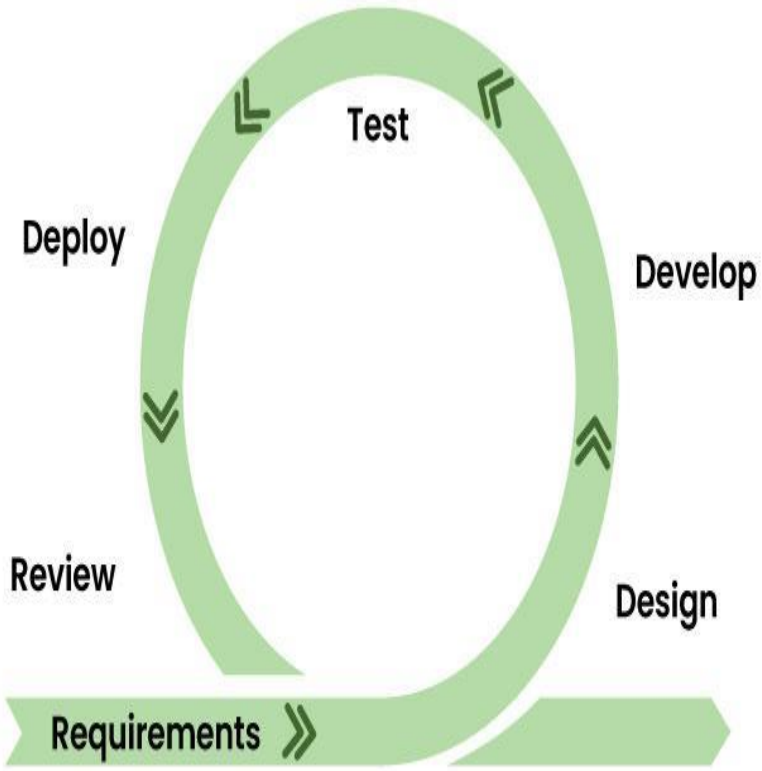
8. Agile Model

Agile model is a combination of iterative and incremental models, that is, it is made up of iterative and incremental models.

- In Agile model, focus is given to process adaptability and customer satisfaction.
- In earlier times, iterative waterfall model was used to create software. But in today's time developers have to face many problems. The biggest problem is that in the middle of software development, the customer asks to make changes in the software. It takes a lot of time and money to make these changes.

So to overcome all these shortcomings, the agile model was proposed in the 1990s.

The agile model was created mainly to make changes in the middle of software development so that the software project can be completed quickly.



Agile Model

Agile Model

- In the agile model, the software product is divided into small incremental parts. In this, the smallest part is developed first and then the larger one.
- And each incremental part is developed over iteration.
- Each iteration is kept small so that it can be easily managed. And it can be completed in two-three weeks. Only one iteration is planned, developed and deployed at a time.

Principles of Agile model

- There is a customer representative in the development team to maintain contact with the customer during software development and to understand the requirement. When an iteration is completed, stakeholders and customer representatives review it and re-evaluate the requirements.
- Demo of working software is given to understand the customer's requirements. That is, it does not depend only on documentation.
- Incremental versions of the software have to be delivered to the customer representative after a few weeks.
- In this model it is advised that the size of the development team should be small (5 to 9 people) so that the team members can communicate face to face.
- Agile model focuses on the fact that whenever any changes have to be made in the software, it should be completed quickly.
- In agile development, two programmers work together. One programmer does the coding and the other reviews that code. Both the programmers keep changing their tasks, that is, sometimes one does coding and sometimes someone reviews.

Agile has the following models

1. Scrum
2. Crystal methods
3. DSDM
4. Feature driven development (FDD
5. Lean software development
6. Extreme programming (xp)

Advantages of Agile Model

- In this, two programmers work together due to which the code is error free and there are very few mistakes in it.
- In this the software project is completed in a very short time.
- In this the customer representative has an idea of each iteration so that he can easily change the requirement.
- This is a very realistic approach to software development.
- In this, focus is given on teamwork.
- There are very few rules in this and documentation is also negligible.
- There is no need for planning in this.
- It can be managed easily.
- It provides flexibility to developers.

Disadvantages of Agile Model

- It cannot handle complex dependencies.
- Due to lack of formal documentation in this, there is confusion in development.
- It mostly depends on the customer representative, if the customer representative gives any wrong information then the software can become wrong.
- Only experienced programmers can take any decision in this. New programmers cannot take any decision.
- In the beginning of software development, it is not known how much effort and time will be required to create the software.

Why companies are shifting toward agile Software Development models?

In earlier times, iterative waterfall model was used to create software. But in today's time developers have to face many problems. The biggest problem is that in the middle of software development, the customer asks to make changes in the software. It takes a lot of time and money to make these changes. The agile model was created mainly to make changes in the middle of software development so that the software project can be completed quickly.

Agile focuses on customer-centric approach that constantly take feedback from customer and make changes in the middle of software developer according to their needs and expectation. Agile Methodologies like scrum adopt iterative and incremental approach this accelerate the time-to-market for new features and product.

Conclusion

Choosing the right software development model is crucial for the success of a project. Each model offers unique advantages and is suitable for different types of projects. Agile methodologies, due to their flexibility and customer-centric approach, are increasingly popular in the industry. By adopting Agile, companies can accelerate time-to-market and better adapt to changing requirements, leading to higher customer satisfaction and successful project outcomes.

Rapid Application development

The RAD model or Rapid Application Development model is a type of software development methodology that emphasizes quick and iterative release cycles, primarily focusing on delivering working software in shorter timelines. Unlike traditional models such as the Waterfall model, RAD is designed to be more flexible and responsive to user feedback and changing requirements throughout the development process.

In this article, we will break down the key principles and phases of the RAD model, highlighting its advantages and potential challenges.

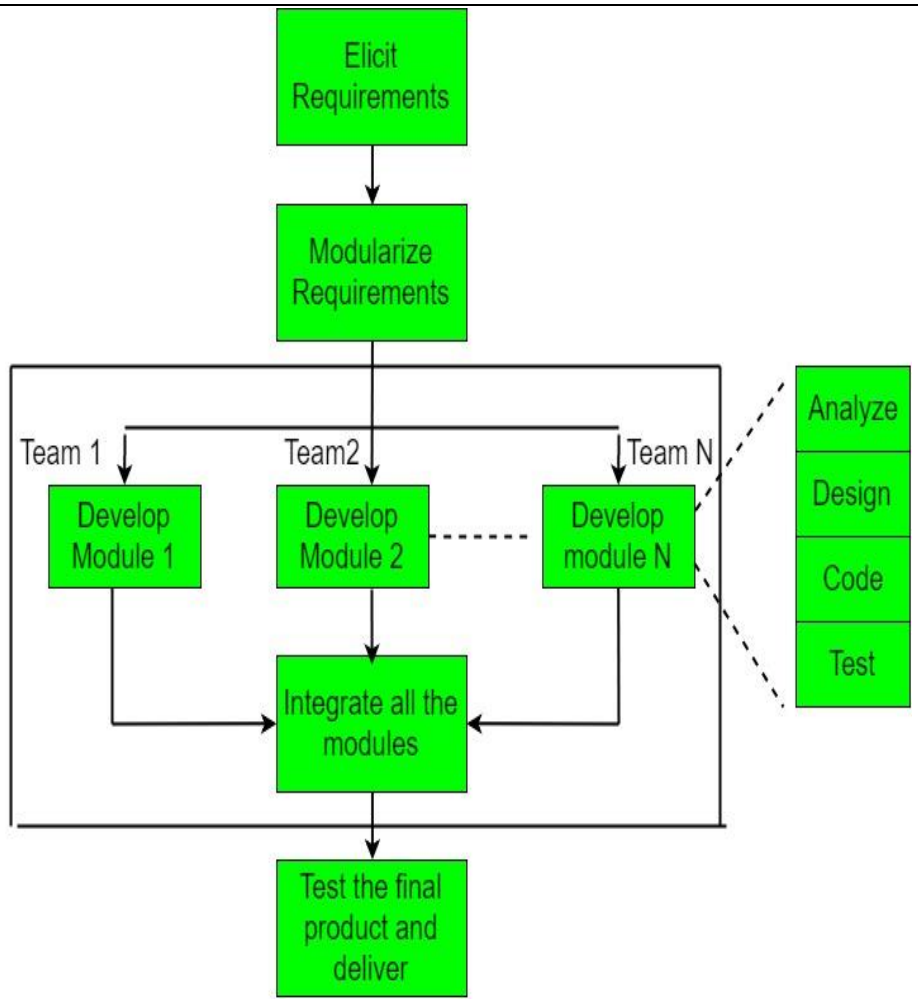
Table of Content

What is RAD Model in Software Engineering?

IBM first proposed the Rapid Application Development or RAD Model in the 1980s. The RAD model is a type of incremental process model in which there is a concise development cycle. The RAD model is used when the requirements are fully understood and the component-based construction approach is adopted. Various phases in RAD are Requirements Gathering, Analysis and **Planning**, **Design**, **Build** or **Construction**, and finally **Deployment**.

The critical feature of this model is the use of powerful development tools and techniques. A software project can be implemented using this model if the project can be broken down into small modules wherein each module can be assigned independently to separate teams. These modules can finally be combined to form the final product. Development of each module involves the various basic steps as in the waterfall model i.e. analyzing, designing, coding, and then testing, etc. as shown in the figure. Another striking feature of this model is a short period i.e. the time frame for delivery(time-box) is generally 60-90 days.

Multiple teams work on developing the software system using the RAD model parallelly.



Rapid application development model (RAD)

The use of powerful developer tools such as JAVA, C++, Visual BASIC, XML, etc. is also an integral part of the projects. This model consists of 4 basic phases:

1. **Requirements Planning** – This involves the use of various techniques used in requirements elicitation like brainstorming, task analysis, form analysis, user scenarios, FAST (Facilitated Application Development Technique), etc. It also consists of the entire structured plan describing the critical data, methods to obtain it, and then processing it to form a final refined model.
2. **User Description** – This phase consists of taking user feedback and building the prototype using developer tools. In other words, it includes re-examination and validation of the data collected in the first phase. The dataset attributes are also identified and elucidated in this phase.
3. **Construction** – In this phase, refinement of the prototype and delivery takes place. It includes the actual use of powerful automated tools to transform processes and data models into the final working product. All the required modifications and enhancements are to be done in this phase.
4. **Cutover** – All the interfaces between the independent modules developed by separate teams have to be tested properly. The use of powerfully automated tools and subparts makes testing easier. This is followed by acceptance testing by the user.

The process involves building a rapid prototype, delivering it to the customer, and taking feedback. After validation by the customer, the SRS document is developed and the design is finalized.

When to use the RAD Model?

1. **Well-understood Requirements:** When project requirements are stable and transparent, RAD is appropriate.
2. **Time-sensitive Projects:** Suitable for projects that need to be developed and delivered quickly due to tight deadlines.
3. **Small to Medium-Sized Projects:** Better suited for smaller initiatives requiring a controllable number of team members.
4. **High User Involvement:** Fits where ongoing input and interaction from users are essential.
5. **Innovation and Creativity**: Helpful for tasks requiring creative inquiry and innovation.
6. **Prototyping:** It is necessary when developing and improving prototypes is a key component of the development process.
7. **Low technological Complexity:** Suitable for tasks using comparatively straightforward technological specifications.

Objectives of Rapid Application Development Model (RAD)

1. Speedy Development

Accelerating the software development process is RAD's main goal. RAD prioritizes rapid prototyping and iterations to produce a working system as soon as possible. This is especially helpful for projects when deadlines must be met.

2. Adaptability and Flexibility

RAD places a strong emphasis on adapting quickly to changing needs. Due to the model's flexibility, stakeholders can modify and improve the system in response to changing requirements and user input.

3. Stakeholder Participation

Throughout the development cycle, RAD promotes end users and stakeholders' active participation. Collaboration and frequent feedback make it possible to make sure that the changing system satisfies both user and corporate needs.

4. Improved Interaction

Development teams and stakeholders may collaborate and communicate more effectively thanks to RAD. Frequent communication and feedback loops guarantee that all project participants are in agreement, which lowers the possibility of misunderstandings.

5. Improved Quality via Prototyping

Prototypes enable early system component testing and visualization in Rapid Application Development (RAD). This aids in spotting any problems, confirming design choices, and guaranteeing that the finished product lives up to consumer expectations.

6. Customer Satisfaction

Delivering a system that closely satisfies user expectations and needs is the goal of RAD. Through rapid delivery of functioning prototypes and user involvement throughout the development process, Rapid Application Development (RAD) enhances the probability of customer satisfaction with the final product.

Advantages of Rapid Application Development Model (RAD)

- The use of reusable components helps to reduce the cycle time of the project.
- Feedback from the customer is available at the initial stages.
- Reduced costs as fewer developers are required.
- The use of powerful development tools results in better quality products in comparatively shorter periods.
- The progress and development of the project can be measured through the various stages.
- It is easier to accommodate changing requirements due to the short iteration time spans.
- Productivity may be quickly boosted with a lower number of employees.

Disadvantages of Rapid application development model (RAD)

- The use of powerful and efficient tools requires highly skilled professionals.
- The absence of reusable components can lead to the failure of the project.
- The team leader must work closely with the developers and customers to close the project on time.
- The systems which cannot be modularized suitably cannot use this model.
- Customer involvement is required throughout the life cycle.
- It is not meant for small-scale projects as in such cases, the cost of using automated tools and techniques may exceed the entire budget of the project.
- Not every application can be used with RAD.

Applications of Rapid Application Development Model (RAD)

1. This model should be used for a system with known requirements and requiring a short development time.
2. It is also suitable for projects where requirements can be modularized and reusable components are also available for development.
3. The model can also be used when already existing system components can be used in developing a new system with minimum changes.
4. This model can only be used if the teams consist of domain experts. This is because relevant knowledge and the ability to use powerful techniques are a necessity.
5. The model should be chosen when the budget permits the use of automated tools and techniques required.

Drawbacks of Rapid Application Development

- It requires multiple teams or a large number of people to work on scalable projects.
- This model requires heavily committed developers and customers. If commitment is lacking then RAD projects will fail.
- The projects using the RAD model require heavy resources.
- If there is no appropriate modularization then RAD projects fail. Performance can be a problem for such projects.
- The projects using the RAD model find it difficult to adopt new technologies. This is because RAD focuses on quickly building and refining prototypes using existing tools. Changing to new technologies can disrupt this process, making it harder to keep up with the fast pace of development. Even with skilled developers and advanced tools, the rapid nature of RAD leaves little time to learn and integrate new technologies smoothly.

Conclusion

The Rapid Application Development (RAD) model offers a powerful approach to software development, focusing on speed, flexibility, and stakeholder involvement. By enabling quick iterations and the use of reusable components, RAD ensures the fast delivery of functional prototypes, enhancing user satisfaction and project adaptability. However, its reliance on highly skilled developers, modular design, and automated tools presents challenges, particularly for projects with complex requirements or limited resources.

Agile methods

The Agile methodology is a project management and software development approach that emphasizes flexibility, collaboration, and customer-centricity. It is the latest model used by major companies today like Facebook, google, amazon, etc. It follows the iterative as well as incremental approach that emphasizes the importance of delivering of working product very quickly. This article focuses on discussing Agile Methodology in detail.

Table of Content

What is Agile?

Agile is a project management and software development approach that aims to be more effective.

1. It focuses on delivering smaller pieces of work regularly instead of one big launch.
2. This allows teams to adapt to changes quickly and provide customer value faster.

Agile is not just a methodology; it's a mindset. Agile isn't about following specific rituals or techniques. Instead, it's a bunch of methods that show a dedication to quick feedback and always getting better.

What is the Agile Methodology?

Agile methodologies are iterative and incremental, which means it's known for breaking a project into smaller parts and adjusting to changing requirements.

1. They prioritize flexibility, collaboration, and customer satisfaction.
2. Major companies like Facebook, Google, and Amazon use Agile because of its adaptability and customer-focused approach.

History of Agile

- In 1957, people started figuring out new ways to build computer programs. They wanted to make the process better over time, so they came up with iterative and incremental methods.
- In the 1970s, people started using adaptive software development and evolutionary project management. This means they were adjusting and evolving how they built software.
- In 1990s, there was a big change. Some people didn't like the strict and super-planned ways of doing things in software development. They called these old ways "waterfall." So, in response, lighter and more flexible methods showed up. These included:
    - Rapid Application Development (RAD) in 1991.
    - Unified Process (UP), Dynamic Systems Development Method (DSDM) in 1994.
    - Scrum in 1995.
    - Crystal Clear and Extreme Programming (XP) in 1996.
    - Feature-Driven Development (FDD) in 1997.

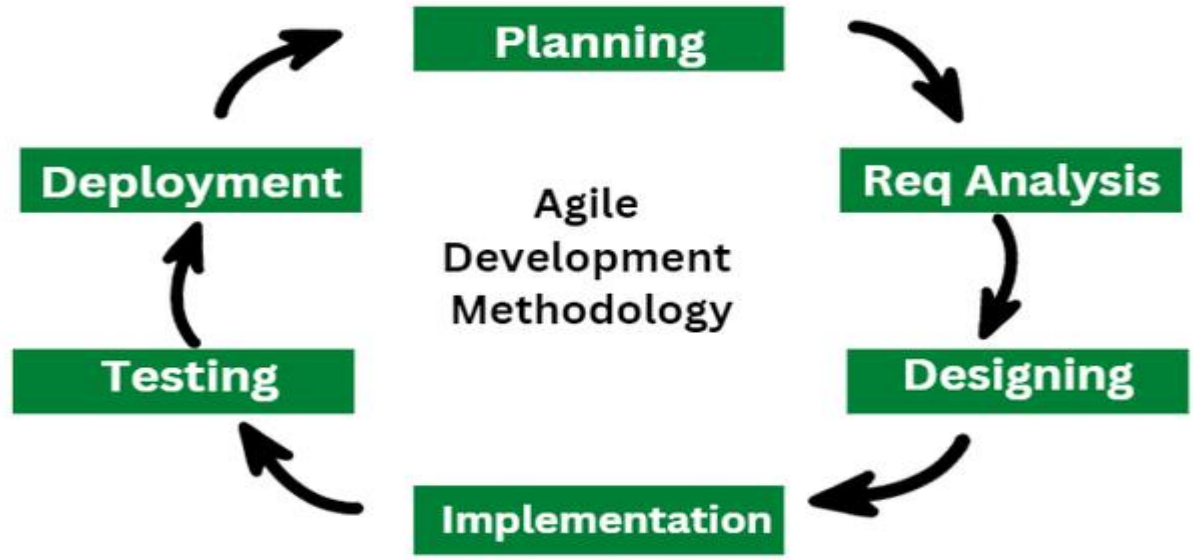
Even though these came before the official "Agile Manifesto", we now call them agile software development methods.

*In **2001**, seventeen software developers met at a resort in Snowbird, Utah to discuss lightweight development methods. They were: Kent Beck (Extreme Programming), Ward Cunningham (Extreme Programming), Dave Thomas (PragProg, Ruby), Jeff Sutherland (Scrum), Ken Schwaber (Scrum), Jim Highsmith (Adaptive Software Development), Alistair Cockburn (Crystal), Robert C. Martin (SOLID), Mike Beedle (Scrum), Arie van Bennekum, Martin Fowler (OOAD and UML), James Grenning, Andrew Hunt (PragProg, Ruby), Ron Jeffries (Extreme Programming), Jon Kern, Brian Marick (Ruby, TDD), and Steve Mellor (OOA). They wrote something important called the Manifesto for Agile Software Development. This was a big moment that set the stage for the agile movement.*

- In 2005, Alistair Cockburn and Jim Highsmith added more ideas about managing projects, creating the PM Declaration of Interdependence.
- Then, in 2009, a group, including Robert C. Martin, added principles about software development. They called it the Software Craftsmanship Manifesto, focusing on being professional and skilled.
- In 2011, the Agile Alliance, a group of agile enthusiasts, made the Guide to Agile Practices (later called Agile Glossary). This was like a shared document where agile people from around the world put down their ideas, terms, and guidelines. It's a bit like a dictionary for how to do agile things.

Life cycle of Agile Methodology

The Agile software development life cycle helps you break down each project you take on into six simple stages:



1. Requirement Gathering

- In this stage, the project team identifies and documents the needs and expectations of various stakeholders, including clients, users, and subject matter experts.
- It involves defining the project's scope, objectives, and requirements.
- Establishing a budget and schedule.
- Creating a project plan and allocating resources.

2. Design

- Developing a high-level system architecture.
- Creating detailed specifications, which include data structures, algorithms, and interfaces.

- Planning for the software's user interface.

3. Development (Coding)

Writing the actual code for the software. Conducting unit testing to verify the functionality of individual components.

4. Testing

This phase involves several types of testing:

- **Integration Testing:** Ensuring that different components work together.
- **System Testing:** Testing the entire system as a whole.
- **User Acceptance Testing:** Confirming that the software meets user requirements.
- **Performance Testing:** Assessing the system's speed, scalability, and stability.

5. Deployment

- Deploying the software to a production environment.
- Put the software into the real world where people can use it.
- Make sure it works smoothly in the real world.
- Providing training and support for end-users.

6. Review (Maintenance)

- Addressing and resolving any issues that may arise after deployment.
- Releasing updates and patches to enhance the software and address problems.

Manifesto for Agile Software Development

The Manifesto for Agile Software Development is a document produced by 17 developers at Snowbird, Utah in 2001. This document consists of 4 Agile Values and 12 Agile Principles. These 12 principals and 4 agile values provide a guide to Software Developers. The Manifesto for Agile Software Development emerged as a transformative guide to **Software Development**.

What are the 12 Agile Principles?

There are 12 agile principles mentioned in the **Agile Manifesto**. Agile principles are guidelines for flexible and efficient software development. They emphasize frequent delivery, embracing change, collaboration, and continuous improvement. The focus is on delivering value, maintaining a sustainable work pace, and ensuring technical excellence.

for more: Agile Software Process and its Principles

Agile Software Development

Agile Software Development is a software development methodology that values flexibility, collaboration, and customer satisfaction. It is based on the Agile Manifesto, a set of principles for software development that prioritize individuals and interactions, working software, customer collaboration, and responding to change.

Agile Software Development is an iterative and incremental approach to software development that emphasizes the importance of delivering a working product quickly and frequently. It involves close collaboration between the development team and the customer to ensure that the product meets their needs and expectations.

What are Agile frameworks?

Agile frameworks are methods of organizing and dealing with software program development initiatives that follow the principles and values of the Agile Manifesto. Agile frameworks intend to supply value to clients faster and extra often, even also allowing groups to conform to converting requirements and remarks.

Types of Agile Frameworks

1. Kanban
2. Scrum
3. Lean
4. DSDM or Dynamic Systems Development Method ·

5. XP or Extreme Programming
6. FDD or Feature Driven Development
7. Crystal
8. Scaled Agile Framework (SAFe)

Agile Development Models

The Agile Development Model was primarily designed to help a project adapt quickly to change requests. 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 and removing activities that may not be essential for a specific project. Also, anything that is a waste of time and effort is avoided.

The Agile Model refers to a group of development processes. These processes share some basic characteristics but do have certain subtle differences among themselves.

Agile Software Development Methodology

Agile Software Development Methodology in software development is an efficient methodology that helps teams produce high-quality software quickly and with flexibility. Agile is not just a methodology; it's a mindset**.** At its core, Agile values individuals and interactions, working solutions, and customer collaboration over strict processes and comprehensive documentation. It acknowledges that the needs and priorities of a project may change, emphasizing the importance of adaptability and continuous improvement.

What is Agile Product Management?

In simple terms, Agile Product Management is a concept and approach that focuses on speed while taking the demands of customers into account throughout the product development life cycle. As compared to traditional Waterfall approaches, which follow a 'linear and sequential' process, Agile uses iterative cycles of operation; developers can return to customers for feedback or modifications constantly. That is a mindset that prioritizes people and interactions over processes and systems, working products over detailed documentation, and customer collaboration over contract negotiation.

What is Agile Project Management?

Agile Project Management is a revolutionary approach, that is aimed at continuously delivering solutions for the changing requirements of the project in a spiral way.

1. It was established by the Agile Manifesto's principles, calling for iterative and incremental development of the project into manageable sprints that create potentially shippable increments.
2. Agile Project Management is the concept of being agile and agile method in the new challenges of advanced projects, concerning its community-oriented spirit, planning flexibility, and ongoing progress.

Agile Software Testing

Agile Testing is a type of software testing that follows the principles of agile software development to test the software application. All members of the project team along with the special experts and testers are involved in agile testing. Agile testing is not a separate phase and it is carried out with all the development phases i.e. requirements, design and coding, and test case generation.

Agile Methodology Advantage and Disadvantage

The main advantage and disadvantage of agile methodology are:

- **Advantage** : Agile methodologies allow for flexibility and adaptability in responding to changes. Teams can easily adjust their plans and priorities based on evolving requirements or feedback during the project.

- **Disadvantage**: The iterative and adaptive nature of agile can sometimes lead to uncertainty, especially in projects with unclear or rapidly changing requirements. This may pose challenges in estimating timelines and costs accurately.

for more: Agile Methodology Advantage and Disadvantage

Agile vs Waterfall Methodology

If you want to know Agile vs Waterfall, then Agile is like a good fit for projects that need to be flexible and change a lot, such as making computer programs. It works well when talking with customers and making improvements bit by bit are really important. On the other side, Waterfall is just right for projects that have clear steps and simple, straight-ahead plans. It's like following a recipe step by step. So, if your project is all about changes and surprises, go for Agile. But if it's more like following a clear plan without many surprises, Waterfall is the way to go.

When to use the Agile Methodology?

If you want to know when to use the Agile methodology, then it is particularly well-suited for projects and organizations where the following conditions or needs are present:

1. **Unclear or Changing Requirements:** Agile is great for projects with requirements that aren't well-defined or might change.
2. **Complex Projects:** It's good for big, complex projects by breaking them into smaller pieces.
3. **Customer Focus:** Use Agile when making customers happy is a priority and you want to involve them throughout.
4. **Quick Time-to-Market:** If you need to get your product out fast, Agile can help.
5. **Small to Medium Teams:** Agile works well for teams of a few to a few dozen people.
6. **Team Skills:** It's best when you have a mix of skills in your team, like development, testing, design, and more.
7. **Collaboration:** Agile promotes working together and open communication.
8. **Regular Updates:** If you want to check progress often and make changes as needed.
9. **Transparency:** Agile emphasizes being open and clear with everyone involved in the project.
10. **Risk Control:** It helps manage risks by tackling issues as they come up.
11. **Innovation:** If you encourage trying new things and learning from experience, Agile supports that.
12. **Continuous Improvement:** Agile fosters a culture of always getting better over time.

Agile Methodologies vs Traditional Approaches

Parameters	**Agile Methodology**	**Traditional Approach**
Definition	Agile is like building a flexible and adaptable treehouse in stages.	Traditional approaches are like constructing a house with a detailed blueprint.
Chronology of operations	Testing and development processes are performed concurrently.	Testing is done once the development phase is completed.
Organizational structure	It follows iterative organizational structure.	It follows linear organizational structure.

Parameters	Agile Methodology	Traditional Approach
Communication	Agile encourages face-to-face communication.	Traditional approach encourages formal communication.
Number of phases	It consists of only three phases.	It consists of five phases.
Development cost	Less using this methodology.	More using this methodology.
User requirements	Clearly defined user requirements before coding.	Requires interactive user inputs.

Benefits of Agile Methodology

The advantages of the agile model are as follows:

1. **Immediate Feedback:** It allows immediate feedback, which aids software improvement in the next increment.
2. **Adapts to Changing Requirements:** It is a highly adaptable methodology in which rapidly changing requirements, allowing responsive adjustments.
3. **Face-to-Face Communication:** Agile methodology encourages effective face-to-face communication.
4. **Time-Efficient:** It is well-suited for its time-efficient practices, which help in delivering software quickly and reducing time-to-market.
5. **Frequent Changes:** It effectively manages and accommodates frequent changes in project requirements according to stakeholder convenience.
6. **Customer Satisfaction:** It prioritizes customer satisfaction.
7. **Flexibility and Adaptability:** Agile methodologies are known for their flexibility and adaptability.

Limitations of Agile Methodology

The disadvantages of the agile model are as follows:

1. **Less Documentation:** Agile methodologies focus on less documentation; it prioritizes working on projects rather than paperwork.
2. **Challenges in Large Organizations:** Busy schedule of clients can make daily meetup and face-to-face communication difficult.
3. **Need for Senior Programmers:** It may require experienced programmers to make critical decisions during the development of software.
4. **Limited Scope Control:** It has less rigid scope control, which may not be suitable in certain situations.
5. **Predictability**: Compared to more structured project management methods, it may lack predictability.

Popular Agile Tools for Software Development

An Agile Tool for software development is a software application or a platform that enables the teams to manage and track the Agile project more efficiently.

Some popular agile tools are:

1. Jira

2. ClickUp
3. Mural
4. Kanbanize
5. GitHub
6. Monday.com
7. Jenkins
8. Shortcut
9. Asana
10. Planbox

Agile Software Development Interview Questions

In this article on 'Agile Software Development Interview Questions,' we will look at key features of Agile methodology and ask questions that will help you land the job you want in this exciting and in-demand industry.

Conclusion

In conclusion, the Agile model is like building a project in small, flexible steps. It's about being quick to adapt, working closely with customers, and delivering value in small doses. This approach has become popular for many companies because it helps them meet changing needs and make customers happy.

Use the Agile model when your project needs to be flexible, your customers' needs might change, and you want to deliver small parts of your project regularly to make them happy. It's like building a puzzle piece by piece, adapting as needed.

Dynamic System Development Method

The **Dynamic Systems Development technique (DSDM)** is an associate degree agile code development approach that provides a framework for building and maintaining systems. The DSDM philosophy is borrowed from a modified version of the sociologist principle—80 % of An application is often delivered in twenty percent of the time it'd desire deliver the entire (100 percent) application.

DSDM is An iterative code method within which every iteration follows the 80% rule that simply enough work is needed for every increment to facilitate movement to the following increment. The remaining detail is often completed later once a lot of business necessities are noted or changes are requested and accommodated.

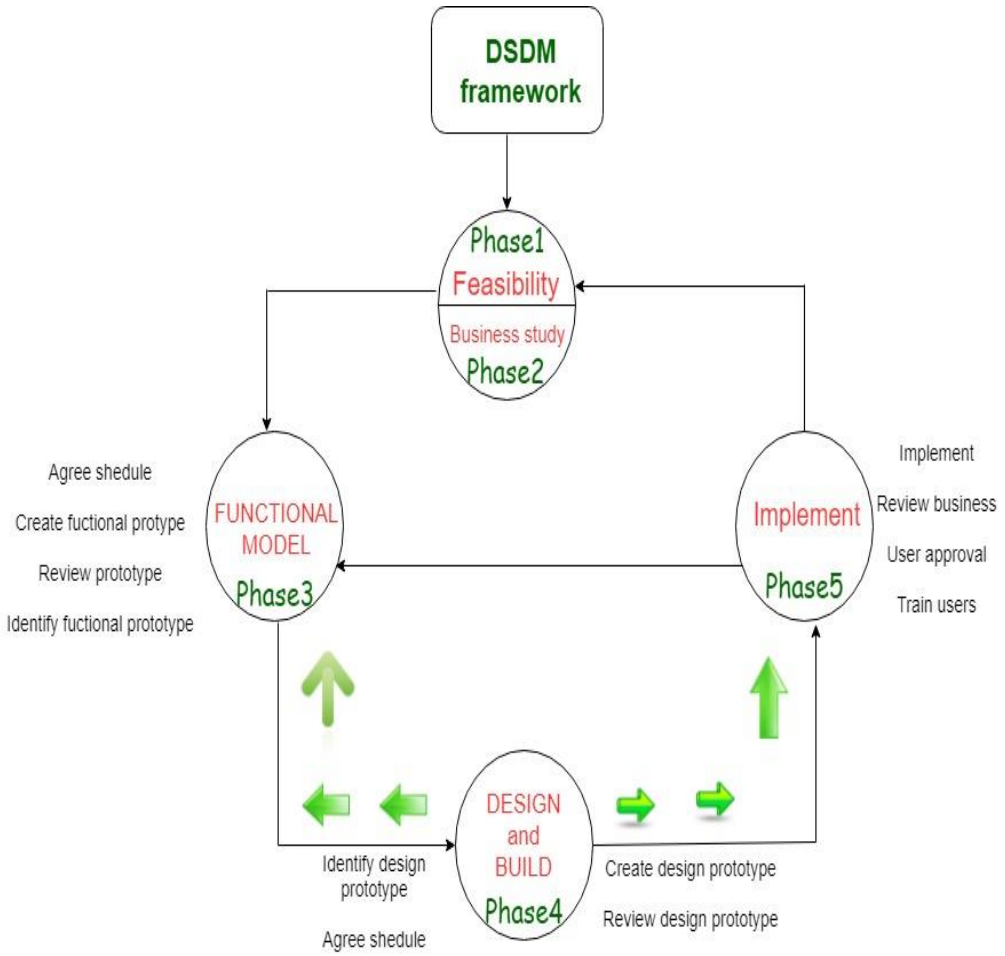
The DSDM tool (www.dsdm.org) could be a worldwide cluster of member companies that put together tackle the role of "keeper" of the strategy. The pool has outlined AN Agile Development Model, known as the DSDM life cycle that defines 3 different unvarying cycles, preceded by 2 further life cycle activities:

1. **Feasibility Study:**
   It establishes the essential business necessities and constraints related to the applying to be designed then assesses whether or not the application could be a viable candidate for the DSDM method.
2. **Business Study:**
   It establishes the use and knowledge necessities that may permit the applying to supply business value; additionally, it is the essential application design and identifies the maintainability necessities for the applying.
3. **Functional Model Iteration:**
   It produces a collection of progressive prototypes that demonstrate practicality for the client.
   (Note: All DSDM prototypes are supposed to evolve into the deliverable application.)

The intent throughout this unvarying cycle is to collect further necessities by eliciting feedback from users as they exercise the paradigm.

4. **Design and Build Iteration:**
   It revisits prototypes designed throughout useful model iteration to make sure that everyone has been designed during a manner that may alter it to supply operational business price for finish users. In some cases, useful model iteration and style and build iteration occur at the same time.
5. **Implementation:**
   It places the newest code increment (an "operationalized" prototype) into the operational surroundings. It ought to be noted that:
   - **(a)** the increment might not 100% complete or,
   - **(b)** changes are also requested because the increment is placed into place. In either case, DSDM development work continues by returning to the useful model iteration activity.

Below diagram describe the DSDM life cycle:



Dynamic Systems Development Method life cycle

DSDM is often combined with XP to supply a mixed approach that defines a solid method

model (the DSDM life cycle) with the barmy and bolt practices (XP) that are needed to create code increments. additionally, the ASD ideas of collaboration and self-organizing groups are often tailored to a combined method model.

Extreme Programming

Extreme programming (XP) is one of the most important software development frameworks of Agile models. It is used to improve software quality and responsiveness to customer requirements.

Table of Content

The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.

What is Extreme Programming (XP)?

Extreme Programming (XP) is an Agile software development methodology that focuses on delivering high-quality software through frequent and continuous feedback, collaboration, and adaptation. XP emphasizes a close working relationship between the development team, the customer, and stakeholders, with an emphasis on rapid, iterative development and deployment.

Extreme Programming (XP)

Agile development approaches evolved in the 1990s as a reaction to documentation and bureaucracy-based processes, particularly the waterfall approach. Agile approaches are based on some common principles, some of which are:

1. Working software is the key measure of progress in a project.
2. For progress in a project, therefore software should be developed and delivered rapidly in small increments.
3. Even late changes in the requirements should be entertained.
4. Face-to-face communication is preferred over documentation.
5. Continuous feedback and involvement of customers are necessary for developing good-quality software.
6. A simple design that involves and improves with time is a better approach than doing an elaborate design up front for handling all possible scenarios.
7. The delivery dates are decided by empowered teams of talented individuals.

Extreme programming is one of the most popular and well-known approaches in the family of agile methods. an XP project starts with user stories which are short descriptions of what scenarios the customers and users would like the system to support. Each story is written on a separate card, so they can be flexibly grouped.

Good Practices in Extreme Programming

Some of the good practices that have been recognized in the extreme programming model and suggested to maximize their use are given below:

Extreme Programming Good Practices

- **Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their work between them every hour.
- **Testing:** Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach, test cases are written even before any code is written.
- **Incremental development:** Incremental development is very good because customer feedback is gained and based on this development team comes up with new increments every few days after each iteration.
- **Simplicity:** Simplicity makes it easier to develop good-quality code as well as to test and debug it.
- **Design:** Good quality design is important to develop good quality software. So, everybody should design daily.
- **Integration testing:** Integration Testing helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.

Basic Principles of Extreme programming

XP is based on the frequent iteration through which the developers implement User Stories. User stories are simple and informal statements of the customer about the functionalities needed. A User Story is a conventional description by the user of a feature of the required system. It does not mention finer details such as the different scenarios that can occur. Based on User stories, the project team proposes Metaphors. Metaphors are a common vision of how the system would work. The development team may decide to build a Spike for some features. A Spike is a very simple program that is constructed to explore the suitability of a solution being proposed. It can be considered similar to a prototype. Some of the basic activities that are followed during software development by using the XP model are given below:

- **Coding:** The concept of coding which is used in the XP model is slightly different from traditional coding. Here, the coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system, and choosing among several alternative solutions.
- **Testing:** The XP model gives high importance to testing and considers it to be the primary factor in developing fault-free software.
- **Listening:** The developers need to carefully listen to the customers if they have to develop good quality software. Sometimes programmers may not have the depth knowledge of the system to be developed. So, the programmers should understand properly the functionality of the system and they have to listen to the customers.
- **Designing:** Without a proper design, a system implementation becomes too complex, and very difficult to understand the solution, thus making maintenance expensive. A good design results elimination of complex dependencies within a system. So, effective use of suitable design is emphasized.
- **Feedback:** One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs. Frequent contact with the customer makes the development effective.
- **Simplicity:** The main principle of the XP model is to develop a simple system that will work efficiently in the present time, rather than trying to build something that would take time and may never be used. It focuses on some specific features that are

immediately needed, rather than engaging time and effort on speculations of future requirements.

- **Pair Programming:** XP encourages pair programming where two developers work together at the same workstation. This approach helps in knowledge sharing, reduces errors, and improves code quality.
- **Continuous Integration:** In XP, developers integrate their code into a shared repository several times a day. This helps to detect and resolve integration issues early on in the development process.
- **Refactoring:** XP encourages refactoring, which is the process of restructuring existing code to make it more efficient and maintainable. Refactoring helps to keep the codebase clean, organized, and easy to understand.
- **Collective Code Ownership:** In XP, there is no individual ownership of code. Instead, the entire team is responsible for the codebase. This approach ensures that all team members have a sense of ownership and responsibility towards the code.
- **Planning Game:** XP follows a planning game, where the customer and the development team collaborate to prioritize and plan development tasks. This approach helps to ensure that the team is working on the most important features and delivers value to the customer.
- **On-site Customer:** XP requires an on-site customer who works closely with the development team throughout the project. This approach helps to ensure that the customer's needs are understood and met, and also facilitates communication and feedback.

Applications of Extreme Programming (XP)

Some of the projects that are suitable to develop using the XP model are given below:

- **Small projects:** The XP model is very useful in small projects consisting of small teams as face-to-face meeting is easier to achieve.
- **Projects involving new technology or Research projects:** This type of project faces changing requirements rapidly and technical problems. So XP model is used to complete this type of project.
- **Web development projects:** The XP model is well-suited for web development projects as the development process is iterative and requires frequent testing to ensure the system meets the requirements.
- **Collaborative projects:** The XP model is useful for collaborative projects that require close collaboration between the development team and the customer.
- **Projects with tight deadlines:** The XP model can be used in projects that have a tight deadline, as it emphasizes simplicity and iterative development.
- **Projects with rapidly changing requirements: The** XP model is designed to handle rapidly changing requirements, making it suitable for projects where requirements may change frequently.
- **Projects where quality is a high priority:** The XP model places a strong emphasis on testing and quality assurance, making it a suitable approach for projects where quality is a high priority.

XP, and other agile methods, are suitable for situations where the volume and space of requirements change are high and where requirement risks are considerable.

Life Cycle of Extreme Programming (XP)

The Extreme Programming Life Cycle consist of five phases:

Life Cycle of Extreme Programming (XP)

1. **Planning:** The first stage of Extreme Programming is planning. During this phase, clients define their needs in concise descriptions known as user stories. The team calculates the effort required for each story and schedules releases according to priority and effort.
2. **Design:** The team creates only the essential design needed for current user stories, using a common analogy or story to help everyone understand the overall system architecture and keep the design straightforward and clear.
3. **Coding:** Extreme Programming (XP) promotes pair programming i.e**.** wo developers work together at one workstation, enhancing code quality and knowledge sharing. They write tests before coding to ensure functionality from the start (TDD), and frequently integrate their code into a shared repository with automated tests to catch issues early.
4. **Testing:** Extreme Programming (XP) gives more importance to testing that consist of both unit tests and acceptance test**.** Unit tests, which are automated, check if specific features work correctly. Acceptance tests, conducted by customers, ensure that the overall system meets initial requirements. This continuous testing ensures the software's quality and alignment with customer needs.
5. **Listening:** In the listening phase regular feedback from customers to ensure the product meets their needs and to adapt to any changes.

Values of Extreme Programming (XP)

There are five core values of Extreme Programming (XP)

Values of Extreme Programming (XP)

1. **Communication:** The essence of communication is for information and ideas to be exchanged amongst development team members so that everyone has an understanding of the system requirements and goals. Extreme Programming (XP) supports this by allowing open and frequent communication between members of a team.
2. **Simplicity:** Keeping things as simple as possible helps reduce complexity and makes it easier to understand and maintain the code.
3. **Feedback:** Feedback loops which are constant are among testing as well as customer involvements which helps in detecting problems earlier during development.
4. **Courage:** Team members are encouraged to take risks, speak up about problems, and adapt to change without fear of repercussions.
5. **Respect**: Every member's input or opinion is appreciated which promotes a collective way of working among people who are supportive within a certain group.

Advantages of Extreme Programming (XP)

- **Slipped schedules:** Timely delivery is ensured through slipping timetables and doable development cycles.
- **Misunderstanding the business and/or domain** − Constant contact and explanations are ensured by including the client on the team.
- **Canceled projects:** Focusing on ongoing customer engagement guarantees open communication with the consumer and prompt problem-solving.
- **Staff turnover:** Teamwork that is focused on cooperation provides excitement and goodwill. Team spirit is fostered by multidisciplinary cohesion.
- **Costs incurred in changes:** Extensive and continuing testing ensures that the modifications do not impair the functioning of the system. A functioning system always guarantees that there is enough time to accommodate changes without impairing ongoing operations.
- **Business changes:** Changes are accepted at any moment since they are seen to be inevitable.

- **Production and post-delivery defects:** the unit tests to find and repair bugs as soon as possible.

Conclusion

Extreme Programming (XP) is a Software Development Methodology, known for its flexibility, collaboration and rapid feedback using techniques like continuous testing, frequent releases, and pair programming, in which two programmers collaborate on the same code. XP supports user involvement throughout the development process while prioritizing simplicity and communication. Overall, XP aims to deliver high-quality software quickly and adapt to changing requirements effectively.

Managing interactive processes

Project management involves several key phases that guide the project from initiation to completion, ensuring that objectives are met efficiently and effectively. It's like having a step-by-step guide to follow, ensuring you stay on track and reach your goals smoothly. These phases typically include initiation, planning, execution, monitoring and control, and closure. Each phase is crucial for managing tasks, resources, timelines, and deliverables, ultimately leading to the successful completion of projects.

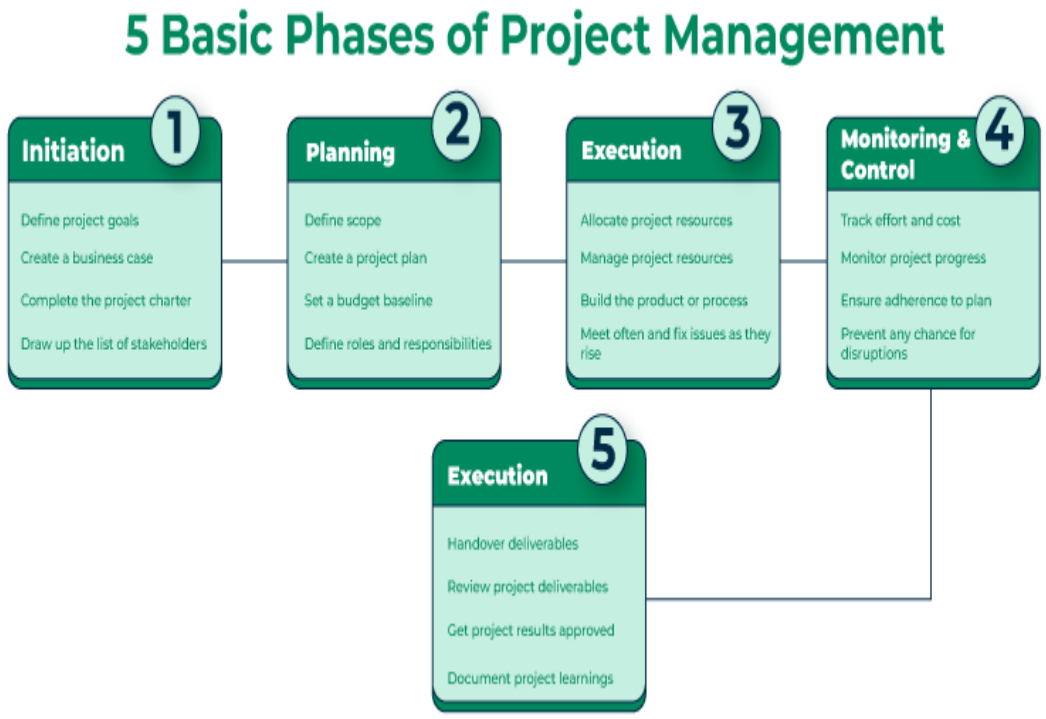
This article explores each phase of project management in detail, highlighting their importance and how they contribute to project success.

Table of Content

What is the Project Management Process?

Project Management is the discipline of planning, monitoring, and controlling software projects, identifying the scope, estimating the work involved, and creating a project schedule. Along with it is also responsible for keeping the team up to date on the project's progress handling problems and discussing solutions.

Phases of Project Management Process



Phases of project management Process

1. Project Initiation/Feasibility Study:

A feasibility study explores system requirements to determine project feasibility. There are several fields of feasibility study including economic feasibility, operational feasibility, and technical feasibility. The goal is to determine whether the system can be implemented or not. The process of feasibility study takes as input the required details as specified by the user and other domain-specific details. The output of this process simply tells whether the project should be undertaken or not and if yes, what would the constraints be. Additionally, all the risks and their potential effects on the projects are also evaluated before a decision to start the project is taken.

This phase of Project Management involves defining the project, identifying the stakeholders, and establishing the project's goals and objectives.

2. Project Planning:

In this phase of Project Management, the project manager defines the scope of the project, develops a detailed project plan, and identifies the resources required to complete the project. A detailed plan stating a stepwise strategy to achieve the listed objectives is an integral part of any project. Planning consists of the following activities:

- Set objectives or goals
- Develop strategies
- Develop project policies
- Determine courses of action
- Making planning decisions
- Set procedures and rules for the project
- Develop a software project plan
- Prepare budget
- Conduct risk management
- Document software project plans

This step also involves the construction of a work breakdown structure(WBS). It also includes size, effort, schedule, and cost estimation using various techniques.

3. Project Execution:

The Project Execution phase of the Project Management process involves the actual implementation of the project, including the allocation of resources, the execution of tasks, and the monitoring and control of project progress. A project is executed by choosing an appropriate software development lifecycle model (SDLC). It includes several steps including requirements analysis, design, coding, testing and implementation, testing, delivery, and maintenance. Many factors need to be considered while doing so including the size of the system, the nature of the project, time and budget constraints, domain requirements, etc. An inappropriate SDLC can lead to the failure of the project.

4. Project Monitoring and Controlling:

This phase of Project Management involves tracking the project's progress, comparing actual results to the project plan, and making changes to the project as necessary. In the project management process, in that third and fourth phases are not sequential in nature. These phase will run regularly with the project execution phase. These phase will ensure that project deliverable are need to meet.

During the monitoring phase of the project management phases. The manager will respond to the proper tracking the cost and effort during the process. This tracking will not ensure that budget is also important for the future projects.

5. Project Closing:

There can be many reasons for the termination of a project. Though expecting a project to terminate after successful completion is conventional, at times, a project may also terminate without completion. Projects have to be closed down when the requirements are not fulfilled according to given time and cost constraints. This phase of Project Management involves completing the project, documenting the results, and closing out any open issues.

Some reasons for failure include:

- **Fast-changing technology**
- **The project running out of time**
- **Organizational politics**
- **Too much change in customer requirements**
- **Project exceeding budget or funds**

Once the project is terminated, a post-performance analysis is done. Also, a final report is published describing the experiences, lessons learned, and recommendations for handling future projects.

Project management is a systematic approach to planning, organizing, and controlling the resources required to achieve specific project goals and objectives. The project management process involves a set of activities that are performed to plan, execute, and close a project. The project management process can be divided into several phases, each of which has a specific purpose and set of tasks.

Principles of Project Management

The project should be effective so that the project begins with well-defined tasks. Effective project planning helps to minimize the additional costs incurred on the project while it is in progress.

1. Planning is necessary: Planning should be done before a project begins.
2. Risk analysis: Before starting the project, senior management and the project management team should consider the risks that may affect the project.
3. Tracking of project plan: Once the project plan is prepared, it should be tracked and modified accordingly.

4. Most quality standards and produce quality deliverables: The project plan should identify processes by which the project management team can ensure quality in software.
5. Description of flexibility to accommodate changes: The result of project planning is recorded in the form of a project plan, which should allow new changes to be accommodated when the project is in progress

Advantages of the Project Management process:

- Provides a structured approach to managing projects.
- Helps to define project objectives and requirements.
- Facilitates effective communication and collaboration among team members.
- Helps to manage project risks and issues.
- Ensures that the project is delivered on time and within budget.

Disadvantages of the Project Management Process:

- Can be time-consuming and bureaucratic
- May be inflexible and less adaptable to changes
- Requires a skilled project manager to implement effectively
- May not be suitable for small or simple projects.

What is a Project life Cycle in Project Management?

The Project Life Cycle is a framework that outlines the phases a project goes through from initiation to closure. It typically includes five main phases. Each phase has specific objectives, activities, and deliverables, and the Project Life Cycle provides a structured approach for managing and executing projects efficiently. The Project Life Cycle helps project managers and teams understand the project's progression, allocate resources effectively, manage risks, and ensure successful project outcomes.

Responsibilities of Software Project Manager:

- Proper project management is essential for the successful completion of a software project and the person who is responsible for it is called the project manager.
- To do his job effectively, the project manager must have a certain set of skills. This section discusses both the job responsibilities of a project manager and the skills required by him.

Job Responsibilities of Software Project Manager:

- Involves the senior managers in the process of appointing team members.
- Builds the project team and assigns tasks to various team members.
- Responsible for effective project planning and scheduling, project monitoring, and control activities to achieve the project objectives.
- Acts as a communicator between the senior management and the other persons involved in the project like the development team and internal and external stakeholders.
- Effectively resolves issues that arise between the team members by changing their roles and responsibilities.
- Modifies the project plan(if required)to deal with the situation.

Conclusion

Project management is a procedure that requires responsibility. The project management process brings all the other project tasks together and ensures that the project runs smoothly. Understanding the phases of project management—initiation, planning, execution, monitoring and control, and closure—is crucial for successfully managing any project. Each phase plays a vital role in ensuring that projects are completed on time, within budget, and to the satisfaction of stakeholders. By meticulously following these phases, project managers can effectively coordinate tasks, resources, and teams, address challenges proactively, and deliver high-quality outcomes.

Basics of Software estimation

In the fast-paced world of Software Engineering, accurately estimating the size of a project is key to its success. Understanding how big a project will be helps predict the resources, time, and cost needed, ensuring the project starts off on the right foot.
Project Size Estimation Techniques are vital because they allow you to plan and allocate the necessary resources effectively. This is a critical step in software engineering that ensures projects are feasible and managed efficiently from the start.

Table of Content

What is Project Size Estimation?

Project size estimation is determining the scope and resources required for the project.

1. It involves assessing the various aspects of the project to estimate the effort, time, cost, and resources needed to complete the project.
2. Accurate project size estimation is important for effective and efficient project planning, management, and execution.

Importance of Project Size Estimation

Here are some of the reasons why project size estimation is critical in project management:

1. **Financial Planning:** Project size estimation helps in planning the financial aspects of the project, thus helping to avoid financial shortfalls.
2. **Resource Planning:** It ensures the necessary resources are identified and allocated accordingly.
3. **Timeline Creation:** It facilitates the development of realistic timelines and milestones for the project.
4. **Identifying Risks:** It helps to identify potential risks associated with overall project execution.
5. **Detailed Planning:** It helps to create a detailed plan for the project execution, ensuring all the aspects of the project are considered.
6. **Planning Quality Assurance:** It helps in planning quality assurance activities and ensuring that the project outcomes meet the required standards.

Who Estimates Projects Size?

Here are the key roles involved in estimating the project size:

1. **Project Manager:** Project manager is responsible for overseeing the estimation process.
2. **Subject Matter Experts (SMEs):** SMEs provide detailed knowledge related to the specific areas of the project.
3. **Business Analysts:** Business Analysts help in understanding and documenting the project requirements.
4. **Technical Leads:** They estimate the technical aspects of the project such as system design, development, integration, and testing.
5. **Developers:** They will provide detailed estimates for the tasks they will handle.

6. **Financial Analysts:** They provide estimates related to the financial aspects of the project including labor costs, material costs, and other expenses.
7. **Risk Managers:** They assess the potential risks that could impact the projects' size and effort.
8. **Clients:** They provide input on project requirements, constraints, and expectations.

Different Methods of Project Estimation

1. **Expert Judgment:** In this technique, a group of experts in the relevant field estimates the project size based on their experience and expertise. This technique is often used when there is limited information available about the project.
2. **Analogous Estimation:** This technique involves estimating the project size based on the similarities between the current project and previously completed projects. This technique is useful when historical data is available for similar projects.
3. **Bottom-up Estimation:** In this technique, the project is divided into smaller modules or tasks, and each task is estimated separately. The estimates are then aggregated to arrive at the overall project estimate.
4. **Three-point Estimation:** This technique involves estimating the project size using three values: optimistic, pessimistic, and most likely. These values are then used to calculate the expected project size using a formula such as the PERT formula.
5. **Function Points:** This technique involves estimating the project size based on the functionality provided by the software. Function points consider factors such as inputs, outputs, inquiries, and files to arrive at the project size estimate.
6. **Use Case Points:** This technique involves estimating the project size based on the number of use cases that the software must support. Use case points consider factors such as the complexity of each use case, the number of actors involved, and the number of use cases.
7. **Parametric Estimation:** For precise size estimation, mathematical models founded on project parameters and historical data are used.
8. **COCOMO (Constructive Cost Model):** It is an algorithmic model that estimates effort, time, and cost in software development projects by taking into account several different elements.
9. **Wideband Delphi:** Consensus-based estimating method for balanced size estimations that combines expert estimates from anonymous experts with cooperative conversations.
10. **Monte Carlo Simulation:** This technique, which works especially well for complicated and unpredictable projects, estimates project size and analyses hazards using statistical methods and random sampling.

Each of these techniques has its strengths and weaknesses, and the choice of technique depends on various factors such as the project's complexity, available data, and the expertise of the team.

Estimating the Size of the Software

Estimation of the size of the software is an essential part of Software Project Management. It helps the project manager to further predict the effort and time that will be needed to build the project. Here are some of the measures that are used in project size estimation:

1. Lines of Code (LOC)

As the name suggests, LOC counts the total number of lines of source code in a project. The units of LOC are:

1. **KLOC:** Thousand lines of code
2. **NLOC:** Non-comment lines of code
3. **KDSI:** Thousands of delivered source instruction

- The size is estimated by comparing it with the existing systems of the same kind. The experts use it to predict the required size of various components of software and then add them to get the total size.
- It's tough to estimate LOC by analyzing the problem definition. Only after the whole code has been developed can accurate LOC be estimated. This statistic is of little utility to project managers because project planning must be completed before development activity can begin.
- Two separate source files having a similar number of lines may not require the same effort. A file with complicated logic would take longer to create than one with simple logic. Proper estimation may not be attainable based on LOC.
- The length of time it takes to solve an issue is measured in LOC. This statistic will differ greatly from one programmer to the next. A seasoned programmer can write the same logic in fewer lines than a newbie coder.

Advantages:

1. Universally accepted and is used in many models like COCOMO.
2. Estimation is closer to the developer's perspective.
3. Both people throughout the world utilize and accept it.
4. At project completion, LOC is easily quantified.
5. It has a specific connection to the result.
6. Simple to use.

Disadvantages:

1. Different programming languages contain a different number of lines.
2. No proper industry standard exists for this technique.
3. It is difficult to estimate the size using this technique in the early stages of the project.
4. When platforms and languages are different, LOC cannot be used to normalize.

2. Number of Entities in ER Diagram

ER model provides a static view of the project. It describes the entities and their relationships. The number of entities in the ER model can be used to measure the estimation of the size of the project. The number of entities depends on the size of the project. This is because more entities needed more classes/structures thus leading to more coding.

Advantages:

1. Size estimation can be done during the initial stages of planning.
2. The number of entities is independent of the programming technologies used.

Disadvantages:

1. No fixed standards exist. Some entities contribute more to project size than others.
2. Just like FPA, it is less used in the cost estimation model. Hence, it must be converted to LOC.

3. Total Number of Processes in DFD

Data Flow Diagram(DFD) represents the functional view of software. The model depicts the main processes/functions involved in software and the flow of data between them. Utilization of the number of functions in DFD to predict software size. Already existing processes of similar type are studied and used to estimate the size of the process. The sum of the estimated size of each process gives the final estimated size.

Advantages:

1. It is independent of the programming language.
2. Each major process can be decomposed into smaller processes. This will increase the accuracy of the estimation.

Disadvantages:

1. Studying similar kinds of processes to estimate size takes additional time and effort.

2. All software projects are not required for the construction of DFD.

4. Function Point Analysis

In this method, the number and type of functions supported by the software are utilized to find FPC(function point count). The steps in function point analysis are:

1. Count the number of functions of each proposed type.
2. Compute the Unadjusted Function Points(UFP).
3. Find the Total Degree of Influence(TDI).
4. Compute Value Adjustment Factor(VAF).
5. Find the Function Point Count(FPC).

The explanation of the above points is given below:

1. Count the number of functions of each proposed type:

Find the number of functions belonging to the following types:

- External Inputs: Functions related to data entering the system.
- External outputs: Functions related to data exiting the system.
- External Inquiries: They lead to data retrieval from the system but don't change the system.
- Internal Files: Logical files maintained within the system. Log files are not included here.
- External interface Files: These are logical files for other applications which are used by our system.

2. Compute the Unadjusted Function Points(UFP):

Categorize each of the five function types as simple, average, or complex based on their complexity. Multiply the count of each function type with its weighting factor and find the weighted sum. The weighting factors for each type based on their complexity are as follows:

Function type	**Simple**	**Average**	**Complex**
External Inputs	3	4	6
External Output	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

3. Find the Total Degree of Influence:

Use the '14 general characteristics of a system to find the degree of influence of each of them. The sum of all 14 degrees of influence will give the TDI. The range of TDI is 0 to 70. The 14 general characteristics are: Data Communications, Distributed Data Processing, Performance, Heavily Used Configuration, Transaction Rate, On-Line Data Entry, End-user Efficiency, Online Update, Complex Processing Reusability, Installation Ease, Operational Ease, Multiple Sites and Facilitate Change.

Each of the above characteristics is evaluated on a scale of 0-5.

4. Compute Value Adjustment Factor(VAF):
Use the following formula to calculate VAF:
*VAF = (TDI * 0.01) + 0.65*

5. Find the Function Point Count:
Use the following formula to calculate FPC:
*FPC = UFP * VAF*

Advantages:
1. It can be easily used in the early stages of project planning.
2. It is independent of the programming language.
3. It can be used to compare different projects even if they use different technologies(database, language, etc).

Disadvantages:
1. It is not good for real-time systems and embedded systems.
2. Many cost estimation models like COCOMO use LOC and hence FPC must be converted to LOC.

When Should Estimates Take Place?
Project size estimates must take place at multiple key points throughout the project lifecycle. It should take place during the following stages to ensure accuracy and relevance:
1. **Project Initiation:** Project is assessed to determine its feasibility and scope.
2. **Project Planning:** Precise estimates are done to create a realistic budget and timeline.
3. **Project Execution:** Res-estimation when there are significant changes in scope.
4. **Project Monitoring and Control:** Regular reviews to make sure that the project is on track.
5. **Project Closeout:** Comparing original estimates with actual outcomes and documenting estimation accuracy.

Challenges in Project Size Estimation
Project size estimation can be challenging due to multiple factors. Here are some factors that can affect the accuracy and reliability of estimates:
1. **Unclear Requirements:** Initial project requirements can be vague or subject to change, thus making it difficult to estimate accurately.
2. **Lack of Historical Data:** Without access to the data of similar past projects, it becomes difficult to make informed estimates, thus estimates becoming overly optimistic or pessimistic and leading to inaccurate planning.
3. **Interdependencies:** Project with numerous interdependent tasks are harder to estimate due to the complicated interactions between components.
4. **Productivity Variability:** Estimating the productivity of resources and their availability can be challenging due to fluctuations and uncertainties.
5. **Risks:** Identifying and quantifying risks and uncertainties is very difficult. Underestimating the potential risks can lead to inadequate contingency planning, thus causing the project to go off track.

Improving Accuracy in Project Size Estimation
Improving the accuracy of project size estimation involves a combination of techniques and best practices. Here are some key strategies to enhance estimation accuracy:
1. **Define Clear Requirements:** Ensure all project requirements are thoroughly documented and engage all stakeholders early and frequently to clarify and validate the requirements.
2. **Use Historical Data:** Use data from similar past projects to make informed estimates.
3. **Use Estimation Techniques:** Use various estimation techniques like Analogue Estimation, Parametric Estimation, Bottom-Up Estimation, and Three-Point Estimation.

4. **Break Down the Project:** Use Work Breakdown Structure (WBS) and detailed take analysis to make sure that each task is specific and measurable.
5. **Incorporate Expert Judgement:** Engage subject matter experts and experienced team members to provide input on estimates.

Future of Project Size Estimation

The future of project size estimation will be shaped by the advancements in technology and methodologies. Here are some key developments that can define the future of project size estimation:

1. **Smarter Technology:** Artificial intelligence (AI) could analyze past projects and code to give more accurate forecasts, considering how complex the project features are.
2. **Data-Driven Insights:** Instead of just lines of code, estimates could consider factors like the number of users, the type of software (mobile app vs. web app), and how much data it handles.
3. **Human-AI Collaboration:** Combining human expertise with AI can enhance the decision-making process in project size estimation.
4. **Collaborative Platforms:** Tools that facilitate collaboration among geographically dispersed teams can help to enhance the project size estimation process.
5. **Agile Methodologies:** The adoption of agile methodologies can promote continuous estimation and iterative refinement.

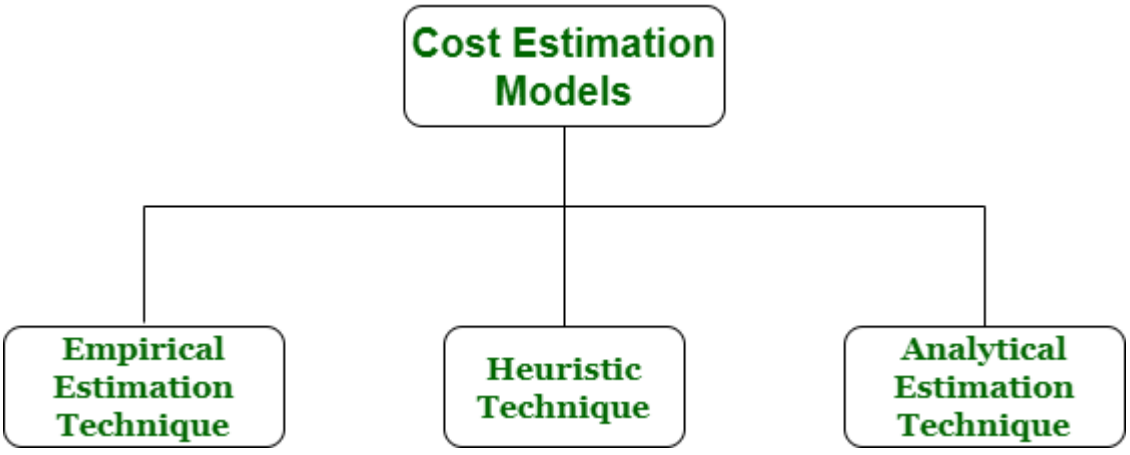
Conclusion

In conclusion, accurate project size estimation is crucial for software project success. Traditional techniques like lines of code have limitations. The future of estimation lies in AI and data-driven insights for better resource allocation, risk management, and project planning.

Effort and cost estimation techniques

Cost estimation simply means a technique that is used to find out the cost estimates. The cost estimate is the financial spend that is done on the efforts to develop and test software in Software Engineering. Cost estimation models are some mathematical algorithms or parametric equations that are used to estimate the cost of a product or a project. Various techniques or models are available for cost estimation, also known as Cost Estimation Models.

Cost Estimation Models as shown below :



Cost Estimation Models

1. **Empirical Estimation Technique –** Empirical estimation is a technique or model in which empirically derived formulas are used for predicting the data that are a required and essential part of the software project planning step. These techniques are usually based on the data that is collected previously from a project and also based on some guesses, prior experience with the development of similar types of projects, and assumptions. It uses the size of the software to estimate the effort. In this technique, an educated guess of project parameters is made. Hence, these models are based on common sense. However, as there are many activities involved in empirical estimation techniques, this technique is formalized. For example Delphi technique and Expert Judgement technique.
2. **Heuristic Technique –** Heuristic word is derived from a Greek word that means "to discover". The heuristic technique is a technique or model that is used for solving problems, learning, or discovery in the practical methods which are used for achieving immediate goals. These techniques are flexible and simple for taking quick decisions through shortcuts and good enough calculations, most probably when working with complex data. But the decisions that are made using this technique are necessary to be optimal. In this technique, the relationship among different project parameters is expressed using mathematical equations. The popular heuristic technique is given by Constructive Cost Model (COCOMO). This technique is also used to increase or speed up the analysis and investment decisions.
3. **Analytical Estimation Technique –** Analytical estimation is a type of technique that is used to measure work. In this technique, firstly the task is divided or broken down into its basic component operations or elements for analyzing. Second, if the standard time is available from some other source, then these sources are applied to each element or component of work. Third, if there is no such time available, then the work is estimated based on the experience of the work. In this technique, results are derived by making certain basic assumptions about the project. Hence, the analytical estimation technique has some scientific basis. Halstead's software science is based on an analytical estimation model.

Other Cost Estimation Models

1. **Function Point Analysis (FPA):** This technique counts the number and complexity of functions that a piece of software can perform to determine how functional and sophisticated it is. The effort needed for development, testing and maintenance can be estimated using this model.
2. **Putnam Model:** This model is a parametric estimation model that estimates effort, time and faults by taking into account the size of the the programme, the expertise of the development team and other project-specific characteristics.
3. **Price-to-Win Estimation:** Often utilized in competitive bidding, this model is concerned with projecting the expenses associated with developing a particular software project in order to secure a contract. It involves looking at market dynamics and competitors.
4. **Models Based on Machine Learning:** Custom cost estimating models can be built using machine learning techniques including neural networks, regression analysis and decision trees. These models are based on past project data. These models are flexible enough to adjust to changing data and project-specific features.
5. **Function Points Model (IFPUG):** A standardized technique for gauging the functionality of software using function points is offered by the International Function

Point Users Group (IFPUG). It is employed to calculate the effort required for software development and maintenance.

COSMIC Full function points

COSMIC FFP – Common Software Measurement International Consortium Full

• Function Point. COSMIC deals with decomposing the system architecture into a hierarchy of

• software layers. Unit is Cfsu(COSMIC functional size units).

• A Data Movement moves one Data Group. A Data Group is a unique cohesive set of data (attributes) specifying an 'object of interest' (i.e. something that is 'of interest' tothe user). Each Data Movement is counted as one CFP (COSMIC function point). COSMIC recognizes 4 (types of) Data Movements: Entry moves data from outside into the process

- Exit moves data from the process to the outside world
- Read moves data from persistent storage to the process
- Write moves data from the process to persistent

• storage. Function Points Function points were defined in 1979 in Measuring Application Development Productivity by Allan Albrecht at IBM. The functional user requirements of the software areidentified and each one is categorized into one of five types: outputs, inquiries, inputs, internal files, and external interfaces. Once the function is identified and categorized into a type, it is then assessed for complexity and assigned a number of function points. Each of these functional user requirements maps to an end-user business function, such as a data entry for an Input or a user query for an Inquiry. This distinction is important because it tends to make the functions measured in function points map easily into user-oriented requirements, but it also tends to hide internal functions (e.g. algorithms), which also require resources to implement.

There is currently no ISO recognized FSM Method that includes algorithmic complexity in the sizing result. Recently there have been different approaches proposed to deal with this perceived weakness, implemented in several commercial software products. The variations of the Albrecht- ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY IT 8075 – SOFTWARE PROJECT MANAGEMENT based IFPUG method designed to make up for this (and other weaknesses) include: Early and easy function points – Adjusts for problem and data complexity with two questions

• that yield a somewhat subjective complexity measurement; simplifies measurement by eliminating the need to count data elements. Engineering function points – Elements (variable names) and operators (e.g., arithmetic,

• equality/inequality, Boolean) are counted. This variation highlights computational function. The intent is similar to that of the operator/operand-based Halstead complexity measures. Bang measure – Defines a function metric based on twelve primitive (simple) counts that affect

• or show Bang, defined as "the measure of true function to be delivered as perceived by the user." Bang measure may be helpful in evaluating a software unit's value in terms of how much useful function it provides, although there is little evidence in the literature ofsuch application. The use of Bang measure could apply when re-engineering (either complete or piecewise) is being considered, as discussed in Maintenance of Operational Systems—An Overview. Feature points – Adds changes to improve applicability to systems with significant internal

• processing (e.g., operating systems, communications systems). This allows accounting for functions not readily perceivable by the user, but essential for proper operation. Weighted Micro Function Points – One of the newer models (2009) which adjusts function

• points using weights derived from program flow complexity, operand and operator vocabulary, object usage, and algorithm. Benefits The use of function points in favor of lines of code seek to address several additional issues: The risk of "inflation" of the created lines of code, and thus reducing the value of the

• measurement system, if developers are incentivized to be more productive. FP advocates refer to this as measuring the size of the solution instead of the size of the problem. Lines of Code (LOC) measures reward low level languages because more lines of code are

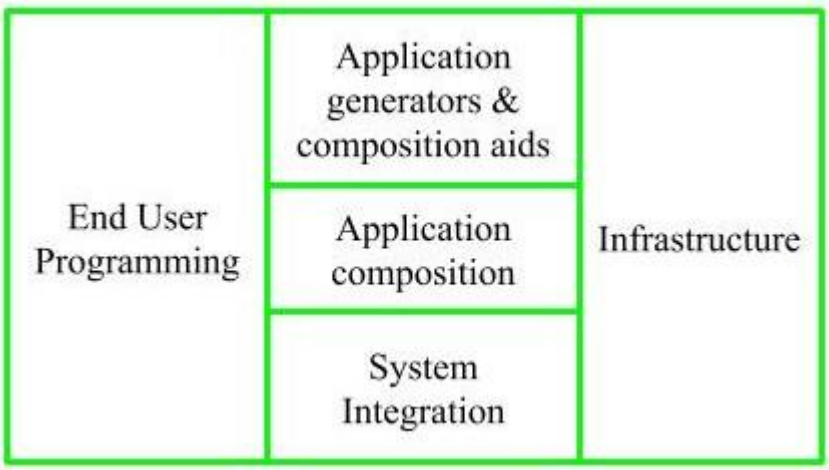
• needed to deliver a similar amount of functionality to a higher level language. C. Jones offersa method of correcting this in his work. LOC measures are not useful during early project phases where estimating the number of lines

• of code that will be delivered is challenging. However, Function Points can be derived from requirements and therefore are useful in methods such as estimation by proxy.

COCOMO II

COCOMO-II is the revised version of the original Cocomo (Constructive Cost Model) and was developed at the University of Southern California. It is the model that allows one to estimate the cost, effort, and schedule when planning a new software development activity.

Sub-Models of COCOMO Model

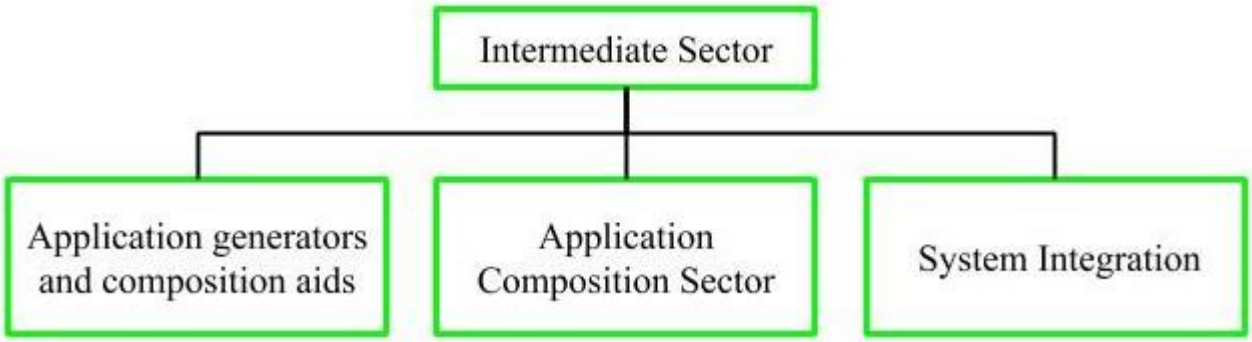


COCOMO Sub-models

1. End User Programming

Application generators are used in this sub-model. End user write the code by using these application generators. For Example, Spreadsheets, report generator, etc.

2. Intermediate Sector



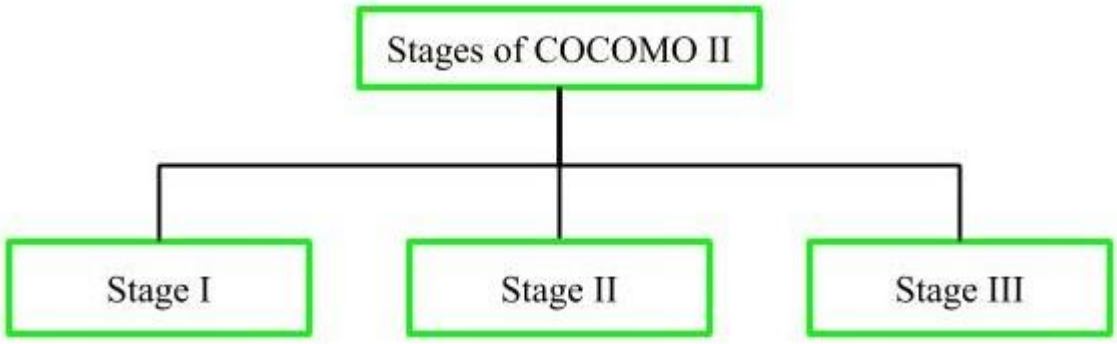
COCOMO Intermediate Sector

- **Application Generators and Composition Aids:** This category will create largely prepackaged capabilities for user programming. Their product will have many reusable components. Typical firms operating in this sector are Microsoft, Lotus, Oracle, IBM, Borland, Novell.
- **Application Composition Sector:** This category is too diversified and to be handled by prepackaged solutions. It includes GUI, Databases, domain specific components such as financial, medical or industrial process control packages.
- **System Integration:** This category deals with large scale and highly embedded systems.

3. Infrastructure Sector

This category provides infrastructure for the software development like Operating System, Database Management System, User Interface Management System, Networking System, etc.

Stages of COCOMO II



Stages of COCOMO

1. Stage-I

It supports estimation of prototyping. For this it uses Application Composition Estimation Model. This model is used for the prototyping stage of application generator and system integration.

2. Stage-II

It supports estimation in the early design stage of the project, when we less know about it. For this it uses Early Design Estimation Model. This model is used in early design stage of application generators, infrastructure, system integration.

3. Stage-III

It supports estimation in the post architecture stage of a project. For this it uses Post Architecture Estimation Model. This model is used after the completion of the detailed architecture of application generator, infrastructure, system integration.

Want to learn **Software Testing** and **Automation** to help give a kickstart to your career? Any student or professional looking to excel in **Quality Assurance** should enroll in our course, ***Complete Guide to Software Testing and Automation***, only on GeeksforGeeks. Get hands-on learning experience with the latest testing methodologies, automation tools, and industry best practices through practical projects and real-life scenarios. Whether you are a beginner or just looking to build on existing skills, this course will give you the competence necessary to ensure the quality and reliability of software products. Ready to be a **Pro in Software Testing**? Enroll now and Take Your Career to a Whole New Level!