Unit – II

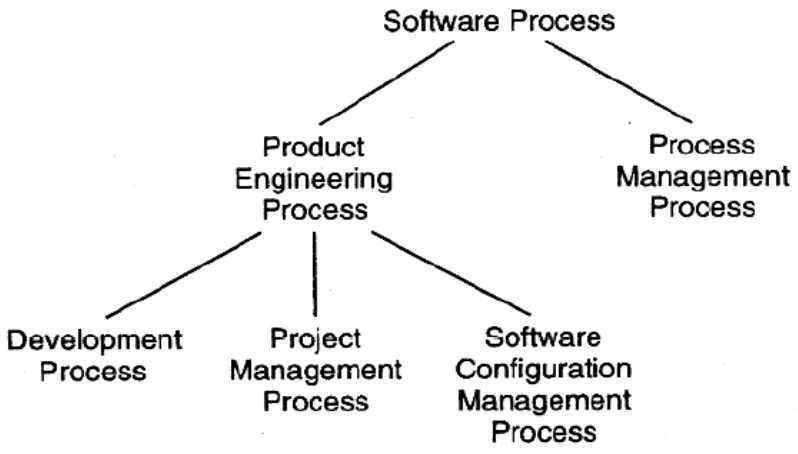
SOFTWARE PROCESS

1. Process and Project

- Software process – is a sequence of activities to be performed to make a software product.
- Several process models have been developed so far for the efficient development of software that meets the user requirements.
- Software process
  - Helps development of software in a systematic and disciplined manner.
  - Helps common understanding of activities among the software developers.
- Software project – is one instance of the software process.

1.2 Component Software Processes

- As defined above, software process is a sequence of steps executed to develop a software product.
- Several component processes exists in a software process. The relationship between the major component processes is shown below as a hierarchy;



- At the bottom, we have development process and project management process.

- The development process specifies all the engineering activities (analysis, design, coding, testing and maintenance) that need to be performed to develop a software product, whereas the management process specifies how to plan and control these activities so that cost, schedule, quality, and other objectives are met.
- During the project, many products are produced which are typically composed of many items (for example, the final source code may be composed of many source files).
- These items keep evolving as the project proceeds, creating many versions on the way.
- So, it is the responsibility of software configuration management process to handle these different versions.
- These three component processes viz. development process, project management process, software configuration management process comes under product engineering process, as their main objective is to produce the desired product.
- The basic objective of the process management process is to improve the software process. By improvement, we mean that the capability of the process to produce quality goods at low cost is improved.

2 Software Development Process Models

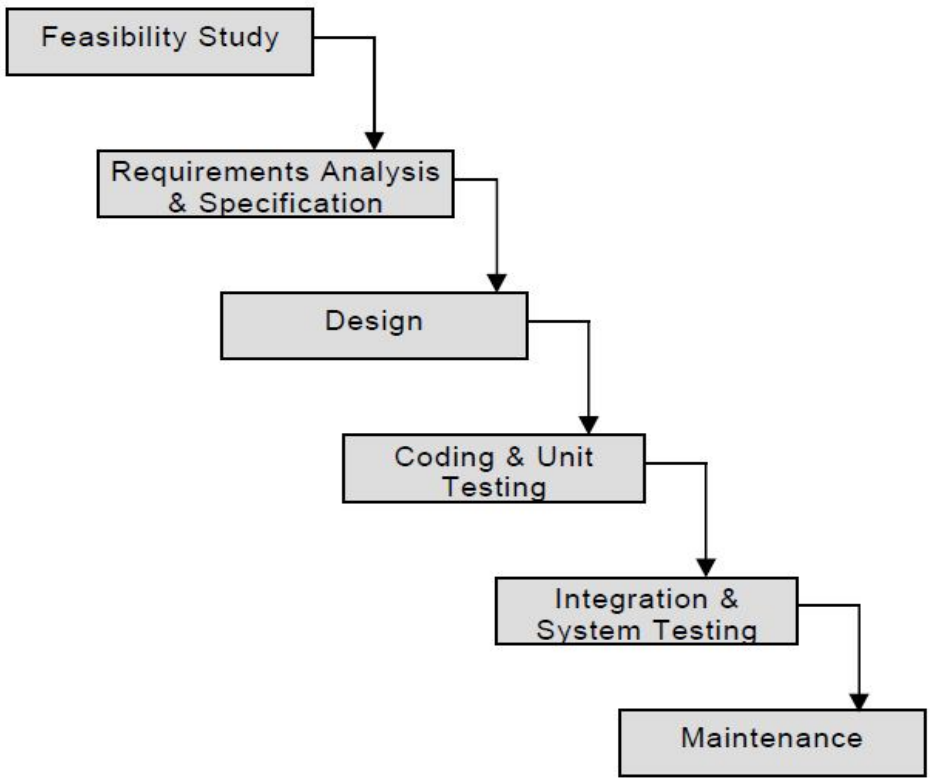
- Software development process defines all the activities undertaken during product development and establishes a precedence ordering among the different activities.
- The development team must identify a suitable process model and then adhere to it.
- The primary advantage of adhering to a life cycle model is that it helps development of software in a systematic and disciplined manner.
- Due to the importance of the development process, various models have been proposed.

2.1 Waterfall Model

- The simplest process model is the waterfall model, which states that the phases are organized in a linear order.
- Though the classical waterfall model is elegant and intuitively obvious, it is not a practical model in the sense that it cannot be used in actual software development projects.
- But all other life cycle models are essentially derived from the classical waterfall model. So, in order to be able to appreciate other life cycle models it is necessary to learn the classical waterfall model.
- Classical waterfall model divides the life cycle into the following phases as shown in the following figure.

Feasibility Study

- The main aim of feasibility study is to determine whether it would be financially and technically feasible to develop the product.



Classical Waterfall Model

- At first project managers or team leaders try to have a rough understanding of what is required to be done by visiting the client side. They study different input data to the system and output data to be produced by the system. They study what kind of processing is needed to be done on these data and they look at the various constraints on the behavior of the system.
- After they have an overall understanding of the problem they investigate different possible solutions. Then they examine each of these solutions in terms of what kind of resources required, what would be the cost of development and what would be the development time for each solution.
- Based on this analysis they pick the best solution and determine whether the solution is feasible financially and technically. They check whether the customer budget would meet the cost of the product and whether they have sufficient technical expertise in the area of development.

The following is an example of a feasibility study undertaken by an organization. It is intended to give you a feel of the activities and issues involved in the feasibility study phase of a typical software project.

Case Study

A mining company named Galaxy Mining Company Ltd. (GMC) has mines located at various places in India. It has about fifty different mine sites spread across eight states. The company employs a large number of mines at each mine site. Mining being a risky profession, the company intends to operate a special provident fund, which would exist in addition to the standard provident fund that the miners already enjoy. The main objective of having the special provident fund (SPF) would be quickly distribute some compensation before the standard provident amount is paid. According to this scheme, each mine site would deduct SPF installments from each miner every month and deposit the same with the CSPFC (Central Special Provident Fund Commissioner). The CSPFC will

maintain all details regarding the SPF installments collected from the miners. GMC employed a reputed software vendor Adventure Software Inc. to undertake the task of developing the software for automating the maintenance of SPF records of all employees. GMC realized that besides saving manpower on bookkeeping work, the software would help in speedy settlement of claim cases. GMC indicated that the amount it can afford for this software to be developed and installed is Rs. 50 millions.

Adventure Software Inc. deputed their project manager to carry out the feasibility study. The project manager discussed the matter with the top managers of GMC to get an overview of the project. He also discussed the issues involved with the several field PF officers at various mine sites to determine the exact details of the project. The project manager identified two broad approaches to solve the problem. One was to have a central database which could be accessed and updated via a satellite connection to various mine sites. The other approach was to have local databases at each mine site and to update the central database periodically through a dial-up connection. These periodic updates could be done on a daily or hourly basis depending on the delay acceptable to GMC in invoking various functions of the software. The project manager found that the second approach was very affordable and more fault-tolerant as the local mine sites could still operate even when the communication link to the central database temporarily failed. The project manager quickly analyzed the database functionalities required, the user-interface issues, and the software handling communication with the mine sites. He arrived at a cost to develop from the analysis. He found that the solution involving maintenance of local databases at the mine sites and periodic updating of a central database was financially and technically feasible. The project manager discussed his solution with the GMC management and found that the solution was acceptable to them as well.

Requirements Analysis and Specification

- The aim of the requirements analysis and specification phase is to understand the exact requirements of the customer and to document them properly. This phase consists of two distinct activities, namely
  - Requirements gathering and analysis, and
  - Requirements specification
- The goal of the requirements gathering activity is to collect all relevant information from the customer regarding the product to be developed. This is done to clearly understand the customer requirements so that incompleteness and inconsistencies are removed.
- The requirements analysis activity is begun by collecting all relevant data regarding the product to be developed from the users of the product and from the customer through interviews and discussions.
- For example, to perform the requirements analysis of a business accounting software required by an organization, the analyst might interview all the accountants of the organization to ascertain their requirements.
- The data collected from such a group of users usually contain several contradictions and ambiguities, since each user typically has only a partial and incomplete view of the system.
- Therefore it is necessary to identify all ambiguities and contradictions in the requirements and resolve them through further discussions with the customer.
- After all ambiguities, inconsistencies, and incompleteness have been resolved and all the requirements properly understood, the requirements specification activity can start. During this activity, the user requirements are systematically organized into a Software Requirements Specification (SRS) document.

- The important components of this document are functional requirements, the non-functional requirements, and the goals of implementation.

Design

- The goal of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language.
- In technical terms, during the design phase the software architecture is derived from the SRS document. Two distinctly different approaches are available: the traditional design approach and the object-oriented design approach.
- **Traditional design approach -** Traditional design consists of two different activities; first a structured analysis of the requirements specification is carried out where the detailed structure of the problem is examined. This is followed by a structured design activity. During structured design, the results of structured analysis are transformed into the software design.
- **Object-oriented design approach -** In this technique, various objects that occur in the problem domain and the solution domain are first identified, and the different relationships that exist among these objects are identified. The object structure is further refined to obtain the detailed design.

Coding and Unit Testing

- The purpose of the coding and unit testing phase (sometimes called the implementation phase) of software development is to translate the software design into source code. Each component of the design is implemented as a program module. The end-product of this phase is a set of program modules that have been individually tested.
- During this phase, each module is unit tested to determine the correct working of all the individual modules. It involves testing each module

in isolation as this is the most efficient way to debug the errors identified at this stage.

Integration and System Testing

- Integration of different modules is undertaken once they have been coded and unit tested.
- During the integration and system testing phase, the modules are integrated in a planned manner.
- The different modules making up a software product are almost never integrated in one shot. Integration is normally carried out incrementally over a number of steps.
- During each integration step, the partially integrated system is tested and a set of previously planned modules are added to it.
- Finally, when all the modules have been successfully integrated and tested, system testing is carried out.
- The goal of system testing is to ensure that the developed system conforms to its requirements laid out in the SRS document. System testing usually consists of three different kinds of testing activities:
    - **α – testing:** It is the system testing performed by the development team.
    - **β – testing:** It is the system testing performed by a friendly set of customers.
    - **Acceptance testing:** It is the system testing performed by the customer himself after the product delivery to determine whether to accept or reject the delivered product.
- System testing is normally carried out in a planned manner according to the system test plan document. The system test plan identifies all testing-related activities that must be performed, specifies the schedule of testing, and allocates resources. It also lists all the test cases and the expected outputs for each test case.

Maintenance

- Maintenance of a typical software product requires much more than the effort necessary to develop the product itself.
- Many studies carried out in the past confirm this and indicate that the relative effort of development of a typical software product to its maintenance effort is roughly in the 40:60 ratio.
- Maintenance involves performing any one or more of the following three kinds of activities:
  - **Corrective maintenance** - Correcting errors that were not discovered during the product development phase.
  - **Perfective maintenance** - Improving the implementation of the system, and enhancing the functionalities of the system according to the customer's requirements.
  - **Adaptive maintenance** - Porting the software to work in a new environment. For example, porting may be required to get the software to work on a new computer platform or with a new operating system.

Limitations of Waterfall Model

- The waterfall model, although widely used, has some strong limitations. Some of the key limitations are:
  1. It assumes that the requirements of a system can be frozen (i.e., baselined) before the design begins. This is possible for small systems. But for large systems, determining the requirements is difficult as the user does not even know the requirements.
  2. Freezing the requirements usually requires choosing the hardware. A large project might take few years to complete. If the hardware is selected early, then due to the speed at which hardware technology is changing, it is likely that the final software will use a hardware that becomes obsolete. This is clearly not desirable for such expensive software systems.

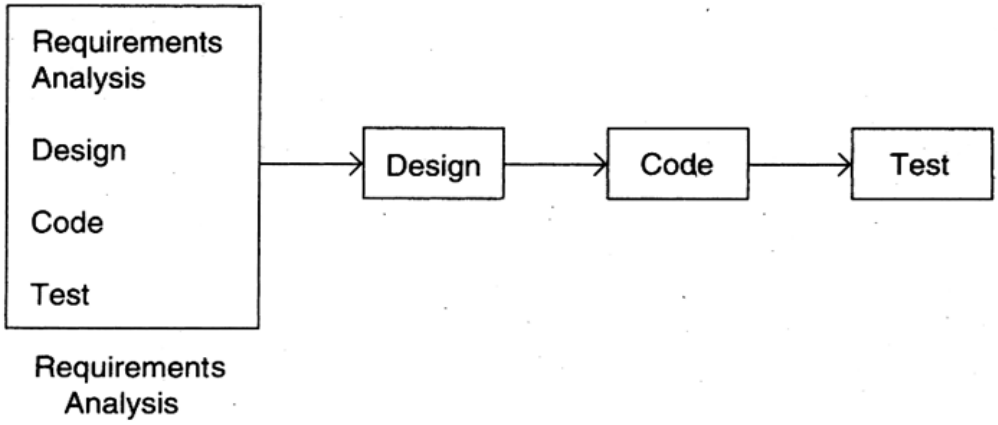
3. It follows the “big bang” approach—the entire software is delivered in one shot at the end. This entails heavy risks, as the user does not know until the end what they are getting. Furthermore, if the project runs out of money in the middle, then there will be no software. That is, it has the “all or nothing” value proposition.
4. It encourages “requirements bloating”. Since all requirements must be specified at the start, it encourages the users and other stakeholders to add unnecessary requirements (which they think that they are needed but, may not be used finally).
5. It is a document-driven process that requires formal documents at the end of each phase.

When to use the Waterfall Model?

- Requirements are well known and stable
- Technology is understood
- Development team have experience with similar projects
- For small projects

2.2 Prototyping

- The goal of a prototyping-based development process is to counter the first limitation of the waterfall model.
- The basic idea here is that instead of freezing the requirements before any design or coding can proceed, a throwaway prototype is built to help understand the requirements.
- A prototype is a toy implementation of the system. It usually exhibits limited functional capabilities.
- This prototype is developed based on the currently known requirements. Development of the prototype obviously undergoes design, coding, and testing, but each of these phases is done more informally.
- By using this prototype, the client can get an actual feel of the system, which can enable the client to better understand the requirements of the desired system. This results in more stable requirements.
- The process model of the prototyping approach is shown below.



- The development of the prototype proceeds as follows; with currently known requirements, a prototype of the system is developed. The prototype is then presented to the clients/users. Based on the feedback given by the clients/users, the prototype is modified to incorporate the suggested changes. This process is continued until the clients/users are satisfied with the prototype.
- Once the prototype is accepted, it means that all the requirements of the system have obtained. After this, the actual design, coding, and testing phases are carried out to develop the software.

Advantages

- Provides a working model to the user early in the process, so, it increases user's confidence.
- The developer gains experience and insight by developing a prototype that results better implementation of requirements.
- The prototyping model serves to clarify requirements which are not clear; hence, it reduces ambiguity and improves communication between the developers and users.
- There is a great involvement of users in software development. Hence, the requirements of the users are met to the greatest extent.
- Helps in reducing risks associated with the software.

Disadvantages

- If the user is not satisfied by the developed prototype, then a new prototype is developed. This process goes on until a perfect prototype is developed. Thus, this model is time consuming and expensive.
- The developer loses focus of the real purpose of prototype and hence, may compromise with the quality of the software. For example, developers may use some inefficient algorithms or inappropriate programming languages while developing the prototype.
- Prototyping can lead to false expectations. For example, a situation may be created where the user believes that the development of the system is finished when it is not.

Where to Use Prototype Model?

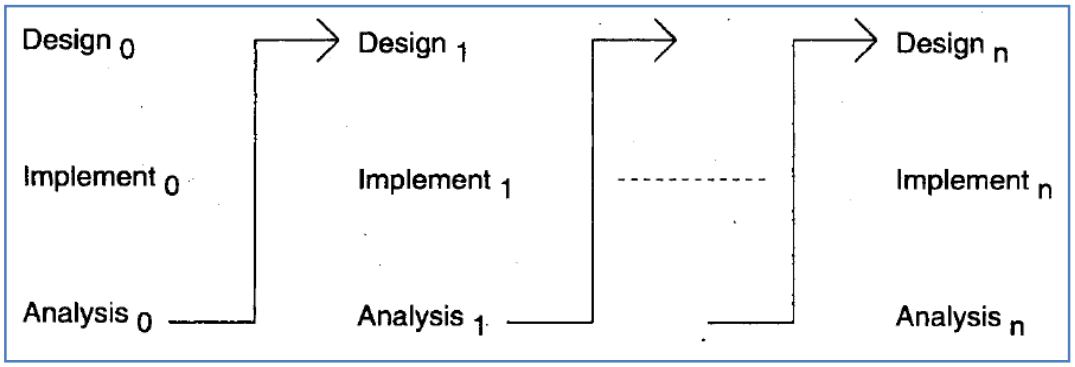
- Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determine the requirements.
- This might be needed for novel systems.

2.3 Iterative Development

- The iterative development process model counters the third and fourth limitations of the waterfall model and tries to combine the benefits of both prototyping and the waterfall model.
- The basic idea is that the software should be developed in increments, each increment adding some functionality to the system until the full system is implemented.
- There are two approaches in iterative model.
- The first approach called the *iterative enhancement model* works as follows;
- A simple initial implementation is done for a subset of requirements that are assumed to be important for the customer.
- A *project control list* is created that contains, in order, all the tasks that must be performed to obtain the final implementation.
- Each step consists of removing the next task from the list, designing the implementation for the selected task, coding and testing the

implementation, and performing an analysis of the partial system obtained. These three phases are called the design phase, implementation phase, and analysis phase.

- The process is iterated until the project control list is empty, at which time the final implementation of the system will be available. The iterative enhancement model is shown below.



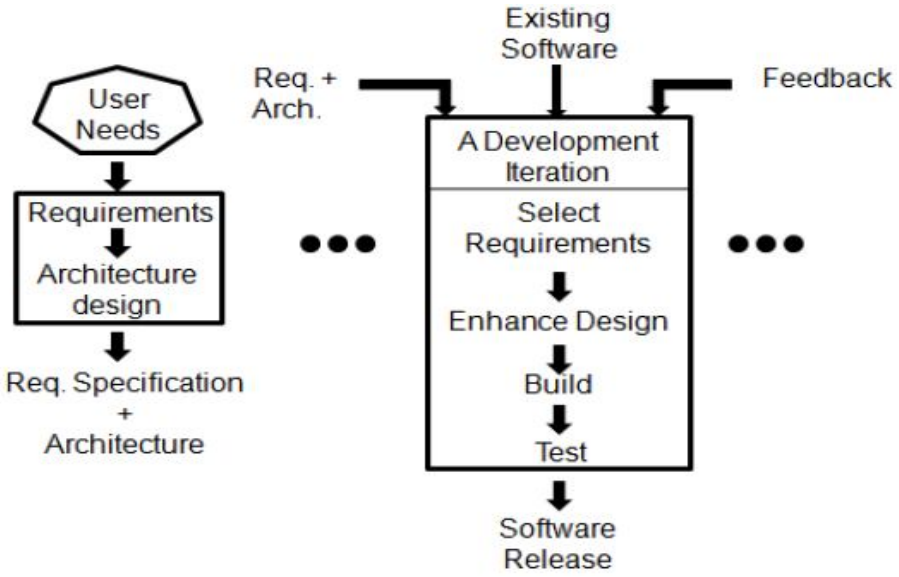
Iterative Enhancement Model

- The second approach for iterative development is to do the requirements and the architecture design in a standard waterfall or prototyping approach, but deliver the software iteratively.

- At each iteration which requirements will be implemented in this release is decided, and then the design is enhanced and code is developed to implement the requirements. The iteration ends with delivery of a working software system.

- This process is repeated until the final system is released.

- Selection of requirements for an iteration is done primarily based on the value the requirement provides to the end users. This approach is shown below.



Iterative delivery approach

Advantages

- In today's world clients do not want to invest too much money without seeing returns. The iterative model permits this—after each iteration some working software is delivered.
- It reduces the rework.
- It has no all-or-nothing risk.
- New requirements can be added easily.

Disadvantages

- Needs good planning and design.
- Becomes invalid when there is time constraint on the project schedule.

2.4 Rational Unified Process

- Rational Unified Process (RUP) is another iterative process model that was designed by Rational Software, now part of IBM.
- It was primarily designed for object-oriented development.

- In this, software development is divided into cycles where each cycle consists of four phases;
  - Inception phase
  - Elaboration phase
  - Construction phase
  - Transition phase

Inception Phase

- Inception phase is first phase of the process
- The objectives of the inception phase are;
  - Establishing the project's software scope and boundary conditions including a clear understanding of what is and is not intended in the product.
  - Illustrating the critical use cases of the system.
  - Demonstrating at least one candidate architecture.
  - Estimating the cost and schedule for the entire project.
  - Estimating the potential risk.

Elaboration Phase

- Elaboration is the second phase of the process
- The objectives of the Elaboration phase are;
  - Baselining the architecture.
  - Baselining the vision.
  - Baselining a plan for the construction phase.
  - Demonstrating that the architecture will support the vision at a reasonable cost in a reasonable time.

Construction Phase

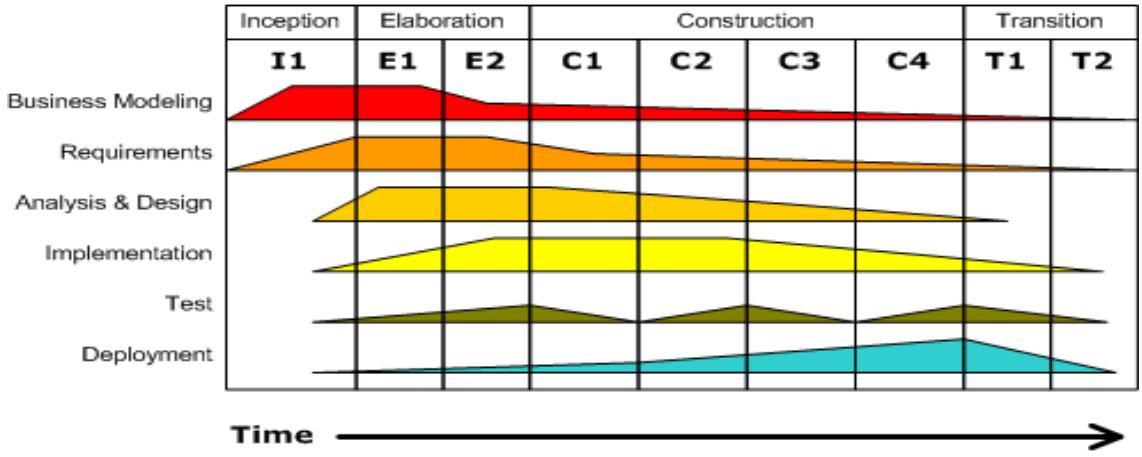
- Construction is the third phase of the process which involves the construction of the software system.

- The objectives of the Construction phase are;
  - Minimizing the development costs by optimizing the resources and avoiding the unnecessary scrap and rework.
  - Achieving adequate quality.
  - Achieving useful versions (alpha, beta and other test releases).

Transition Phase

- Transition is the fourth phase of the process which involves the deployment of the software system to the user community.
- The objectives of the Transition phase are;
  - Beta testing to validate the new system against user expectations.
  - Installing the software at the client sites.
  - Conversion of operational databases.
  - Training of users and maintainers.

The RUP model is shown below.



2.5 Extreme Programming and Agile Process

- To overcome the shortcomings of the waterfall model of, Agile process model was proposed in mid 1990s.

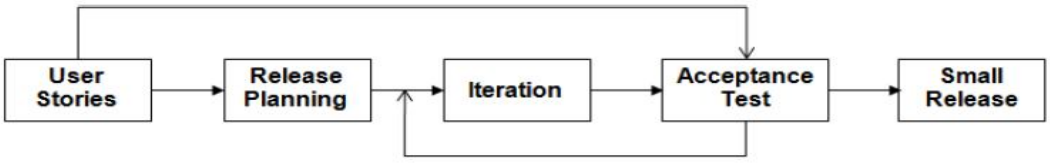
- The agile model was primarily designed to help projects to adapt to change requests.
- Agile approaches are based on some common principles, some of which are;
  - Working software is the key measure of progress in a project.
  - Software should be developed and delivered rapidly in small increments.
  - Even late changes in the requirements should be entertained.
  - Face-to-face communication is preferred over documentation.
  - Continuous feedback and involvement of customer is necessary for developing good-quality software.
  - Simple design which evolves and improves with time is a better approach than doing an elaborate design for handling all possible scenarios.
  - The delivery dates are decided by empowered teams of talented individuals.
- Some popular agile methodologies are XP (Extreme programming), Scrum, Unified Process, Crystal, DSDM and Lean.

Extreme programming (XP)

- An extreme programming project starts with user stories.
- User stories are the short descriptions of the requirements.
- They are different from traditional requirements specification where each requirement is specified in greater detail.
- For example, the following figure shows user stories.

- Students can purchase monthly parking passes online.
- Parking passes can be paid via credit cards.
- Parking passes can be paid via PayPal.
- Professors can input student marks.
- Students can obtain their current seminar schedule.
- Students can order official transcripts.
- Students can only enroll in seminars for which they have prerequisites.
- Transcripts will be available online via a standard browser.

- The development team estimates how long it will take to implement a user story. Generally, implementation of a user story takes one to four weeks.
- Using these estimates and the stories, release planning is done which defines which stories are to be built in which system release, and the dates of these releases.
- Usually, small releases are encouraged and each release is done in some iterations.
- Acceptance tests are performed to test the software before the release.
- The overall process of extreme programming is shown below.



- As shown in the above figure, development is done in iterations by considering the user stories one by one.
- An iteration starts with iteration planning in which the stories to be implemented in this iteration are selected.
- At a time, only one user story is planned, developed and tested using acceptance testing.
- An iteration may not add significant functionality but, still a new release is made at the end of each iteration.
- After some iterations, the entire software is released.

Advantages

- Customer satisfaction by rapid, continuous delivery of useful software.
- Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Facilitates face-to-face conversation.
- Provides close and daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed.

Disadvantages

- Can be misinterpreted...
- Difficult to get external review.
- When project is complete, and team disperses, maintenance becomes difficult.

Where to use Agile Process Model

- XP, and other agile methods, are suitable for situations where the volume and pace of requirements change is high, and where requirement risks are considerable.
- It works well when the customer is willing to involve heavily during the entire development, working as a team member.

UNIT-II
Assignment-Cum-Tutorial Questions
SECTION-A

Objective Questions

1) Which of the following is a characteristic of Agile development? [ ]

a) Shared code ownership

b) Implement the simplest solution to meet today's problem

c) Continual feedback from customer

d) test-driven development

e) All of the above

2) In waterfall model, output of one phase is input to next phase. [ ]

a) True b) False

3) If requirements are easily understandable and defined then which model is best suited? [ ]

a) Waterfall Model b) Iterative Development Model

c) Prototyping d) Extreme Programming

4) Which of the following are advantages of iterative model? [ ]

a) Early revenue generation

b) Simpler to manage

c) Divided workload

d) Early feedback

e) All the above

5) Which phase of the RUP is used to establish a business case for the system [ ]

a) Transition b) Elaboration

c) Construction d) Inception

6) In XP Increments are delivered to customers every _____ weeks. [ ]

a) One b) Two c) Three d) Four

7) In a college, students are asked to develop a software. Which model would be Preferable? [ ]

a) Waterfall model b) Spiral model

c) Agile model d) Code and fix model

8) Which of the following life cycle model can be chosen if the development team has less experience on similar projects? [ ]

a) Spiral b) Waterfall

c) Prototyping d) Iterative Enhancement Model

9) Which four framework activities are found in the Extreme Programming(XP)?

a) analysis, design, coding, testing [ ]

b) planning, analysis, design, coding

c) planning, design, coding, testing

d) planning, analysis, coding, testing

10) An iterative process of system development in which requirements are converted to a working system that is continually revised through close work between an analyst and user is called_____ [ ]

a) Waterfall modeling

b) Iterative modeling

c) Spiral modeling

d) None of these above

11) A company is developing an advance version of their current software available in the market, what model approach would they prefer ? [ ]

a) waterfall

b) Prototyping

c) Iterative Enhancement

d) Spiral

12) Which one of the following statements most accurately identifies the stakeholders in a software development project? [ ]

a) A stakeholder of the organization developing the software

b) Anyone who is interested in the software

c) Anyone who is a source of requirements for the software

d) Anyone who might be affected by the software

SECTION-B

SUBJECTIVE QUESTIONS

1) What is a process model? How do process models differ from one another?
2) What is the oldest paradigm for software engineering? Why does the waterfall model sometimes fail?
3) Write about the Rational unified process model in detail.
4) Compare the waterfall model with the Unified process model.
5) Explain about agile methodology & extreme programming as software development process models.
6) Describe prototyping model in detail. Discuss how to select a particular process model based on characteristics of a project.
7) Categorize the strengths and weaknesses of waterfall, iterative development and prototyping.
8) Analyze why does iterative process makes it easier to manage change.
9) Is it possible to combine the process models? If so explain with an example.
10) Which process model is suitable for medium scale projects, justify it.

SECTION-C

GATE QUESTIONS

1) Match the following: [ ]

1. Waterfall Model	a) Specifications can be developed incrementally
2. Evolutionary Model	b) Requirements compromises are inevitable
3. Component-based Software Engineering	c) Explicit recognition of risk
4. Spiral Development	d) Inflexible partitioning of the project into stages

(a) 1-a, 2-b, 3-c, 4-d **(b)** 1-d, 2-a, 3-b, 4-c
(c) 1-d, 2-b, 3-a, 4-c **(d)** 1-c, 2-a, 3-b, 4-d **(Gate CS 2015)**

2) Which one of the following is TRUE? [ ]

(a) The requirements document also describes how the requirements that are listed in the document are implemented efficiently.

(b) Consistency and completeness of functional requirements are always achieved in practice.

(c) Prototyping is a method of requirements validation.

(d) Requirements review is carried out to find the errors in system design **(GATE CS 2014)**

3) What is the appropriate pairing of items in the two columns listing various activities encountered in a software life cycle? [ ]

P. Requirements Capture	1. Module Development and Integration
Q. Design	2. Domain Analysis
R. Implementation	3. Structural and Behavioral Modeling
S. Maintenance	4. Performance Tuning

a) P-3, Q-2, R-4, S-1 **(b)** P-2, Q-3, R-1, S-4

(c) P-3, Q-2, R-1, S-4 **(d)** P-2, Q-3, R-4, S-1 **(GATE CS 2010)**