UNIT-2

Software Processes

A software process is a set of related activities that leads to the production of software system. There are many different types of software systems, and there is no universal software engineering method that is applicable to all of them. Consequently, there is no universally applicable software process. The process used in different companies depends on the type of software being developed, the requirements of the software customer, and the skills of the people writing the software. However, although there are many software processes, they all must include, in some form, the four fundamental software engineering activities:

- → Software specification
- → Software development
- → Software validation
- → Software evolution.

these points already discussed in unit 1 in short.

⊛. Software Process Models:

A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective. It represents the order in which the activities of software development will be undertaken. The general process models are as follows:

1) Waterfall model: The waterfall model is a sequential approach, where each fundamental activity of a process represented as a seperate phase, arranged in linear order. Each phase is carried out completely before proceeding to the next. The process is strictly sequential - no backing up or repeating phases. This process is strictly documented and predefined with features expected to every phase of SDLC model.

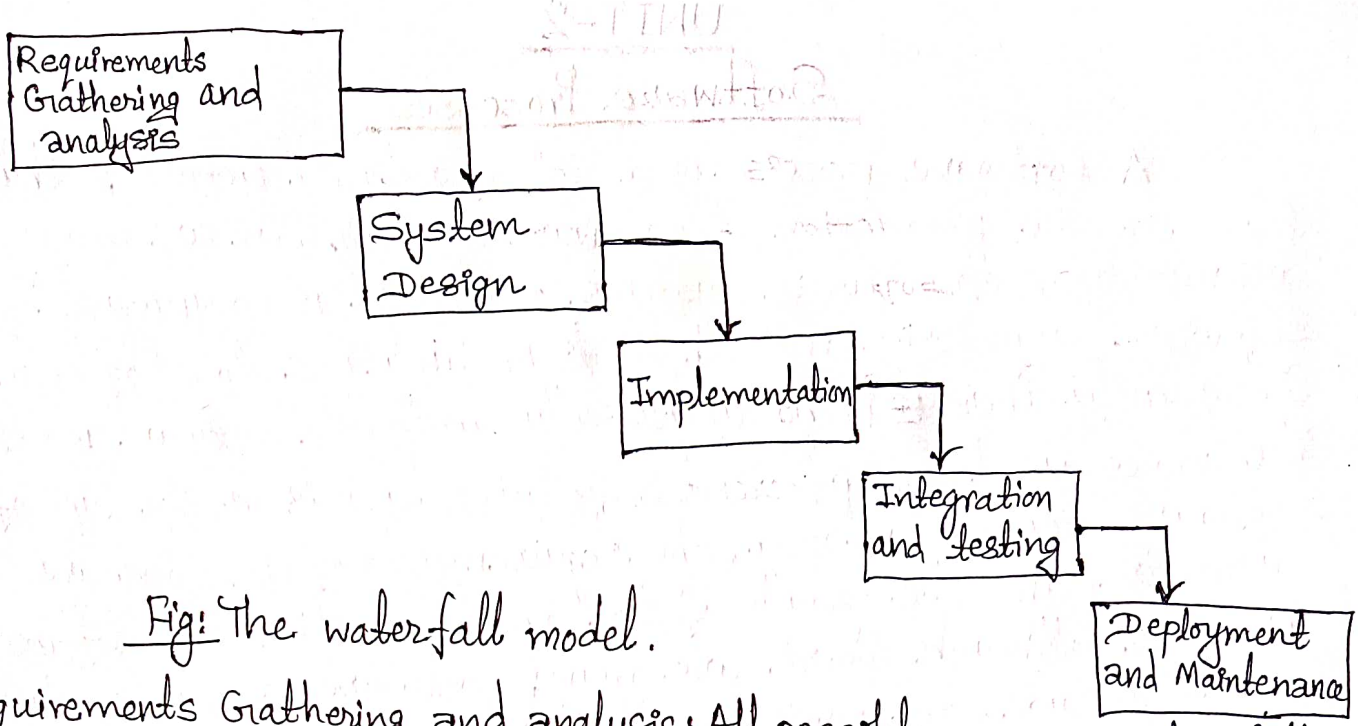

Fig: The waterfall model.

Requirements Gathering and analysis: All possible requirements of the system to be developed are captured in this phase. Then analysis is done and documented in a requirement specification document.

System Design: The requirement specifications from first phase are studied in this phase and design for system is prepared based on requirements gathered and analysed.

Implementation: Now designed system is developed by programmers in this phase. The system is first developed in small programs called units, which are integrated and tested in next phase.

Integration and Testing: All the units developed in the implementation phase are integrated into a system after testing of each unit.

Deployment and Maintenance: Once, the functional and non-functional testing is done; the product is deployed in the customer environment or released in the market. Maintenance is done to fix issues and to enhance the product to some better versions.

Merits of waterfall model:

↳ This model is simple to implement also the number of resources that are required for it is minimal.

↳ The requirements are simple and explicitly declared; they remain unchanged during the entire project development.

↳ The start and end points for each phase is fixed, which makes it easy to cover progress.

↳ Easy to manage due to rigidity of the model.

Demerits of waterfall model:

↳ No working software is produced until late during the life cycle.
↳ This model cannot accept changes in requirements during development.
↳ It becomes tough to go back to the phase.
↳ Since the testing is done at larger stage, risk reduction strategy is difficult to prepare.

2) Incremental development model:

In an incremental model, customers identify, in outline, the services to be provided by the system. They identify which of the services are most important and which are least important to them. A number of delivery increments are then defined, with each increment providing a sub-set of the system functionality. The allocation of services to increments depends on the service priority with the highest priority services delivered first.

Once increment is completed and delivered, customers can put it into services they can experiment with the system that helps to clarify their requirements for later increments and for later versions of the current increment. As new increments are completed, they are integrated with existing increments so that the system functionality improves each delivered increment.

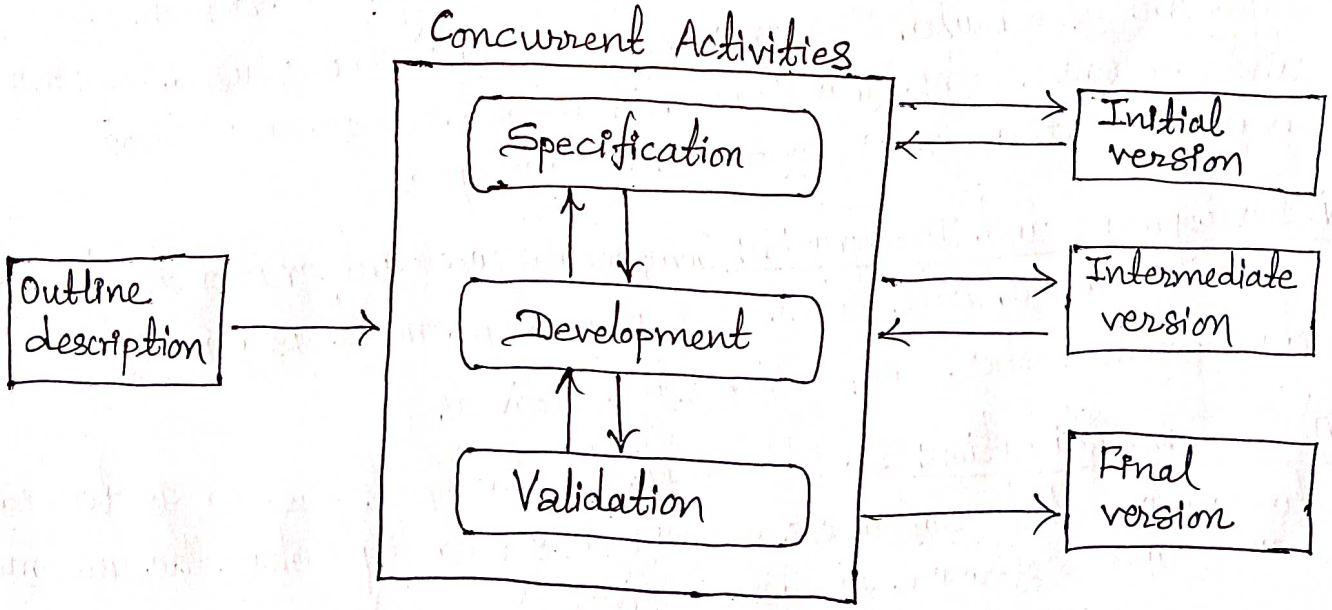

Fig: Incremental development model.

3) Integration and Configuration model:

OR Reuse-oriented development model.

In the majority of software projects, there is some software reuse. This often happens when people working on the project know of or search for code that is similar to what is required. They look for these, modify them as needed, and integrate them with the new code that they have developed.

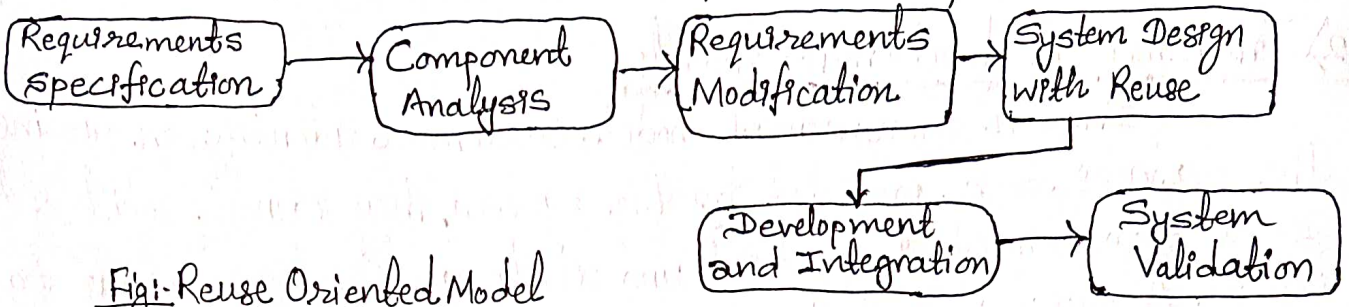

Fig:- Reuse Orienfed Model

Stages:

i) Requirement specification: All possible requirements of the system to be developed are captured in this phase.

ii) Component Analysis: Based on requirement specification, a search is made for components that can implement the given specification. Usually there is no exact match, and the components that may be used only provide some of the functionality required.

iii) Requirements Modification: Requirements are modified according to available components, to reflect the services of available components.

iv) System design with reuse: During this stage the design of system is build. Designer must consider the reused component and organize the framework. If reused component is not available then new software is developed.

v) Development and Integration: Components are integrated to develop new software. Integration in this ~~part is~~ model is part of development rather than seperate activity.

vi) System validation: In this step, the developed system is tested to ensure that software system does exactly what the customer wants and software is defect free.

⊛. Coping with change:

Change is sure to happen in all large software projects. The system requirements change as business respond to external pressures, competition, and changed management priorities. Change adds to the costs of software development because it usually means that work that has been completed has to be redone. Following are the two ways of coping with change and changing system requirements:

1) Prototyping:

In prototyping instead of spending a lot of time producing very detailed specifications, the developers find out only outline of system. The complete system is not developed at once. Instead, a quick prototype is created which contains some portions of the system, and the prototype is refined and extended iteratively until the final specifications. Prototyping approach is used when requirements are not clear or well-understood.

A software prototype can be used in a software development process to help anticipate changes that may be required: (expect or predict)

→ In the requirement engineering process, a prototype can help with the elicitation and validation of system requirements. (hint or suggestion)

→ In the system design process, a prototype can be used to explore software solutions and in the development of a user interface for the system.

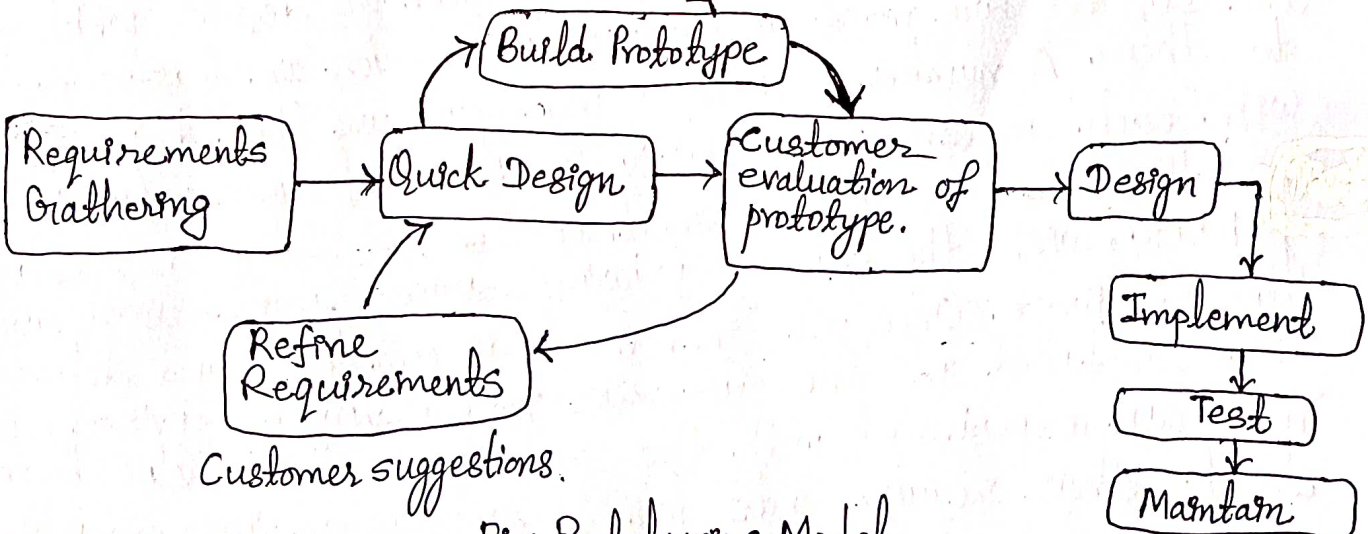

Fig: Prototyping Model

Q1. Discuss evolutionary prototyping and throw-away prototyping in the software process.

Solution:

Evolutionary prototyping: In this method, the prototype developed initially is incrementally refined on the basis of customer feedback till it finally gets accepted. It helps us to save time as well as effort. That's because developing a prototype from scratch for every interaction of the process can sometimes be very frustrating. This model is helpful for a project which uses a new technology that is not well understood. It is also used for complex project where every functionality must be checked once. It is helpful when the requirement is not stable or not understood clearly at the initial stage.

Throw-away prototyping: Throwaway is based on the preliminary requirement. It is quickly developed to show how the requirement will look visually. The customers feedback helps drives changes to the requirement, and the prototype is again created until the requirement is baselined. In this method, a developed prototype will be discarded and will not be a part of the ultimately accepted prototype. This technique is useful for exploring ideas and getting instant feedback for customer requirements.

2) Incremental Delivery: (2nd way of coping with change 1st way is prototyping.)

Incremental delivery is an approach to software development where some of the developed increments are delivered to the customer and deployed for use in their working environment. In an incremental delivery process, customers define which of the services are most important and which are least important to them. A number of delivery increments are then defined, with each increment providing a subset of system functionality. The allocation of services to increments ~~are t~~ depends on service priority, with the highest priority services implemented and delivered first. Once an increment is completed and delivered, it is installed in the customers normal working environment. They can experiment with the system, and this helps them clarify their requirements for later system increments.

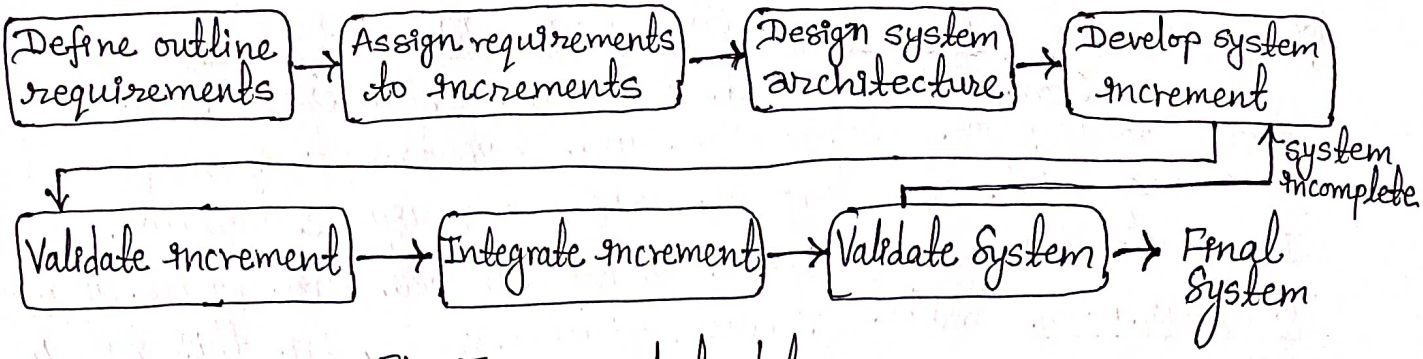

Fig: Incremental delivery

⊛. Process Improvement:

Process improvement means understanding existing processes and changing these processes to increase product quality and/or reduce cost and development time. Two quite different approaches to process improvement and change are used:

i) Process maturity approach: It has focused on improving process and project management and introducing good software engineering practice into an organization. The level of process maturity reflects the extent to which good technical and management practice has been adopted in organizational software development process.

ii) Agile approach: It has focused on iterative development and the reduction of overheads in the software process. The primary characteristic of agile methods are rapid delivery of functionality and responsiveness to changing customer requirements.

⊛. Differentiate between V-shape model and spiral model:

V-Shape model	Spiral model.
i) In V-model testing activities start with the first stage.	i) In spiral model testing is done at the end of engineering phase.
ii) Guarantee of success through V-model is high.	ii) Guarantee of success through Spiral model is low.
iii) It is not iterative	iii) It is iterative.
iv) Cost of V-model is less expensive than spiral model.	iv) Cost of Spiral model is very expensive.
v) In this model development and testing are not concurrent.	v) In this model development and testing are concurrent.

⊛. Component Based Software Engineering (CBSE):

Component-based software engineering is a procedure that focuses on the design and development of computer based systems with the help of existing reusable software components. It is also known as reuse oriented model. Integration and Configuration model is reuse oriented model. In this rather than developing system from scratch, we search for existing reusable components and design the desired system in terms of those components.

Advantages:

→ Provides reusability of components.
→ Software development risk is reduced.
→ Software development process becomes easier and faster.
→ Reduces cost of development of system.
→ Easier to add functionalities to the system.