> ***MSS Module 1 Important Topics***

> ***Table of contents***

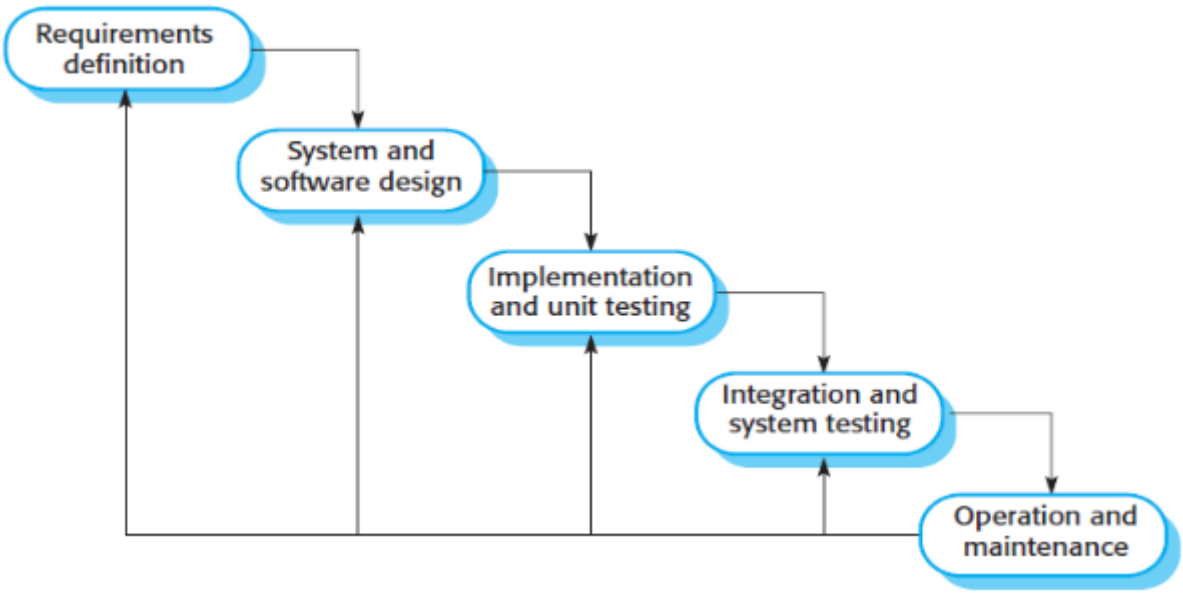
Software process models

1. Waterfall Model

This takes the fundamental process activities of

- specification
- development
- validation
- evolution
  and represents them as separate process phases such as
- requirements specification
- software design,
- implementation
- testing

Stages of Waterfall Model



Requirements analysis and definition

- We figure out what the system should do, what limits it has, and what it aims to achieve by talking to the people who will use it.
- After that, we describe these things in a lot of detail, and that becomes the official plan for how the system should work.

System and software design

- When we design a system, we decide which parts should be done by the physical equipment (hardware) and which parts should be handled by the computer programs (software).

- We create a basic plan for how the entire system should be organized, called the system architecture.
- For the software part, we figure out the main elements of the computer programs and how they relate to each other.

Implementation and unit testing

- During this stage, the software design is realized as a set of programs or program units.
- Unit testing involves verifying that each unit meets its specification

Integration and system testing

- The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met.
- After testing, the software system is delivered to the customer

Operation and maintenance

- Normally, this is the longest life-cycle phase.
- The system is installed and put into practical use. Maintenance involves correcting errors that were not discovered in earlier stages of the life cycle, improving the implementation of system units, and enhancing the system’s services as new requirements are discovered.

Trick to remember waterfall model

- RSSIO
- "Remember, Start Implementing In Order."
- Each word in this phrase corresponds to one stage of the waterfall model:

1. "Remember" for Requirements analysis and definition
2. "Start" for System and software design
3. "Implementing" for Implementation and unit testing
4. "In" for Integration and system testing
5. "Order" for Operation and maintenance

Advantages of Waterfall Model

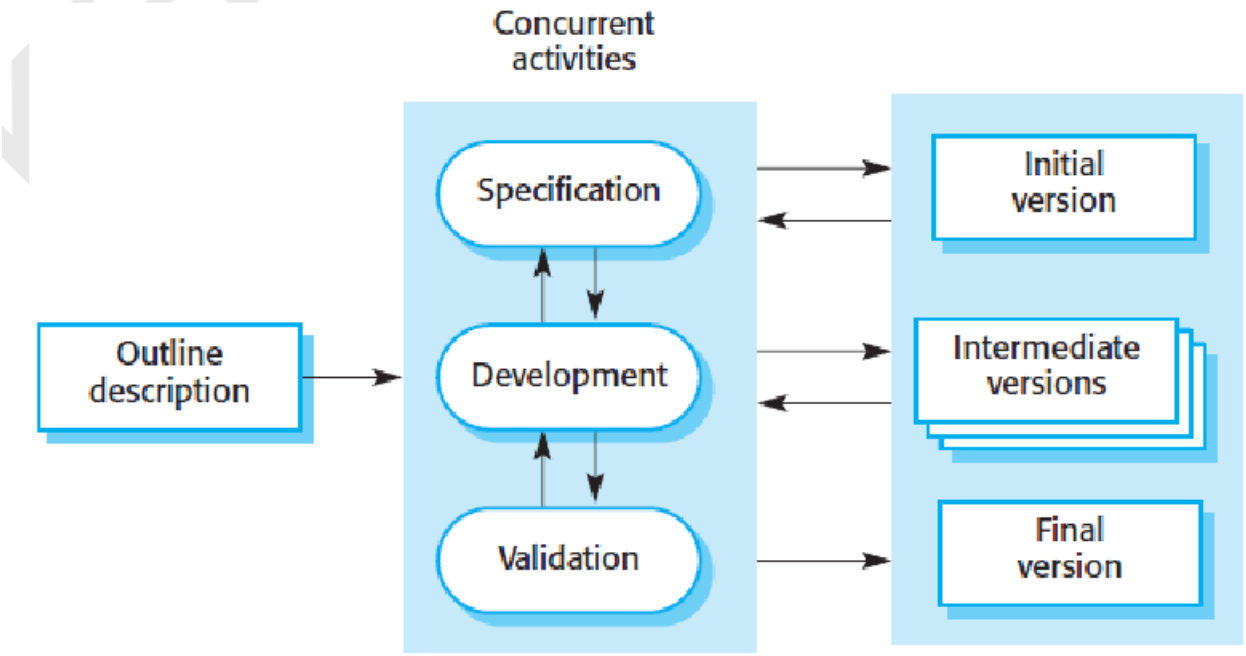
- Used in Large software systems
- Used in Critical systems where there is a need for extensive safety and security analysis of the software specification and design.
- Embedded systems where the software has to interface with hardware systems.

Disadvantages of Waterfall Model

- The waterfall model is not the right process model in situations where informal team communication is possible and software requirements change quickly.
- terative development and agile methods are better for these systems.

2. Incremental Development Model

- Incremental development is based on the idea of developing an initial implementation, getting feedback from users and others, and evolving the software through several versions until the required system has been developed.
- Most common approach for the development of application systems and software products
- Each increment or version of the system incorporates some of the functionality that is needed by the customer
  - Generally, the early increments of the system include the most important or most urgently required functionality



Advantages

- The cost of implementing requirements changes is reduced
- It is easier to get customer feedback on the development work that has been done.
- Early delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included.

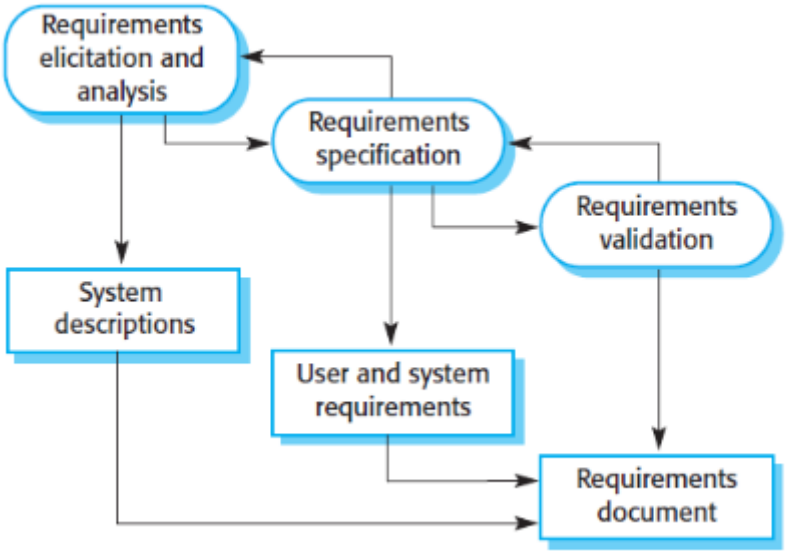
Disadvantages

- If systems are developed quickly, it is not cost effective to produce documents that reflect every version of the system
- System structure tends to degrade as new increments are added.
    - Regular change leads to messy code as new functionality is added in whatever way is possible

---

Software specification

- Software specification or requirements engineering is the process of understanding and defining what services are required from the system.
  There are 3 main activities



Requirements elicitation and analysis

- This is the process of deriving the system requirements through observation of existing systems, discussions with potential users and procurers, task analysis, and so on.

Requirements specification

- Requirements specification is the activity of translating the information gathered during requirements analysis into a document that defines a set of requirement
- There are 2 types of requirements under this
    - User requirements
        - abstract statements of the system requirements for the customer and end-user of the system

- System requirements
  - more detailed description of the functionality to be provided

Requirements validation

- This activity checks the requirements for realism, consistency, and completeness
- During this process, errors in the requirements document are inevitably discovered. It must then be modified to correct these problems

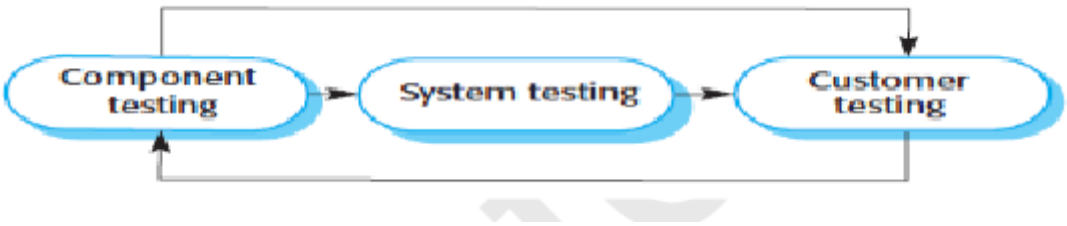
Trick to Remember

- E -> Elicitation
- S -> Specification
- V -> Validation
- ESV
- "Evaluating Software Visions."

Software validation

- Software Validation intended to show that a system both conforms to its specification and meets the expectations of the system customer.
- Program testing, where the system is executed using simulated test data, is the principal validation technique
- Validation may also involve checking processes, such as inspections and reviews, at each stage of the software process from user requirements definition to program development

Stages in testing process



- Component testing
  - The components making up the system are tested by the people developing the system.

    - Each component is tested independently, without other system components. Components may be simple entities such as functions or object classes
- System testing
    - System components are integrated to create a complete system.
    - This process is concerned with finding errors that result from unanticipated interactions between components
- Customer testing
    - This is the final stage in the testing process before the system is accepted for operational use. The system is tested by the system customer (or potential customer) rather than with simulated test data

---

*Alpha testing and beta testing

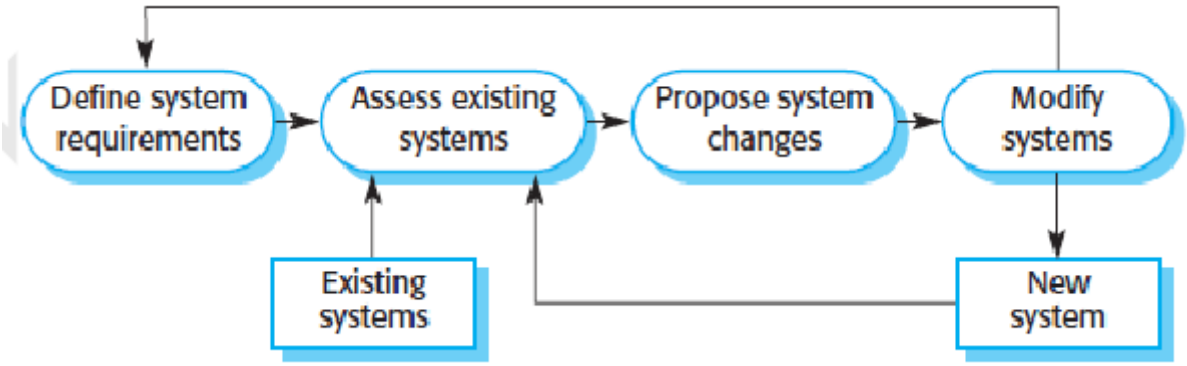
Difference between Alpha and Beta testing

ALPHA TESTING	BETA TESTING
It is performed by the developers	It is performed by the customers
It is performed before the software is released to the end user	It is performed after the software is released to the customer
The developer keeps track of all the errors(if any)	The customer keeps track of all the errors. In case of any error it is reported to the developer.
It involves both Blackbox testing and Whitebox testing	It involves only Blackbox testing
It is performed in virtual environment	It is performed in real environment

00:04

Software evolution

- The flexibility of software is one of the main reasons why more and more software is being incorporated into large, complex systems.
- Once a decision has been made to manufacture hardware, it is very expensive to make changes to the hardware design.

- Changes can be made to software at any time during or after the system development. Even extensive changes are still much cheaper than corresponding changes to system hardware
- 

Coping with change

- Change is inevitable in all large software projects.
- The system requirements change as businesses respond to external pressures, competition, and changed management priorities. As new technologies become available, new approaches to design and implementation become possible.
- Change adds to the costs of software development because it usually means that work that has been completed has to be redone, This is called rework

2 ways to reduce costs of rework

- Change anticipation
  - Where the software process includes activities that can anticipate or predict possible changes before significant rework is required.
- Change tolerance
  - Where the process and software are designed so that changes can be easily made to the system. This normally involves some form of incremental development

Two ways of coping with change

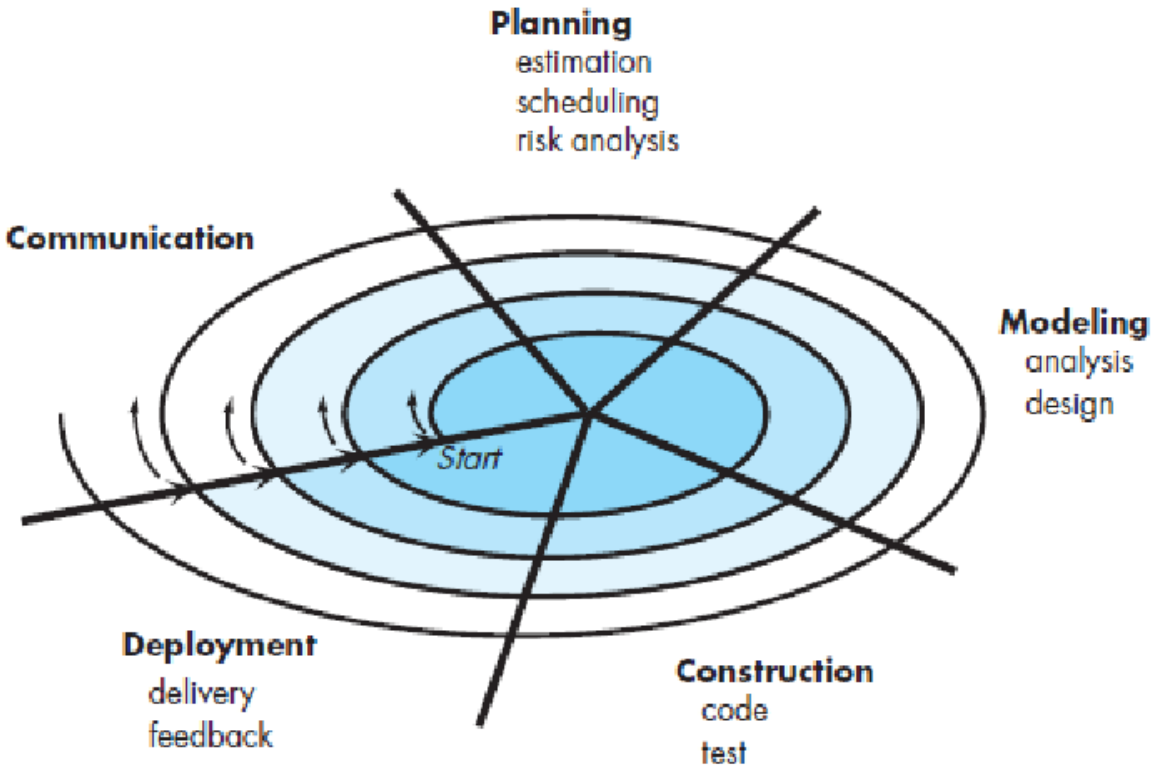
- System prototyping,
  - Version of the system or part of the system is developed quickly to check the customer's requirements and the feasibility of design decisions.
  - allows users to experiment with the system before delivery and so refine their requirements.
- Incremental delivery

- system increments are delivered to the customer for comment and experimentation. This supports both change avoidance and change tolerance

---

Boehm's Spiral Model

- The spiral model is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model



- Each cycle in spiral is divided into 4 parts
  - Objective setting
    - Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists
  - Risk Assessment and reduction
    - The next phase in the cycle is to calculate these various alternatives based on the goals and constraints
  - Development and validation:
    - The next phase is to develop strategies that resolve uncertainties and risks.

        - This process may include activities such as benchmarking, simulation, and prototyping.
    - Planning
        - Finally, the next step is planned
        - The project is reviewed, and a choice made whether to continue with a further period of the spiral.

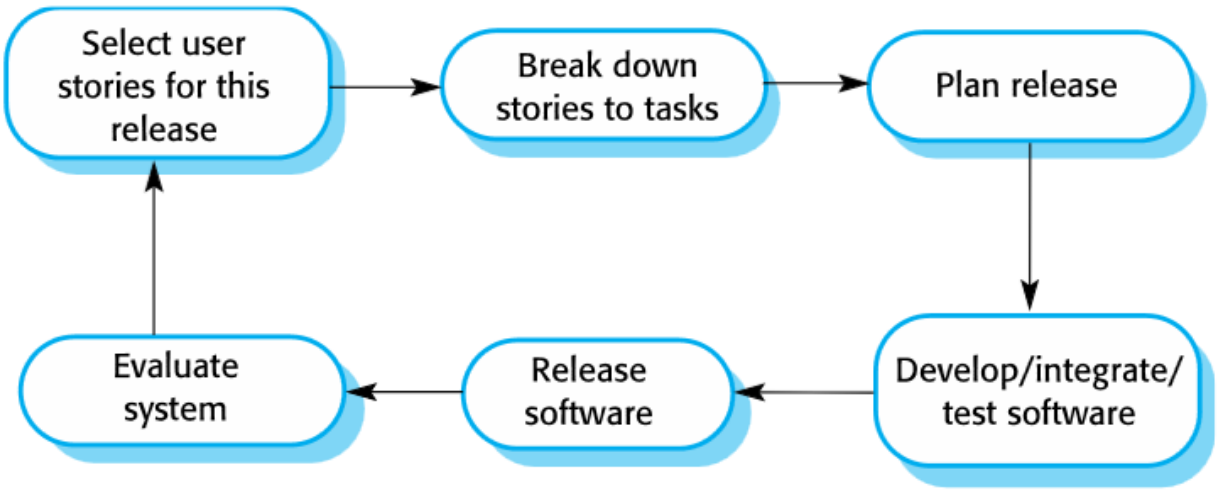
Agile software development

- Agile methods are incremental development methods in which the increments are small, and, typically, new releases of the system are created and made available to customers every two or three weeks.
- They involve customers in the development process to get rapid feedback on changing requirements.
- They minimize documentation by using informal communications rather than formal meetings with written documents.

Principles of Agile Software development

- Customer involvement
    - Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system
- Incremental delivery
    - The software is developed in increments with the customer specifying the requirements to be included in each increment
- People not process
    - The skills of the development team should be recognized and exploited.
    - Team members should be left to develop their own ways of working without prescriptive processes
- Embrace change
    - Expect the system requirements to change and so design the system to accommodate these changes.
- Maintain simplicity
    - Focus on simplicity in both the software being developed and in the development process.
    - Wherever possible, actively work to eliminate complexity from the system

Extreme Programming



XP release cycle

- A very influential agile method, developed in the late 1990s, that introduced a range of agile development techniques
- Extreme Programming (XP) takes an ‘extreme’ approach to iterative development.
  - New versions may be built several times per day
  - Increments are delivered to customers every 2 weeks;
  - All tests must be run for every build and the build is only accepted if tests run successfully