Unit 2. Software Process Model

2.1 Waterfall Model
2.2 Prototype Model
2.3 Incremental Model
2.4 Spiral Model

1) **Software Process:** The process that deals with the technical and management issues of s/w development is called a s/w process.
2) **What is Software Model?:** It is the blueprint for software development which provides informative and technical guidelines to develop the software in an efficient manner.

3) **Software Development Process Model:** In the software engineering there are many models for the development of the software. Following is the list of main four models for software development.

   1) **Waterfall Model/Linear Sequential Model**
   2) **Prototype Model**
   3) **Incremental Model**
   4) **Spiral Model**

   Each of the above models is discussed in the following section:

WATERFALL MODEL/THE LINEAR SEQUENTIAL MODEL

Sometimes called the **classic life cycle** or **the waterfall model**, the linear sequential model suggests a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing, and support.

Although the original waterfall model proposed by Winston Royce.

Modeled after a conventional engineering cycle, the linear sequential model encompasses the following activities:

System/information engineering and modeling: Because software is always part of a larger system, work begins by establishing requirements for all system elements and then allocating some subset of these requirements to software.

This system view is essential when software must interact with other elements such as hardware, people, and databases. System engineering and analysis encompass requirements gathering at the system level with a small amount of top level

Information engineering encompasses requirements gathering at the strategic business level and at the business area level.

Software requirements analysis: The requirements gathering process is intensified and focused specifically on software. To understand the nature of the program(s) to be built, the software engineer ("analyst") must understand the information domain for the software, as well as **required function, behavior, performance, and interface**. Requirements for both the system and the software are documented and reviewed with the customer.

Design: Software design is actually a multistep process that focuses on four distinct attributes of a program: **data structure, software architecture, interface representations, and procedural (algorithmic) detail**. The design process translates requirements into a representation of the software that can be assessed for quality before coding begins. Like requirements, the design is documented and becomes part of the software configuration.

Code generation: The design must be translated into a machine-readable form. The code generation step performs this task. If design is performed in a detailed manner, code generation can be accomplished mechanistically.

Testing: Once code has been generated, program testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals; that is, conducting tests to uncover errors and ensure that defined input will produce actual results that agree with required results.

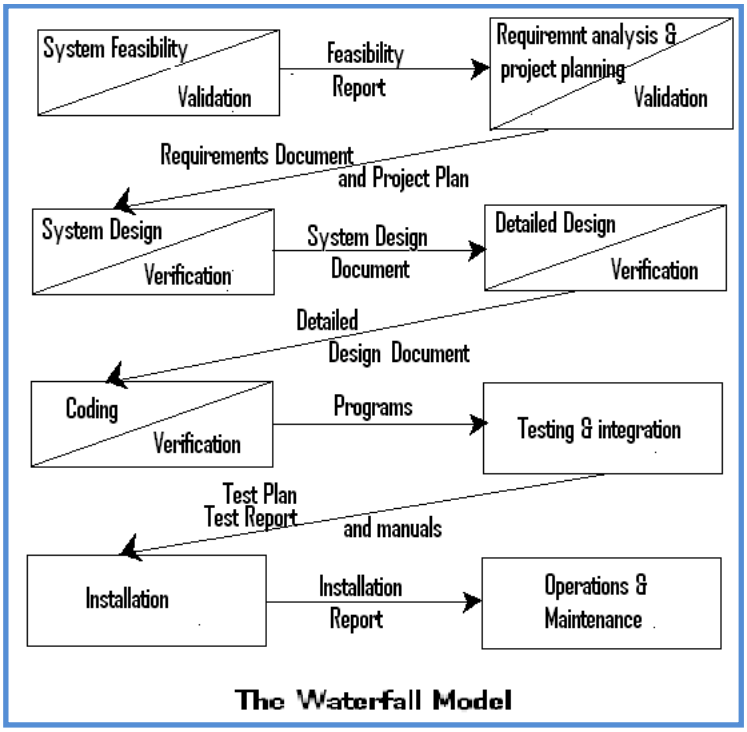
Support: Software will undoubtedly undergo change after it is delivered to the customer. Change will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment, or because the customer requires functional or performance enhancements. Software support/maintenance reapplies each of the preceding phases to an existing program rather than a new one.

Type of Project in which this Model is used: This Model is suited for well understood problems, short duration project, and automation of existing manual system.

Advantages: It is simple, Easy to execute and Intuitive and logical.

Limitations

1 Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly.
2 It is often difficult for the customer to state all requirements explicitly. The linear sequential model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.
3 The customer must have patience. A working version of the program(s) will not be available until late in the project time-span.
4 It is a document driven process that requires formal documents at the end of each phase.
5 It follows the “big-bang” approach. The entire s/w is delivered at the end and therefore entails heavy risk, as the user does not know until the very end what they are getting.
6 It assumes the requirements of a system can be frozen before the design begins. This is possible for systems designed to automate an existing systems. But for new systems determining the requirement is difficult.
7 Freezing the requirements usually requires choosing the hardware.



The Waterfall Model

THE PROTOTYPING MODEL

Often, a customer defines a set of general objectives for software but **does not identify detailed input, processing, or output requirements.** In other cases, the developer may be **unsure of the efficiency of an algorithm, the adaptability of an operating system, or** the form that **human/machine interaction** should take. In such situations, a prototyping paradigm may offer the best approach.

The prototyping paradigm begins with requirements gathering. Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory. A "quick design" then occurs. The quick design focuses on a representation of those aspects of the software that will be visible to the customer/user (e.g.,input approaches and output formats). The quick design leads to the construction of prototype. The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed. Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time **enabling the developer to better understand what needs to be done.**

Ideally, **the prototype serves as a mechanism for identifying software requirements**.

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. This prototype is based on the currently specified requirements.

In most projects, the first system built is barely usable (throwaway). It may be too slow, too big, and awkward in use or all three. There is no alternative but to start again and build a redesigned version in which these problems are. The prototype can serve as "the first system" known as throwaway prototype.

The development process using throwaway prototyping typically starts when the preliminary requirement specification document has been developed where a reasonable understanding of the system and its needs and which needs are clear or likely to change. After the prototype has been developed, the end users and clients are given an opportunity to use the prototype, based on their experience feedback is provided to the developer and if any changes are needed, prototype is modified and clients are again allowed to use the system until the client is satisfied.

Type of Project in which this Model is used: Systems with novice (beginner) users, When uncertainities in requirements, When UI very important.

Advantages: Helps in requirements elicitation, Reduce risk, Leads to a better system, 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.

Limitations of prototype model:

1. The customer sees what appears to be a working version of the software. When informed that the product must be rebuilt so that high levels of quality can be maintained, the customer cries foul and demands that "a few fixes" be applied to make the prototype a working product.

2. The developer often makes implementation compromises in order to get a prototype working quickly. An inappropriate operating system or programming language may be used simply because it is available and known; an inefficient algorithm may be implemented simply to demonstrate capability.

Although problems can occur, prototyping can be an effective paradigm for software engineering. **The customer and developer must both agree that the prototype is built to serve as a mechanism for defining requirements. It is then discarded and the actual software is engineered with an eye toward quality and maintainability.**

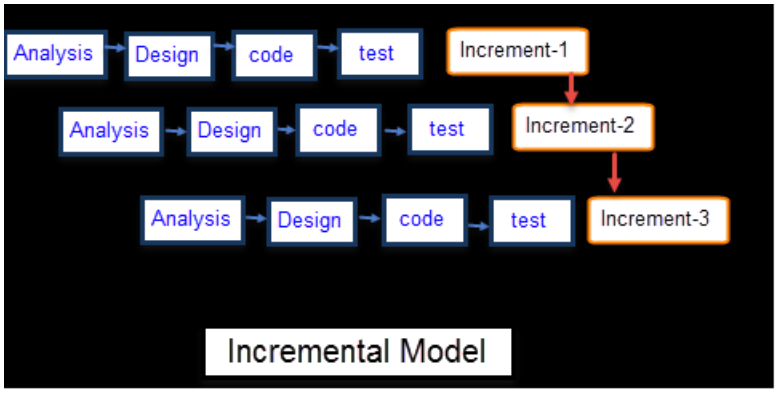
Software Development Life Cycle (SDLC)

1) Requirement Gathering
2) Requirement Analysis
3) Design/Plan Solution
4) Develop Solution
5) System Integration and Testing
6) Implementation and Customer Acceptance
7) Support and Maintenance

INCREMENTAL MODEL

Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance.

Each iteration passes through the **requirements, design, coding and testing phases**. And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented.



The system is put into production when the first increment is delivered. The first increment is often a core product where the basic requirements are addressed, and supplementary features are added in the next increments. Once the core product is analyzed by the client, there is plan development for the next increment.

Characteristics of an Incremental module includes

- System development is broken down into many mini development projects
- Partial systems are successively built to produce a final total system
- Highest priority requirement is tackled first
- Once the requirement is developed, requirement for that increment are frozen

Incremental Phases	**Activities performed in incremental phases**
Requirement Analysis	• Requirement and specification of the software are collected
Design	• Some high-end function are designed during this stage
Code	• Coding of software is done during this stage
Test	• Once the system is deployed, it goes through the testing phase

When to use Incremental models?

- Requirements of the system are clearly understood
- When demand for an early release of a product arises
- When software engineering team are not very well skilled or trained
- When high-risk features and goals are involved
- Such methodology is more in use for web application and product based companies

Advantages and Disadvantages of Incremental Model

Advantages	Disadvantages
• The software will be generated quickly during the software life cycle	• It requires a good planning designing
• It is flexible and less expensive to change requirements and scope	• Problems might cause due to system architecture as such not all requirements collected up front for the entire software lifecycle
• Throughout the development stages changes can be done	• Each iteration phase is rigid and does not overlap each other
• This model is less costly compared to others	• Rectifying a problem in one unit requires correction in all the units and consumes a lot of time
• A customer can respond to each building	
• Errors are easy to be identified	

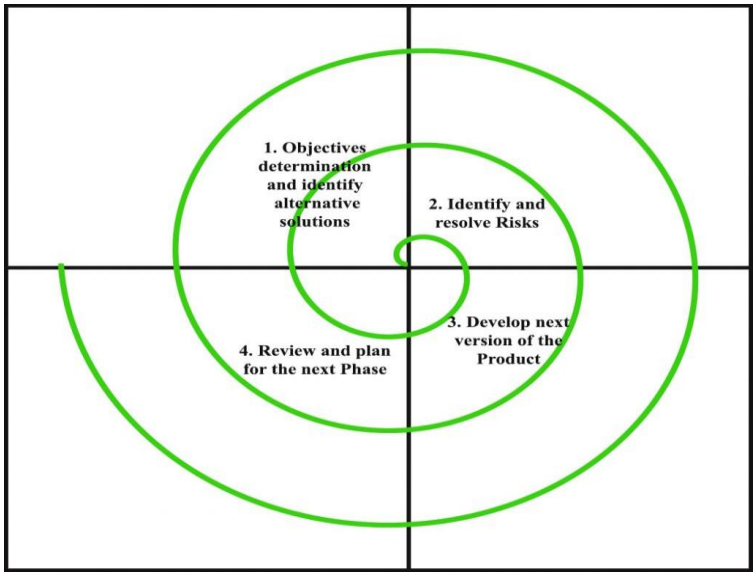
SPIRAL MODEL

Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using the spiral model.

The Radius of the spiral at any point represents the expenses (cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

Each phase of spiral model in software engineering begins with a design goal and ends with the client reviewing the progress. The spiral model in software engineering was first mentioned by Barry Boehm in his 1986 paper.

The development process in Spiral model in SDLC, starts with a small set of requirement and goes through each development phase for those set of requirements. The software engineering team adds functionality for the additional requirement in every-increasing spirals until the application is ready for the production phase. The below figure very well explain Spiral Model: ***The below diagram shows the different phases of the Spiral Model: –***



Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

2. **Identify and resolve Risks:** During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

3. **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

When to use Spiral Model?

- A Spiral model in software engineering is used when project is large
- When releases are required to be frequent, spiral methodology is used
- When creation of a prototype is applicable
- When risk and costs evaluation is important
- Spiral methodology is useful for medium to high-risk projects
- When requirements are unclear and complex, Spiral model in SDLC is useful
- When changes may require at any time
- When long term project commitment is not feasible due to changes in economic priorities

Advantages of Spiral Model: Below are some advantages of the Spiral Model.

1. **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
2. **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
3. **Flexibility in Requirements:** Change requests in the Requirements at later phase can be incorporated accurately by using this model.
4. **Customer Satisfaction:** Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

Disadvantages of Spiral Model: Below are some main disadvantages of the spiral model.

1. **Complex:** The Spiral Model is much more complex than other SDLC models.
2. **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
3. **Too much dependability on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.
4. **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.

When to use waterfall model and prototype model? (2 marks)

Ans: When the requirement fixed and project is of small duration then, waterfall model model

is usefull. When the requirement is not fixed and project is of long duration then, prototype model model is usefull.

Write short note on Waterfall model.

Write short note on prototype model.

Differentiate Waterfall model and prototype model.

Differential Spiral Model and Incremental Model.