Choice of Process Models

A Software product development process usually starts when a request for the product is received from the customer.

- Starting from the inception stage:
    - A product undergoes a series of transformations through a few identifiable stages
    - Until it is fully developed and released to the customer.
- After release:
    - The product is used by the customer and during this time the product needs to be maintained for fixing bugs and enhancing functionalities. This stage is called Maintenance stage.
- This set of identifiable stages through which a product transits from inception to retirement form the life cycle of the product.
- Life cycle model (also called a process model) is a graphical or textual representation of its life cycle.

Choice of Process Models

The no. of inter related activities to create a final product can be organized in different ways and we can call these Process Models.

A software process model is a simplified representation of a software process. Each model represents a process from a specific perspective. These generic models are abstractions of the process that can be used to explain different approaches to the software development.

Any software process must include the following four activities:

1. **Software specification** (or requirements engineering): Define the main functionalities of the software and the constrains around them.
2. S**oftware design and implementation**: The software is to be designed and programmed.
3. **Software verification and validation**: The software must conform to its specification and meets the customer needs.

4. **Software evolution** (software maintenance): The software is being modified to meet customer and market requirements changes.

The various models are

Water Fall Model

Spiral Model

Prototype Model

Incremental Delivery

i) Water fall model

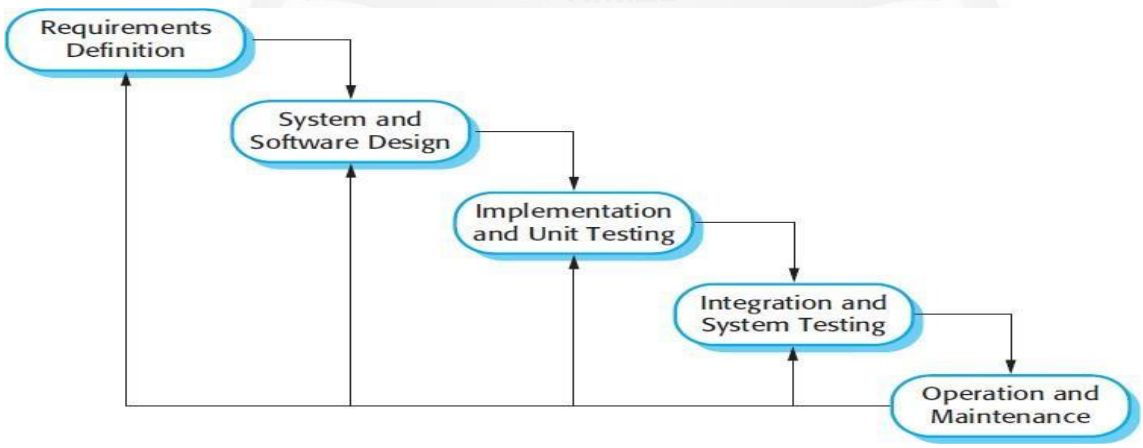
The waterfall model is a sequential approach, where each fundamental activity of a process represented as a separate phase, arranged in linear order.

In the waterfall model, you must plan and schedule all of the activities before starting working on them (plan-driven process).

Plan-driven process is a process where all the activities are planned first, and the progress is measured against the plan. While the agile process, planning is incremental and it's easier to change the process to reflect requirement changes.

The phases of the waterfall model are:

- Requirements
- Design
- Implementation
- Testing
- Maintenance



In principle, the result of each phase is one or more documents that should be approved and the next phase shouldn't be started until the previous phase has completely been finished. In practice, however, these phases overlap and feed information to each other. For example, during design, problems with requirements can be identified, and during coding, some of the design problems can be found, etc.

The software process therefore is not a simple linear but involves feedback from one phase to another. So, documents produced in each phase may then have to be modified to reflect the changes made.

ii) Spiral Model

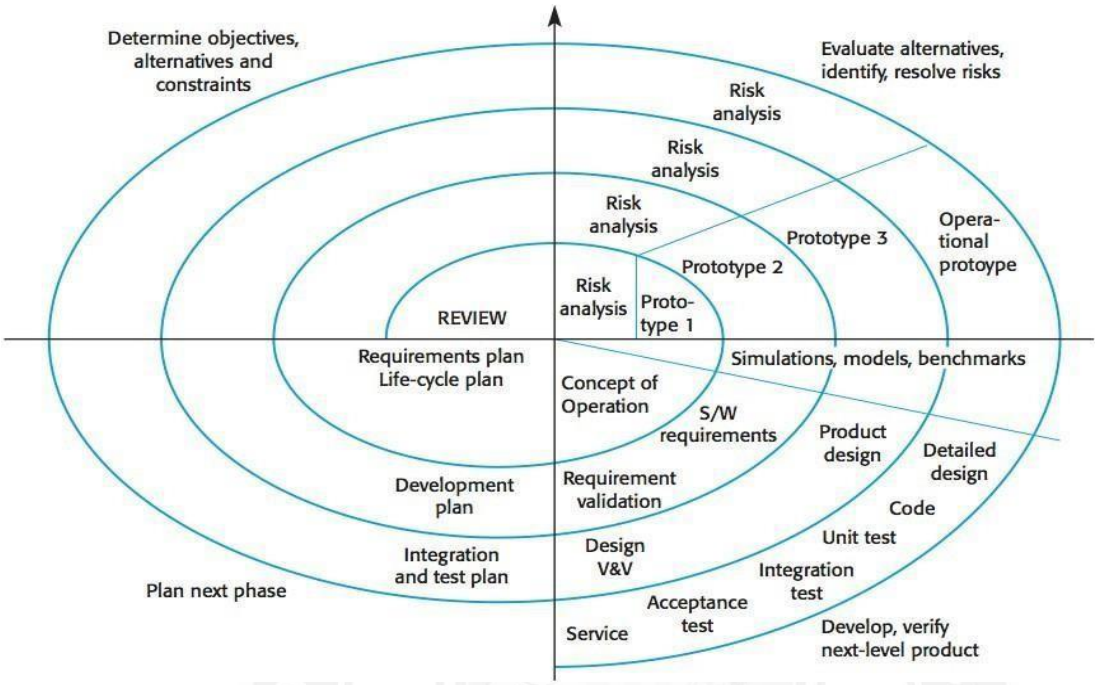
The spiral model is similar to the incremental model, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements is gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. It is one of the software development models like Waterfall, Agile, V-Model.

Planning Phase: Requirements are gathered during the planning phase. Requirements like 'BRS' that is 'Business Requirement Specifications' and 'SRS' that is 'System Requirement specifications'.

Risk Analysis: In the **risk analysis phase**, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis, then alternate solutions are suggested and implemented.

Engineering Phase: In this phase software is **developed**, along with **testing** at the end of the phase. Hence in this phase the development and testing is done.

Evaluation phase: This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.



Advantages of Spiral model:

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the software life cycle.

Disadvantages of Spiral model:

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

iii) Software Prototyping

A prototype is a version of a system or part of the system that's developed quickly to check the customer's requirements or feasibility of some design decisions. So, a

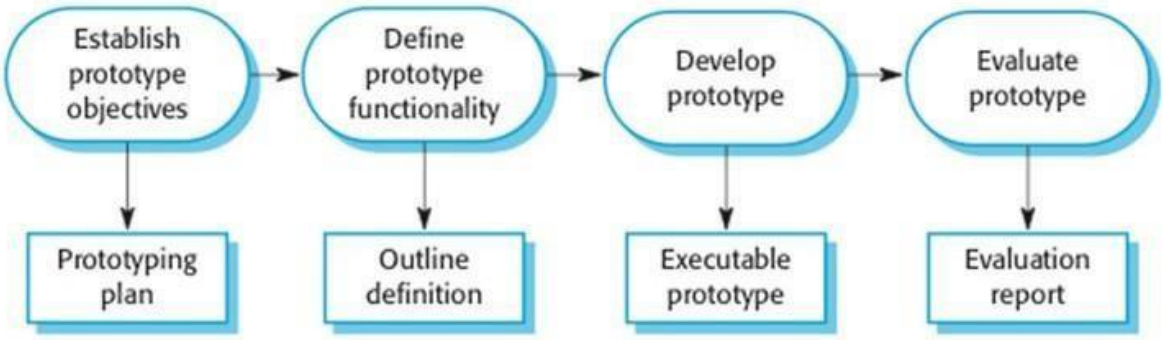
prototype is useful when a customer or developer is not sure of the requirements, or of algorithms, efficiency, business rules, response time, etc.

In prototyping, the client is involved throughout the development process, which increases the likelihood of client acceptance of the final implementation. While some prototypes are developed with the expectation that they will be discarded, it is possible in some cases to evolve from prototype to working system.

A software prototype can be used:

[1] In the **requirements engineering**, a prototype can help with the elicitation and validation of system requirements. It allows the users to experiment with the system, and so, refine the requirements. They may get new ideas for requirements, and find areas of strength and weakness in the software. Furthermore, as the prototype is developed, it may reveal errors and in the requirements. The specification maybe then modified to reflect the changes.

[2] In the **system design**, a prototype can help to carry out deign experiments to check the feasibility of a proposed design. For example, a database design may be prototype-d and tested to check it supports efficient data access for the most common user queries.



The phases of a prototype are:

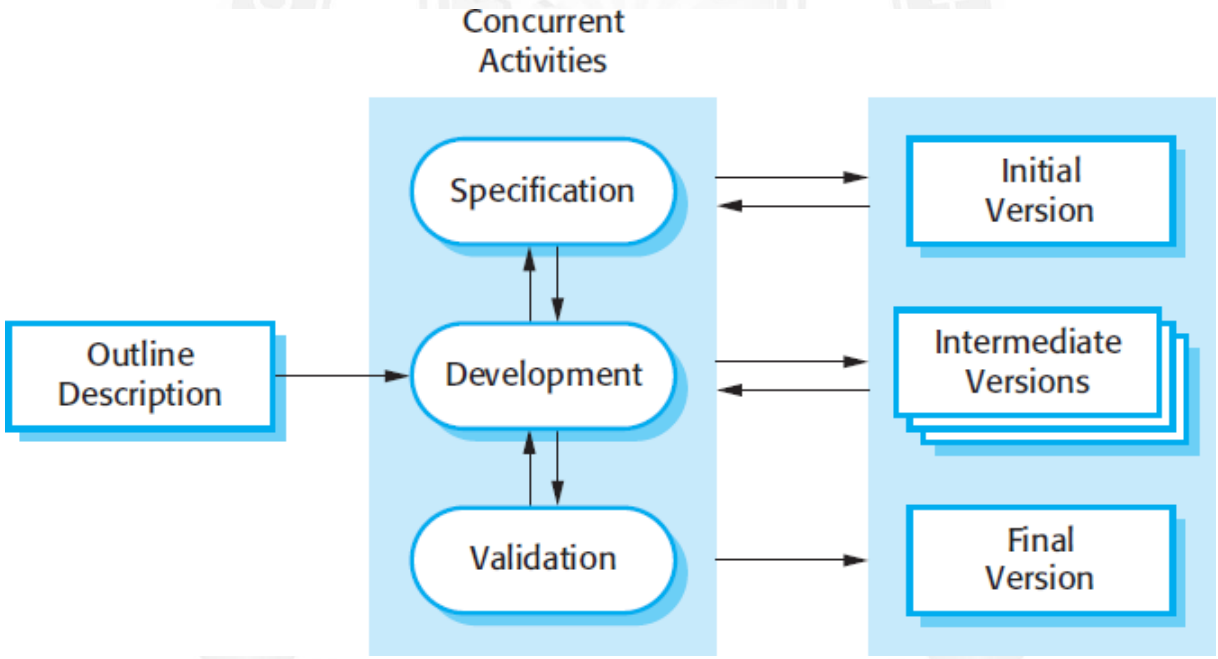
1. **Establish objectives**: The objectives of the prototype should be made explicit from the start of the process. Is it to validate system requirements, or demonstrate feasibility, etc.
2. **Define prototype functionality**: Decide what are the inputs and the expected output from a prototype. To reduce the prototyping costs and accelerate the delivery schedule, you may ignore some functionality, such as response time and memory utilization unless they are relevant to the objective of the prototype.

3. **Develop the proto type**: The initial prototype is developed that includes only user interfaces.
4. **Evaluate the proto type:** Once the users are trained to use the prototype, they then discover requirements error. Using the feedback both the specifications and prototypes can be improved. If changes are introduced, then a repeat of steps 3 and 4 may be needed.

Prototyping is not a standalone, complete development methodology, but rather an approach to be used in the context of a full methodology (such as incremental, spiral, etc.).

iv) Incremental Delivery

Incremental development is based on the idea of developing an initial implementation, exposing this to user feedback, and evolving it through several versions until an acceptable system has been developed. The activities of a process are not separated but interleaved with feedback involved across those activities.



Each system increment reflects a piece of the functionality that is needed by the customer. Generally, the early increments of the system should include the most important or most urgently required functionality.

This means that the customer can evaluate the system at early stage in the development to see if it delivers what's required. If not, then only the current increment has to be changed and, possibly, new functionality defined for later increments.

Incremental Vs Waterfall Model

Incremental software development is better than a waterfall approach for most business. E-commerce, and personal systems. By developing the software incrementally, it is cheaper and easier to make changes in the software as it is being developed.

Compared to the waterfall model, incremental development has three important benefits:

1. The **cost of accommodating changing** customer requirements is reduced. The amount of analysis and documentation that has to be redone is much less than that's required with waterfall model.
2. It's easier to get **customer feedback** on the work done during development than when the system is fully developed, tested, and delivered.
3. More **rapid delivery** of useful software is possible even if all the functionality hasn't been included. Customers are able to use and gain value from the software earlier than it's possible with the waterfall model.