Software Process Models:

- To solve actual problems in an industry setting, a software engineer or a team of engineers must incorporate a development strategy that encompasses the process, methods, and tools layers.
- This strategy is often referred to as a process model or a software engineering paradigm.
- A process model for software engineering is chosen based on the nature of the project and application, the methods and tools to be used, and the controls and deliverables that are required.

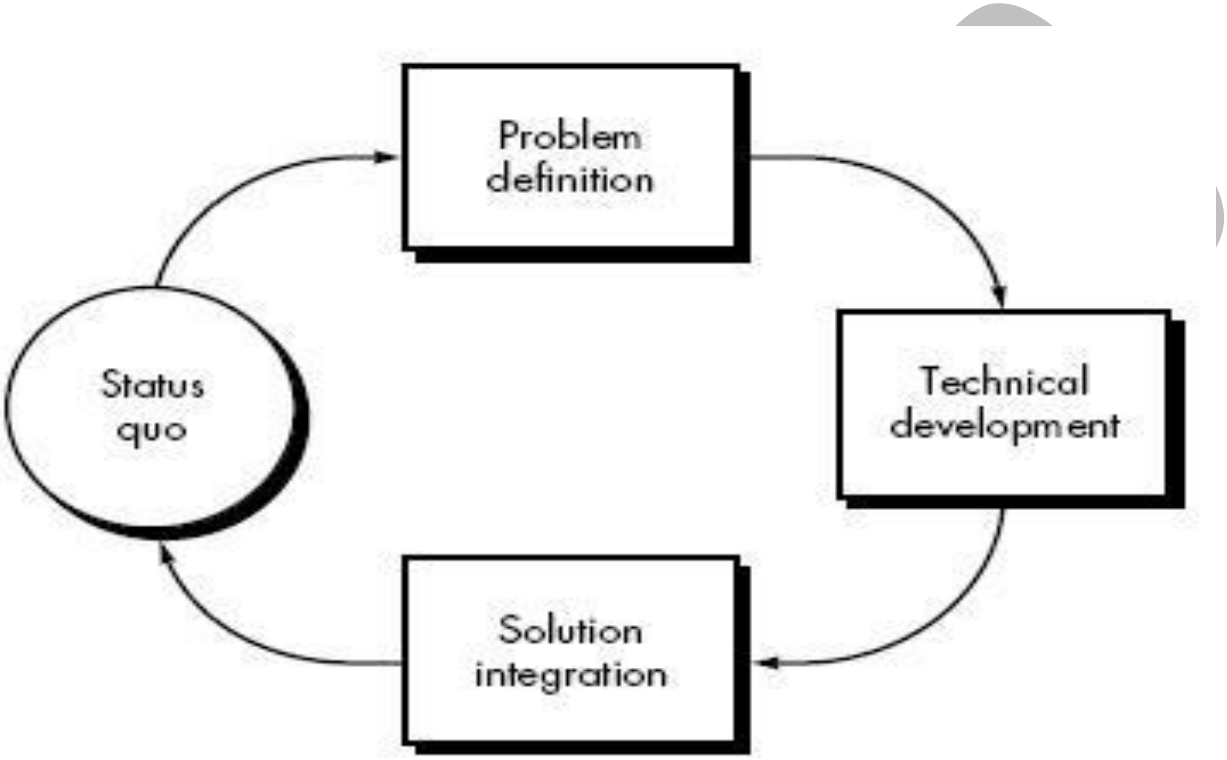


Figure 1.4(Process Model)

Process Models:

- The Waterfall Model
- The Prototyping Model
- The Spiral Model

Concurrent/Incremental Model

2.1 Waterfall Model:

It is also known as Linear Sequential model or Classic Life cycle model.

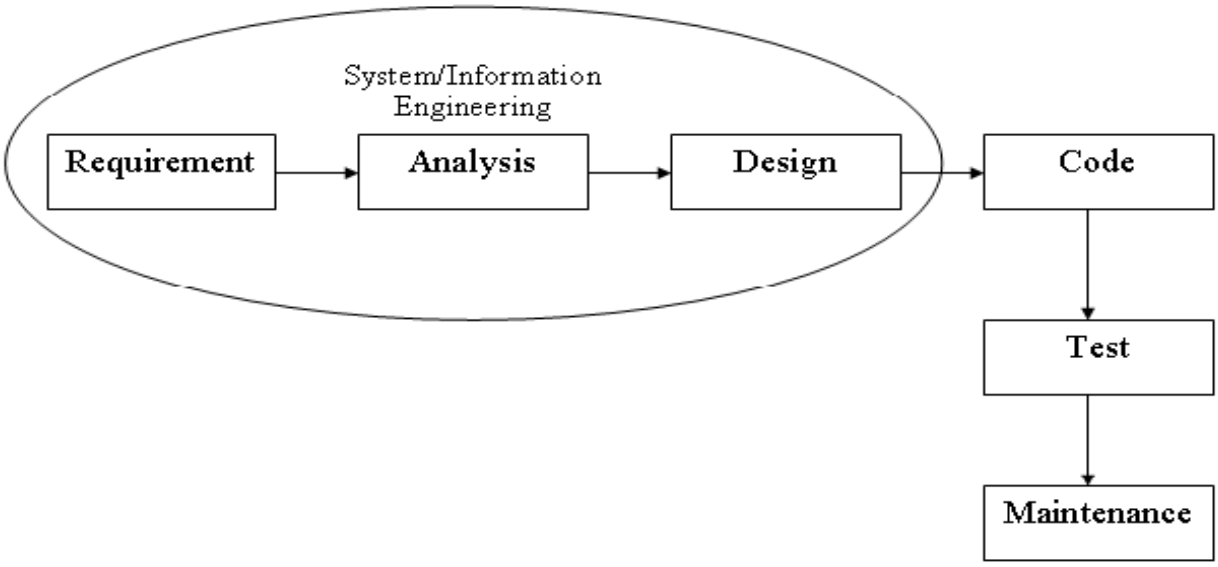


Figure 1.1 (Waterfall Model)

Requirements:

Requirements are set of functionalities and constraints that the end-user (who will be using the system) expects from the system.

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.

Coding:

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.

Maintenance:

Software will undoubtedly undergo change after it is delivered to the customer (a possible exception is embedded software). Change will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment (e.g., a change required because of a new operating system or peripheral device), 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.

Advantages of Waterfall Model

- Documentation is produced at every stage of the waterfall model development.
- This makes the understanding of the product designing procedure simpler.
- After every major stage of software coding, testing is done to check the correct running of the code.
- Organized approach provides robust separation of phases.
- Reflects common engineering practice.

Limitation of Waterfall Model

- You cannot go back, if the design phase has gone wrong, things can get very complicated in the implementation phase.
- Small changes or errors that arise in the completed software may cause a lot of problem.
- Not all requirements are received at once, the requirements from customer goes on getting added to the list even after the end of.
- As the requirements of the customer goes on getting added to the list, not all the requirements are fulfilled, this results in development of almost unusable system.

Characteristics

- Also called classic software life cycle or sequential model.
- Process activities (phases/stages) are clearly separated.

Waterfall Strengths

- Easy to understand, easy to use.
- Provides structure to inexperienced staff.
- Milestones are well understood.
- Sets requirements stability.
- Good for management control (plan, staff, track).
- Works well when quality is more important than Cost or schedule.

Use of Waterfall Model

- Requirements are very well known.
- Product definition is stable.
- Technology is understood.
- New version of an existing product.

2.2 Prototype Model:

"A Prototyping Model is the system development method in which a prototype is built, tested then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can be developed."

"Prototyping is becoming increasingly used for system development where rapid development is essential"

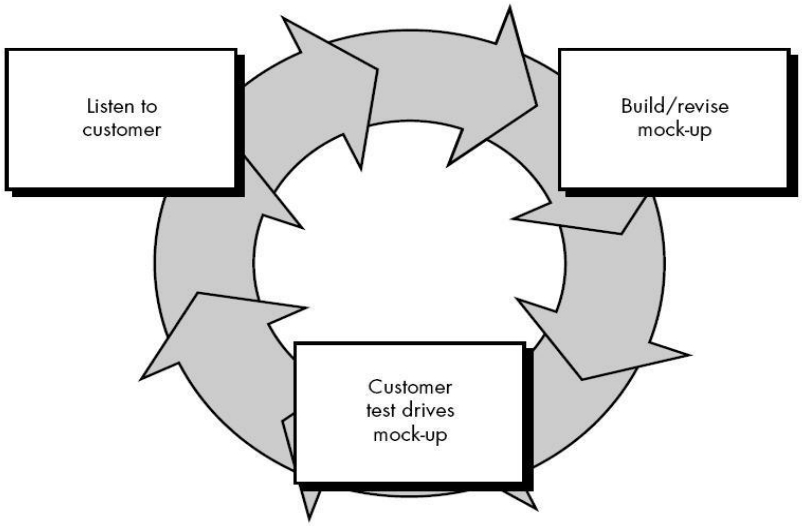


Figure 1.2 (Prototype Model)

Components of Prototyping Model:

1. Communication
2. Quick Plan
3. Modeling quick design
4. Construction of prototype
5. Deployment delivery and feedback

1. *Communication*
   - ❖ Prototyping model is begins with requirements.
   - ❖ Meet stakeholders to define the overall objectives for the software that will be visible to end user.
   - ❖ Define outline area.

2. *Quick Plan*
   - ❖ Make a plan for overall software which is going to be developed.
   - ❖ Define all the criteria which were mention in the requirements.
   - ❖ Highlight all the functionality of system or software.

3. *Modeling quick design*
   - ❖ It focuses on all aspects which will be visible to end users.
   - ❖ Example:
     - o Human interface layout
     - o Output display
   - ❖ Quick design leads to the construction of Prototype.

4. *Construction of prototype*
   - ❖ In this stage we include coding part and that time decide which technology and architecture going to be used.
   - ❖ Quality must be attended to :
     - o Usability
     - o Reliability
     - o Robustness
     - o Maintainability

- Integrity
- Portability
- Efficiency

- Taking the prototype and create whole project.

5. *Deployment delivery and feedback*
   - Here we will deploy our system/software
   - So that end user can use easily.
   - User provides feedback so it used for further refinement and enhancements.

Benefits:

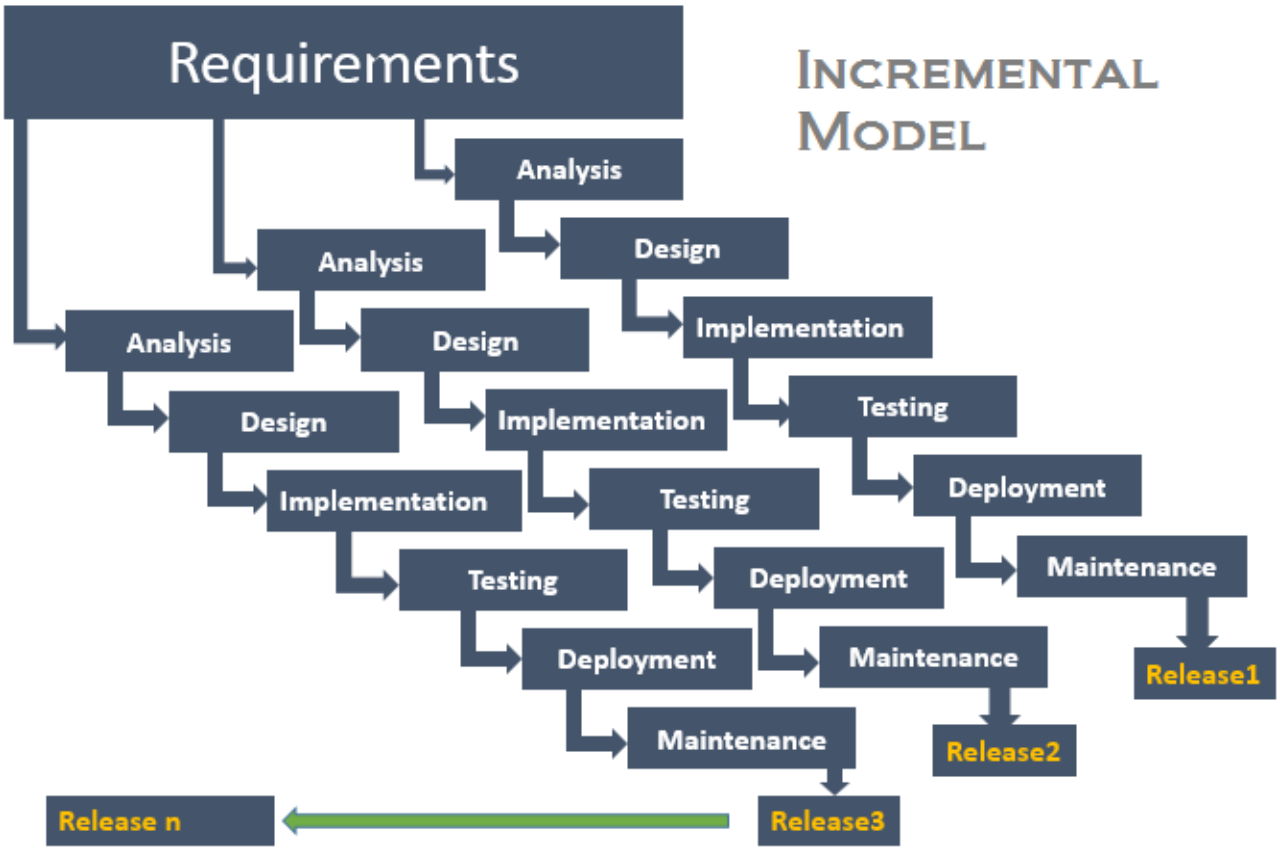
- Improved system quality
- Improved design quality
- Improved maintainability
- Reduced overall development effort
- Misunderstanding between may be detected and confusing services may be identified
- The system can support user training and system testing.
- Strong dialogues users and developer
- Missing functionality can be identified easily
- Confusion and difficult function can be easily identified
- Cost effective development (Cost reduced)
- Quick implementation of incomplete project requirements.
- Increase system development speed
- Helps to deliver the product with high quality easily

Limitation:

- Identifying non-functional elements difficult to document
- Incomplete or inadequate problem analysis
- Client may be unknowledgeable
- Approval process and requirement is not strict
- Requirements may frequently change

2.3 Incremental Model:

- Incremental build model.
- The incremental build model is a method of software development where the product is designed, implemented and tested incrementally (a little more is added each time) until the product is finished.
- It involves both development and maintenance.



Strengths

- You can develop prioritized requirements first.
- Initial product delivery is faster.
- Customers gets important functionality early.
- Lowers initial delivery cost.
- Each release is a product increment, so that the customer will have a working product at hand all the time.
- Customer can provide feedback to each product increment, thus avoiding surprises at the end of development.
- Requirements changes can be easily accommodated.

Weaknesses

- Requires effective planning of iterations.
- Requires efficient design to ensure inclusion of the required functionality and provision for changes later.
- Requires early definition of a complete and fully functional system to allow the definition of increments.
- Well-defined module interfaces are required, as some are developed long before others are developed.
- Total cost of the complete system is not lower.

2.4 Spiral model:

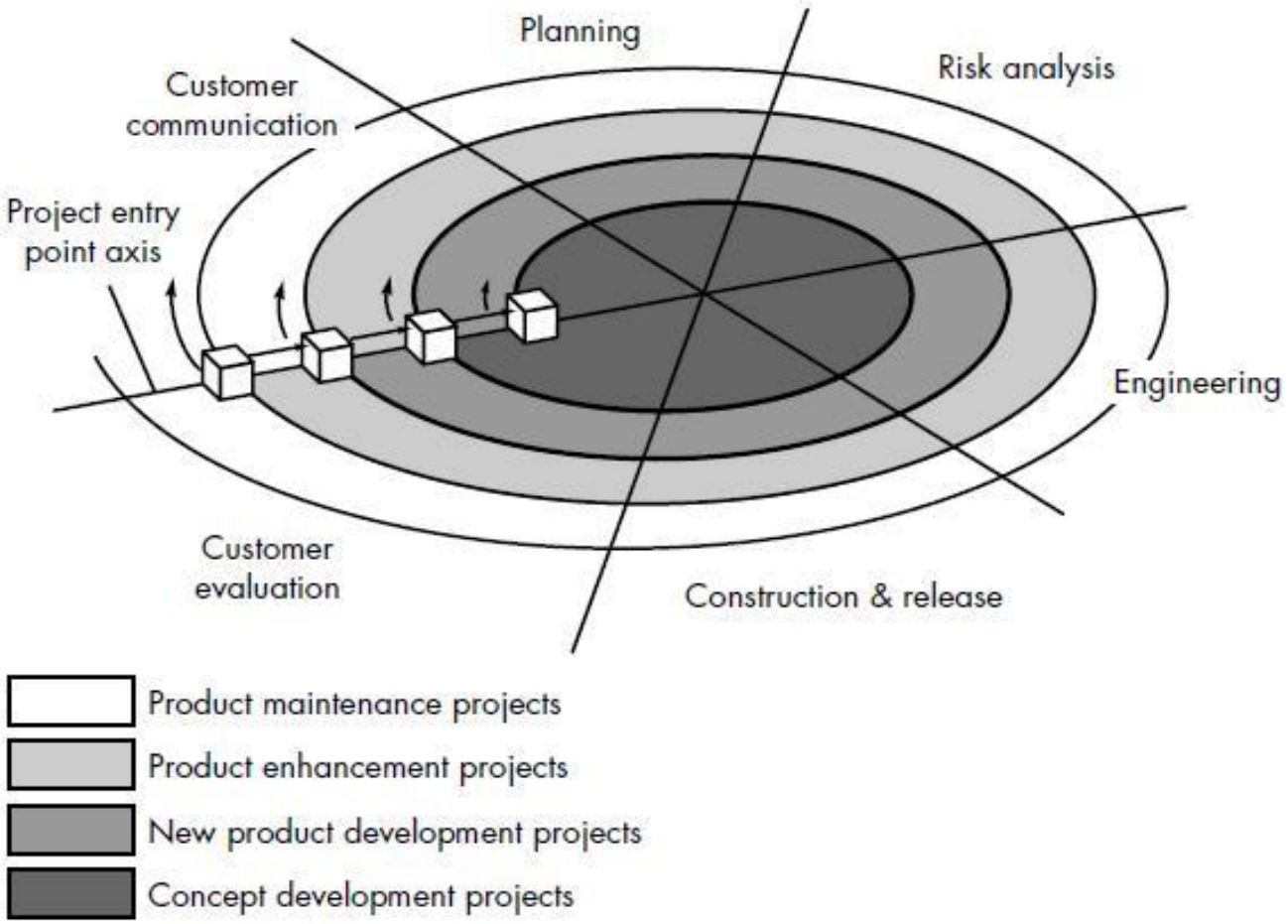
Spiral Model is an evolutionary software process model that combines the iterative nature of prototyping with the controlled and systematic aspects of linear sequential model.

The spiral model is intended for large, expensive and complicated projects.

Definition:
A software life-cycle model which supposes incremental development, using the waterfall model for each step, with the aim of managing risk. In the spiral model, developers define and implement features in order of decreasing priority.

History of Spiral Model

- The spiral model was defined by Barry Boehm in his 1986 article "A Spiral Model of Software Development and Enhancement"
- This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration matters
- As originally envisioned, the iterations were typically 6 months to 2 years' long
- Each phase starts with a design goal and ends with the client reviewing the progress
- Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project



A spiral model is divided into a number of framework activities, also called *task regions*.

- **Customer communication**
  - Tasks required to establish effective communication between developer and customer

- **Planning**
  - In this phase, the objectives, alternatives and constraints of the project are determined and are documented
  - Tasks required to define resources, timelines, and other project related information
  - The objectives and other specifications(terms) are fixed in order to decide which strategies / approaches to follow during the project life cycle

- **Risk analysis**
  - This phase is the most important part of "Spiral Model"
  - All possible (and available) alternatives, which can help in developing a cost effective project are analyzed and strategies are decided to use them in this phase
  - This phase has been added specially in order to identify and resolve all the possible risks in the project development
  - An analytical (logical) assessment of selected programs, to consider how to identify and eliminate risk

- **Engineering**
  - The actual development of the project is carried out
  - The output of this phase is passed through all the phases iteratively in order to obtain improvements same in this phase
  - Tasks required to build one or more representatives of the application

- **Construction and release**
  - Tasks required to construct, test, install and provide user support
  - e.g., documentation and training
  - Construction is divided into iterations
  - Each iteration is composed of: analysis, design, implement, test
  - Usually, specialists say that evaluation should be 30% of the project and construction 50%

- **Customer evaluation**
  - Developed product is passed on to the customer in order to receive customer's comments and suggestions
  - It can help in identifying and resolving possible problems/errors in the software which we have developed
  - This phase is very much similar to TESTING phase

Spiral Model Circuits
- The first circuit may result in a **requirement specification**

- The second will result in a **prototype**
- The next one will result in another prototype or **sample of a product**
- The last circuit leads to a product which is suitable to be sold

Advantages

- Introduces risk management
- Prototyping control cost
- Evolutionary development
- Release builds for data testing
- Critical high risk functions are developed first
- Early and frequent feedback from users
- Cumulative (increasing) costs assessed frequently

Disadvantages

- Highly customized limiting re-usability
- Applied differently for each application
- Risk of not meeting budget or schedule
- Lack of risk management experience
- Lack of milestone
- Management is doubtful (unsure) of spiral process
- Complex process model

When to use Spiral Model

- When creation of a prototype is appropriate
- When costs and risk evaluation is important
- For medium to high risk projects
- Users are unsure of their needs
- Requirements are complex
- Significant changes are expected

Purpose of Spiral Model

- Suitable for large project
  - The spiral module make the developer and customer to understand better and requirement of the system
  - Developer uses a prototype to reduce risk and cost
  - By the increase of the spiral, the prototype becomes a real system

➢ **Comparison with other models**

Water fall Model	Prototype Model	Spiral Model

When the customer requirements are clear and complete	When the customer requirements are not clear ambiguous or our project team is following this model with sample first	When the requirements of the customer are enthusing (Improving and extending)
Risk factor is considered in the spiral model but in water fall model it is not considered	When the requirement of the client is not clear and it is supposed to be changed. It doesn't cover risk management	Spiral model takes special care about risk analysis. Whereas it is not given importance in prototyping model