CHAPTER 5 –SOFTWARE ENGINEERING

What is Software Engineering?

(***Engineering*** means applying well–defined concepts, methodologies, scientific principles and methods)

- It is a structured, systematic approach for the design, development and maintenance of software systems.

Need for Software Engineering?

- Modern day software are large and complex, handles large amount of data.
- In most cases, software is not delivered on time or did not conform to the specifications provided by Customer.

Importance of Software Engineering?

1. **Correct Specifications –** As large software systems are complex, and have numerous functionality, it is very important to devise its specifications in a scientific, methodical manner.
2. **Scalability scope –** Software should be scalable with the current design. And to design such software, scientific methodology should be adapted.
3. **Cost Control –** If software not designed using scientific methodology, faults may occur at later stages. A fault detected later costs much higher as compared to early fault-detection.
4. **Quality –** Engineering methodology ensures that correct problem and specifications for the new software are determined. This leads to efficient design and overall enhanced quality of the software.

Software Process Activities/SDLC (Software development Life Cycle)

- It refers to set of **logically related activities**, which are carried out in a systematic order that leads to the production of 'the software to be delivered'.
- **Four important activities** in SDLC are:
  1. **Software Specification (Requirement Engineering)**
     - This activity is responsible for **defining the main functionalities** of the software as per the customer's requirements and expectations.
     - It also defined **constraints** on its operation.
     - It is carried out in *four phases*:
       (i) **Feasibility study**
       - **Cost estimation** of proposed software.
       - Determines whether the software will be **cost-effective** from a business point of view.
       - Economic, technical and operational **feasibility** of the project.

       (ii) **Requirements Elicitation and analysis**
       - **Determining the System requirements**, by studying existing system and discussion with potential users.

       (iii) **Requirement Specification**
       - Determined **requirements are documented** for **user** (functionality features) **requirements** and **system** (non-functional, e.g. security, performance, interface etc.) **requirements**.

       (iv) **Requirements Validation**
       - Determine whether the requirements defined are complete and all user problems, needs and expectations etc, have been taken care of.

  2. **Software Design and Implementation**
     - This activity is responsible for the **design and programming** of proposed software.
     - Sub activities carried out in this phase are:

(i) Architectural and abstract design

The sub-system of system and their relationships are identified.

(ii) Interface design

The interface is designed and documented for each sub-system.

(iii) Data Structure and algorithm design

- The required data structures used in the system implementation are designed.
- The algorithms used to provide services are designed.

(iv) Programming and Implementation

- The programming is done with determined data structures and algorithms as per the design.
- The program code is implemented.

3. **Software Verification and Validation**
   - This activity is responsible for ensuring that **software conforms to all the specifications** and works as per the proposed design.
   - **Verification –** It means that set of activities that are carried out to confirm that the software correctly implements the specific functionality.
   - **Validation –** It means that the set of activities that ensure that the software that has been built is satisfying the customer's requirements.
   - For verification and validation, system testing is carried out in three forms:
     - **System components are tested –** For components, usually, components development and testing are carried out simultaneously with one another.
     - **Integrated components** of the system are tested together.
     - **Entire system** is tested in actual conditions.

4. **Software Evolution/Software Maintenance**

   This activity ensures that:

   (i) The developed software meets the customer requirements.

   (ii) The software design ensures adaptability and scalability, i.e., it must evolve to meet changing customer needs.

Software Process Model

- It is a simplified representation of a software process.
- Three Generic Models:
  1. The Waterfall model
  2. Evolutionary model
  3. Component-based model

The Waterfall Model

- The waterfall model is a linear and sequential approach of software development where software develops systematically from one phase to another in a downward fashion. This model is divided into different phases and the output of one phase is used as the input of the next phase. It means that every phase has to be completed before the next phase starts and there is no overlapping of the phases.
- Most suitable for projects where:
  1. Requirements are clearly determined.
  2. Each phase fully completes and makes available its output as input of next phase.

- This model consist of 5 phases:

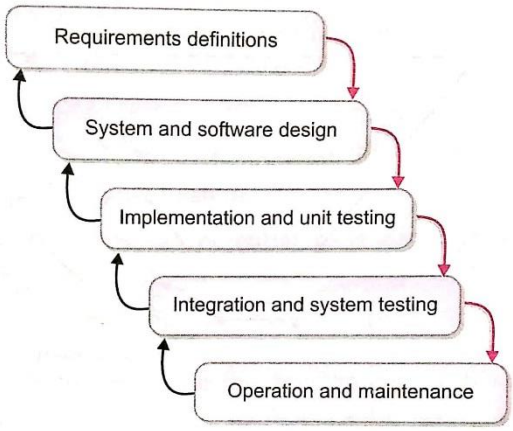

Figure 5.1 The Waterfall model.

1. **Requirement specification**
   - Requirements are gathered.
   - Estimating *Project costs, risks and schedule*.
2. **Analysis and System Design**
   - Design the overall software architecture.
   - Identifying fundamental software system components and their relationships.
3. **Implementation and unit testing**
   - Software design is realised as a set of software components through programming, coding and documentation.
   - Each component tested along with its development.
4. **Integration and System testing**
   - components are integrated and tested as a complete system.
5. **Operation and Maintenance**
   - System is delivered and deployed for the client.
   - The real and practical use of the software system begins.

Advantages of Waterfall Model

1. **Departmentalisation:** It divides the whole process into departments. This allows for separate schedules and deadlines for each department.
2. Easy to understand and explainable Model
3. It is **easy to manage model** as each phase has specific deliverables and a review process.
4. No overlapping of phases → makes this model **simple.**

Disadvantages of Waterfall Model

1. **No estimation of time and cost:** not give idea of estimated time and cost for each phase of the development process.
2. **Difficult to incorporate changes:** once an application is in the testing stage, it is very difficult to go back and change something.
3. **Not for complex systems:** Not a good model for large and complex software system, as they have many overlapping modules.

V-Model

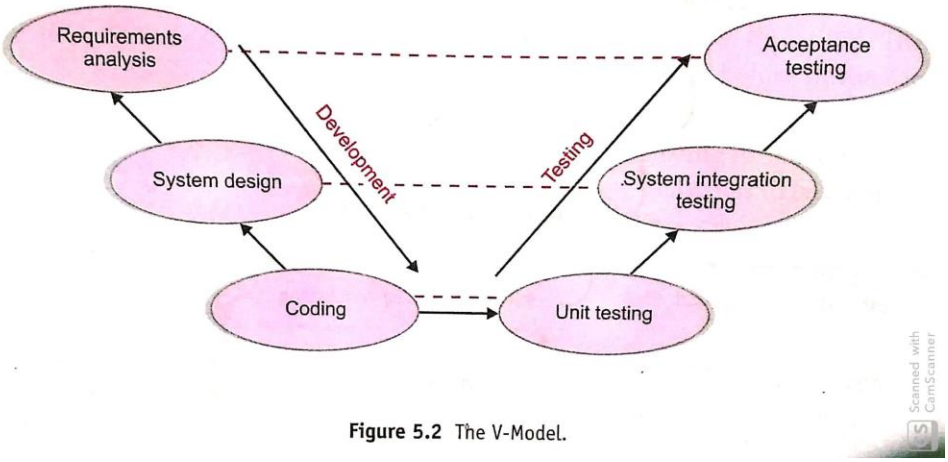

Figure 5.2 The V-Model.

- a variation of waterfall model.
- The processes/phases are same as waterfall model , but phases are bent upwards after the implementation phase, to form the typical V-Shape.
- This model allows parallel execution of activities.

The Evolutionary Model

- It is a rapid-software development model where an *initial software implementation is rapidly developed* from very abstract specifications, which is then iteratively modified according to the user's appraisal of the software.
- It is therefore also called ***Prototype model.***
- Most suitable for projects where:
    1. High technical risks are involved.
    2. Timeline is aggressive.

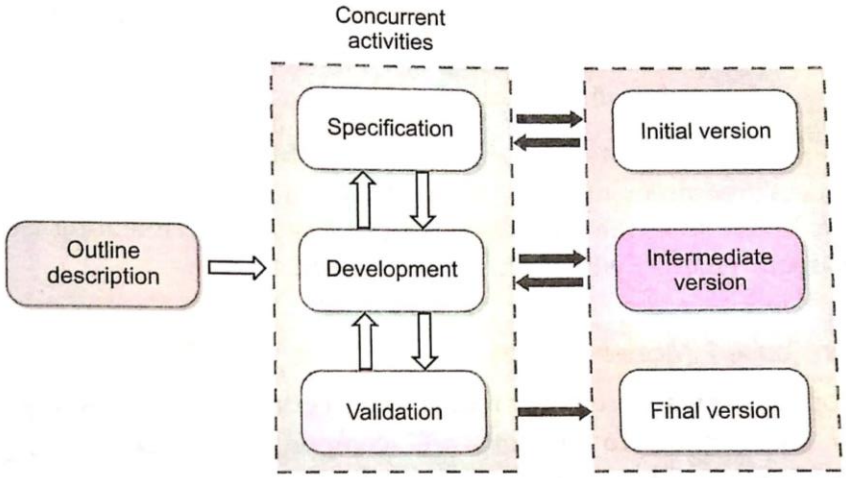

Figure 5.3 Evolutionary model.

Advantages of Evolutionary Model

(i) It ***reduces risk of failure***, as potential risks can be identified early by continuous appraisal of the software implementation and corrective steps can be taken.

(ii) The development takes place with combined ***contribution of the development team and the client*.**

(iii) ***User gets a fair idea of final product's*** working as a working model of the system is provided, the user get a better understanding of the system being developed and can suggest changes and modifications.

(iv) ***User feedback*** is available at an early stage leading to better solution.

Disadvantages of Evolutionary Model

(i) If ***changes suggested are too many*** then it may ***disturb the rhythm of the development team***.

(ii) Too many changes may ***increase the complexity of the system***. Also, the scope of the system may expand beyond original plans.

(iii) Repeated changes may ***increase the cost of development*** and it may derail the budgeting of the software system.

Component-based Model

Component/Software component – It is a ***software element*** that confirms to a software model and can be ***independently deployed*** and composed without modification according to a composition standard.

- **Component-based software development model** is a software development process that develops software systems by selecting appropriate off-the-shelf components and then assembles them with a well-defined software architecture.
- It is based on ***idea of incorporating and reuse existing software components*** in current software development if feasible and possible.

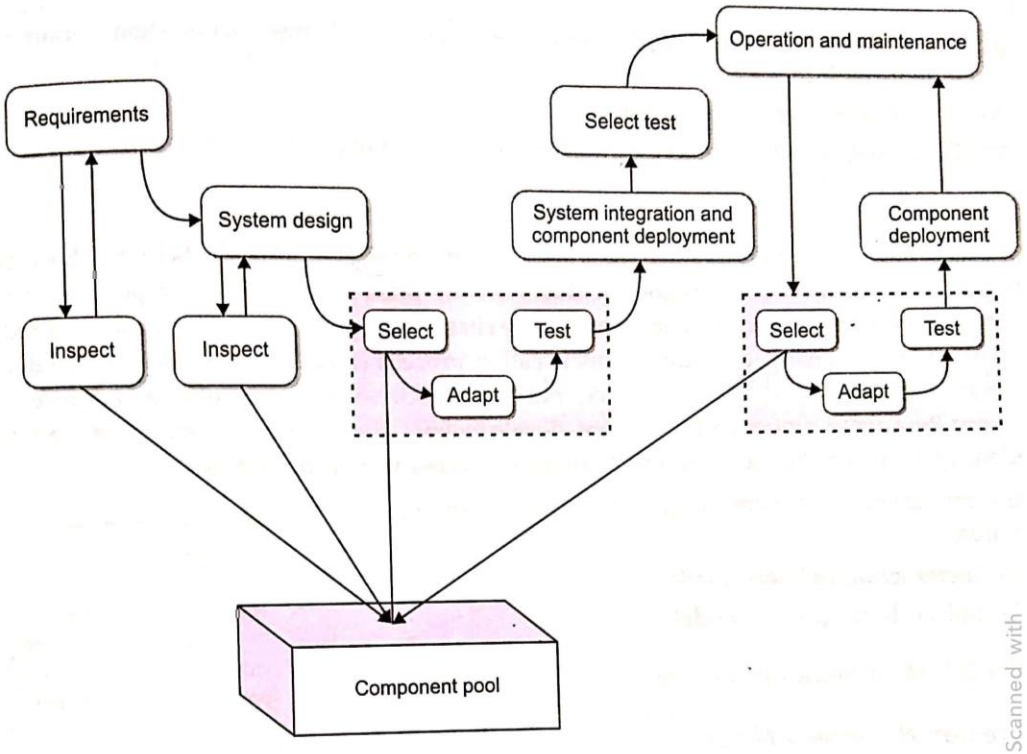

Figure 5.4 The component-based model.

Various phases in this model are:

1. **Requirement Specification:** the basic software specifications (not full specifications) for the new system are determined.
2. **Component Analysis:** From the available software-components pool or library, a search is made for the components that can implement specification as per specifications decided for the new system.
   If a component meets a specification partly (say about 60-70%), the components may be used.
3. **Requirements modification:** The software requirements are re-analysed using information about the available reusable components.
4. **System design with reuse:**
   Two activities:
   (i) System design after taking into account the reusable component available.
   (ii) If no reusable components are available, the new software (components) may have to be designed.
5. **Development and integration:**
   - The ***fresh development*** of the software components takes place for which no reusable components were found.
   - Then the freshly ***developed software components and the reusable systems are integrated*** to create the new system.

Advantages of Component-based Model

(i) It ***reduces the amount of software*** to be developed.
(ii) It also leads to ***faster delivery*** of the software.

Disadvantages of Component-based Model

(i) Sometimes ***requirements are compromised***.
(ii) This may lead to a ***system that does not meet the real (original) needs of users***.

Delivery Models

Why delivery models needed?

(a) *Changing environment and conditions* result in changes in system requirements (e.g. growth in user base)
(b) *Policies and management* priorities may change.
(c) *Technological advancement* may lead to forced changes.

Delivery models is not just about delivery, but it is about development process that delivers iteratively along with software development.

Two important delivery models: (i) Incremental delivery model **(ii)** Spiral development model

Incremental Delivery Model

- The Incremental Model is designed, implemented and tested as a series of ***incremental software builds*** until the product is finished. A build consist of pieces of code from various modules that interact together to provide a specific service or functionality.

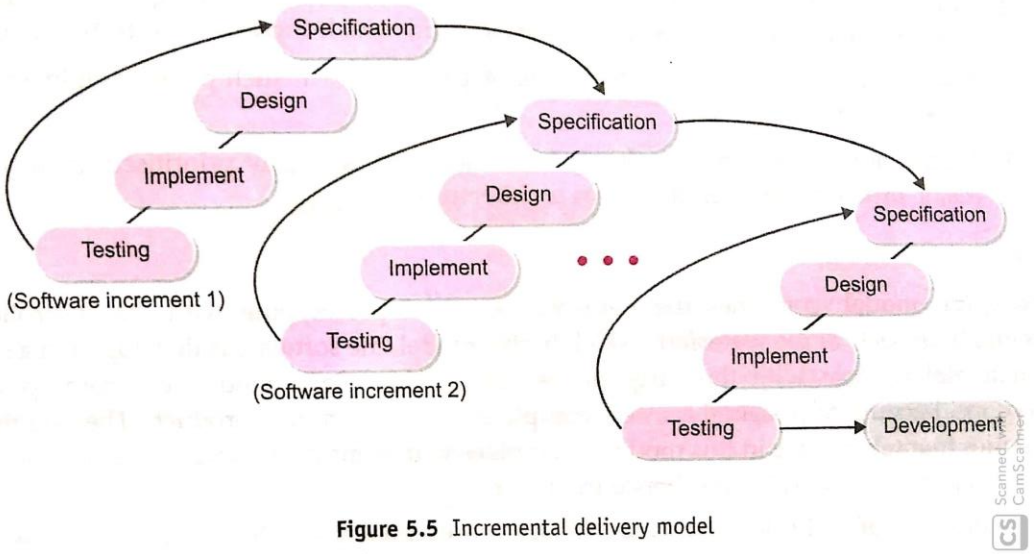

Figure 5.5 Incremental delivery model

- Phases in incremental model are:

1. **Determine overall services to be delivered**
   This phase ***identifies all types of services*** that the proposed system needs to deliver.

2. **Determine software incremental builds**
   In order to determine software builds or Software Increments, the following mechanism is carried out:
   (i) The customer is asked to prioritise the required services as the **'most important service(s)'** to the **'least important service(s)'.**
   (ii) As per the customer decided priority levels, ***the number of software-delivery increments (also called software builds) are defined*** where each software increment provides each priority level service(s).

3. **Development and delivery of each software incremental build**
   The following activities are carried out for each software increment, in the order of their priority:
   (i) The ***requirements for each software increment are defined*** in detail, and that increment is developed using the best suited software process.
   (ii) Once developed and tested, the ***software build is delivered*** and implemented for the client, ***integrating it with already installed increments***.
   (iii) The work for next priority software build starts and the whole process is repeated.

Advantages of Incremental Delivery Model

1. It ***generates working software quickly***. Customers do not have to wait until the entire system is delivered.
2. It is ***easier to test and debug*** during a smaller iteration.
3. It **cost less to change** scope and requirements.
4. There is a ***lower risk of overall project failure*** as tested increments are added.
5. ***Easier to manage risk*** because risky pieces are identified and handled first of all.

Disadvantages of Incremental Delivery Model

1. After each software increment is added with the system, the integration testing to test the system working as a whole, is carried out, which increases the testing-load.
2. If all requirements are not clearly identified and not aptly prioritised, it may lead to major problems in overall system architecture.

The Spiral Model

- It **combines** the iterative nature of **prototyping** with the systematic approach of **waterfall model**.
- Software is developed in a series of incremental release, where in early stage, prototypes are used and in later iterations, it becomes a more complete version of the product.
- In this model, the sequence of activities takes place with some backtracking from one activity to another, just like a spiral – and hence the name.
- Most suitable for large, complex and expensive software systems.
- The following activities are carried out during each phase of spiral model:
  (i) Objective setting (First Quadrant) – The ***objective of the phase*** are determined and ***associated risks*** are examined.
  (ii) Risk Assessment and Reduction (Second Quadrant) – Detailed Analysis of the risk that determined in previous phase. This phase is also responsible for risk reduction.
  (iii) Development and Validation (Third Quadrant) – This phase is for the development and validation of the next level of the product after resolving the identified risks.
  (iv) Review and Planning (Fourth Quadrant) – During this phase, the results achieved so far are reviewed with the customer. And the planning for the next iteration also takes place.

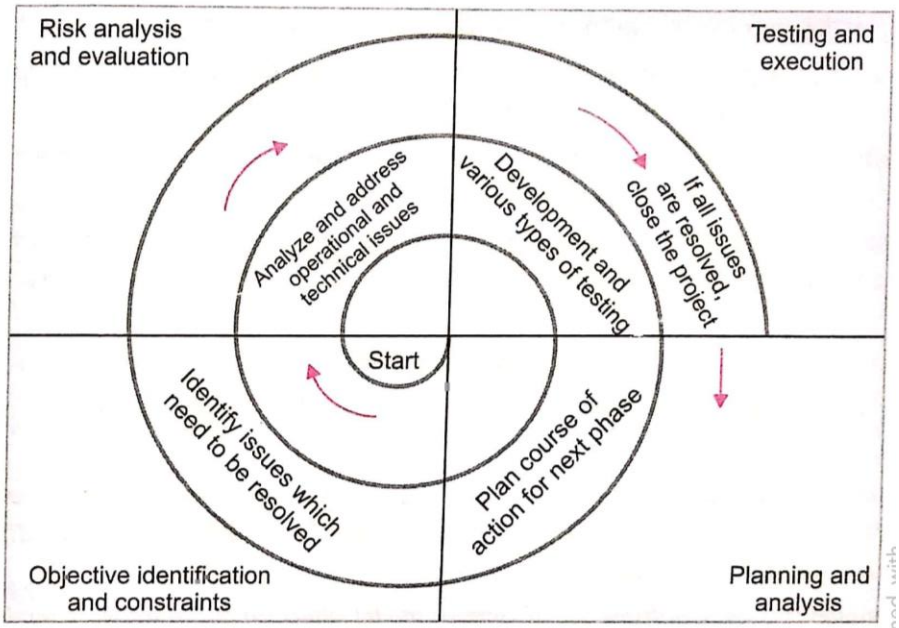

Figure 5.6 The four phases of spiral model

****