Module-1, SDLC (SE)

Software Engineering is a systematic, disciplined, quantifiable study and approach to the design, development, operation, and maintenance of a software system.

Classification of Software:

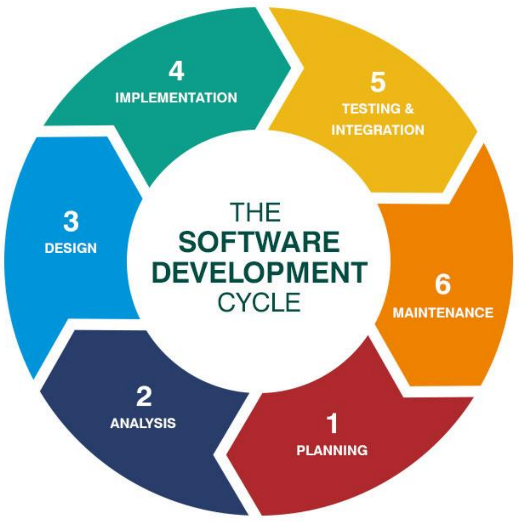
- System Software
  - Operating System
  - System Development Software
  - Utility Programs
- Application Software
  - User Developed Programs
  - Application Package

Software Development Life Cycle (SDLC):

It is a process used by the software industry to design, develop and test high quality software.

The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

SDLC is a framework defining tasks performed at each step in the software development process.



Types pf Software Development Process Model:

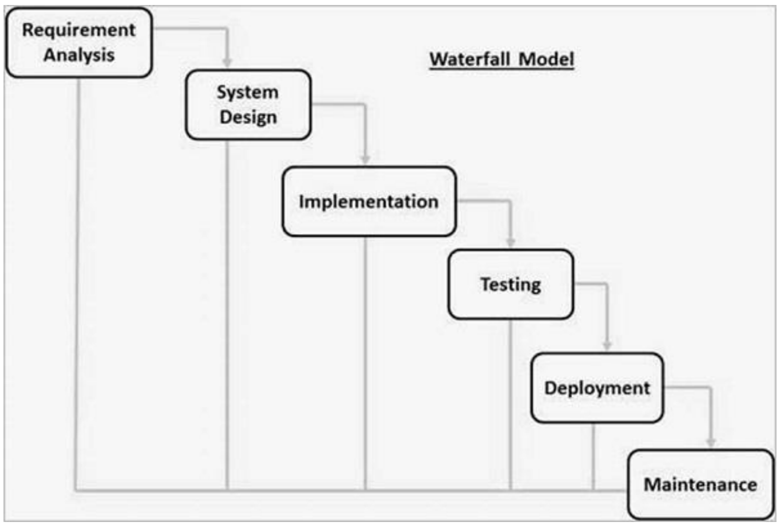
- Waterfall Model
- Prototype Model
- RAD Model
- Incremental Model
- Spiral Model

Waterfall Model:

The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model".

This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.



The sequential phases in Waterfall model are as follows:

- **Requirement Gathering and Analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Applications of Waterfall Model:

- Product definition is stable
- Technology is understood and is not dynamic

Advantages of Waterfall Model:

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model
- Phases are processed and completed one at a time
- Clearly defined stages
- Well understood milestones
- Easy to arrange tasks

Disadvantages of Waterfall Model:

- No working software is produced until late during the life cycle
- High amounts of risk and uncertainty
- Poor model for long and ongoing projects
- Difficult to measure progress within stages
- Cannout accommodate changing requirements

Prototype Model

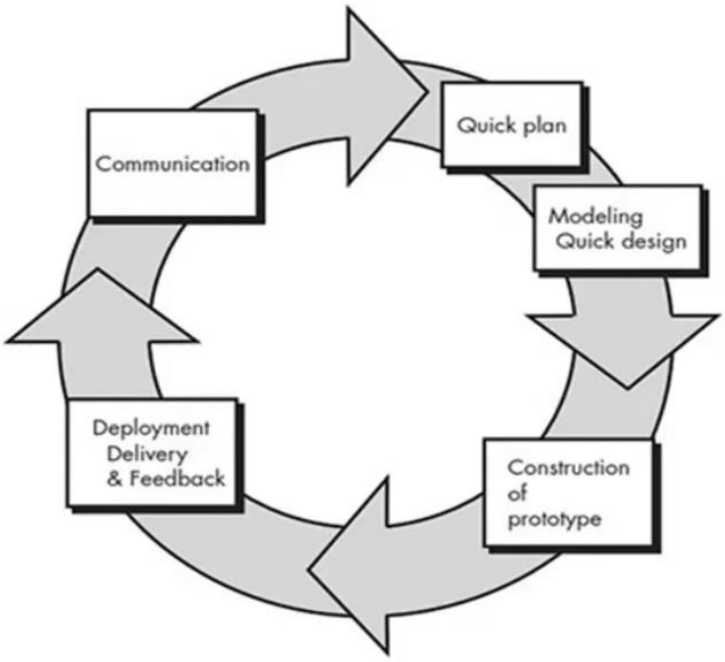
This model is used when the customers do not know the exact project requirements beforehand.

In this model, a prototype of the end product is first developed, tested and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.

In this process model, the system is partially implemented before or during the analysis phase thereby giving the customers an opportunity to see the product early in the life cycle.

The process starts by interviewing the customers and developing the incomplete high-level paper model. This document is used to build the initial prototype supporting only the basic functionality as desired by the customer.

Once the customer figures out the problems, the prototype is further refined to eliminate them. The process continues until the user approves the prototype and finds the working model to be satisfactory.



Advantages of Prototype Model:

- The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.
- New requirements can be easily accommodated as there is scope for refinement.
- Missing functionalities can be easily figured out.
- Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
- Flexibility in design.

Disadvantages of Prototype Model:

- There may be too much variation in requirements each time the prototype is evaluated by the customer.
- It is very difficult for developers to accommodate all the changes demanded by the customer.
- There is uncertainty in determining the number of iterations that would be required before the prototype is finally accepted by the customer.
- Developers in a hurry to build prototypes may end up with sub-optimal solutions.

- The customer might lose interest in the product if he/she is not satisfied with the initial prototype.

Use of Prototype Model:
The Prototyping Model should be used when the requirements of the product are not clearly understood or are unstable. It can also be used if requirements are changing quickly.
It is also a very good choice to demonstrate the technical feasibility of the product.

Incremental Model

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases.

Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

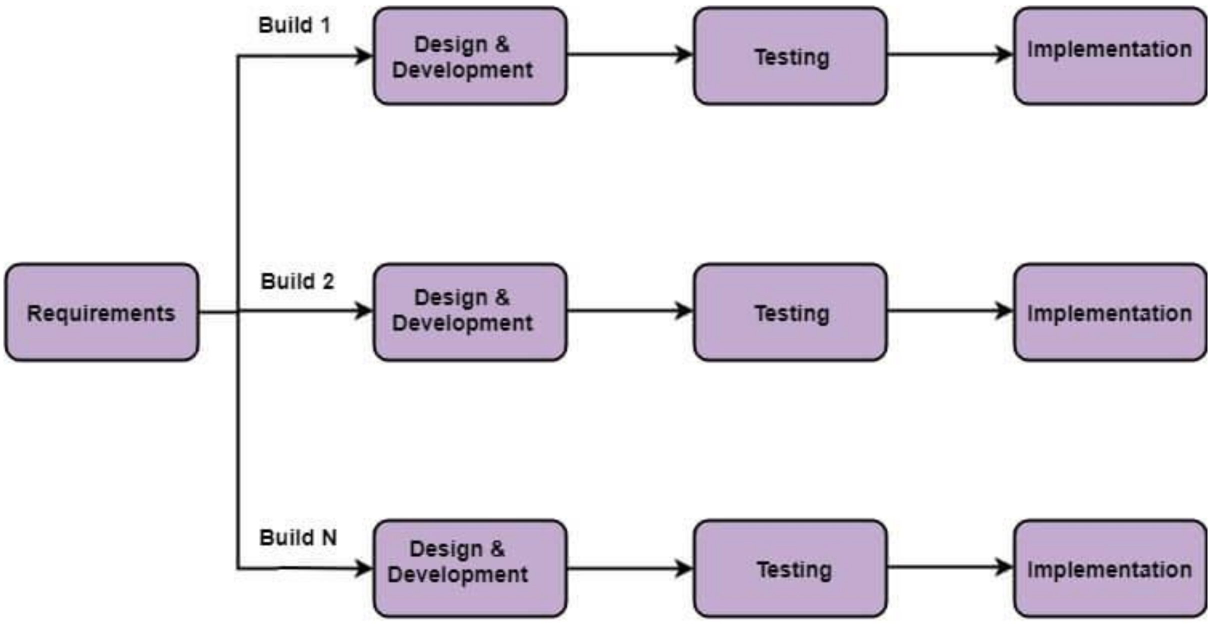


Fig: Incremental Model

Various Phases of Incremental Model:

- **Requirement Analysis:** In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.
- **Design and Development:** In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are

finished with success. When software develops new practicality, the incremental model uses style and development phase.

- **Testing:** In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.
- **Implementation:** Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

Use of Incremental Model

- When the requirements are superior.
- A project has a lengthy development schedule.
- When Software team are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.

Advantages of Incremental Model:

- Errors are easy to be recognized
- Easier to test and debug
- Uses divide and conquer breakdown of tasks
- More flexible
- Simple to manage rusk because it hadnles durings its iteration
- Client get important functionality early

Disadvantages of Incremental Model:

- Requires good planning and design
- Total cost is high
- Well-defined modules interfaces are need

RAD Model:

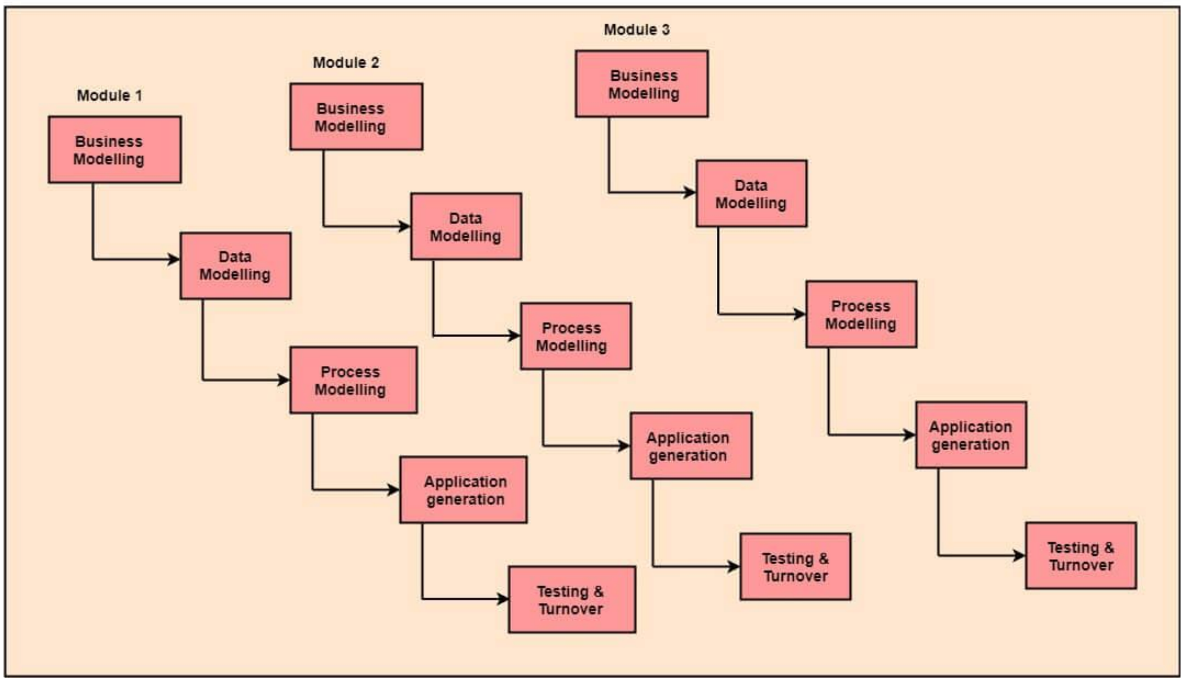
RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element-based construction approach.

It is a concept that products can be developed faster and of higher quality through:

- Gathering requirements using workshops or focus groups
- Prototyping and early, reiterative user testing of designs

- The reuse of software components
- A rigidity paced schedule that refers improvements to the next product version
- Less formality in reviews and other team communication

Fig: RAD Model



The various phase of RAD model are as follows:

1.Business Modelling: The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.

2. Data Modelling: The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.

3. Process Modelling: The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

4. Application Generation: Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.
5. Testing & Turnover: Many of the programming components have already been tested since RAD emphasis reuse. This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised.

When to use RAD Model:

- When the system should need to create the project that modularizes in a short span time (2-3 months).
- When the requirements are well-known.
- When the technical risk is limited.
- When there's a necessity to make a system, which modularized in 2-3 months of period.
- It should be used only if the budget allows the use of automatic code generating tools.

Advantages of RAD Model:

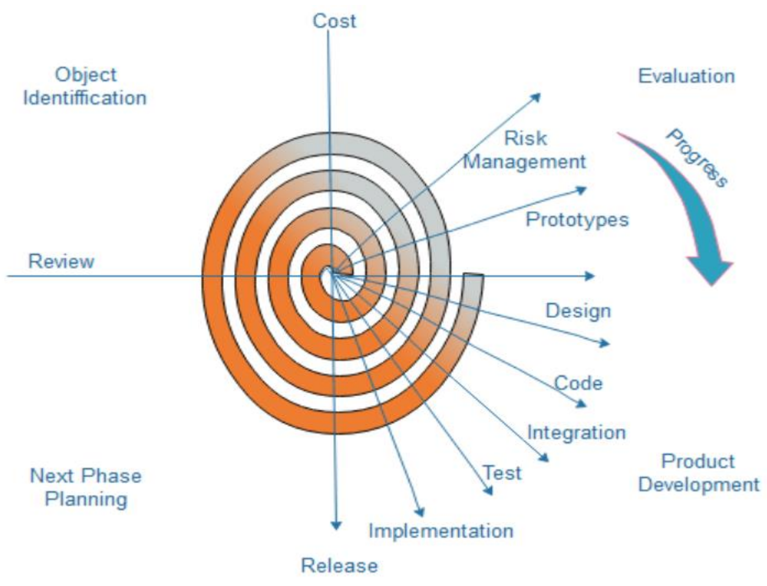
- This model is flexible for change.
- In this model, changes are adoptable.
- Each phase in RAD brings highest priority functionality to the customer.
- It reduced development time.
- It increases the reusability of features.

Disadvantages of RAD Model:

- It required highly skilled designers.
- All application is not compatible with RAD.
- For smaller projects, we cannot use the RAD model.
- On the high technical risk, it's not suitable.
- Required user involvement.

Spiral Model:

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



It implements the potential for rapid development of new version of the software.

Using the spiral model, the software is developed in a series of incremental releases.

<u>Each cycle in the spiral is divided into four parts:</u>

- **Objective setting:** Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieveing 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.

When to use Spiral Model?

- When the project is large
- When requirements are unclear and complex
- When changes may require at any time
- Large and high budget projects
- When deliverance is required to be frequent

Advantages of Spiral Model:

- High amount of risk analysis
- Useful for large and mission-critical projects

Disadvantages of Spiral Model:

- Can be a costly model to use
- Risk analysis needed highly particular expertise
- Doesn’t work well for smaller projects

COCOMO Model (Constructive Cost Estimation Model):

COCOMO Model is one of the most generally used software estimation models in the world. It predicts the efforts and schedule of a software product based on the size of the software.

Basic COCOMO Model:

The effort equation is as follows: **$E = a*(KLOC)^b$** **$D = c*(E)^d$**

where:

E = effort applied by per person per month

D = development time in consecutive months

KLOC = estimated thousand of lines of code delivered for the project

In COCOMO, projects are categorised into three types:

- **Organic:** A development project can be treated of the organic type, if the project deals with developing a well-understood application program, the size of the development size is really small, and the team members are experiences in developing similar methods of projects.
  Example: Simple Business System, Simple Inventory Management System.

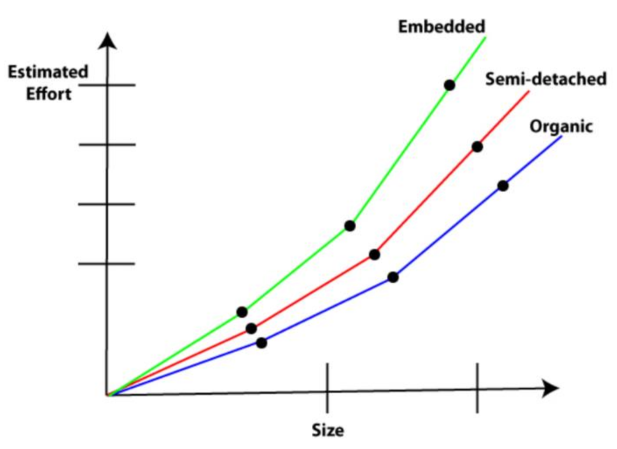
- **Semidetached:** A development project can be treated with semidetached type if the development consists of a mixture of experienced and inexperienced staff.

  Example: Developing a new Operating System (OS), Database Management System (DBMS), and Complex Inventory Management System.

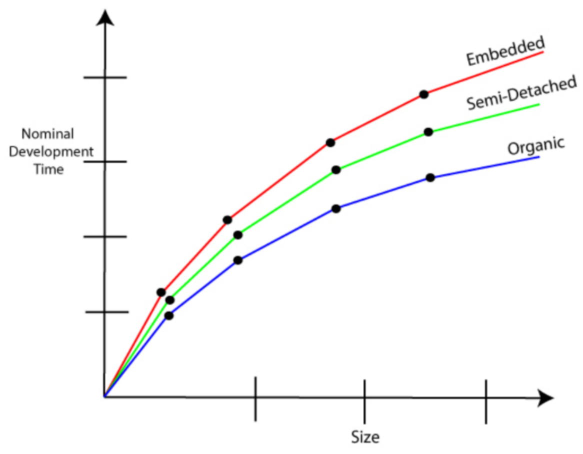
- **Embedded:** A development project is treated to be of an embedded type, if the software being developed is strongly coupled to complex hardware on the operational method exist.
  Example: ATM, Air Traffic Control

Basic COCOMO Model Equation:

Mode	**Effort**	**Schedule**
Organic	$E = 2.4*(KDSI)^{1.05}$	$TDEV = 2.5*(E)^{0.38}$
Semidetached	$E = 3.0*(KDSI)^{1.12}$	$TDEV = 2.5*(E)^{0.35}$
Embedded	$E = 3.6*(KDSI)^{1.20}$	$TDEV = 2.5*(E)^{0.32}$



Effort versus product size



Development time versus size

Numerical Problems:

1. Suppose a project has estimated to be 400 KLOC. Calculate the effort and the development time for each of the three model i.e., organic, semidetached and embedded

Ans:

Effort E

Organic E = 2.4 * $(400)^{1.05}$ PM = 1295.312 PM

Semidetached E = 3.6 * $(400)^{1.12}$ PM = 2462.79 PM

Embedded E = 3.6 * $(400)^{1.2}$ PM = 4772.814 PM

O = T_{dev} = 38.075 M

S = T_{dev} = 38.45 M

E = T_{dev} = 37.597 M

2. Average: 2500 PM (Salary). Calculate expected cost of software project.

Ans:

O = T_{dev} = 25000 * 38.075 = 951750

S = T_{dev} = 25000 * 38.45 = 961250

E = T_{dev} = 25000 * 37.597 = 939750

3. Project size of 200 KLOC is to be developed. Software development team has average experience on similar type of project. Two project schedule is not very tight. Calculate the effort, Tdev, average stuff size and productivity of the project.

Ans:

Semidetached

$E = 3 * (200)^{1.12} = 1133.18$ PM
$T_{dev} = 2.5 (1133.18)^{0.35} = 29.31$ M
Avg stuff size = 1133.18/29.31 = 38.66 E/T

Productivity = Size(KLOC)/Effort = 200/1133.18=0.176 KLOC/PM=1.76 LOC/PM

4. Let us assume that the software product is containing an estimated to be 62,000 lines of source code. The average salary of software engineers is Rs 20000 per month. Calculate effort and development cost of the software product.

Ans:

Effort = $2.4(KLOC)^{1.05}$

= $2.4(62)^{1.05}$

= 2.4 * 76.2099

= 183 PM

TDEV = $2.5(Effort)^{0.38}$ months

= 2.5 * 7.2399

= 18 months

Therefore, Cost required to develop the product = 18 * 20,000 = Rs 360,000/-