**SDLC Models**

SDLC stands for *‘Software development life cycle’.* It describes various phases involved in the software development process using which we can create and maintain a software efficiently.

Without SDLC models it becomes tough to monitor progress of the project.

Factors that are to be considered while choosing SDLC Model:

* Project Requirements
* Team Size
* Development Timeline
* Level of Flexibility

Different Phases Of SDLC



Requirement Analysis & Defining: Main activities included.

* Requirements Gathering
* Requirements Organization
* Requirements Documenting
* Requirement Verification & Validation

It is the process of collecting actual data, understanding process involved, defining the problem, and providing documentation for further use.

Once problem is understood, it is described in the requirements specification document or the SRS.

The document describes the product to be delivered not the process how it is to be developed.

Software Design: The goal I to transform the collected data into a structure that is suitable to implement in programming languages.

Physical and logical design i.e., HLD and LLD.

Main activities included.

* Developing architecture of system
* Developing algorithms for each component
* Outlining hierarchical structure
* Designing E-R diagrams, DFD’s, UML diagrams, etc.

Coding: Concerned with the development of source code that will implement the design.

Testing: Main activities included.

* Detecting design errors, Requirement errors, Coding errors
* Fixing/Correcting/Removing bugs
* Preparing test cases, test plans and test reports

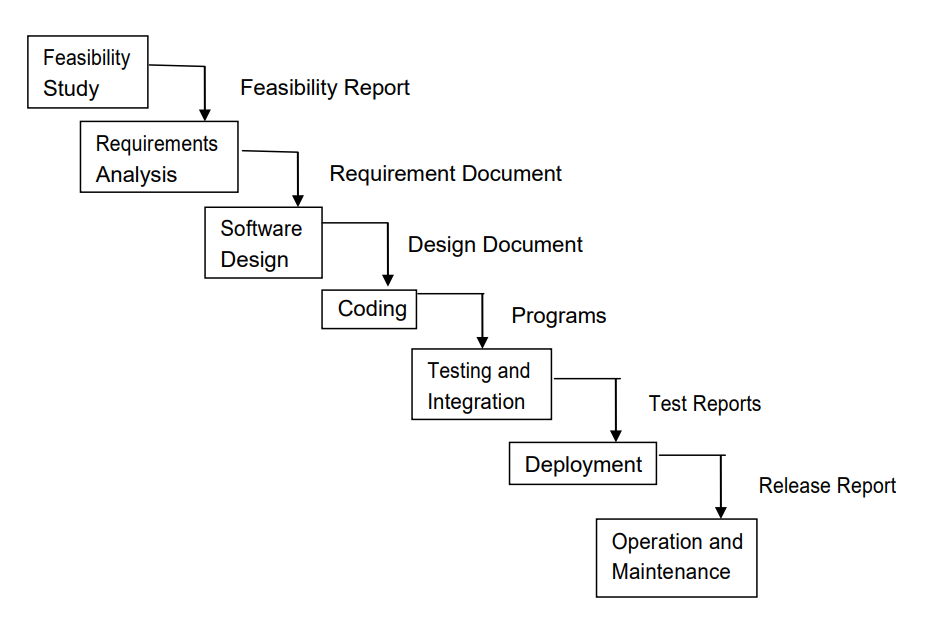
Deployment: Making software available for operational use. Includes delivery of software to the client, installation at client site, training to use the software, providing user manuals & documentation to client.

Maintenance: To adapt with the changing environment, new features added, bugs removed, improving performance, preventing problems in future.

**Different SDLC Models with their advantages & disadvantages**

1. Waterfall Model
2. Iterative Waterfall Model
3. Incremental Model
4. Spiral Model
5. Evolutionary Model
6. Prototyping Model
7. Rapid Application Development (RAD) Model

**Waterfall Model**

****

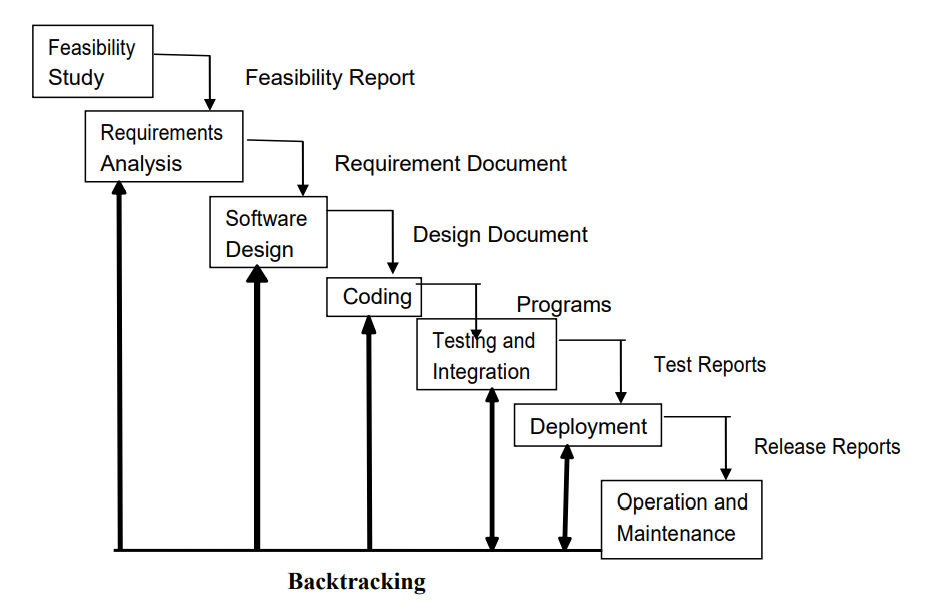
Advantages:

* Simple & Easy to understand and use.
* Works well with small projects where requirements are very well understood.
* Number of resources required is minimal.
* Sequential development so less chance to rework.
* Document driven process.

Disadvantages:

* Assumes requirements will not change during project.
* Once application is in testing, it is difficult to go back and change that is not well constructed.
* Very difficult to estimate time and cost.
* Less effective if all requirements are not known at beginning.

**Iterative Waterfall Model**

****

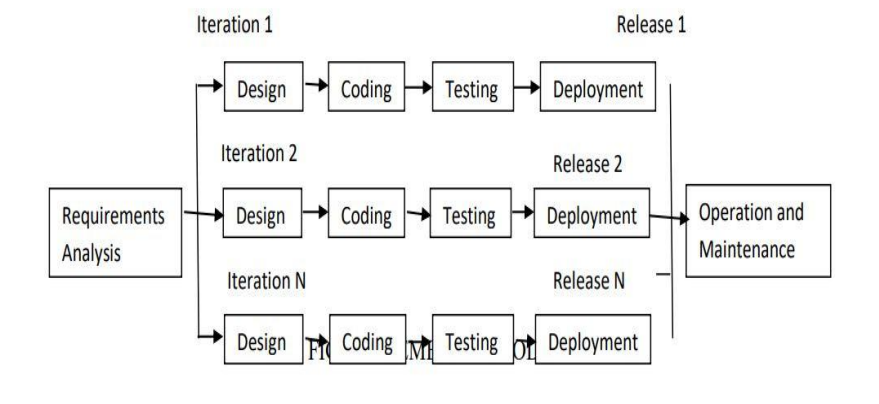
Advantages:

* Track defects at early stages.
* Reliable user feedback.
* Less time on documenting and more time to designing.
* Output is generated after each stage therefore it has high visibility.

Disadvantages:

* Each phase is rigid with no overlaps.
* Costly system architecture or design issues may arise because not all requirements are gathered up front for entire lifecycle.
* Real projects rarely follow sequential flow and iterations are handled indirectly. These changes can cause confusion as the project proceeds.
* Often difficult to get customer requirements explicitly.

**Incremental Model**

****

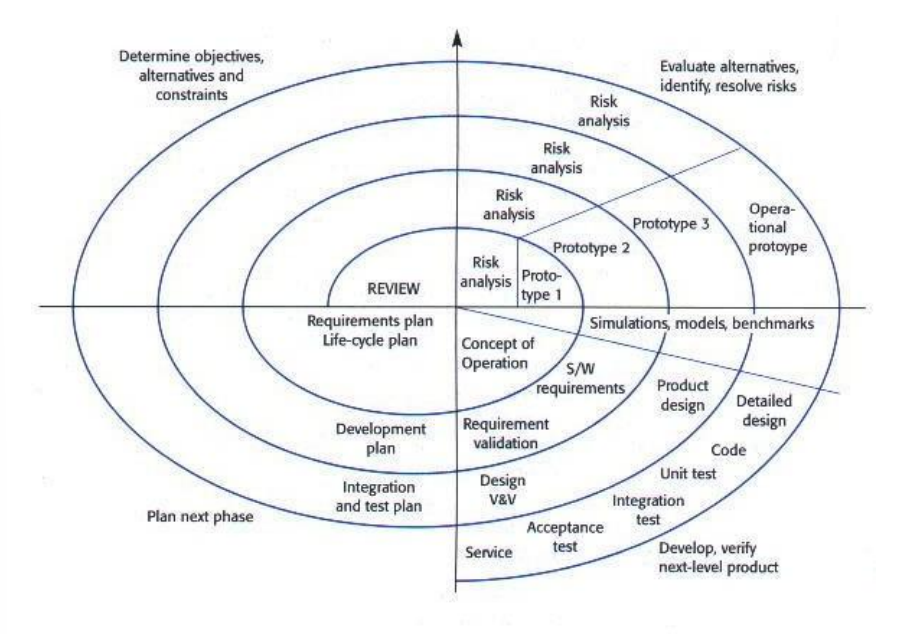
Advantages:

* Generates working software quickly and early during software life cycle.
* This model is more flexible-less costly to change scope and requirement.
* Easier to test and debug during a smaller interaction.
* In this model customer can respond to each built.
* Lowers initial delivery cost.
* Easier to manage risk because risky pieces are identified and handled during the iterations.

Disadvantages:

* Needs good planning and design.
* Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
* Total cost is higher than waterfall.

**Spiral Model**

****

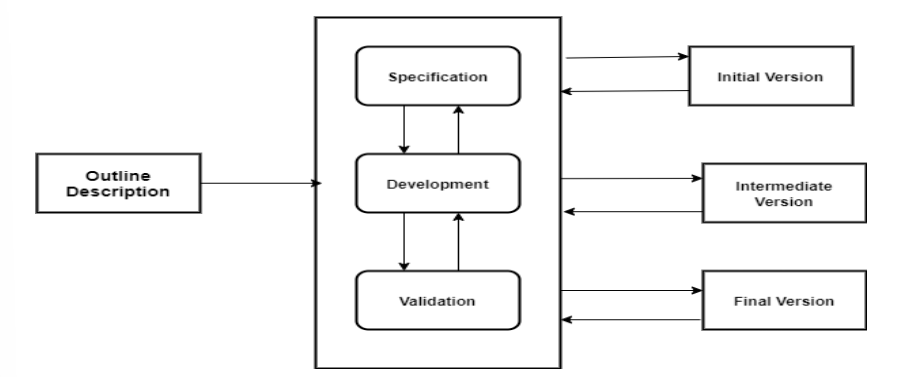
Advantages:

* High amount of risk analysis hence, avoidance of Risk is enhanced.
* Good for large and mission critical projects.
* Strong approval and documentation control.
* Project estimates in terms of schedule, cost etc become more and more realistic as the project moves forward and loops in spiral get completed.
* Additional Functionality can be added at later date.

Disadvantages:

* Can be costly model to use.
* Risk analysis require highly specific expertise. Success is dependent on risk analysis phase.
* Doesn’t work well for smaller projects.
* Not suitable for low-risk projects.
* Spiral may continue indefinitely.

**Evolutionary Model**

****

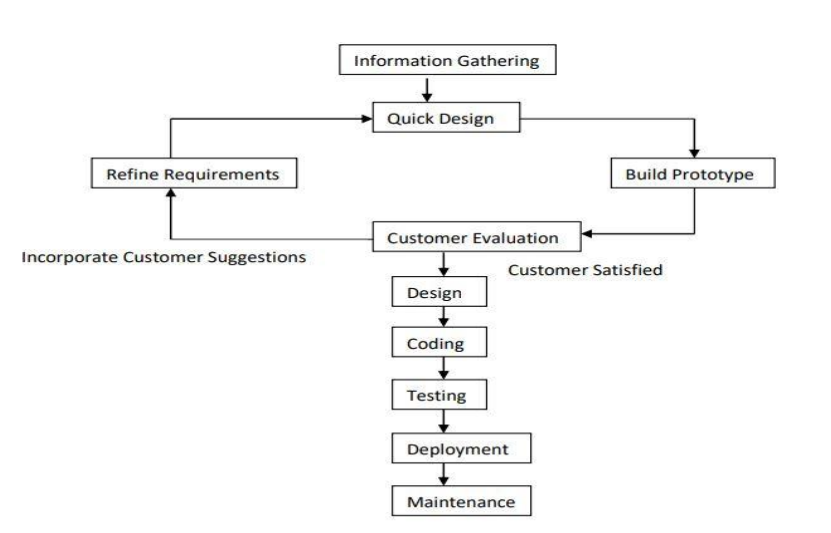
Advantages:

* Risk analysis is better.
* It supports changing environment.
* Initially operating time is less.
* Better suited for large mission-critical projects.
* During lifecycle software is produced early which facilitates customer evaluation feedback.

Disadvantages:

* Management complexity is more.
* Not suitable for smaller projects.
* Can be costly to use.
* Highly skilled resources are required for risk analysis.

**Prototyping Model**

****

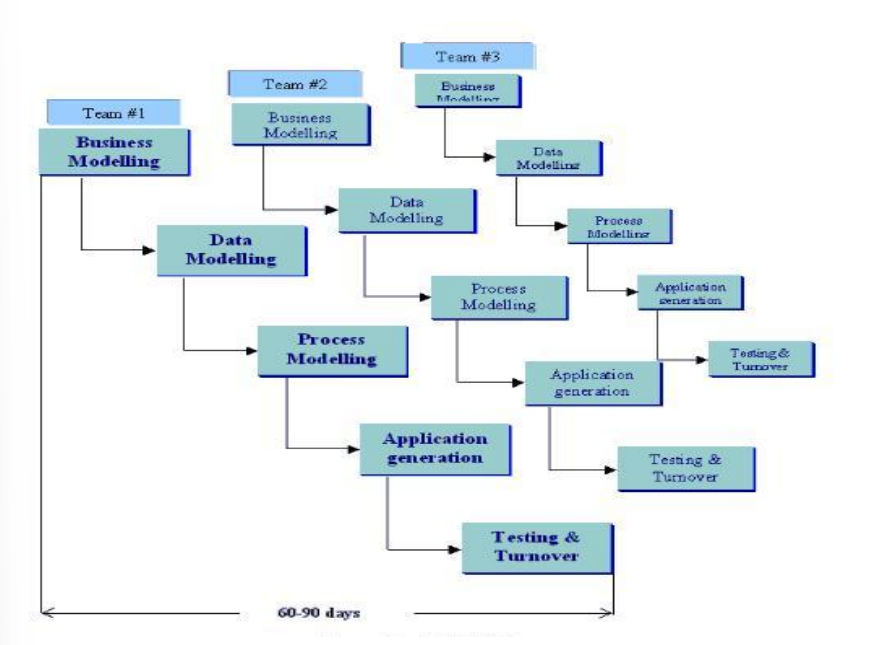
Advantages:

* It minimalizes the change requests from the customer side and the associated redesign and redevelopment costs.
* The overall development cost might turn out to be lower than equivalent Software development using Waterfall Model.
* Using prototype mode, customers can get a feeling of the prototype version of the final product very early.

Disadvantages:

* This model requires exclusive involvement of the customer, this is not always possible.
* Sometimes bad design decisions during prototype development may propagate to the real product.
* Software development in this way might include extra cost for prototype development.

**Rapid Application Development (RAD) Model**

****

Advantages:

* It reduces development time.
* Increases reusability of components.
* Quick initial reviews occur.
* Encourages customer feedback.
* Integration from very beginning solves a lot of integration issues.

Disadvantages:

* Depends on strong team and individual performance for identifying business requirements.
* Only system that can be modularized can be built using RAD.
* Requires highly skilled developers/designers.
* High dependency on modelling skills.
* Inapplicable to cheaper projects as cost of modelling and automated code generation is high.