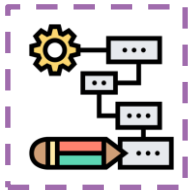
SDLC Models

Waterfall Model

Iterative Model

Sprial Model

RAD Model



Prototype Model

Agile Model

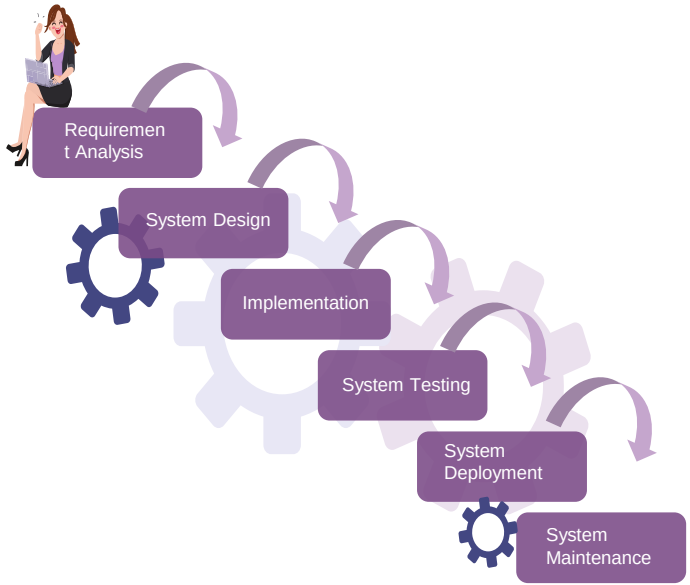
Waterfall Model (SDLC)

Waterfall Model: The earliest and most popular structured SDLC model.

The software development process is broken down into various sequential phases.

Each phase has its own set of activities and goals.

One phase's output becomes the next phase's input.



Pros

1. straightforward and simple to use
2. Each phase is processed and finished separately
3. Works effectively for small projects

Cons

1. Not suitable for intricate and object-oriented projects
2. unable to adapt to changing requirements
3. Risk and uncertainty are very high.

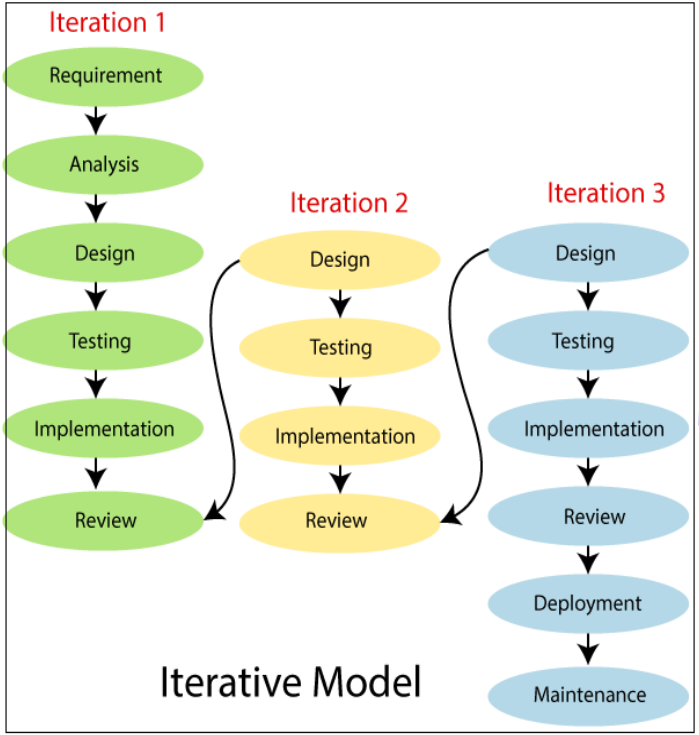
Iterative Model (SDLC)

Development begins with specific set of requirements that iteratively improves and develops the software.

The total software development is divided into iterations and each iteration has design, development and review.

After each iteration, this procedure is repeated to create a new version of the software.

The planning and requirement analysis, deployment and maintenance are one time ,and doesn't involve in iterations.



Pros

1. Parallel development can be planned.
2. Results are obtained early and periodically
3. Less costly to change the scope/requirements.

Cons

1. Management complexity is more.
2. More resources may be required
3. Not suitable for smaller projects.

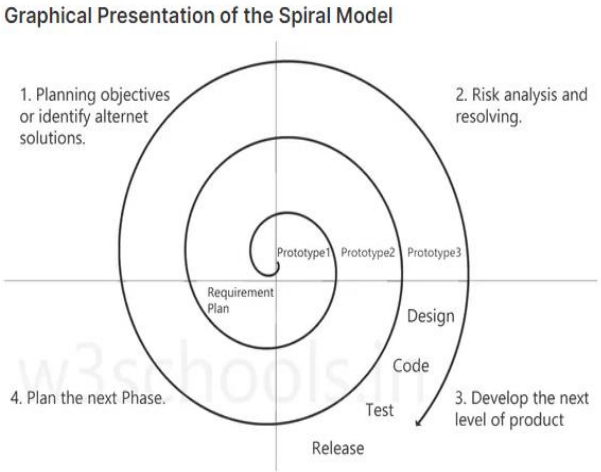
Spiral Model (SDLC)

Spiral Model: Couples the idea of iterative development with the systematic, controlled aspects of the waterfall model.

The software is created through a series of incremental releases using the spiral methodology.

The additional release during the initial rounds could be a paper model or prototype.

Later iterations result in ever-more-complete versions of the engineered system.



Pros

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

Cons

1. Can be a costly model to use.
2. Risk analysis needed highly particular expertise
3. Doesn't work well for smaller projects.

Prototype Model (SDLC)

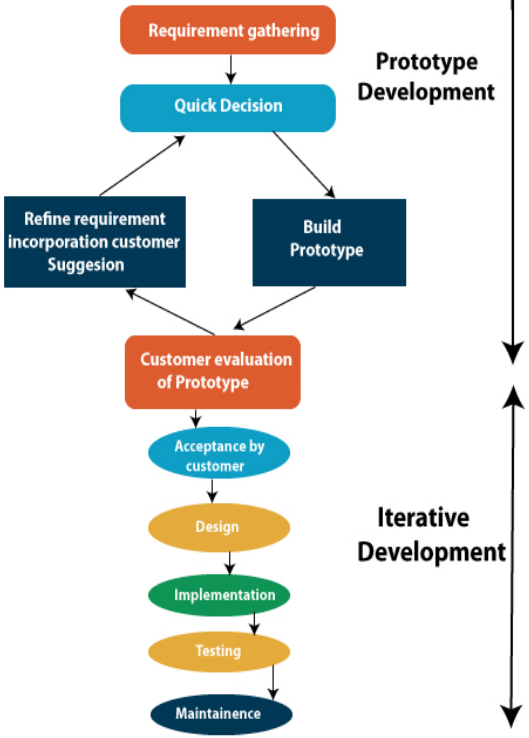
A prototype is created, tested, and modified necessarily so that the full system or product can be developed.

This model performs best in situations where not all of the project needs are known in detail beforehand.

A functioning system prototype must be created before the actual software is developed.

A system prototype is a toy version of the actual system.

Fig: Prototype Model



Pros

1. Increased user involvement in the product even before its implementation.
2. Reduces time and cost
3. Missing functionality can be identified easily.

Cons

1. Users may get confused in the prototypes and actual systems.
2. Developers may try to reuse the existing prototypes to build the actual system

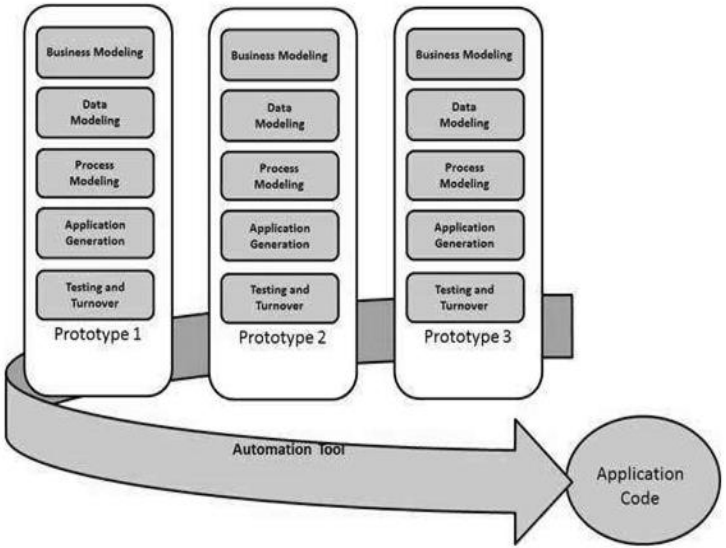
RAD (Rapid Application Development) Model (SDLC)

RAD model is based on prototyping and iterative development with no specific planning involved.

Functional modules are developed concurrently and then combined to create the finished product

It is simpler to accommodate modifications throughout the development.

The main features of the RAD model are that it emphasizes the reuse of templates, tools, processes, and code.



Pros

1. Progress can be measured.
2. It reduced development time.
3. It increases the reusability of features.

Cons

1. Only system that can be modularized can be built
2. High dependency on Modeling skills.
3. Suitable for projects requiring shorter development times.

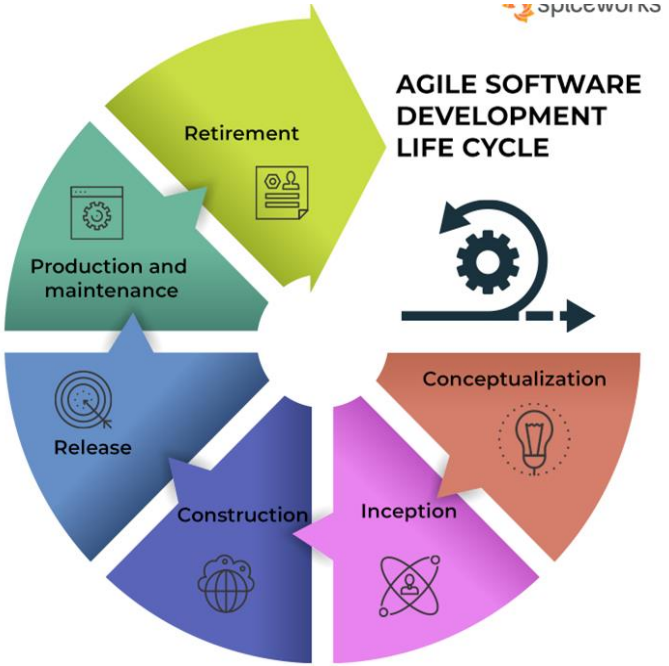
Agile Model (SDLC)

A method of software development based on iterative development.

Agile methodologies minimize long-term planning by breaking down projects into smaller iterations or parts.

At the beginning of the development process, the project's requirements and scope are established.

Each iteration's duration, length, and scope are all precisely predetermined in advance.



Pros

1. Satisfied customers
2. Higher quality product
3. Easily and Quickly adapt to change
4. Predictable Delivery Dates

Cons

1. Short term planning
2. Lack of necessary documentation
3. Cost is less predictable
4. Projects easily fall off track


Thanks