**SDLC**

Software Development Life Cycle is an integrated sequence of steps in software engineering to develop a software product. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.



**Phases in an SDLC**

**Requirement Analysis**

In this phase, we are to gather requirements and to analyse the possibility of creating the ideas.

**Design**

In this phase, the system analyst and designer come together to create the layout of the system.

**Testing**

In this phase, the system would undergo an automated testing tool to check that every part of the system is working.

**Implementation**

In this phase, the developers will do a final run of the system and check for any flaws that are in the system before launching it.

**Evolution**

In this phase, the system is consider to be working and is ready to loaded to other devices to be used

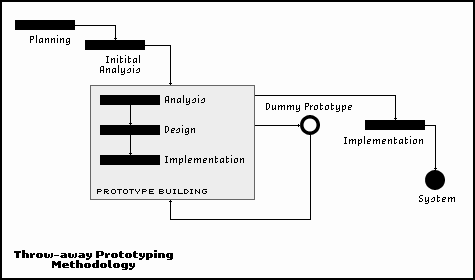
**Software development models in SDLC**

**Prototyping Method**

In Prototyping, creation of prototypes of the software application is done. In doing this step the software developers will be able to visualize some of the component of the software to constrain the gap of misunderstanding of the customer requirements. This additionally will reduce the iterations that may occur in waterfall approach and hard to be implemented due to inflexibility of the waterfall approach. So, when the final prototype is developed, the requisite is considered to be frozen.

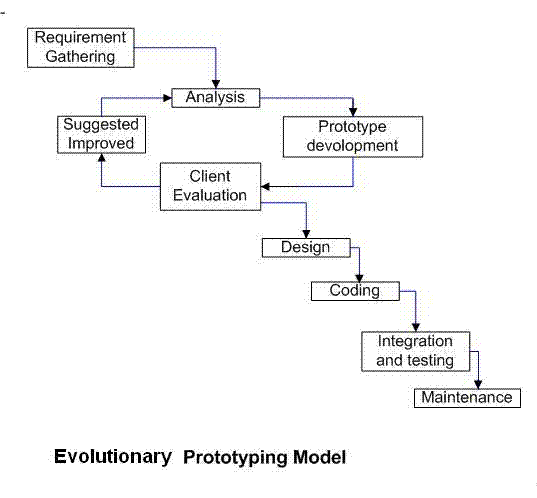
Types of Prototyping:

**Throwaway prototyping**



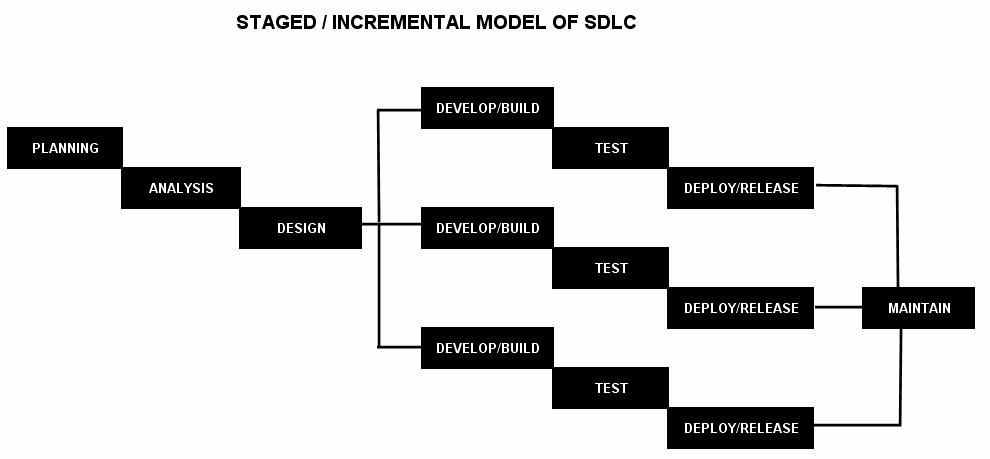
Prototypes that are eventually discarded rather than becoming a component of the conclusively distributed software.

**Evolutionary prototyping**



Prototypes that evolve into the final system through iterative incorporation of utilizer feedback.

**Incremental prototyping**



The final product is built as discrete prototypes. At the terminus the separate prototypes are merged in an overall design.

**Extreme prototyping**

Utilized at web applications mainly. Fundamentally, it breaks down web development into three phases, each one predicted on the preceding one. The first phase is a static prototype that consists mainly of HTML pages. In the second phase, the screens are programmed and plenary functional utilizing a simulated accommodations layer. In the third phase the accommodations are implemented.

Advantages

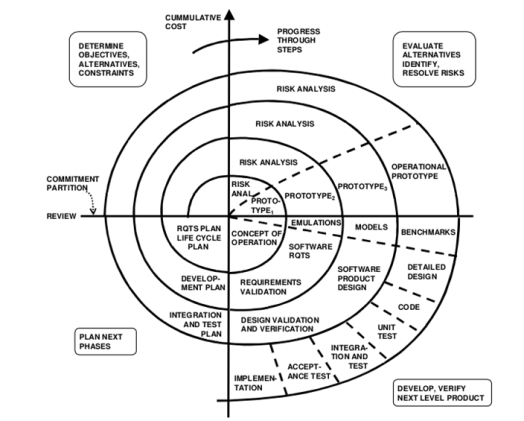
* Reduced time and costs, but can be disadvantage if developer loses time in developing the prototypes. Improved and increased user involvement.

Disadvantages

* Insufficient analysis
  + User confusion of prototype and finished system
* Development misunderstanding of user objectives
  + Excessive development time of the prototype
* Expense of implementing prototyping

**Spiral Method**

This model of development combines the features of the prototyping and waterfall model. The spiral model is used for large, expensive and complicated projects. This model adapts many of the same steps as the waterfall model, in the same order, only separated by planning, risk assessment and building of prototypes and simulators.



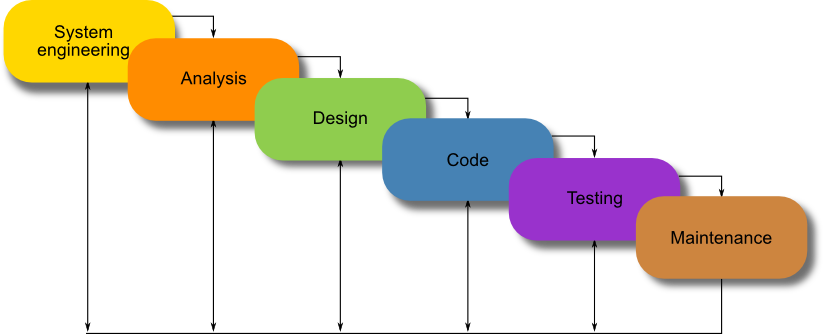
Advantages

* Budgets and Schedules are more realistic as the work progresses as issues are found early. Manages risks and develops system into phases

Disadvantages

* Expensive and timely to reach the targeted product. Skills needed for evaluation of risks and assumptions. Highly customized limiting re-usability

**Waterfall Model**



Waterfall model is basically used for the project which is small and there are no uncertain requirements. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model software testing starts only after the development is complete. In waterfall model phases do not overlap.

Advantages

* This model is simple and easy to understand and use.
* It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
* In this model phases are processed and completed one at a time. Phases do not overlap.
* Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages

* Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing.