Software and Software Engineering

Software development life cycle (SDLC) models

The life cycle of software represents the series of identifiable stages through which it evolves during its lifetime.

previous activities when graphically modeled and textually described are called as software development life cycle (SDLC) model.

All software process models can accommodate the generic framework activities, but each applies a different emphasis to these activities.

- **The Waterfall Model**

The waterfall model, sometimes called the linear sequential model, is the oldest paradigm for software engineering that suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment. It is used when requirements are well-defined and reasonably stable.

Although the model is simple, it has some disadvantages. Real projects rarely follow the sequential flow that the model proposes, also it is difficult for the customer to state all requirements clearly at the beginning of the project, and finally, defects may not be detected until the working program is reviewed.

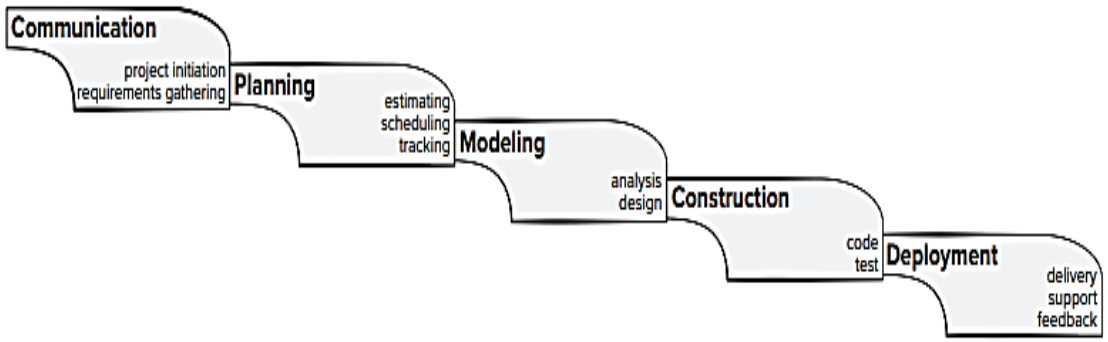

Figure 3.1: Waterfall Model.

the iterative waterfall model can be thought of as incorporating the necessary changes to the classical waterfall model to make it usable in software development projects.

The feedback paths allow for correcting errors during some phase, as and when these are detected in a later phase. For example, if during the testing phase a design error is identified, then the feedback path allows the design to be reworked and the changes to be reflected in the design documents and all other subsequent documents.

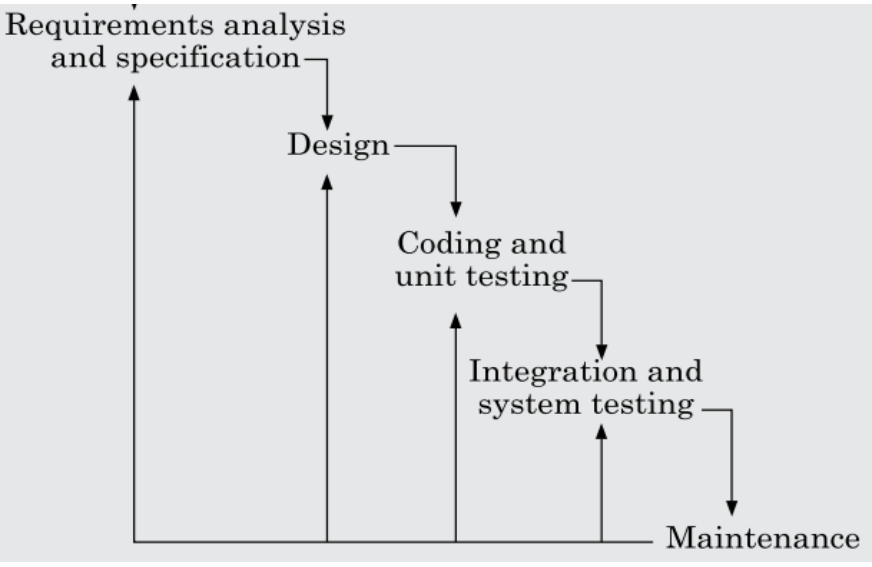

Figure 3.2: Iterative Waterfall Model.

- **Prototyping Process Model**

the prototyping paradigm assists in a better understanding of what is to be built when requirements are fuzzy.

The prototyping paradigm begins with communication. stakeholders are met to define the overall objectives for the software. A prototyping iteration is planned quickly, and modeling (in the form of a "quick design") occurs. The quick design leads to the construction of a prototype. The prototype is deployed and evaluated by stakeholders, who provide feedback that is used to further refine requirements. Iteration occurs as the prototype is tuned to satisfy the needs of stakeholders.

The prototype model is especially useful in developing the graphical user interface (GUI) part of a system.

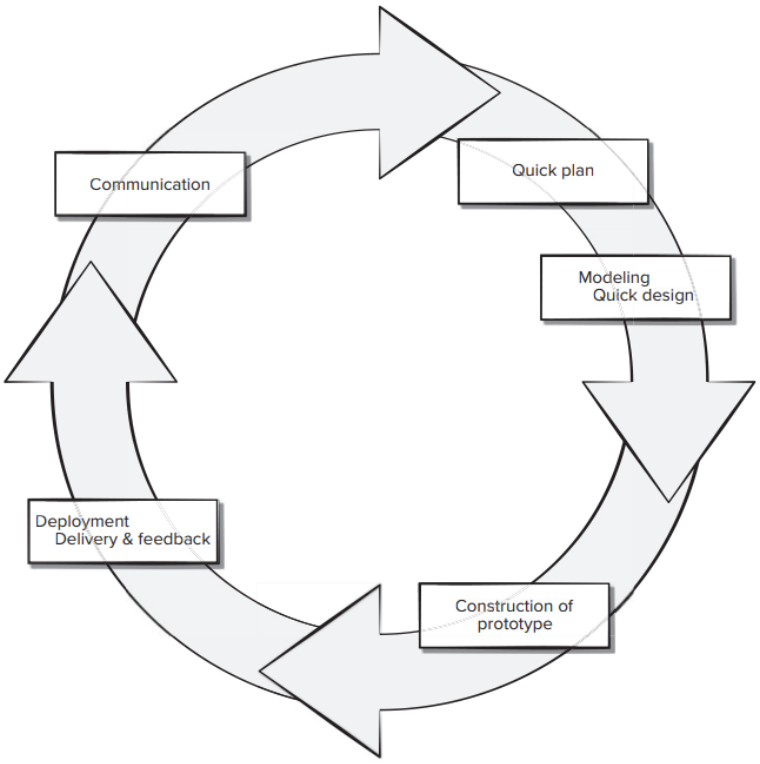

Figure 3.3: Prototyping Process Model

- **Incremental Development Model**

In the incremental life cycle model, the software is developed in increments. At any time, a plan is made only for the next increment, and no long-term plans are made.

The development team first develops the core features of the system. The core or basic features are those that do not need to invoke any services from the other features. On the other hand, non-core features need services from the core features. Once the initial core features are developed, new functionalities are added in the next increment.

After the requirements gathering and specification, the requirements are split into several increments. Starting with the core (increment 1), and ending with (increment n) which delivers fully developed software.

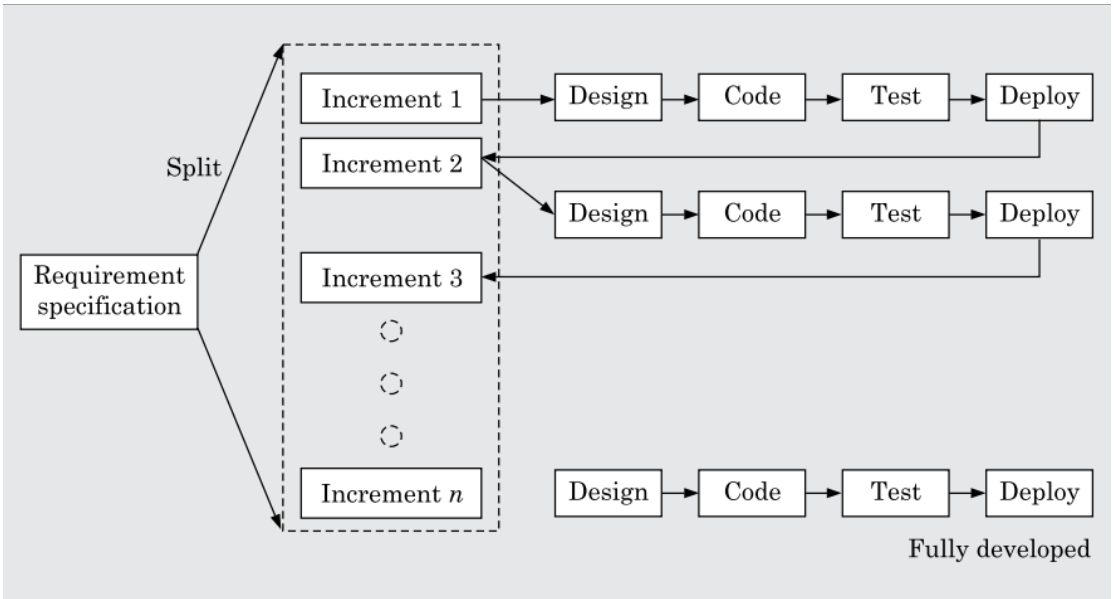

Figure 3.4: Incremental Development Model

this model offers error reduction, the core modules are used by the customer from the beginning and therefore these get tested thoroughly.

This reduces the chances of errors and leads to more reliability of the software.

- **Spiral Model**

The spiral model is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model. During early iterations, the release might be a model or prototype. During later iterations, more complete versions are produced.
A spiral model is divided into a set of framework activities. Each of the framework activities represents one segment of the spiral path illustrated in Figure 3.5. As this evolutionary process begins, the software team performs activities by a circuit around the spiral in a clockwise direction, beginning at the center.

Risk is considered as each evolution is made. It may be difficult to convince customers that the evolutionary approach is controllable. It demands risk assessment expertise for success. If a major risk is not uncovered and managed, problems will occur.

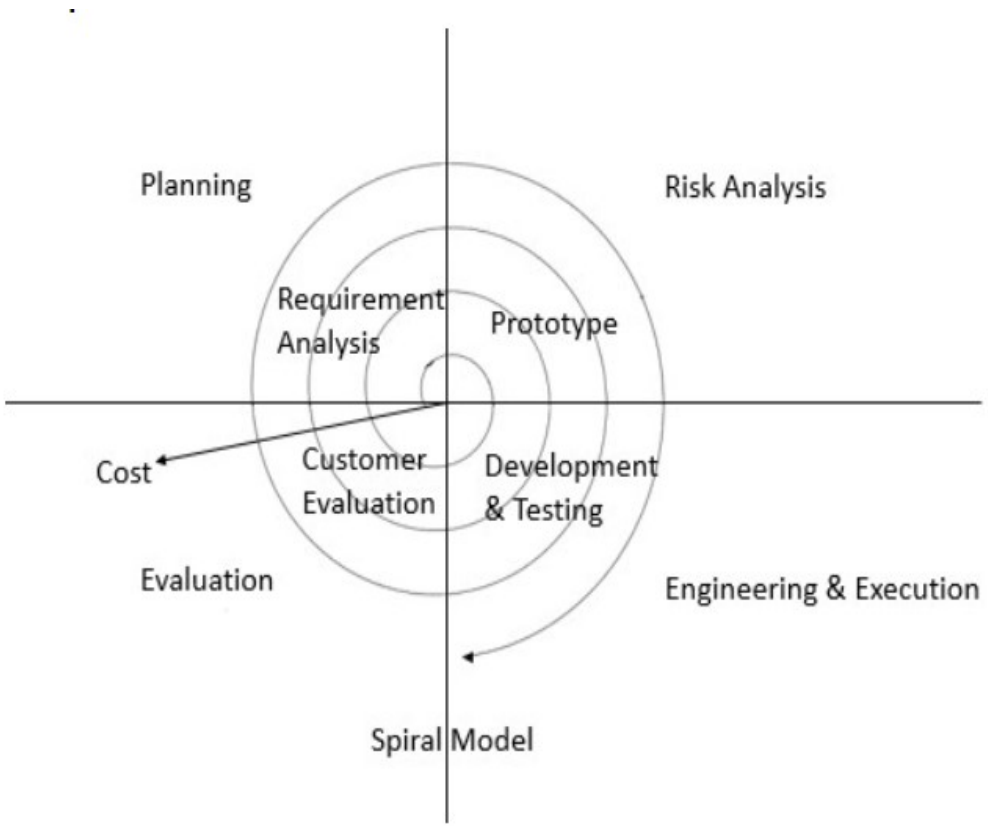

Figure 3.5: Spiral Model