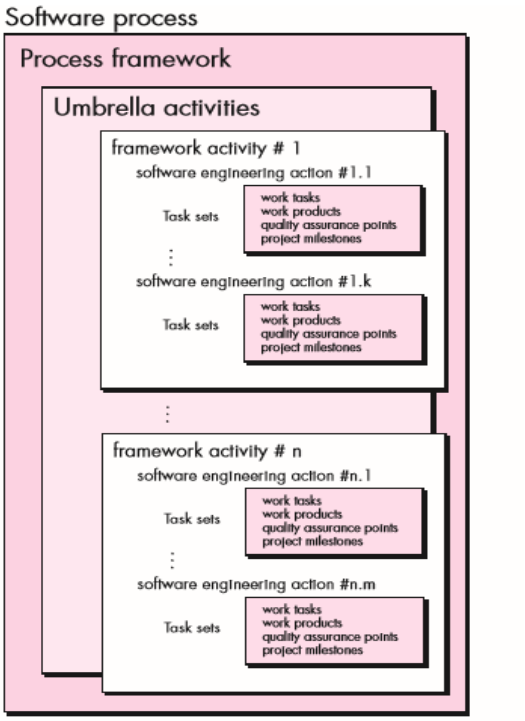
Chapter-2

A GENERIC PROCESS MODEL

A process was defined as a collection of work activities, actions, and tasks that are performed when some work product is to be created. Each of these activities, actions, and tasks reside within a framework or model that defines their relationship with the process and with one another.

The software process is represented schematically in Figure 2.1. Referring to the figure, each framework activity is populated by a set of software engineering actions. Each software engineering action is defined by a task set that identifies the work tasks that are to be completed, the work products that will be produced, the quality assurance points that will be required, and the milestones that will be used to indicate progress.

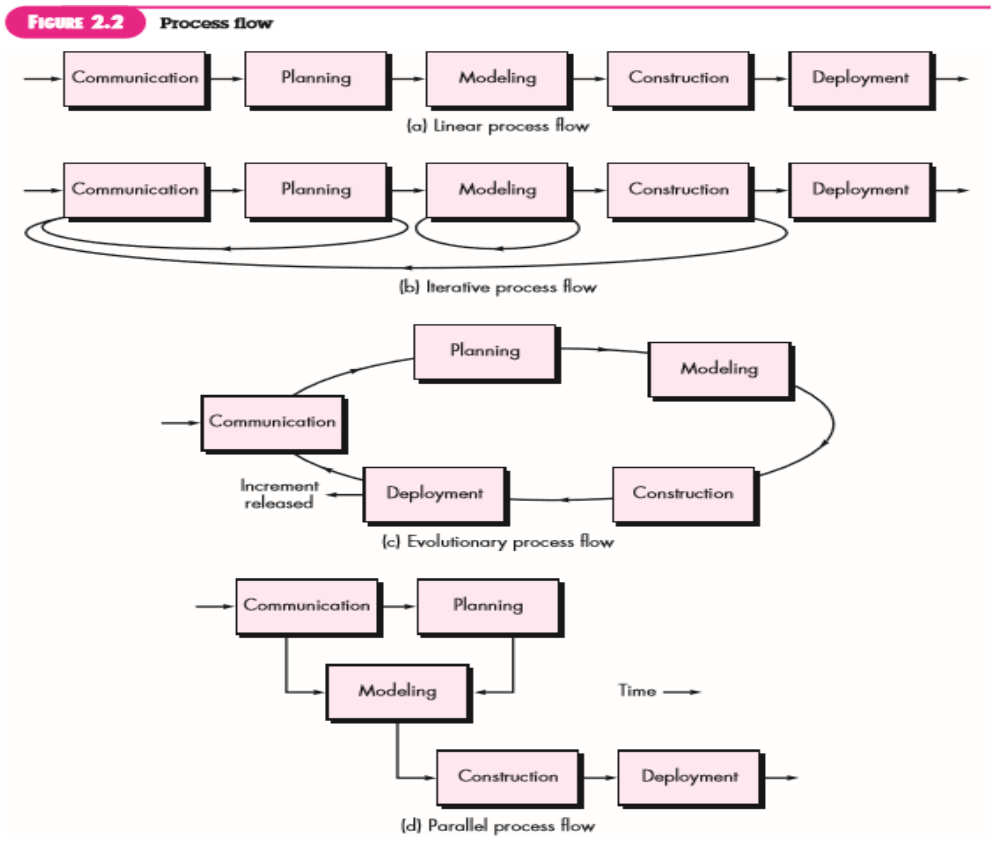
FIGURE 2.1
A software process framework



A generic process framework for software engineering defines five framework activities—communication, planning, modeling, construction, and deployment. In addition, a set of umbrella activities—project tracking and control, risk management, quality assurance, configuration management, technical reviews, and others—are applied throughout the process.

You should note that one important aspect of the software process has not yet been discussed. This aspect—called process flow—describes how the framework activities and the actions and tasks that occur within each framework activity are organized with respect to sequence and time and is illustrated in Figure 2.2.

- A linear process flow executes each of the five framework activities in sequence, beginning with communication and culminating with deployment (Figure 2.2a).
- An iterative process flow repeats one or more of the activities before proceeding to the next (Figure 2.2b).
- An evolutionary process flow executes the activities in a "circular" manner. (Figure 2.2c).
- A parallel process flow (Figure 2.2d) executes one or more activities in parallel with other activities.



Identifying a Task Set

You should choose a task set that best accommodates the needs of the project and the characteristics of your team. A task set defines the actual work to be done to accomplish the objectives of a software engineering action.

- A list of the task to be accomplished
- A list of the work products to be produced
- A list of the quality assurance filters to be applied

Process Pattern

A process pattern describes a process-related problem that is encountered during software engineering work, identifies the environment in which the problem has been encountered, and suggests one or more proven solutions to the problem.

In more general terms, a process pattern provides you with a template a consistent method for describing problem solutions within the context of the software process. By combining patterns, a software team can solve problems and construct a process that best meets the needs of a project.

Process pattern types-

- *Stage patterns* — defines a problem associated with a framework activity for the process.
- *Task patterns* — defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice
- *Phase patterns* — define the sequence of framework activities that occur with the process, even when the overall flow of activities is iterative in nature.

PROCESS ASSESMENT AND IMPROVEMENT

A number of different approaches to software process assessment and improvement have been proposed over the past few decades:

- ***Standard CMMI Assessment Method for Process Improvement (SCAMPI)*** — provides a five step process assessment model that incorporates five phases: initiating, diagnosing, establishing, acting and learning.

- ***CMM-Based Appraisal for Internal Process Improvement (CBA IPI)—***provides a diagnostic technique for assessing the relative maturity of a software organization; uses the SEI CMM as the basis for the assessment.

- ***SPICE—The SPICE (ISO/IEC15504)*** standard defines a set of requirements for software process assessment. The intent of the standard is to assist organizations in developing an objective evaluation of the efficacy of any defined software process.

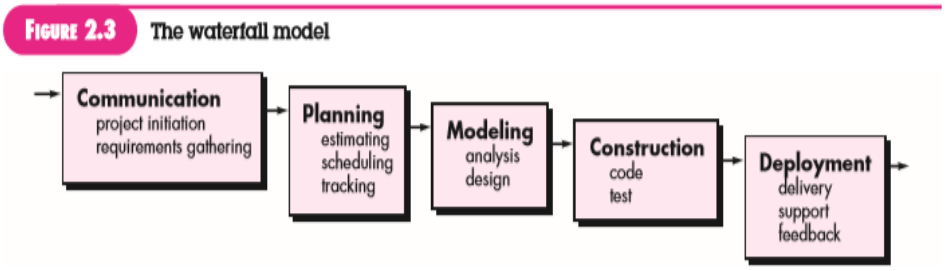
- ***ISO 9001:2000 for Software***—a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides. Therefore, the standard is directly applicable to software organizations and companies.

PRESCRIPTIVE PROCESS MODELS

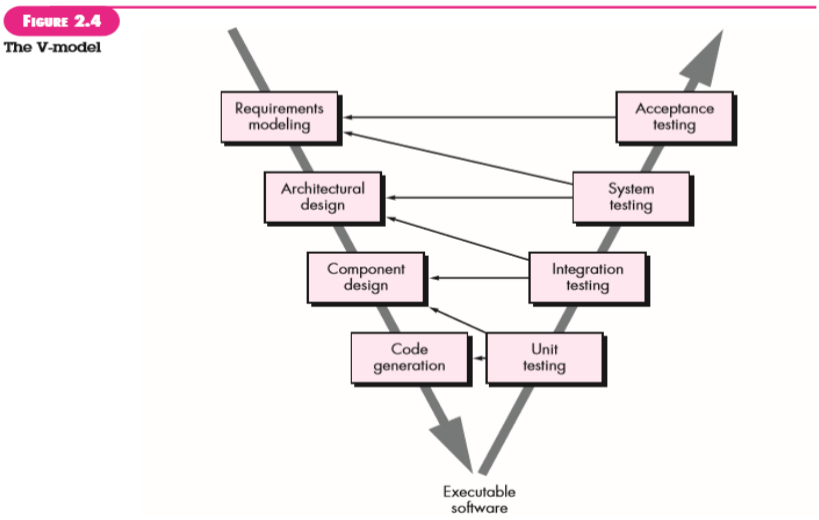
1) The Waterfall Model
2) Incremental Process Models
3) Evolutionary Process Models
4) Concurrent Models

1) The Waterfall Model

The waterfall model, sometimes called the classic life cycle, suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment.



A variation in the representation of the waterfall model is called the V-model. Represented in Figure 2.4. As a software team moves down the left side of the V, basic problem requirements are refined into progressively more detailed and technical representations of the problem and its solution. Once code has been generated, the team moves up the right side of the V, essentially performing a series of tests (quality assurance actions) that validate each of the models created as the team moved down the left side.



Advantages of waterfall model-

- This model works for small projects because the requirements are understood very well.
- The waterfall model is simple and easy to understand, implement, and use.
- All the requirements are known at the beginning of the project, hence it is easy to manage.

Disadvantages of the waterfall model

- The problems with this model are uncovered, until the software testing.
- The amount of risk is high.
- This model is not good for complex and object oriented projects.

Incremental Process Models

The incremental model combines elements of linear and parallel process flows Referring to Figure 2.5, the incremental model applies linear sequences in a staggered fashion as calendar time progresses. Each linear sequence produces deliverable "increments" of the software.

When an incremental model is used, the first increment is often a core product. That is, basic requirements are addressed but many supplementary features (some known, others unknown) remain undelivered. The core product is used by the customer (or undergoes detailed evaluation)

As a result of use and/or evaluation, a plan is developed for the next increment. The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced.

The incremental process model focuses on the delivery of an operational product with each increment. Early increments are stripped-down versions of the final product, but they do provide capability that serves the user and also provide a platform for evaluation by the user.

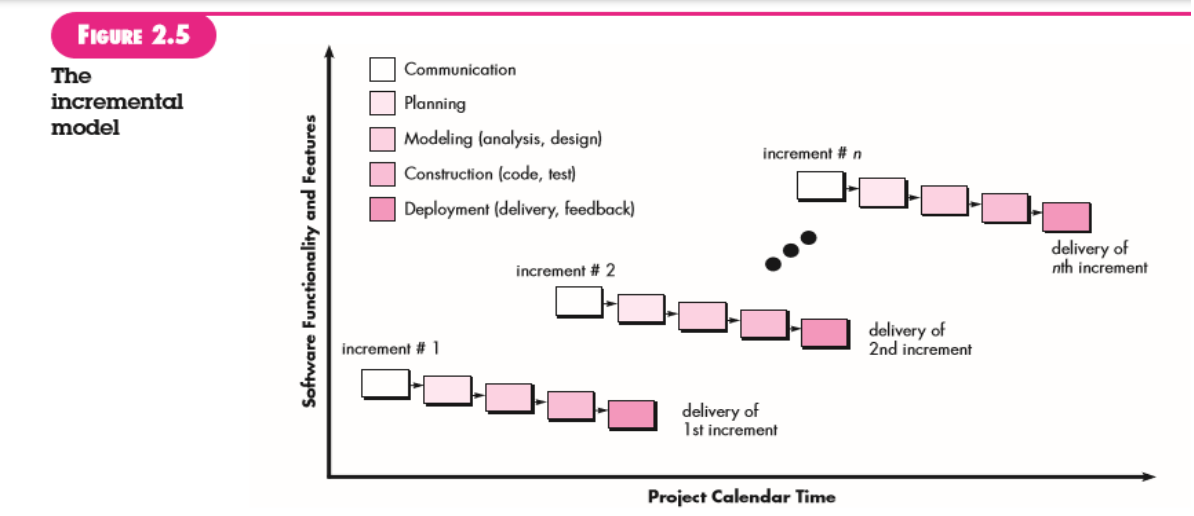


FIGURE 2.5 The incremental model

Advantages of incremental model

- This model is flexible because the cost of development is low and initial product delivery is faster.
- It is easier to test and debug during the smaller iteration.

- The working software generates quickly and early during the software life cycle.
- The customers can respond to its functionalities after every increment.

Disadvantages of the incremental model

- The cost of the final product may cross the cost estimated initially.
- This model requires a very clear and complete planning.
- The planning of design is required before the whole system is broken into small increments.
- The demands of customer for the additional functionalities after every increment causes problem during the system architecture.

Evolutionary Process Models

- Evolutionary models are iterative type models.They allow to develop more complete versions of the software.

Following are the evolutionary process models.

1. The prototyping model
2. The spiral model
3. Concurrent development model

1. The Prototyping model

- Prototype is defined as first or preliminary form using which other forms are copied or derived.
- Prototype model is a set of general objectives for software.
- It does not identify the requirements like detailed input, output.
- It is software working model of limited functionality.
- In this model, working programs are quickly produced.

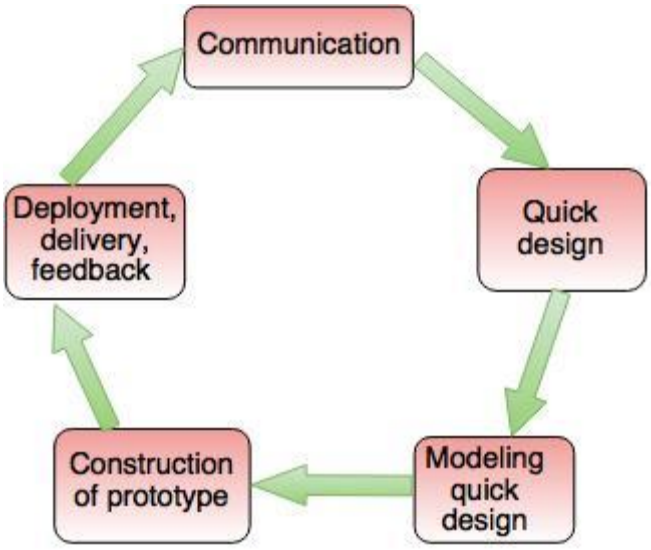


Fig. - The Prototyping Model

The different phases of Prototyping model are:

1. Communication

In this phase, developer and customer meet and discuss the overall objectives of the software.

2. Quick design

- Quick design is implemented when requirements are known.
- It includes only the important aspects like input and output format of the software.
- It focuses on those aspects which are visible to the user rather than the detailed plan.
- It helps to construct a prototype.

3. Modeling quick design

- This phase gives the clear idea about the development of software because the software is now built.
- It allows the developer to better understand the exact requirements.

4. Construction of prototype

The prototype is evaluated by the customer itself.

5. Deployment, delivery, feedback

- If the user is not satisfied with current prototype then it refines according to the requirements of the user.

- The process of refining the prototype is repeated until all the requirements of users are met.
- When the users are satisfied with the developed prototype then the system is developed on the basis of final prototype.

Advantages of Prototyping Model

- Prototype model need not know the detailed input, output, processes, adaptability of operating system and full machine interaction.
- In the development process of this model users are actively involved.
- The development process is the best platform to understand the system by the user.
- Errors are detected much earlier.
- Gives quick user feedback for better solutions.
- It identifies the missing functionality easily. It also identifies the confusing or difficult functions.

Disadvantages of Prototyping Model:

- The client involvement is more and it is not always considered by the developer.
- It is a slow process because it takes more time for development.
- Many changes can disturb the rhythm of the development team.
- It is a thrown away prototype when the users are confused with it.

2. The Spiral model

- Spiral model is a risk driven process model.
- It is used for generating the software projects.
- In spiral model, an alternate solution is provided if the risk is found in the risk analysis, then alternate solutions are suggested and implemented.
- It is a combination of prototype and sequential model or waterfall model.
- In one iteration all activities are done, for large project's the output is small.

The framework activities of the spiral model are as shown in the following figure.

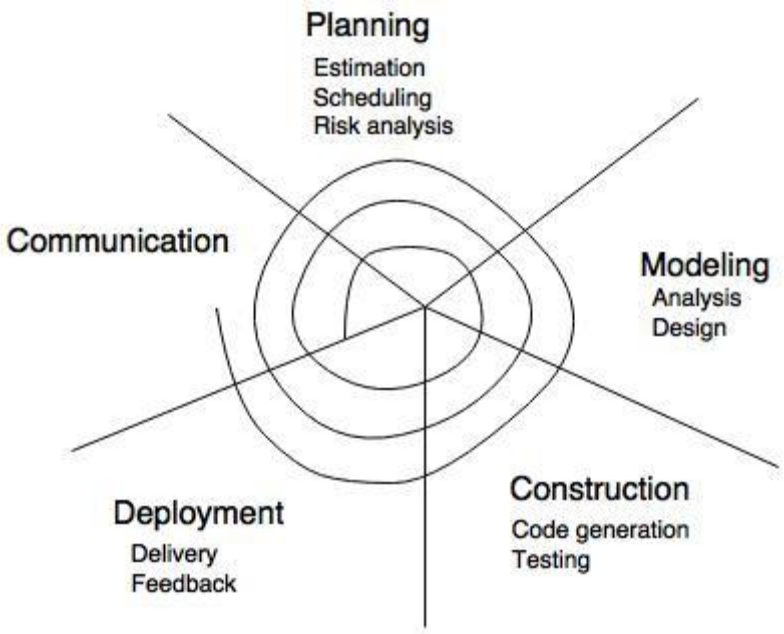


Fig. - The Spiral Model

NOTE: The description of the phases of the spiral model is same as that of the process model.

Advantages of Spiral Model

- It reduces high amount of risk.
- It is good for large and critical projects.
- It gives strong approval and documentation control.
- In spiral model, the software is produced early in the life cycle process.

Disadvantages of Spiral Model

- It can be costly to develop a software model.
- It is not used for small projects.

3. The concurrent development model

- The concurrent development model is called as concurrent model.
- The communication activity has completed in the first iteration and exits in the awaiting changes state.
- The modeling activity completed its initial communication and then go to the underdevelopment state.
- If the customer specifies the change in the requirement, then the modeling activity moves from the under development state into the awaiting change state.
- The concurrent process model activities moving from one state to another state.

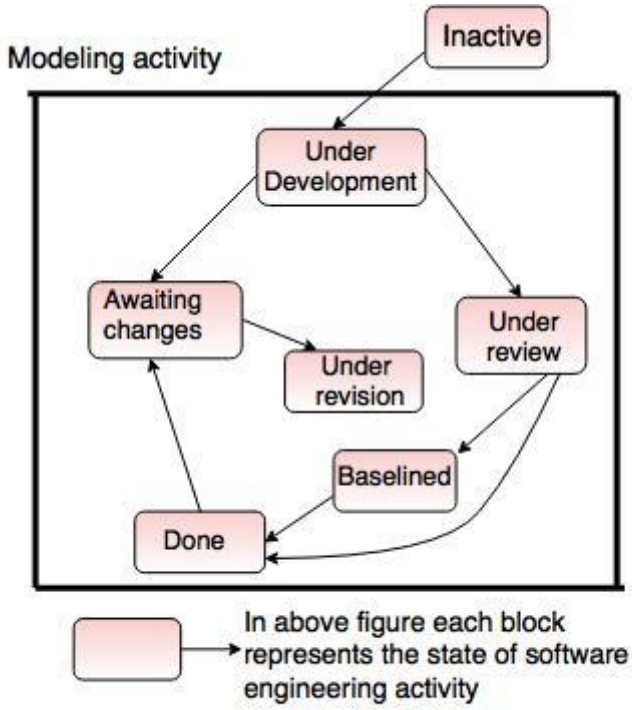


Fig. - One element of the concurrent process model

Advantages of the concurrent development model

- This model is applicable to all types of software development processes.
- It is easy for understanding and use.
- It gives immediate feedback from testing.
- It provides an accurate picture of the current state of a project.

Disadvantages of the concurrent development model

- It needs better communication between the team members. This may not be achieved all the time.
- It requires to remember the status of the different activities.

PERSONAL SOFTWARE PROCESS(PSP) AND TEAM SOFTWARE PROCESS(TSP)

Personal software process (psp)-

The Personal Software Process (PSP) model is good from the perspective that an individual software engineer can use it to improve his or her personal productivity and work product quality.

PSP process model defines five framework activities: planning, high-level design, high-level design review, development, and postmortem. It stresses the need to identify errors early and to understand the types of errors.

Planning: it isolates requests. And a project schedule is created.

High-level design: Prototypes are built when uncertainty exists.

High-level design review: Formal verification methods are applied to uncover errors in the design.

Development: Code is generated, reviewed, compiled, and tested.

Postmortem: using the measures and metrics collected, the effectiveness of the process is determined.

Team Software Process (TSP):

The goal of TSP is to build a "self-directed" project team that organizes itself to produce high-quality s/w.

Each project is "launched" using a "script" that defines the tasks to be accomplished.

Teams are self-directed.

Measurement is encouraged.Measures are analyzed with the intent of improving the team process.