SOFTWARE ENGINEERING

Process Models: A Generic Process Model like Waterfall Models, Agile Model etc. Process Assessment and Improvement, Prescriptive Process Models, Specialized Process Models, The Unified Process, Personal and Team Process Models, Process Technology, Product and Process.

PART-2: PROCESS MODELS

A GENERIC PROCESS MODEL

The software process is represented schematically. Framework activity is populated by a set of software engineering actions. The Generic Process Model (GPM) is a formal framework for process analysis. It uses concepts from Bunge's ontology and extends this ontology with process-related concepts. The GPM-based process analysis methods are language-independent, and can be used for process models at different modeling languages mapped to GPM.

The main concepts employed by GPM are domain, state, event, goal, transition, and law. In GPM an enacted process is represented as a sequence of state transitions in the process domain.

Generic process framework for software engineering defines five framework activities—

1. Communication:
The software development starts with the communication between customer and developer.
2. Planning:
It consists of complete estimation, scheduling for project development and tracking.
3. Modeling:

- Modeling consists of complete requirement analysis and the design of the project like algorithm, flowchart etc.
- The algorithm is the step-by-step solution of the problem and the flow chart shows a complete flow diagram of a program.

4. Construction:

- Construction consists of code generation and the testing part.
- Coding part implements the design details using an appropriate programming language.
- Testing is to check whether the flow of coding is correct or not.

5. Deployment:

- Deployment step consists of delivering the product to the customer and take feedback from them.

If the customer wants some corrections or demands for the additional capabilities, then the change is required for improvement in the quality of the software.

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 as follows-

PROCESS ASSESSMENT AND IMPROVEMENT

The existence of a software process is no guarantee that software will be delivered on time, that it will meet the customer's needs, or that it will exhibit the technical characteristics that will lead to long-term quality characteristics. In addition, the process itself should be assessed to be essential to ensure that it meets a set of basic process criteria that have been shown to be essential for a successful software engineering.

A number of different approaches to software process assessment and improvement have been proposed.

- **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. The SCAMPI method uses the SEI CMMI as the basis for assessment.
- **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 (ISO/IEC15504)—**a standard that 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.

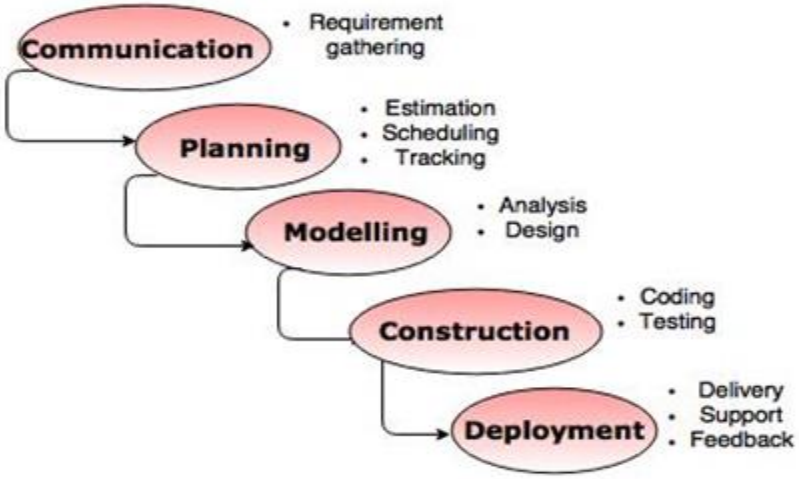
PRESCRIPTIVE PROCESS MODELS

Prescriptive process models define a prescribed set of process elements and a predictable process work flow. Prescriptive process models were originally proposed to bring order to the chaos of software development. All software process models can accommodate the generic framework activities, but each applies a different emphasis to these activities and defines a process flow that invokes each framework activity in a different manner.

❖ WATERFALL MODEL:

- The waterfall model is also called as **'Linear sequential model'** or **'Classic life cycle model'.**
- In this model, each phase is fully completed before the beginning of the next phase.

- This model is used for the small projects.
- In this model, feedback is taken after each phase to ensure that the project is on the right path.
- Testing part starts only after the development is complete.



Advantages of waterfall model

- 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.
- It avoids overlapping of phases because each phase is completed at once.
- This model works for small projects because the requirements are understood very well.
- This model is preferred for those projects where the quality is more important as compared to the cost of the project.

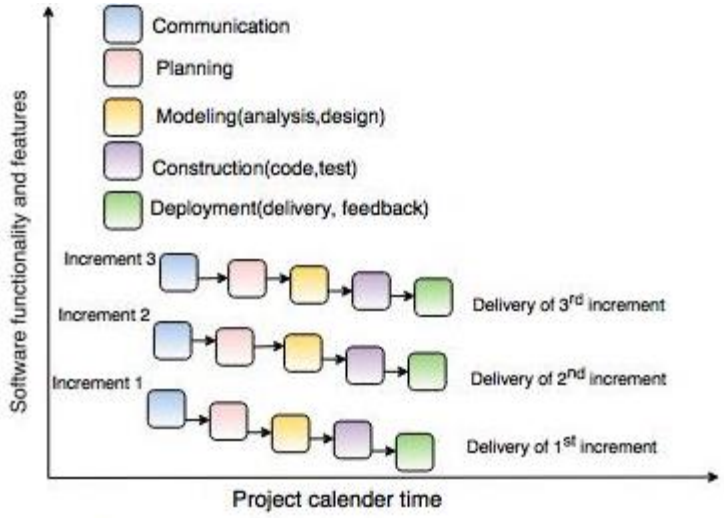
Disadvantages of the waterfall model

- This model is not good for complex and object oriented projects.
- It is a poor model for long projects.
- The problems with this model are uncovered, until the software testing.

❖ INCREMENTAL PROCESS MODEL:

- The incremental model combines the elements of waterfall model and they are applied in an iterative fashion.

- The first increment in this model is generally a core product.
- Each increment builds the product and submits it to the customer for any suggested modifications.
- The next increment implements on the customer's suggestions and add additional requirements in the previous increment.
- This process is repeated until the product is finished.



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.

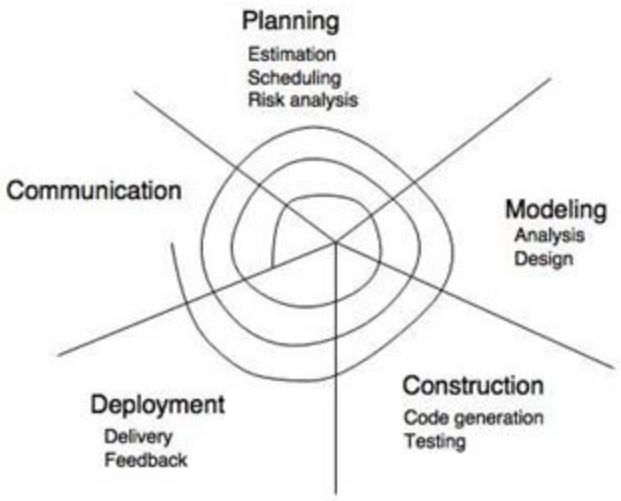
Disadvantages of the incremental model

- 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.

❖ 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.



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.
- The concurrent development 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.

SPECIALIZED PROCESS MODELS

These models tend to be applied when a specialized or narrowly defined software engineering approach is chosen.

- ✓ Component Based Development
- ✓ Formal Methods Model
- ✓ Aspect-Oriented Software Development

Component-based Development Model

Consists of the following process steps

- Available component-based products are researched and evaluated for the application domain in question
- Component integration issues are considered
- A software architecture is designed to accommodate the components
- Components are integrated into the architecture
- Comprehensive testing is conducted to ensure proper functionality
- Relies on a robust component library
- Capitalizes on software reuse, which leads to documented savings in project cost and time

Formal Methods Model

Encompasses a set of activities that leads to formal mathematical specification of computer software

- Enables a software engineer to specify, develop, and verify a computer-based system by applying a rigorous, mathematical notation
- Ambiguity, incompleteness, and inconsistency can be discovered and corrected more easily through mathematical analysis
- Offers the promise of defect-free software
- Used often when building safety-critical systems

Aspect-Oriented Software Development

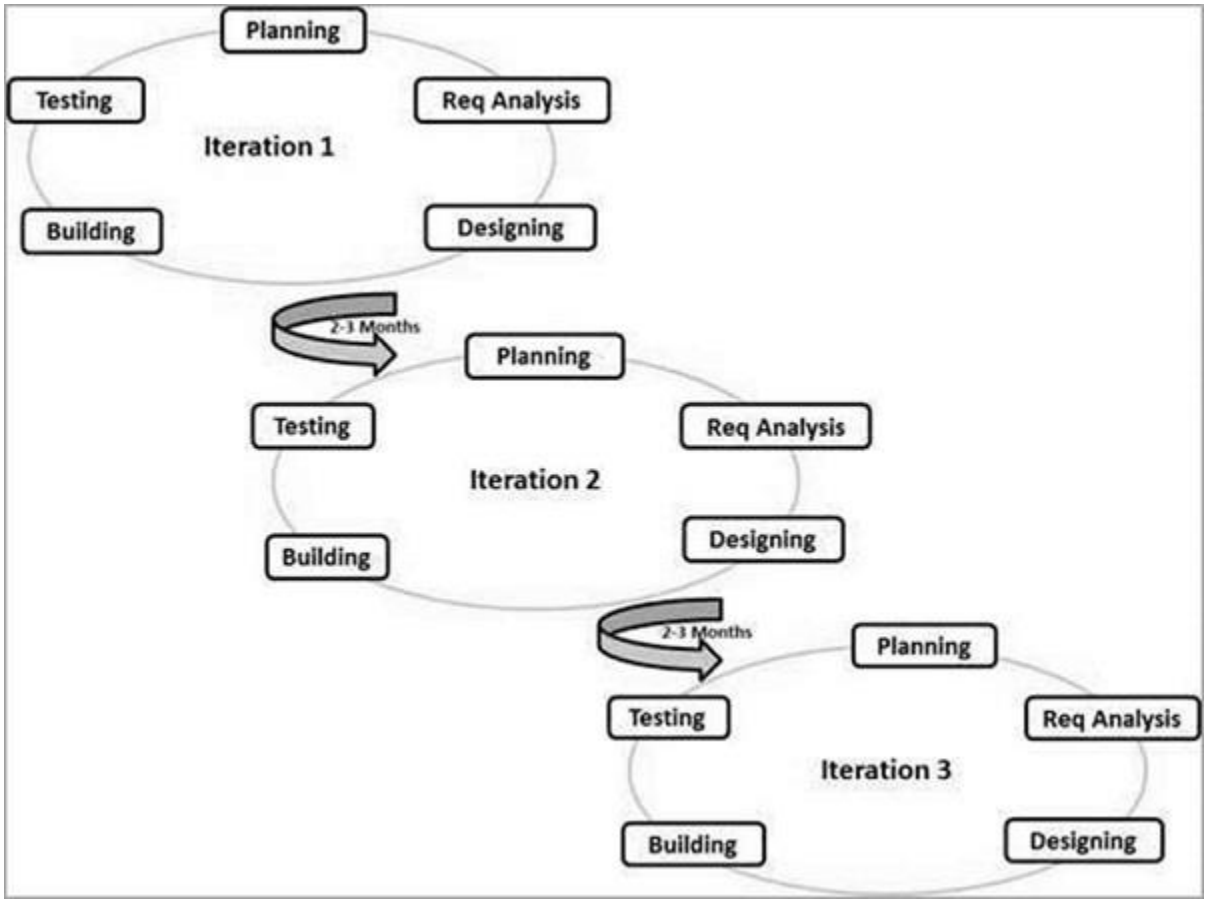
- Regardless of the software process that is chosen, the builders of complex software invariably implement a set of localized features, functions, and information content.
- These localized software characteristics are modeled as components (e.g., object oriented classes) and then constructed within the context of a system architecture.
- Aspectoriented software development (AOSD), often referred to as aspect-oriented programming (AOP), is a relatively new software engineering paradigm that provides a process and methodological approach for defining, specifying, designing, and constructing aspects.

AGILE PROCESS MODEL

Agile model is a combination of incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like −

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and

Here is a graphical illustration of the Agile Model –



Agile Manifesto Principles:

- **Individuals and interactions** − In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

- **Working software** − Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
- **Customer collaboration** − As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
- **Responding to change** − Agile Development is focused on quick responses to change and continuous development.

THE UNIFIED PROCESS

The Unified Process recognizes the importance of customer communication and streamlined methods for describing the customer's view of a system. It emphasizes the important role of software architecture and "helps the architect focus on the right goals, such as understandability, reliance to future changes, and reuse".

Phases of the Unified Process:

The Unified Process is with five basic framework activities depicted as follows-

The Unified Process divides the project into four phases:

- Inception
- Elaboration (milestone)
- Construction (release)
- Transition (final production release)

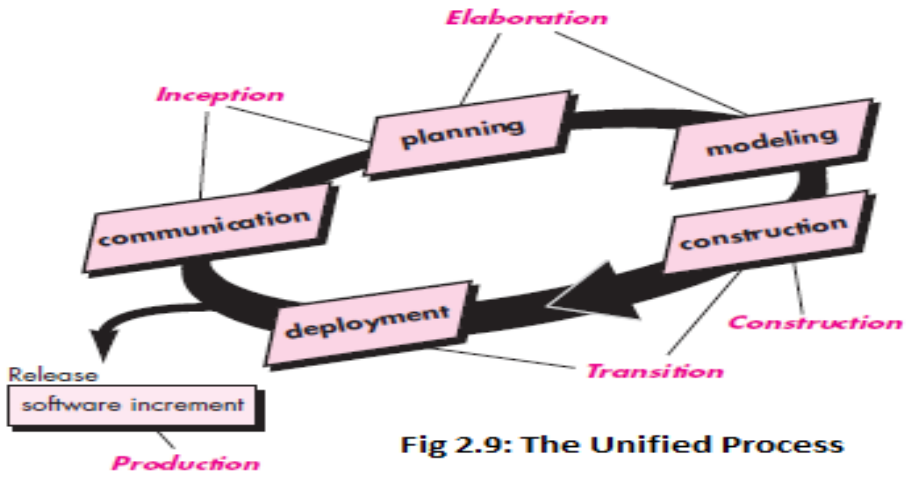


Fig 2.9: The Unified Process

Inception phase

Inception is the smallest phase in the project, and ideally it should be quite short. If the Inception Phase is long then it may be an indication of excessive up-front specification, which is contrary to the spirit of the Unified Process.

The following are typical goals for the Inception phase:

- Establish

- Prepare a preliminary project schedule and cost estimate
- Feasibility
- Buy or develop it

The Lifecycle Objective Milestone marks the end of the Inception phase.

Elaboration phase

- During the Elaboration phase the project team is expected to capture a healthy majority of the system requirements.
- ,Primary goals of Elaboration are to address known risk factors and to establish and validate the system architecture.
- The architecture is validated primarily through the implementation of an Executable Architecture Baseline.
- This is a partial implementation of the system which includes the core most architecturally significant components.
- The final Elaboration phase deliverable is a plan (including cost and schedule estimates) for the Construction phase.

Construction phase

- Construction is the largest phase in the project. In this phase the remainder of the system is built on the foundation laid in Elaboration.
- System features are implemented in a series of short, timeboxed iterations. Each iteration results in an executable release of the software.
- It is customary to write full text use cases during the construction phase and each one becomes the start of a new iteration.
- Iterative implementation for the lower risks and easier elements are done.
- The final Construction phase deliverable is software ready to be deployed in the Transition phase.

Transition phase

- The final project phase is Transition.
- In this phase the system is deployed to the target users.
- Feedback received from an initial release (or initial releases) may result in further refinements to be incorporated over the course of several Transition phase iterations.
- The Transition phase also includes system conversions and user training.

PERSONAL AND TEAM PROCESS MODELS

If a software process model has been developed at a corporate or organizational level, it can be effective only if it is agreeable to significant adaptation to meet the needs of the project team that is actually doing software engineering work.

The PSP and TSP both use proven techniques to improve individual performance and team in software development. Used together they help reduce costs, produce defect-free and inside deadline by empowering development teams. These technologies are based on the premise that a defined and structured process can improve individual work quality and efficiency.

The **Personal Software Process** (PSP) provides engineers with a disciplined personal framework for doing software work. The PSP process consists of a set of methods, forms, and scripts that show software engineers how to plan, measure, and manage their work.
. PSP defines the following framework activities: **Planning, High-level design, High-level design review, Development, Postmortem**

The **Team Software Process** (TSP) guides engineering teams that are developing software-intensive products. Using TSP helps organizations establish a mature and disciplined engineering practice that produces secure, reliable software in less time and at lower costs.
TSP defines the following framework activities: **project launch, high-level design, implementation, integration and test, and postmortem.**

PROCESS TECHNOLOGY

Process technology tools allow a software organization to build an automated model of the process framework, task sets, and umbrella activities. The model, normally represented as a network, can then be analyzed to determine typical workflow and examine alternative process structures that might lead to reduced development time or cost.

PRODUCT AND PROCESS.

If the process is weak, the end product will undoubtedly suffer. All of human activity may be a **Process**. And the outcome of the process is **Product.**

Software is more than just a program code. Software engineering is an engineering branch associated with development of **software product** using well defined scientific principles, methods and procedures which is **process**. The outcome of software engineering is an efficient and reliable software product.