Module 1

1.1

- Software Engineering ✓
- Layered Technology ✓
- Process Framework ✓
- Capability Maturity Model (CMMI) ✓

1.2

- Prescriptive Models ✓
- Waterfall Models ✓
- Incremental ✓
- RAD ✓
- Evolutionary Process Models ✓
- Prototyping ✓
- Spiral ✓
- test Driven Development ✓

1.3

- Agile Process ✓
- Scrum - Industry Perspective ✓
- DevOps Dev Practice.

1.1

* Software and Software Engineering

→ Software comprises of computer programs that provide desired features, functionality and performance when executed.

→ It includes data structures that facilitate the manipulation of info. by the programs.

→ Software encompasses descriptive information, available in both hard copy and virtual forms.

& Types of software.

① System Software

- These are programs written to service other programs.
- For example: compilers, editors etc.
- Processes complex and determinate information structures.

② Engineering software.

- Characterized by number crunching algorithms.
- Used in astronomy, volcanology, molecular biology etc.

③ Application software.

- Stand-alone programs which perform a specific task. Used for addressing business needs.
- Eg. Management decision-making tools.

④ Embedded software.

- Resides within a product or a system, controlling features for end users and the system itself.
- Eg. Automobile digital functions.

⑤ Product-Line software.

- Designed for use by many customers, addressing specific capabilities.

- Eg. Inventory control products, word processing.

⑥ Web Applications (WebApp)

→ Network-centric software offering various functionalities.

- Examples - Simple HTML files to sophisticated computing environments.

⑦ Artificial Intelligence Software

- Utilizes non-numerical algorithms for solving complex problems.

- Example : Robotics, expert systems, pattern recognition etc.

* Challenges in Software Development.

① Growing User Interest. : No. of stakeholders interested in software have significantly increased.

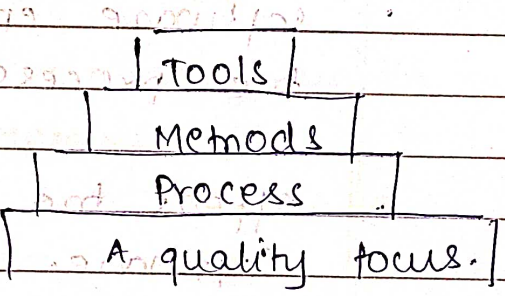
② Complexity of Systems: Modern Tech and products are highly complex, requiring meticulous attention to software elements.

⑤ Consequences of software Failure: Software failures can lead to consequences ranging from minor inconvenience to catastrophic failure.

④ Increasing Percieved value: As the percieved value of specific application grows, their user base longevity increases.

* Layered Framework / Technology

① Quality Focus Layer



- Serves as the bedrock for software engineering
- Embodies organizational commitment to quality, promoting total quality management, Six Sigma.
- A quality focused culture fosters continuous process improvement, leading to the development of increasingly effective approaches.

② Process Layer

- The process layer forms the foundation of software engineering.

- Provides framework for the rational and timely development of computer software.
- Process defines management control, establishes milestones, ensures quality, and manages changes through out the software.

③ Methods layer.

- Software engineering methods, situated above the process layer.
- Offers the technical 'how-tos' for building software.
- Methods encompass various tasks such as communication, requirement analysis, design modelling, programming, testing etc.
- Rely on basic principles covering every area of technology.

④ Tools layer.

- Software Engineering tools provide automated or semi-automated support for software development process and methods.

- These tools facilitate various tasks such as requirements management, design modelling, coding, testing and project management.

- Integration of tools allows for a cohesive system called computer-aided software engineering (CASE).

* Process Framework.

- A generic process framework for software engineering defines five frameworks

a) Communication
b) Planning
c) modelling
d) construction
e) deployment

1) Communication

→ Identify stakeholders and their involvement in the solution

→ Determine unknowns and essential data, functions and features.

→ Consider compartmentalizing the problem for better understanding.

→ Explore graphical representation and analysis model creation.

④ Planning

→ Try seeking patterns and similarities in previous solutions.

→ Evaluate existing software for potential reuse.

→ Define sub-problems and assess their solvability

→ Create an implementation-friendly representation of the solution.

⑤ Modelling

→ Ensure solution aligns with the plan.

→ Verify traceability of source code to design models.

→ Confirm correctness of each component through reviews or ~~proofs~~ proofs.

→ Develop a comprehensive testing strategy, validate against requirements.

* Umbrella activities:

① Project tracking and control.

② Risk management.

③ Quality Assurance.

④ Configuration Management.

→ Task set: Defines the specific work to be done within software engineering actions to achieve objectives.

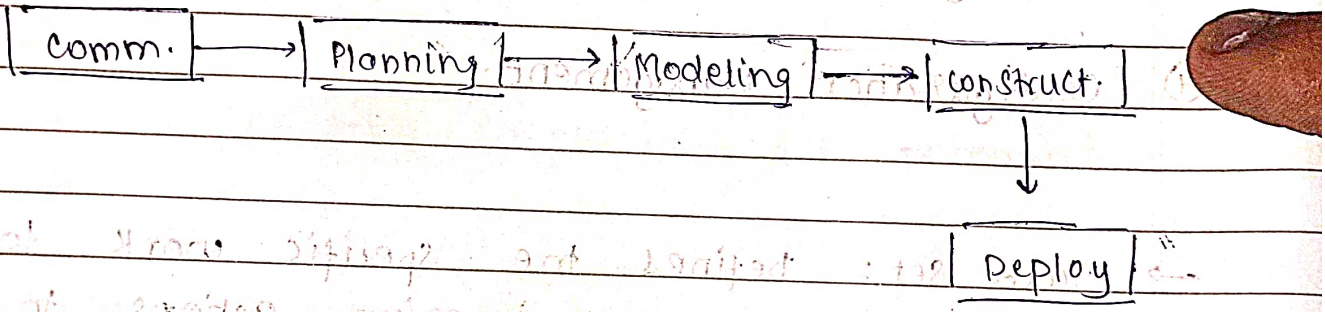
→ Process Pattern: Describes a problem encountered in software engineering work, identifies the environment where it occurs, and suggest proven solutions.

→ Process Flow: Describes how framework activities, actions and tasks are organized in terms of sequence and time.

* Types of Process Flow.

① Linear.

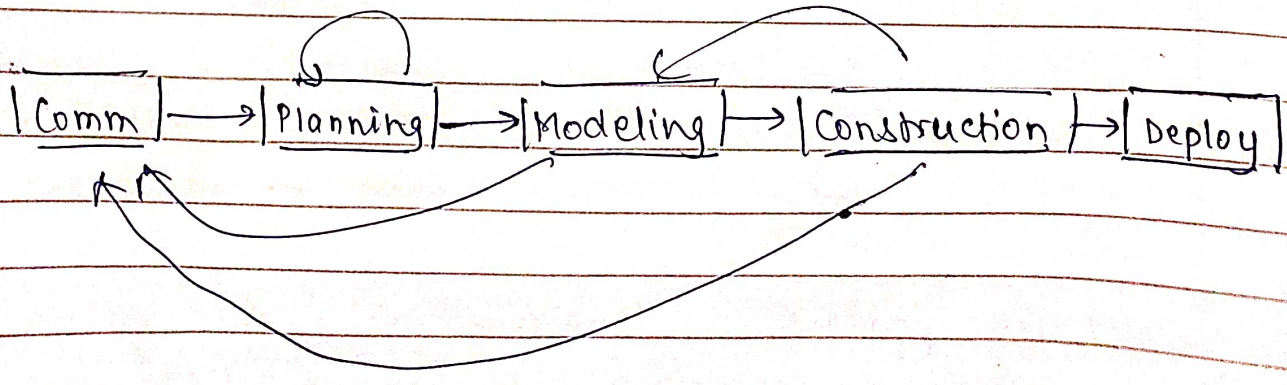
- Executes each of the five framework activities in sequence.



- Follows a step-by-step process without revisiting previous activities.

② Iterative

- Repeats one or more activities before proceeding to the next.
- Allows for feedback and refinement by revisiting activities multiple times.



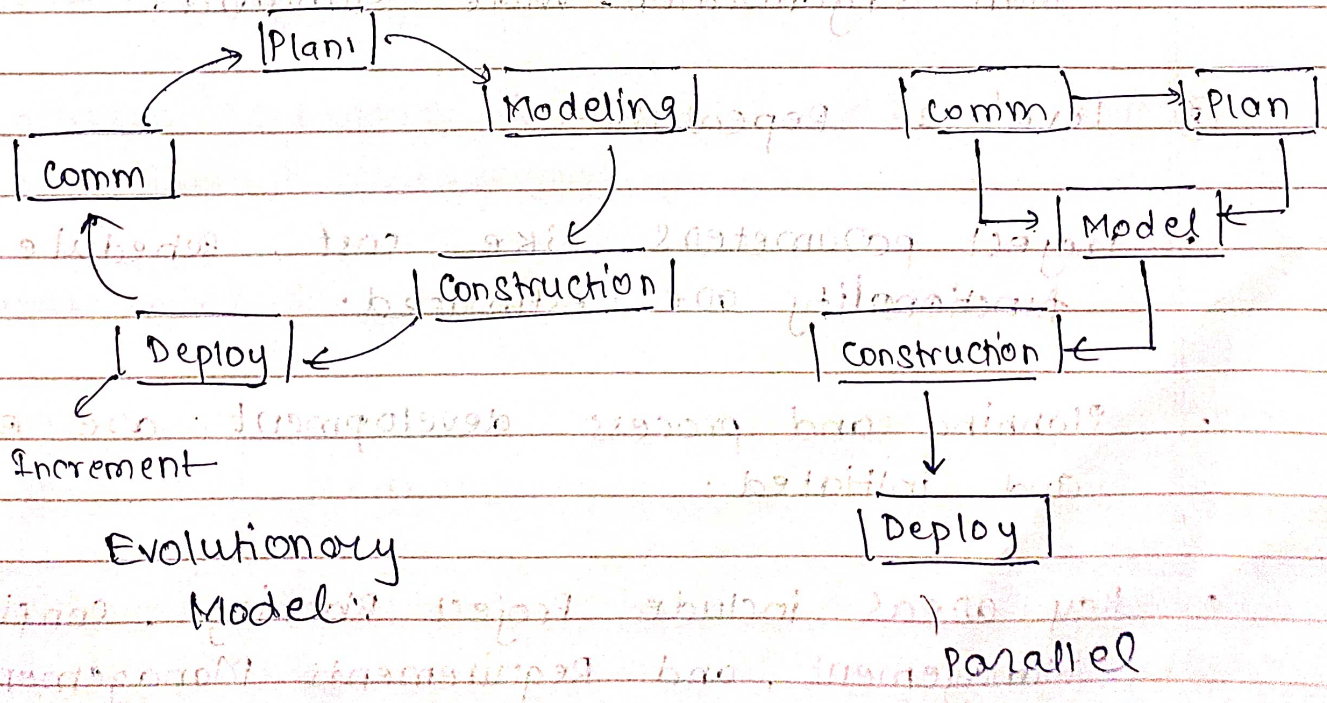
classmate
Date
Page

(3) Evolutionary

- Executes activities in a circular manner, with each circuit through the five activities resulting in a more complete version of the software.
- Focuses on incremental development and continuous improvement.

(4) Parallel.

- Executes one-or-more activities concurrently with others, rather than sequentially.
- Enables teams to accelarate development process.



* CMMI (Capability Maturity Model Integration)

- CMM is used as a benchmark to measure the maturity of an organization's software process.
- Used to analyze the approach and techniques followed by any organization to develop software products.
- Has a 5 level process.

① Level 1 - Initial

- Process is completely unpredictable and ad-hoc.
- Capability is based on individual rather than organization-wide standard.

② Level 2 - Repeatable

- Project parameters like cost, schedule, and functionality are estimated.
- Planning and process development are estimated and initiated.
- Key areas include Project Planning, Configuration Management, and Requirements Management.

classmate
Date
Page

③ Level 3 - Defined

- Standard guidelines and procedures documented
- Integrated set of project specific processes are defined.
- Notable practices like Peer Reviews, Intergroup coordination, organization Process Definition, organization Process Focus, and Training Process.

④ Level 4 - Managed

- Quantitative quality goals are established for both software products and processes.
- Measurement aids in predicting product and process quality within defined limits.
- Includes software Quality Management and Quantitative Management.

⑤ Level 5 - Optimize.

- Focus on continuous process improvement using quantitative feedback.
- Adoption of new tools, techniques and process evaluations to prevent recurrence of known defects.

- Emphasis on Process Change Management, Technology change Management and Defect Prevention.

1.2

* Prescriptive Process Models:

- Prescribe a set of process elements and a predictable process workflow for each project

Types

- Waterfall Model.
- Incremental Process Model.
- RAD Model.

① Waterfall Model.

- Follows a linear sequential approach, progressing through defined phases from comm. to deploy.

- Also known as classic life cycle Model.

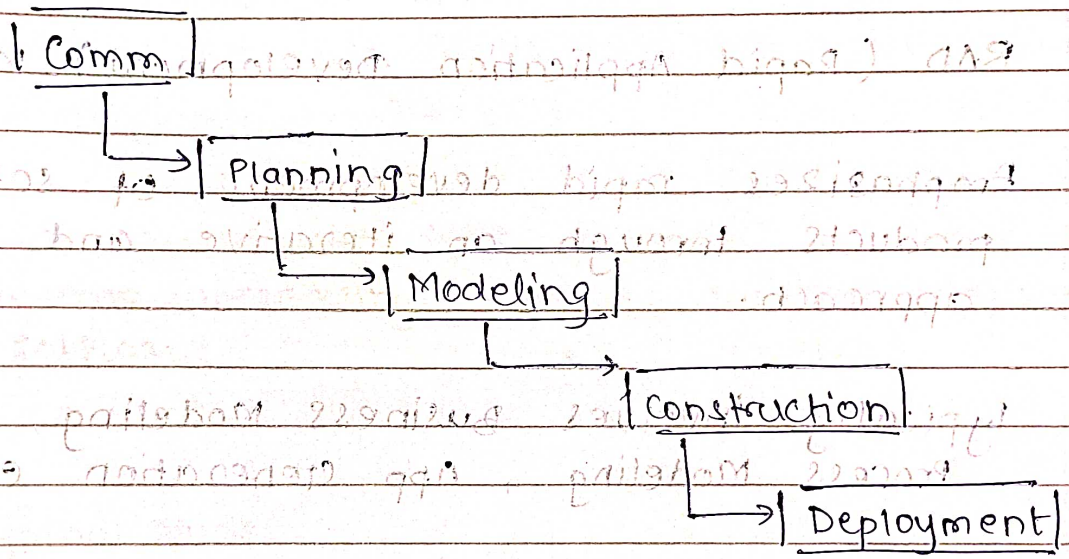
- Best suited for small projects with well-understood requirements at the outset.

Adv.

① Simple and easy to understand
② Prevents overlapping of phases

Disadv

① Not suitable for complex projects
② High risk due to testing being deffered.



Waterfall Model

⑤ Incremental process Model.

- Combines elements of the waterfall model in an iterative fashion, building software in successive increments.
- Each increment delivers a portion of functionality allowing for feedback and adjustment.

Adv.

① Allows flexibility
② Enables frequent testing

Disadv.

① High cumulative costs
② Requires careful management

③ RAD (Rapid Application Development) Model.

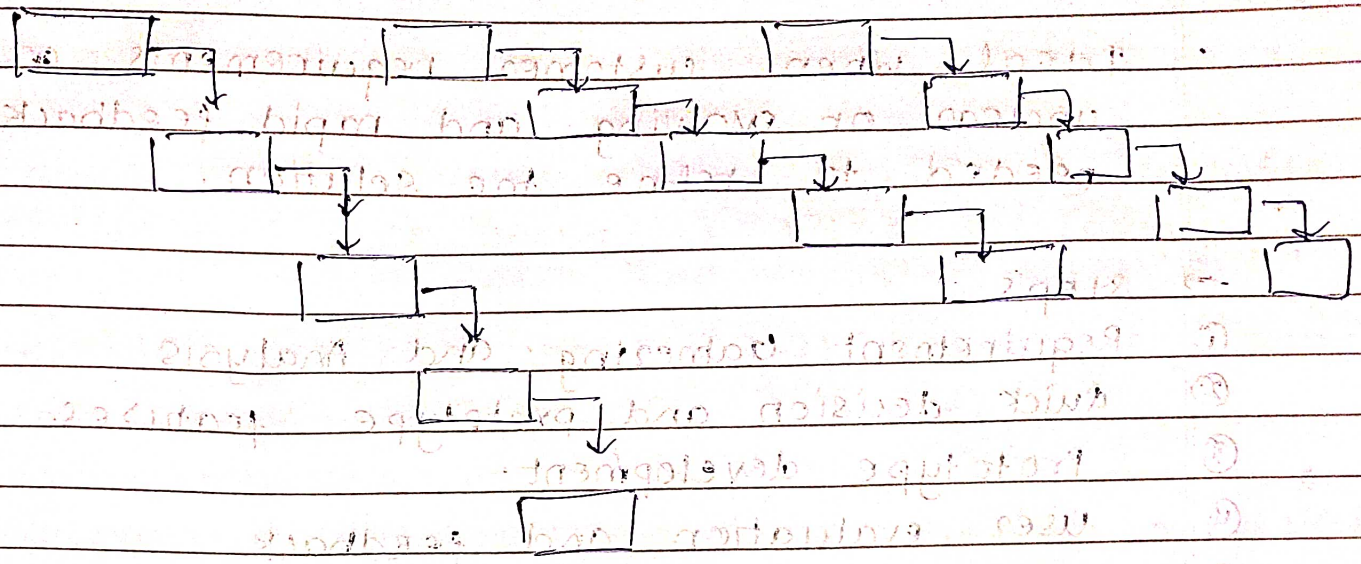
- Emphasizes rapid development of software products through an iterative and incremental approach.
- Typically involves Business Modeling, Data Modeling Process Modeling, App Generation etc.
- Focuses on quick prototyping, frequent user feedback and fast turnaround times.

Adv.

① Allows for swift development
② Early working prototypes

Disadv.

① Requires thorough business analysis
② Complex data and process modeling



* Evolutionary Models

- Iterative in nature that enables software engineers to develop increasingly more complete versions of the software.

Types
- Prototyping Model
- Spiral Model.

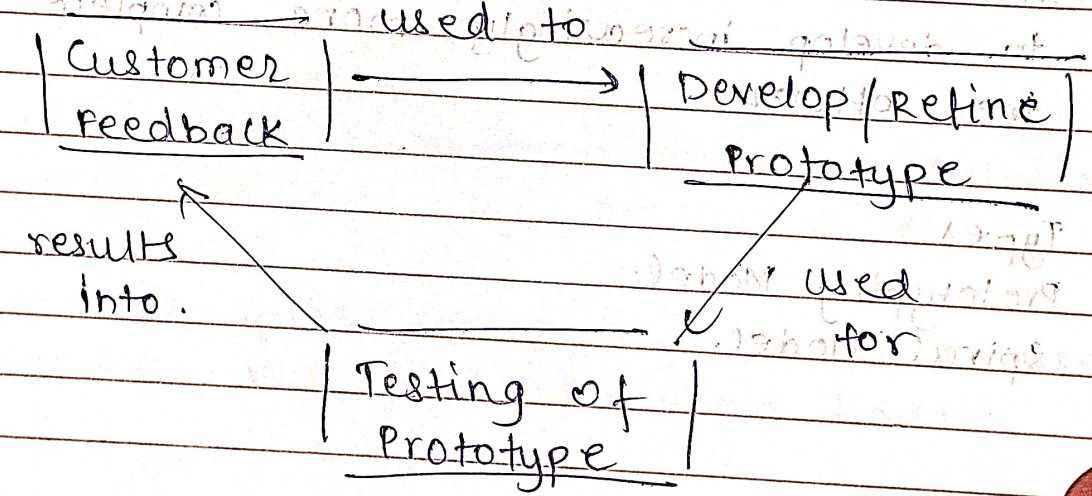
① Prototyping Model.

- Involves developing a working replica of the product to gather feedback and refine requirements iteratively.

Ideal when customer requirements are unclear or evolving and rapid feedback is needed to refine the solution.

→ Steps

① Requirement Gamering and Analysis
② Quick decision and prototype features.
③ Prototype development.
④ User evaluation and feedback
⑤ Prototype refinement based on feedback.
⑥ Engineering final product based on new requirements.



→ Adv.

① Early visibility to customers
② Flexibility in accomodating changes

Disadv

① Time and cost-intensive
② Poor docs due to evolving requirements.

(2) Spiral Model.

- Enables rapid development of increasingly complete software while managing risks effectively.
- Suitable for high risk projects where requirements are complex and subject to change.
- Software development progresses through a series of evolutionary releases, with each iteration providing more complete version of the product.

→ Adv.

(1) Enhanced risk analysis and management
(2) Flexibility to accomodate additional functionality.

→ Disadv.

(1) Costly to use.
(2) Requires specific expertise.

* Test-Driven Development (TDD)

- An iterative development approach where tests are developed first to specify and validate code functionality.

- Process sequence.

a) Add test
b) Ensure new test
c) Write code to pass test
d) Run test case to verify functionality.
e) Refactor code to improve design.
f) Repeat cycle for each new functionality.

- Context of Testing: Include valid and invalid inputs, errors, exceptions, boundary condition and all scenarios that might cause failure.

1:3

* Agile Process. (Khudse aur pabh lena)

- Iterative and incremental approach to software development that prioritises flexibility, collab, customer feedback.

- Emphasizes on delivering small, incremental releases of software rather than large releases.

* Scrum. (Khudse aur Padh lena)

- Scrum is an Agile framework, that provides structure for organizing, planning, and executing Agile projects. It is widely adopted across industries for its simplicity, flexibility and focus on iterative dev.