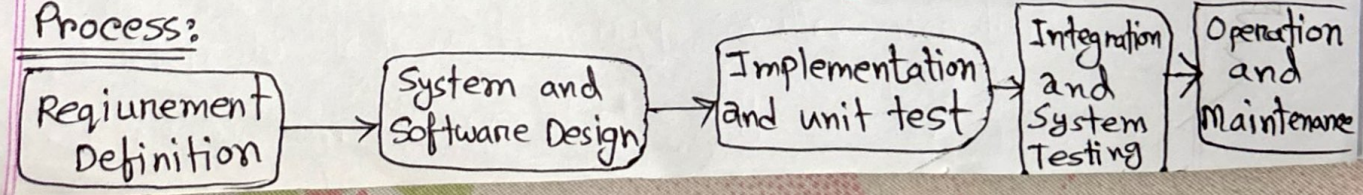
Waterfall Model

- linear and sequential development approach
- A phase cannot begin until previous one is completed.
- No overlapping of phase
- A step-by-step flow
- Output of one phase → input of next phase
- Used for small, well defined project with clear requirements.

Advantages:

- Easy to understand, and implement
- Works well for small projects → Not suitable for large or complex system.
- Easy to manage → changes are difficult to implement.
- End goal is determined early, avoiding scope creep. → Requires complete, accurate requirement which is unrealistic.
- Each phase is well-documented and follows a logical order → No overlapping of phases (slows down development)
- Testing is done after ~~deployment~~ development, reducing confusion. → makes error detection costly
- Basic framework for other models. → Risks are not identified, leading to high uncertainty.
- Predictable timeline due to its structured nature. → Working software is not available for a long time.

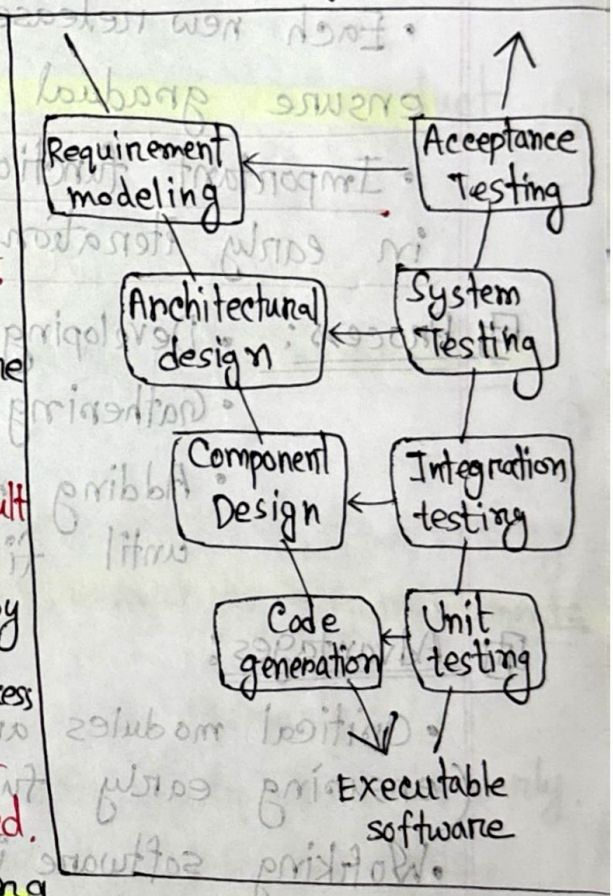
Process:



V Model: (Verification and Validation Model)

- structured and disciplined SDLC approach
- sequential execution - (each phase starts after previous ends)
- Testing is integrated with development (each phase has corresponding testing phase)
- Model forms a V-shape.

Left-side : Represents development activities
Right-side : " testing "
Coding-phase: Acts as the connection of both sides.



Advantages:

- Easy to understand.
  - ↳ Not suitable for complex project.
- Testing activities start before coding, ensuring better quality.
  - ↳ Changes in requirement are difficult to accomodate.
- Follows a structured phase-by phase development and testing process
  - ↳ High risk and uncertainty if requirements are not well-defined.
- Bugs are caught early, reducing cost and effort.
  - ↳ Not flexible for iterative or ongoing project.
- Works well for small and medium scale development.
  - ↳ Not suitable for large/complex project.

Incremental Model

(modules / builds)

- divides development phases into multiple increments
- Each increment follows SDLC phases
- Development happens in small parts, making it suitable for complex projects.
- Modules are delivered in stages, incorporating customer feedback.
- Each new release builds on the previous one, ensure gradual improvement.
- Important functionalities are prioritized and developed in early iterations.

Process:
- Developing initial version with core functionalities.
- Gathering user feedback and refine the system
- Adding more modules in each increment until final system is ready.

Advantages:
- Critical modules are developed first (ensuring early functionalability) → Requires complete clarity / clear requirement
- Working software is available early. → Needs continuous management and coordination
- Easy to modify scope and requirements → requires careful planning and design
- Users can review each module and suggest changes. → Late stage changes may impact overall system integration.
- Project Progress is easily trackable. → total cost may be higher
- Easier to test and debug smaller iterations. → may need to rework
- Errors can be identified early → integration is challenging

Iterative Model

Prototyping Model

Spiral Model.

- Involves repeated cycles (iterations)
- Allowing frequent reviews and changes.

- Process:
  1. Build initial / basic version
  2. Review the version
  3. identify required changes or additional features
  4. Implement changes in next version
  5. Repeat the process until final product meets all requirements.

- Allows continuous improvement.
- Develops in small parts (iterations), flexible for making adjustment.

- Goal: Address drawbacks of waterfall by providing flexibility to adapt changing requirements.

- Advantages:-
  1. Errors / bugs are identified early.
  2. Basic working software is available early.
  3. Easy to test and debug, in iterations.
  4. Customer / User provide feedback
  5. Improves final product quality.
  6. adapts well to changing requirements.
  7. Risks are identified and resolved early.
  8. focus on development, so gives less time on documentation.

- lead to increase effort.
- continuous changes can increase project budget and timeline.
- Managing is complex
- Hard to predict completion date.
- requires careful planning and coordination
- not suitable for small projects

① Prototyping Model

- Throwaway Prototyping
- Evolutionary "
- Incremental "

- create a prototype
- useful when requirement are unclear/not well-defined.
- prototype serves as working model and allows for iterative refinement

Key Features:

- prototype is created based on initial requirements
- final product is developed based on feedback on prototype.
- Can be combined with other SDLC model or standalone approach (use independently)

Challenges:

- If end-user unsatisfied → new version must be create which increase costs and time
- Require careful management of user expectation to avoid excessive iterations.

Advantages:

1. Best for human-machine interaction applications
2. Handles unclear requirements (general obj are known but details are unclear) → first version may include many compromises
3. Help to identify potential risks early.
4. Early user feedback and review. → too many iteration can delay final product delivery
5. Prototype helps developers to evaluate algorithm performance and OS adaptability.
   - Developer may take shortcuts that harm the final system quality to get prototype working.
   - Not suitable for projects requiring strict documentation and planning

Rapid Prototyping Techniques

(Quick low-quality code to demonstrate functionality, test algorithms or interaction)

(i) sketches
(ii) Paper prototypes
(iii) Wireframes and Digital Mockups
(iv) Throwaway Code
(v) 3D protoyping (For Hardware & VR systems, IOT, game development)

~~(2)~~ Spiral Model

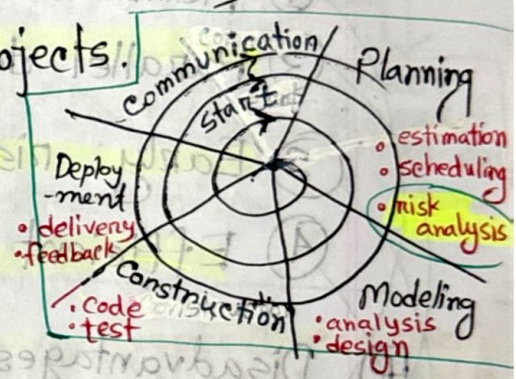
(Risk analysis/Handling) + prototyping + Waterfall) Models.

Helps to identify potential risks early.

- combines feature of (Iterative
- Ideal for large and complex projects.
- Process is visualized as spiral shape starting from central point, make multiple loops around it.

→ number of loops depend on project size and changing requirements.

→ Each loop is a phase of development, with more complete version of software.



Key feature: Risk Handling

→ Risks are identified and resolved using prototyping.

Advantages:
- flexible - additional changes and functionalability can be made at later stages → difficult to manage.
- Best for large and complex projects. → Not suitable for small projects.
- Easier to estimate project cost. → High development cost
- Risk analysis at each phase, reducing uncertainities. → success depends on accurate risk analysis

same as previous

Concurrent Model

- Allows different phases of s to happen in parallel.
- Each activity (analysis, design, coding, testing) is represented as a state —
  - → Under development
  - → " review
  - → Awaiting changes
  - → Approved

Advantages :-

1. Flexible, Allows requirement changes during development.
2. Parallel development (faster progress, multiple team works)
3. Early risk identification and mitigation.
4. Efficient resource utilization.

Disadvantages :-

- Needs strong project co-ordination.
- Requires careful planning of integration
- Syncing parallel tasks can be difficult.
- High communication overhead. (lot of time & effort spent on communication)

When to Use?

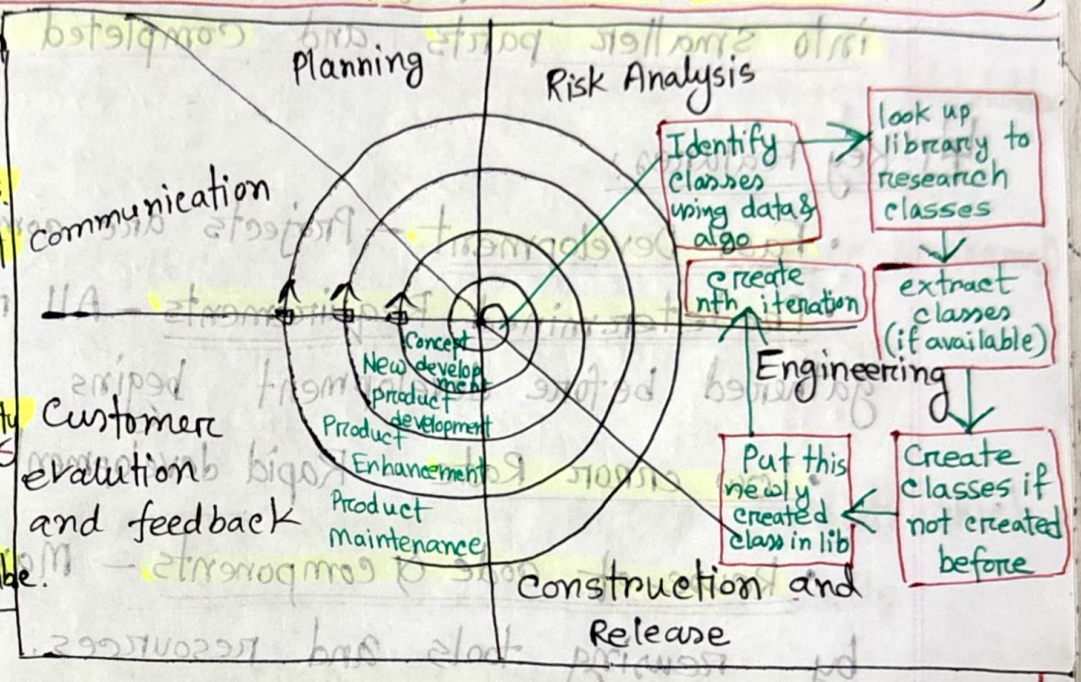
- Best for Large, complex system (Aerospace, Defense, Banking)
- Agile environments, where teams develop and release updates continuously.
- Projects requiring frequent changes and quick iterations.

Component Assembly Model

- Base on reusing existing software components
- It follows "develop once, reuse multiple times"
- This model assembles software systems using pre-made components (modules, APIs, frameworks, libraries)

When to Use?

- When components fits the project needs.
- When fast development is required
- For large, scalable system where modularity is important.
- Where framework and APIs are widely available.



Advantages:

- Faster development by reusing components. → hard to find perfect component match.
- Reduces cost as components are pre-built. → licencing fees may apply for 3rd party components.
- Components are tested and reliable. → (may not integrated well) ← Compatibility issues when integrating multiple components.
- Easier updates (Maintenance) by replacing/upgrading components. → Risk of dependency on third-party providers. (may become outdated or unsupported)
- Flexibility, ~~modula~~ allows easier modification → less control over 3rd-party components internals. (may have to adapt to how the component works)

Rapid Application Development (RAD) Model :-

- fast-paced approach (দ্রুত গতি)
- Similar to Incremental and Waterfall Models, but focused on speed & flexibility, quick delivery.
- Best for small projects. Large projects are broken into smaller parts and completed iteratively.

Key Features:

- **Fast Development** – Projects are completed quickly
- **Predetermined Requirements** – All requirements are gathered before development begins
- **Low error Rate** – Rapid development reduces errors.
- **Reuse of code & components** – Maximizes efficiency by reusing tools and resources.
- Ensures timely delivery while maintaining quality and flexibility.

Advantages:

- **Speed:** • Reduces development time significantly.
- Components are reused, reducing redundancy
- Flexibility: easy to modify and update
- Best for small, well defined projects.
- Prototype-based approach minimizes defects/errors.
- Achieve high productivity with fewer people.
- Reduces overall project costs.

Disadvantages:

- Requires highly skilled developers and designers.
- → Not suitable for complex, long term project
- → difficult to manage.
- Requires constant customer feedback
- → Works well for component-based systems
- → Not ideal for unclear requirement and frequently changing requirement

• Uses high level code, make it portable (easy to transfer) → automated code generation tools can be expensive

Agile Models → XP, SCRUM

Agility is the ability to adapt quickly and effectively to changing requirements.

Key Character: encourages effective communication among stakeholders
- Involves customer onto the development team.
- Organized team (they are in control of work performed)
- Rapid, incremental delivery of software.

Why agility necessary?

- **Uncertainity of requirements**: hard to predict which requirement will change or which will be stable.
- Design and construction are interleaving. (মানে হচ্ছে, construction ও coding একসাথে করা) difficult to predict how much design necessary before construction is ~~des~~ used. Design and construction / implementation happen side by side.
- **Unpredictability**: Planning for analysis, design, coding and testing often inaccurate.

Agile Process:
- Driven by customer-described scenario
- Recognizes that 'plans are short lived'
- Iteratively develops with focus on construction activities
- Deliver multiple software increment
- Adapts as change occur

Agility Principles:

- early and continuous delivery to satisfy customer.
- Welcome changing requirements, even late in development.
- Deliver working software frequently. (frequent delivery)
- Business stakeholders and developers must work together. (collaboration)
- Working software is the primary measure of progress. (progress)
- Promote sustainable development.
- most effective method to share information is face-to-face conversation.
- Provide support and trust to the individuals.

Human Factor:

- Competence → skilled and capable team members
- Common Focus → Goals and objectives.
- Collaboration
- Decision-making ability
- Fuzzy problem solving ability
- Mutual Trust and respect
- Self organization → Team manages their own tasks and responsibilities

Extreme Programming (XP)

An agile framework that emphasizes high-quality software through

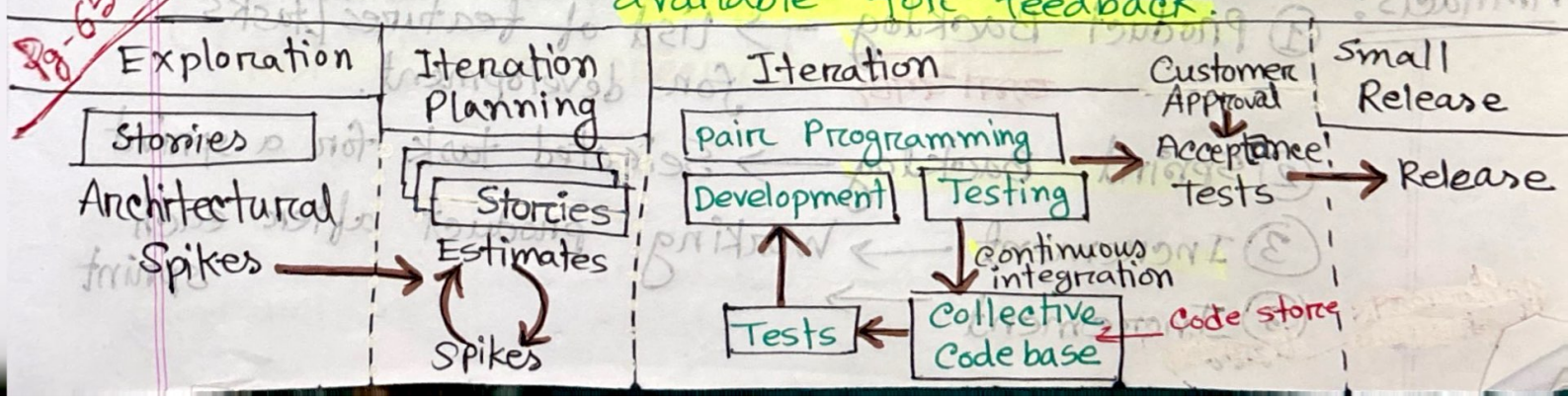
- Frequent Releases
- Continuous customer feedback
- Technical excellence.

- Best for small to medium-sized teams with rapidly changing requirements.

Key-Engineering Practices:

- (TDD) Test Driven Development – Write test before writing code.
- Pair Programming – Two developers work together at one same code/workstation
- Continuous Integration (CI) – Merge code frequently to avoid integration issues
- Simple Design
- Refactoring – Continuous improvement on existing code.
- Collective Code Ownership – Anyone can improve the codebase.
- Sustainable Pace (40 hrs/work week) – Workload is balanced to prevent burnout.
- On-site Customer – Customer/representive always available for feedback.

Pg-65



Advantages:

- High customer involvement (ensures better product alignment) → requires highly skilled and disciplined developers
- Improves software quality through continuous testing and refactoring → Pair Programming can be costly
- Faster adaption to changing requirements. → Difficult to scale for large teams.

SCRUM

- SCRUM is an agile framework that organizes work in short cycles (sprints).
- emphasizes team collaboration, roles, and structured meetings.

Key elements:

Roles:
1. Product owner → Defines and prioritizes the product backlog (করণীয় কাজ)
2. SCRUM Master → Facilitates the process and removes impediments (বাধা)
3. Development Team → Self-organizing team that delivers the products.

Artifacts:
1. Product Backlog (করণীয় কাজ) → list of features/tasks for development.
2. Sprint Backlog → Selected task for a sprint
3. Increment → Working product after each sprint

Events:
- Sprint Planning → Define what will be worked in the sprint.
- Daily Standup → Short 15 minute meeting for progress updates.
- Sprint Review → Present the completed work.
- Sprint Retrospective → Reflect on what went and what can improve.

Advantages

- Increases productivity and faster delivery of features.
  ↗ Requires dedicated commitement from the team.
- Improves collaboration and transparency.
  ↳ can be difficult to implement in rigid organizational structure.
- Enhances adaptibility to changing requirements.
  ↳ Risk of scope creep if backlog is not well managed.

XP Vs SCRUM

Feature	XP	SCRUM
Focus	Engineering practices	Project management
Iteration length	1-2 weeks	2-4 weeks
Team size	Small to medium team	Medium to large team
Customer involvement	High (on-site customer)	Indirect (via product owner)
Key Practice	Pair Programming TDD	Daily stand-up Sprint planning.

XP Vs Scrum Roles:-

Role	XP	Scrum
Customer	On-site Customer (actively involved daily)	Product owner (manages backlog, represented stakeholders)
Coach	Guides the team on XP Practices	Scrum Master (ensures Scrum process, removes impediments)
Tracker	Monitors project progress and team performance	Not a defined role in Scrum
Developer	Write codes, follows TDD, pair programming and collective ownership.	Development team
Tester	No seperate tester role. (Developers test using TDD)	No separate tester role (Developer tests).

When to use XP?

→ You need strong engineering practices (TDD, refactoring, pair programming)

→ Customer involvement is very high

→ Requirement change frequently.

When to use Scrum?

→ You need structured project management approach.

→ The team is larger and cross functional.

→ Buisness stakeholders drive priorities.

Advantages of Agile Models:

- Flexibility for changes on customer feedback,
- Customer satisfaction, so that final product meet their nee
- In short iterations, allow faster delivery
- Improve quality because of frequenting iteration.
- Better risk management (Early error detection)
- Enhanced Team collaboration - Daily stand-ups, feedback loops improve communication.
- Better efficiency for regularly.

Disadvantages:

- lack of Predictability — changing requirements can make project timelines and cost difficult to estimate
- Scope creep — constant changes may lead to uncontrolled growth in project scope.

- Requires skilled Team members (self-organizing, disciplined, experienced team)
- Less document → can cause issues in large projects.
- Not suitable for All projects → large scale projects will strict regulatory / security requirement may not fit well with agile.
- Highly customer Involvement Required → may not be feasible.
- Integration Challenges → continuous changes may lead to integration issues, requiring more testing and refactoring.